CCN Forwarding Strategies

A Dissertation Presented
by
Ahmed Waliullah Kazi

to
The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of
Doctor of Philosophy
in
Computer Science
Stony Brook University

August 2015

Stony Brook University
The Graduate School

Ahmed Waliullah Kazi

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Dr. Hussein G. Badr - Dissertation Advisor
Associate Professor, Department of Computer Science

Dr. Samir R. Das - Chairperson of Defense
Professor, Department of Computer Science

Dr. Ellen Liu - Committee Member
Research Assistant Professor, Department of Computer Science

Dr. Thomas G. Robertazzi - Outside Committee Member
Professor, Department of Electrical & Computer
Engineering, Stony Brook University

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

CCN Forwarding Strategies

by

Ahmed Waliullah Kazi

Doctor of Philosophy

in

Computer Science

Stony Brook University

2015

An active area of Future Internet Architectures research is Information Centric Networking (ICNs). Content Centric Networking (CCN) is one of the widely known ICN proposals. ICNs propose a communication paradigm in which named content is the focal point. A fundamental component of ICNs is “in-network caching”, whereby routers act as content caches. The routers implement a “forwarding strategy” for packets that makes intelligent decisions taking several factors into account, not the least of which is leveraging the presence of cached content in the network. From early on, two basic forwarding strategies were proposed for CCN, CCN-Flooding and CCN-Publication.

We present a performance study of CCN-Flooding and CCN-Publicationbehavior and bandwidth consumption. We analyze the interplay between two important components of CCN, in-network caches and the Pending Interest Table (PIT), under CCN-Publication. We show that the effect of PIT aggregation increases commensurately with network load and the Betweenness Centrality attribute of the network topology. While both PIT aggregation and in-network caching contribute significantly to savings in network bandwidth consumption, we show that they do not do so in an additive fashion. As network load increases, gains from cache hits diminish and are replaced by gains from PIT aggregation, without dramatically impacting the level of overall savings achieved.We also identify and analyze various issues pertaining to CCN-Flooding, such as calibration of PIT timeouts; a PIT-induced isolation effect that negatively impacts bandwidth consumption and response time; and the effects of adopting FIB routes based on volatile in-network cache entries. Finally, we analyze CCN-Publication taking into account, unlike the current research literature, the bandwidth costs of populating the FIBs, and compare that to CCN-Flooding.

Next, we propose a new lightweight forwarding strategy for CCN. One of the primary objectives for a forwarding strategy is to improve user experience by exploiting in-network caching, while minimizing the associated costs. To this effect, we compare the new forwarding strategy, called CCN-FOH, against CCN-Publication and CCN-Flooding. The results show that CCN-FOH provides for a better user experience while keeping its overheads comparable to, if not better than, the less costly CCN-Publication.

Dedicated to Ami and Daddy Saeen; Maahera, Khadijah, and Ismaeel; Humera, Ayesha, and Umer; andFaiz.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Limitations of the Internet Architecture

The design principles which defined the underlying architecture of the Internet were conceived in the 70's [1, 2]. It is safe to say that the Internet has far exceeded the expectations of its creators. Even they could not have foreseen how the Internet would become a global phenomenon that would revolutionize the world, and impact every phase of our daily lives. Nevertheless, from the hindsight of forty years' experience, we can now see that these design principles were both a major reason for the success of the Internet, but also contribute to its limitations as a platform for further development into the future.

Since the inception of the Internet, it has evolved over time to overcome various challenges. The research community had to address these challenges within the constraints of the Internet's underlying architecture. This so-called "Evolutionary" approach takes the current Internet as the starting point for tackling its shortcomings and challenges [3]. Encrypted sockets (*i.e.*, TLS) [4], DiffServ [5], IntServ [6], CDNs [7] [8], NAT [9], MobileIPv4 [10], *etc.*, are all ad-hoc solutions which exemplify the Evolutionary approach. Over the years many such solutions, which are sometimes no more than "patches", have been added to the architecture which has gradually evolved into the shape shown in Figure 1-1. This figure illustrates an important point: the Internet architecture is complex. A lot of the protocols, especially in the Control Plane, overlap with different layers which makes it difficult to grasp the overall architectural structure. Designing further modifications to plug into such an architecture is a daunting task.

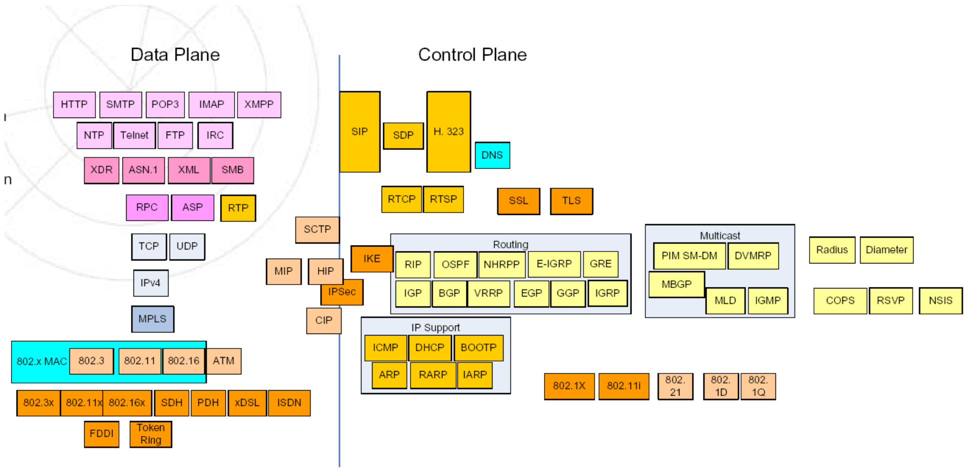


Figure 1-1. Internet Architecture Out of Shape (taken from [11])

One research direction that seeks to address such challenges is the so-called “Clean Slate” approach. There is a wide debate within the research community as to the appropriate approach forward (Evolutionary *vs.* Clean Slate) for future network architecture [12, 13]. Proponents of the Clean Slate approach [14, 15] argue that the challenges facing the Internet today and for the foreseeable future need to be addressed without the design constraints of the current architecture. The Clean Slate approach allows for the designing of new basic protocols and network architectures for the future Internet which would otherwise not be possible. This approach is not necessarily contradictory, but rather can actually be complimentary, to the Evolutionary approach. The solutions available under a Clean Slate approach are much broader and more novel than for an Evolutionary approach, and so can also assist in guiding the redesign of the current Internet architecture.

1.2 Information Centric Networks

The Internet is popular due to the content that can be accessed on it. One Clean Slate approach that is currently the focus of much research interest is Information Centric Networking (ICNs) [16-20]. In ICNs, the focus shifts from the physical machines where content is located to the content itself. This involves a shift from the traditional host-to-host communication paradigm to an object-based communication paradigm in which content is the focal point. Consequently, we no longer deal with IP addresses but only with content names. There are several potential benefits to such architectures: security; multicasting; elimination of redundant traffic; no mobility management – which is inherently built into the network; routing to the nearest content copy; etc.

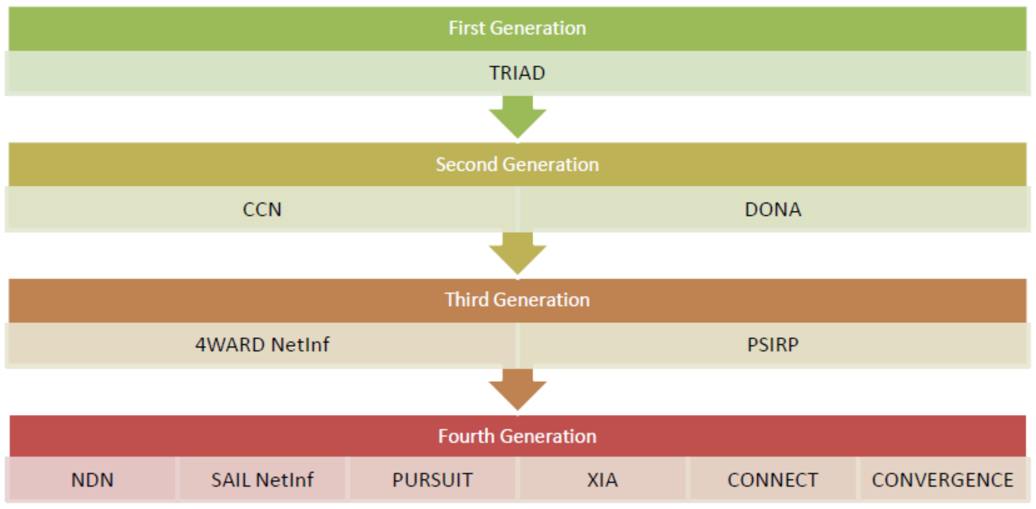


Figure 1-2. Classification of ICN Initiatives

There are several projects aimed at developing this new content-based paradigm. We classify these as First, Second, Third, and Fourth Generation on the basis of their: (a) time of origin, and (b) contribution to subsequent ICN proposals (Figure 1-2). In the First, precursor, Generation, TRIAD (1999) [21] was one of the earliest proposals to adopt something of an Information Centric approach, but not in a radically fundamental way. It introduced the concept of content routing on top of the IP layer. CCN (2006) [22] is a Second Generation initiative which introduced the fundamental Information Centric paradigm shift in which the focus moved from content location (“where”) to the content itself (“what”), and proposed the replacement of the IP layer by a content-centric network layer. DONA (2007) [23] is also a Second Generation initiative, inspired by TRIAD. It proposed the notion of flat, self-certifying names for content which are resolved using anycast over an overlay network on top of IP. Prominent Third Generation proposals (2008 – 2010), such as PSIRP [24] and 4WARD NetInf [25], build on the efforts of the previous two generations by proposing scalable mechanisms for the intradomain and interdomain levels. Direct descendants of each of PSIRP, 4WARD NetInf, and CCN in the Fourth Generation (2010 –) are PURSUIT (http://www.fp7-pursuit.eu), SAIL NetInf (http://www.sail-project.eu/), and NDN (http://www.named-data.net/), respectively. These, along with other similar initiatives such as Expressive Internet Architecture (http://www.cs.cmu.edu/~xia/), CONNECT (www.anr-connect.org/) (which complements CCN), and CONVERGENCE (http://www.ict-convergence.eu/) may be classified as Fourth Generation to the extent that they build on preceding efforts.

While an object-based approach defines, and is therefore common, to all ICNs, different proposals adopt varying other components as fundamental building blocks of their individual design architecture. Security, for example, is one such component. Two tightly-coupled components, however, that are at the heart of all ICN architectures are naming and name resolution [26]. In an architecture that places content at its core, naming objects becomes the central element, analogous to the centrality of IP addressing in today’s Internet for example. Names define how objects are identified, and in principle may be loosely classified as flat *vs*.

hierarchical, persistent *vs*. non-persistent, aggregatable *vs*. non-aggregatable, human-readable *vs*. human-unreadable, *etc*. Name resolution (*i.e*., locating an object given its name), on the other hand, needs to handle billions of object-to-location bindings in a scalable, speedy, and efficient manner. In a given ICN architecture, the potential solutions available for name resolution are a direct function of the specific naming scheme adopted for objects.

We shall use examples from various ICN proposals to highlight the range of naming and name resolution design alternatives. In line with the work presented in this thesis, we limit our discussion to salient considerations of naming and name resolution only, and leave aside a discussion of other aspects of these ICNs. As stated above, many ICN initiatives have put security as a central design concern, which has implications for the kind of naming (*e.g*., self-certifying names, *etc*.) and name resolution schemes they propose. This security aspect is well-surveyed in [27] and will not be included in our discussion.

1.2.1 Translating Relaying Internet Architecture Integrating Active Services (TRIAD)

Naming. Objects in TRIAD are identified using URLs. This implies non-persistent names. Object names are dependent on their location; if the location changes, the name changes accordingly. The names are hierarchical and are not aggregatable beyond two-level domain names, *e.g*., www.ieee.org.

Name Resolution. In TRIAD, content routers (CRs) perform conventional IP routing tasks and also act as name servers. When the Content Layer (immediately above the IP layer) of the client requests for a URL to be resolved, the request traverses a series of CRs which maintain name-to-next-hop mappings similar to traditional IP routers. TRIAD is based on two protocols: (a) Internet Name Resolution Protocol (INRP), which performs the lookup based on the URL requested, to find the IP address of the “best” content server; and (b) Name-Based Routing Protocol (NBRP), similar to BGP except that it advertises URL reachability information to the CRs, and is responsible for maintaining the routing tables needed by the INRP. Certain name servers can operate as content caches, directly providing the content requested.

1.2.2 Content Centric Networks (CCN)

Naming: CCN proposes usage of URI-like, hierarchical, aggregatable, globally-unique names for content. For example,

/cs.stonybrook.edu/akazi/paper.pdf/_v<timestamp>/_s3,

is a valid name which is self-explanatory in terms of what it represents: a PDF file named “paper”, owned by user akazi. The name is divided into three components: (a) a globally-routable name (/cs.sunsysb.edu); (b) an organizationally-unique object name (/akazi/paper.pdf); and (c) versioning and segmentation information (/_v<timestamp>/_s3).

Name Resolution. There are two types of packets in CCN: (a) interest packets, which issue a request for content using the hierarchical content name; and (b) data packets that contain the content requested. In place of IP routers there are Content Routers (CRs). The forwarding model at a CR makes use of three main data structures as seen in Figure 1-3. The Content Store (CS) is used to cache content. The Pending Interest Table (PIT) stores Interest packets that could not be served by the CR, and were therefore forwarded upstream. An entry in the PIT consists of the content name in the interest packet and the router interface that packet arrived on. Subsequent interest packets for the same content, but received on different interfaces of the CR, are maintained in the same PIT entry by adding the new interface to the entry. The Forwarding Information Base (FIB) is similar to the IP forwarding table; but since content can be simultaneously located at multiple locations, a FIB entry identifies multiple outgoing interfaces known to lead to copies of the content.

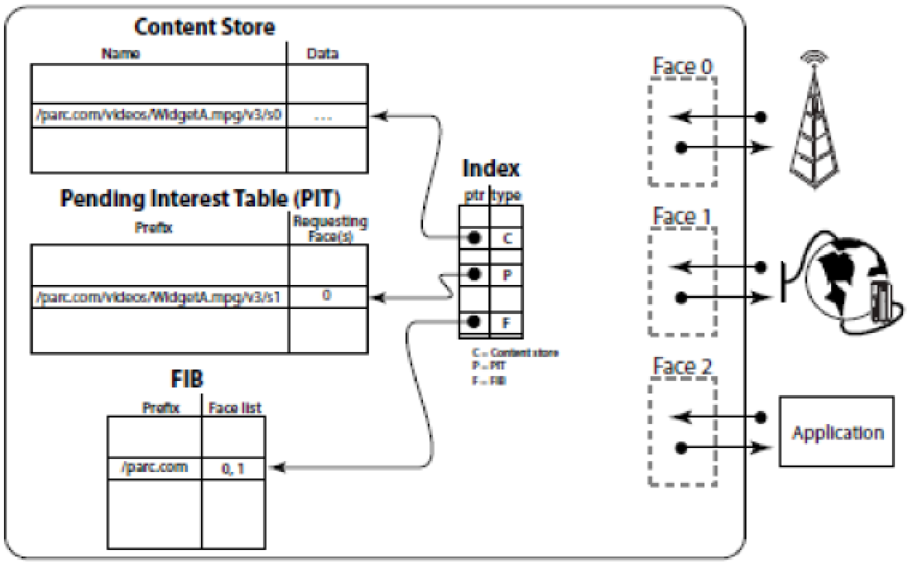


Figure 1-3. Data Structure in CCN Nodes (taken from [22])

Content forwarded along the path between the providing and requesting nodes is cached in the CS at intermediate CRs. Thus, subsequent requests for the same object may be resolved by cached content at intermediate CRs serving an interest packet, instead of forwarding the packet all the way to the original provider node for the content. Furthermore, to control redundant upstream request traffic, interest packets for the same content are suppressed at the first intermediate CR with a PIT entry for that content. Details of the forwarding operation for Interest packets are shown in Figure 1-4.

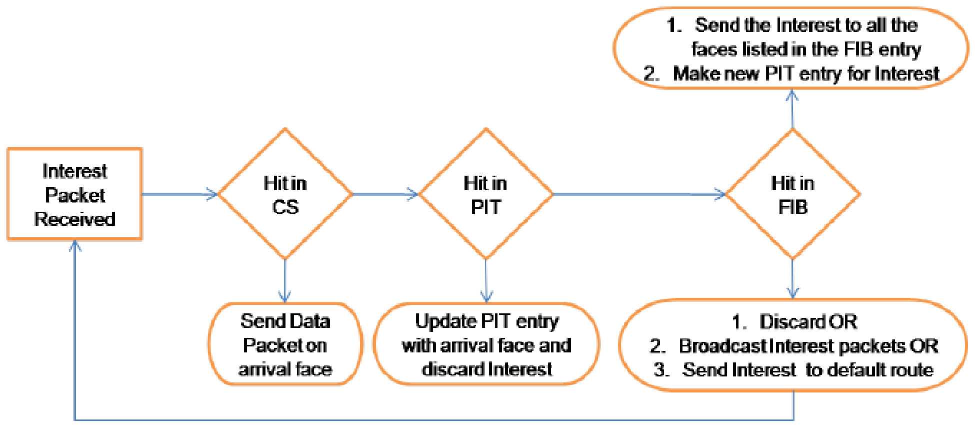

1. Send the Interest to all the faces listed in the FIB entry
2. Make new PIT entry for Interest
Interest Packet Received
Hit in CS
Hit in PIT
Hit in FIB
Send Data Packet on arrival face
Update PIT entry with arrival face and discard Interest
1. Discard OR
2. Broadcast Interest packets OR
3. Send Interest to default route

Figure 1-4. Flow Chart for the CCN Node Forwarding Operation

Returning data packets are forwarded along all interfaces registered in the PIT entry for that content; the PIT entry is purged; and an entry is made in the FIB for the content and the interface off which the data packet arrived. Name resolution and routing operations are thus amalgamated. Interest packets are routed (or flooded, if there is no entry in the FIB for the content), while data packets piggyback using the PIT entries.

1.2.3 Data-Oriented Network Architecture (DONA)

Naming. DONA uses globally unique, flat, self-certifying names for content. The components of a name are <**P**rincipal: **L**abel>, where Label (*e.g.*, abc.html) is chosen by the owner, while Principal is a cryptographic hash of the owner's public key. A name has no association with where its content is located. If that object were moved to a different location, it would still retain the same name. In this respect, a name is persistent and does not change due to mobility/migration. However, if the owner of the object changes, the object name would have to change even though it remains the same object with respect to its content, and in this more fundamental respect names cannot be said to be truly persistent.

Name Resolution. DONA uses name-based routing to locate the closest copy of the content, using anycast service. Instead of DNS, it has Resolution Handlers (RHs) which are similar to Content Routers in TRIAD. Every domain needs at least one RH, similar to a local DNS server. The resolution system consists of two operations: (a) FIND <P: L>, and (b) REGISTER <P: L>. The former is used to locate content while the latter registers content. The REGISTER message is sent to the local RH – determined in the same way that we locate the local DNS server.

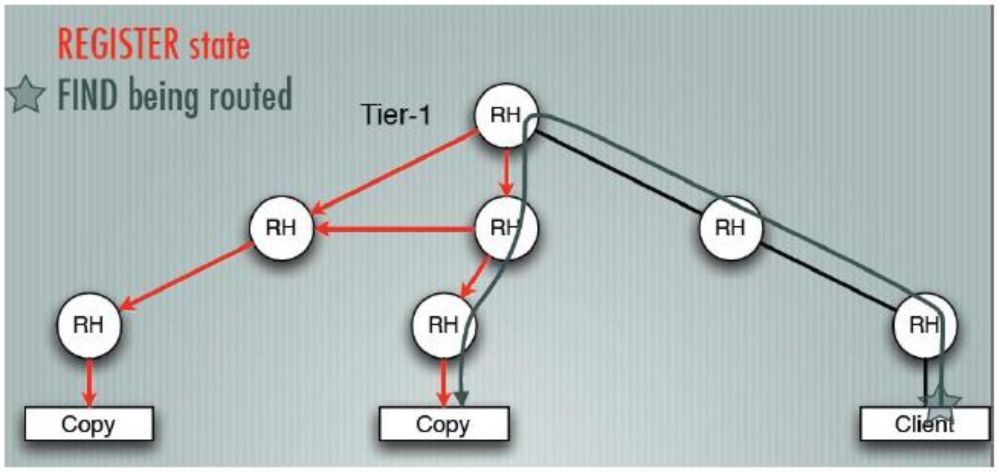


Figure 1-5. FIND and REGISTER Operations (taken from [23])

Each RH maintains a registration table that stores the name and the next-hop RH. The RH updates the registration table and sends the REGISTER message to its upstream peers for global distribution. FIND messages are forwarded to the local RH. If the content is within the local domain then the RH directs the client to the content. If it is not, then the RH forwards the FIND to its upstream peers. RHs route FIND messages to the closest copy, but the corresponding content packets use the IP layer for forwarding. Both operations are shown in Figure 1-5.

1.2.4 Publish Subscribe Internet Routing Paradigm (PSIRP)

Naming. PSIRP proposes an architecture in which Publish/Subscribe operations are supported at every level (network level, application level, *etc*.). There are four identifiers across the various levels of the architecture, as shown in Figure 1-6.

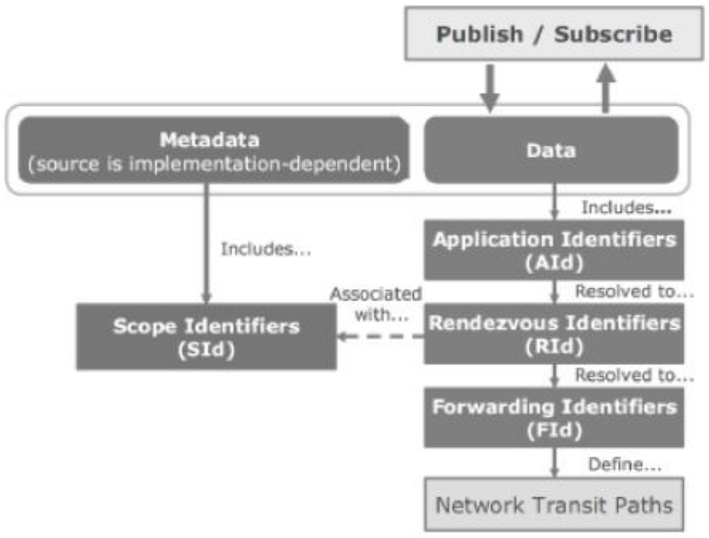


Figure 1-6. PSIRP Identifiers with Network Functionality (taken from [28])

Application Identifiers (AId): PSIRP gives applications the freedom to provide application-specific names for content which are not globally unique. Hence, any naming scheme can be mapped on to the PSIRP architecture, for example, hierarchical naming as in CCN.

Rendezvous Identifiers (RId): These are flat identifiers which define content (*e.g*., a file) in the network. For example, a PDF file would be assigned a unique RId within a scope.

Scope Identifiers (SId): These are also flat identifiers which define aggregations of RId objects. Content is uniquely identified globally by its RId and the SId of the scope in which it resides. Hence, PSIRP has a “structured” (two-level hierarchical) naming scheme at the global level. Scopes are analogous to network-oriented topologies, but are much richer since they can designate not only IP-like physical scopes, *e.g.*, a university campus, but also logical scopes, *e.g.*, Facebook friends, *etc*. Content can be published in multiple scopes. The same content in different scopes will have a different SId (and possibly RId). Thus, if content migrates it will have a different global Id, rendering names at this level non-persistent.

Forwarding Identifiers (FId): These are flat identifiers which assist in routing the content to the subscriber and are created dynamically when the name for a given object is resolved.

Name Resolution. Published content is represented by RId's. Content is published (with meta-data pertaining to it included) within a scope represented by a SId. A SId thus represents all the RId's within the scope. For each scope, there is at least one Rendezvous Node (RN) which is responsible for managing the scope's RId's. These RId's will potentially be in the billions, since each individual object is assigned one. To achieve scalable reachability information, only the SId's are advertized, and not the individual RId's. SId's are advertized (published) at RNs that are further upstream, *e.g.*, the RNs of the service provider's domain, and intermediate domains until a Tier-1 RN is reached.

A subscriber interested in an item of published content will first need to acquire a SId and RId for the item through search engines, similar to the way we search for relevant website content today. To locate the item, a Subscription with the relevant SId:RId:metadata is relayed to the local RN of the subscriber. If the subscriber is within the same scope (local domain) as the item, then the RN will resolve the Subscription. If not, the RN forwards the Subscription to the next RN upstream, and this process continues recursively until the RN for the scope is found. This RN will then initiate the path discovery between the publisher and subscriber.

1.2.5 4WARD NetInf

Naming. NetInf names are persistent. An object is represented mainly at two levels:

Information Object (IO): Represents information about the object without any association as to its location. Essentially, it is a name which is not topologically bound to a location. It is the higher of the two levels.

Data Object (DO): The actual object itself or the bit-level representation of that object.

Both information and data representations of an object are named using flat identifiers. The components of an identifier are: a tag (to identify the object type); an optional parameter that is a cryptographic hash of the owner's public key; and a mandatory label which is unique.

Name Resolution. NetInf design incorporates flexibility by allowing several levels of indirection before a name is resolved to its locater. It supports different types of bindings which are maintained to inter-relate different representations of objects and their locations: IO → IO, IO → DO, and DO → locaters. Thus, NetInf deals with a lot of information that will be required to resolve an object name and find its location. NetInf introduces the notion of Multiple Distributed Hash Tables (MDHTs) [29] and Late Location Constructors (LLC) for intradomain name resolution; and Resolution Exchange (REX) systems for global resolution. Due to shortage of space, we only discuss the MDHT and REX.

As its name implies, a MDHT maintains several DHTs at various levels in an Autonomous System (AS): Access Node HT; POP DHT; and AS DHT. The binding of object to locater is replicated at each of these levels. Whenever an object name is requested, it is looked up in the following order (see Figure 1-7): (a) Access Node HT, (b) POP DHT, and (c) AS DHT. If the request is resolved at a lower level, it does not need to be propagated upward. On the other hand, if it remains unresolved at the AS DHT level, then the object is not present in the domain, and the request will be forwarded to the REX. MDHTs were chosen for the intradomain level because they are highly scalable and, in a single AS setting, do not pose any issues of mistrust or non-cooperation for key placement.

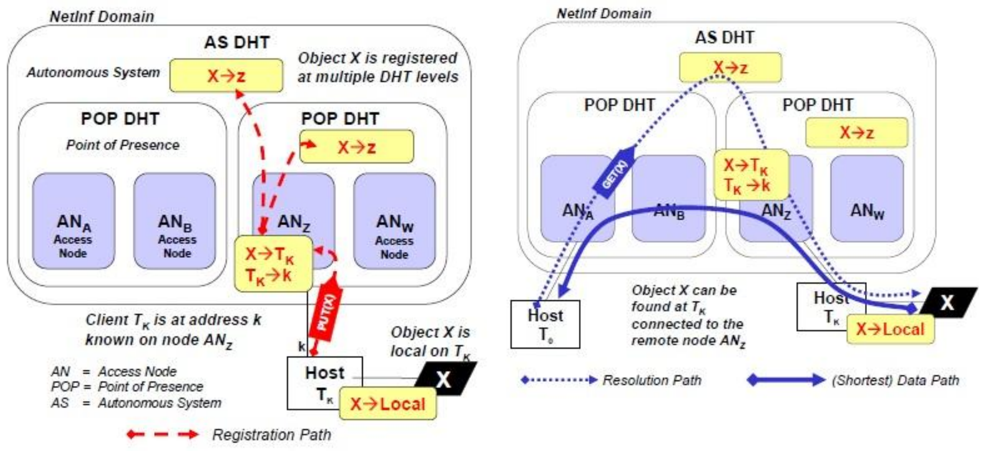


Figure 1-7. 4WARD NetInf Name Resolution at the Intradomain Level (taken from [29])

MDHT is an intradomain solution. It requires global (interdomain) resolution which is provided by the REXs. These are maintained by third parties, similar to Top Level Domain (TLD) DNS servers. A resolution request not satisfied within the AS will be forwarded to the corresponding REX which will either have the binding or will contact other REXs for it. The REXs maintain all bindings for the ASes under their responsibility. The REXs themselves can be implemented using, for example, a DHT.

1.3 Discussion

The current Internet architecture is fundamentally oriented towards physical nodes: “naming” involves providing unique identifiers to nodes, and “name resolution” deals with locating them (and, ultimately, establishing routing paths to them).

Thus, naming and name resolution are integrated. The main reason for this is IP semantics, in which an address represents a single “locater-identifier” that both names/identifies a node and gives its network location within the topology. The dominance and ubiquity of the Internet tends to make it the “normative” standard for network design. However, it is important to keep in mind that naming and name resolution are two conceptually and logically distinct operations. ICN initiatives make a clean break from IP-like semantics by focusing on naming and locating network content rather than physical nodes. Furthermore, like all Future Internet initiatives, ICN schemes support locater-identifier separation.

1.3.1 Naming

As mentioned earlier, names can be categorized as flat or hierarchical, and possess certain properties such as persistence, aggregatability, human-readability, *etc*. In general, flat names support persistence, while hierarchical names are inherently aggregatable, and may be human-readable. Persistent names have the advantage of inherently accommodating content mobility. Flat names, however, are not aggregatable – this is a price that would have to be paid if one stringently insists on having persistent names. Hierarchical names, because they are aggregatable, have the potential advantage of in-built scalability. Scalability is thus a more acute challenge for flat naming schemes, especially given the number of named objects an ICN would have to support.

IP semantics makes names topologically bound. These names are consequently non-persistent and are not immune to relocation/migration of the named object. Most ICN initiatives propose topologically-independent names. They also lean strongly towards using persistent names. NetInf and DONA use flat naming schemes that are indeed persistent. PSIRP supports persistent names (AId’s) at the application level. Hence, reference to an object name in, say, a webpage, would not have to be updated even if this referenced object moves to a new location. However, the representation of this same object in terms of its (SId:RId) name would change if the object were moved to a new scope, and hence is non-persistent with respect to this level. PSIRP's persistent AId names necessitate an extra resolution step (AId $\longrightarrow$ SId:RId) which increases the complexity and impacts the scalability of the resolution system. CCN uses hierarchical names precisely in order to support name aggregation, and so relies on a hierarchical structure for name resolution. Content with the same name can be advertized from any location in the network, so CCN names are nevertheless topologically independent. Topological independence, however, is only a necessary but not sufficient condition for persistence, and CCN names are not persistent. For example, the globally-routable component of a CCN name is typically an organizational ID which changes with the owner's relocation to a different organization; thus, the globally-routable component in the content name would need to be replaced. On the other hand, CCN can be made to support persistent names if this globally-routable component is made persistent (*e.g.*, a personal ID instead of an organizational ID). Finally, as previously mentioned, TRIAD uses URL names, which are inherently non-persistent.

1.3.2 Name Resolution

The properties of a given ICN naming scheme (flat/hierarchical, persistence, aggregatability, human-readability, and so on) directly impact the kind of name resolution techniques that can be applied. The central concern of any name resolution scheme in ICNs has to be scalability.

Scalability is already a dominant issue that threatens the current Internet, in which, let it be recalled, "named objects" are IP networks and "name resolution" corresponds to resolving IP network addresses. IP prefix aggregation has proven a successful strategy to address scalability in the Internet; however, IP prefix disaggregation due to multihoming and traffic engineering has tended to counteract this. Already in 2011, BGP routing tables maintained approximately 350,000+ entries and this number has risen quite rapidly in recent years. With ICNs the problem will be magnified to the extent that the number of named objects will be orders of magnitude larger. The more salient name resolution techniques presented in the ICN literature are, in order of most scalable to least scalable: (a) DHTs (both global, GDHT, and multi-level, MDHT); (b) hierarchically-structured techniques analogous to the current DNS system (*e.g.*, REX in NetInf); and (c) flooding.

Flat naming schemes yield a very large search space (especially for NetInf which incorporates several additional bindings that burden the resolution system). ICNs that need to resolve flat names tend to rely on structured networks (DHTs) for more efficient searching. At the intradomain level, DHTs can resolve names in constant time. NetInf's MDHT structure, for example, resolves names within an AS in constant time.

Interdomain resolution with structured networks, such as DHTs, has been viewed by some as the ideal scheme from the perspective of scalability, and is a vigorous subject of debate in the research community [30]. While DHTs have the property of being scalable in terms of the asymptotic growth rate for the system, this comes at the cost of resolution latency. Moreover, a scalable DHT scheme for the interdomain level (GDHT) would involve an increasingly large number of nodes under distributed management. Due to the nature of flat-name resolution by distributed hashing, the bindings for the flat object names of a given autonomous system (AS) could be under the management of any number of other ASes, which raises some disabling trust issues. While the PSIRP initiative does not specify how SId’s are to be maintained and resolved at the interdomain level, PSIRP using GDHT can be made to overcome the trust issue because it uses a two-level hierarchical (SId:RId) naming scheme. Because a SId does not constitute the complete hashed object name, it can be assigned in such a way as to ensure that its resolution bindings are maintained by a GDHT node within the AS to which the object belongs. However, as already mentioned above, PSIRP's (SId:RId) names are not persistent.

In light of the trust issues outlined above, a GDHT scheme for NetInf-like persistent flat names is not a viable option. Flat naming schemes preserving persistence therefore resort to DNS-like solutions, *e.g.*, the REX system in NetInf. However, there are a more limited number of nodes responsible for resolution in such systems as compared to GDHT, yielding a correspondingly much higher number of bindings per node.

CCN's design philosophy presupposes that, because of CR caching, requested content is more likely to be found closer to the requesting node than further away, and so it initially proposed using flooding to locate the closest copy. CCN has been strongly critiqued for adopting this strategy as flooding is not a scalable option. A variant of "pure" CCN has been proposed (to be implemented on top of IP) using OSPF and BGP for content name prefix advertisement at the intradomain and interdomain levels, respectively.

The role of names at the user-level, *i.e.*, user perception, has not been effectively addressed in ICN schemes. It is common behavior amongst users to memorize and exchange object names, which mandates human-readable names. More importantly, name resolution for human-readable names depends on whether these names are required to be globally unique; if not, simple cryptographic hashing will not suffice and additional resolution steps will be required. Some ICN schemes (*e.g.*, NetInf) propose metadata or keywords to identify objects, which is not a satisfactory solution for the user-level. In this respect, ICN schemes that employ URI names have an advantage in as much as they are both human-readable and do not require additional resolution steps.

Flat names are not aggregatable and therefore cannot be grouped according to inherent mutual associations they might possess. If the current Internet model, in which web pages reference multiple embedded objects, is considered, this lack of associability is an extra burden on the resolution system: each flat name, though it might refer to an object on the same server, needs to be individually explicitly resolved. Hierarchical names for such objects, on the other hand, need only be explicitly resolved once, since they would possess a common, globally-routable name component that makes their mutual association explicit.

Our discussion on names and name resolution is summarized in Table I below.

Table I. Summary of Name and Name Resolution Techniques

	Property	TRIAD	DONA	CCN	4WARD NetInf	PSIRP
Naming	Flat/ Hierarchical	Hierarchical	Flat	Hierarchical	Flat	Flat & Hierarchical
	Aggregatability	Yes	Yes (over Principal)	Yes	No	No (over RId's) Yes (over SId's)
	Persistency	No	No	No	Yes	Yes (over AId's) No (over SId's)
	Topological Independence	No	No	Yes (advertize from anywhere)	Yes	Yes
	Human-readability	Yes	Yes	Yes	No	Yes
	Layer(s)	Above IP layer	Above IP layer	Network layer	≥ Network layer[1]	
Name Resolution	Intradomain	URL resolution in the Content Layer	RHs resolve <P:L>	Flooding or OSPF	MDHT & LLC	RID resolution using RNs
	Interdomain			BGP to advertize name prefixes	REX	GDHT

[1] 4WARD NetInf addresses naming and name resolution at a “dictionary” layer which is above the IP layer, while PSIRP addresses them at the Application (AId) and Network Layers (RId and SId).

Scalable mechanisms constitute *a sine qua non* for the success of ICN architectures. It can be argued that aggregation, or some equivalent compaction mechanism, is essential for scalability even for flat-naming schemes. Various data structures have been advocated in the literature to that end. Probabilistic data structures, such as Bloom filters for example, have been used to reduce the size of routing tables by aggregating content names [31], and can also assist in controlling flooding in CCN [32].

ICNs employ name resolution to find the location of the named content in order to retrieve it: the locaters returned from the name resolution stage are used to establish a path to the content. In principle, name resolution to locate the content and building a path to that content could be accomplished as a single integrated step. DONA and TRIAD do not attempt to do this because they rely on IP forwarding to route to the content location. PSIRP also does not integrate name resolution with routing in order not to overburden the RNs, which would then also have to handle data packets as well as control packets that resolve subscriber requests. It uses a separate Topology Function instead to build a path to the content. NetInf does offer this integrated service as an option, but in practice prefers to keep the two operations distinct in order to take advantage of different transfer protocols for retrieving content. Like PSIRP, it also prefers not to direct content traffic through the DHT resolution systems so as not to overwhelm them. CCN is the only architecture which has an integrated name resolution and routing mechanism as a central component of its design, using reverse-paths built during the name resolution phase to retrieve data packets.

1.4 Summary

This chapter provides the broad background to our thesis by presenting an overview of ICNs in general, together with a taxonomy and brief survey of the various ICN proposals from the perspective of naming and name resolution.

Our thesis works deals with forwarding in the specific context of the CCN architecture. In Chapter 2 we give a detailed presentation of the related work in this field, and provide the reader with an outline of the structure for the remainder of the thesis.

Chapter 2

Forwarding in Content Centric Networks

2.1 Introduction

The work of this dissertation deals with forwarding in CCN. In this chapter, we outline the related work in this field, and present an outline of the remainder of the thesis.

First, we introduce the two basic forwarding strategies of CCN that are commonly used in Section 2.2. In-network caching is a fundamental component of ICN architectures in general, and of CCN in particular [33]. We therefore next discuss CCN research that solely focuses on in-network caching in Section 2.3. However, in-network caching is correlative to a forwarding strategy. Given a capability for in-network caching, we need a forwarding strategy that makes use of and leverages that capability; on the other hand, the dynamics of the forwarding strategy tend to influence, where they do not actually fully determine, cache content placement. In Section 2.4, we focus on research conducted on forwarding strategies, for which we have identified two broad approaches. The first of these, which came as a natural outgrowth of research into Content Distribution Networks, takes the optimization of content placement in the in-network caches as its point of departure. The focus and driving motivation is on the effective utilization of the caches. Forwarding strategy then follows as a consequence of cache content placement and distribution, aimed at leveraging the latter. The second approach takes the design of the forwarding strategy itself as the starting point and attempts to enhance the delivery of content to the user, leveraging in-network cache content along the way as best it can. Cache content placement here is a consequence of the forwarding strategy. Finally, in Section 2.5, we present an outline of the remainder of this dissertation.

2.2 CCN Forwarding Strategies

From early on, two basic forwarding strategies were proposed for CCN: CCN-Flooding came first, followed shortly by CCN-Publication. In CCN-Flooding, interest packets are flooded into the network, and data packets are received in response. These data packets are then used to populate the FIBs with routing information. In contrast, CCN-Publication, which has since gained wide currency, pre-populates the FIBs. Repository nodes advertize/publish the names of their objects using an OSPF-like mechanism, by means of which routing information can be derived to populate the FIBs.

Some authors, however, focusing on maximizing the usage of in-network cache content, have adopted an approach in which there is no notion of maintaining FIBs and use CCN-Flooding for every request. This version of CCN-Flooding, together with CCN-Publication, can be viewed as opposite ends of a spectrum from the perspective of how aggressively a CCN forwarding strategy might search for cached content. CCN-Publication takes a minimalist approach. Essentially, it is similar to IP forwarding, but with in-network caching along the path. CCN-Flooding, on the other hand, both carries out an exhaustive search for cached and repository content, and ubiquitously populates the in-network caches with the content it finds.

2.3 Research in In-network Caching

In the CCN literature, there is a considerable body of research that centers on in-network caching as a primary, if not sole, concern, setting aside other aspects of CCN such as, in particular, the forwarding strategy. The focus is on in-network caching and issues related to maximizing its performance, such as cache location in the network, cache replacement policy based upon content popularity, cache storage size, content placement, and so on. Reference [34] is a performance study examining the potential of in-network caching in CCN using BitTorrent user request patterns running on real-world topologies. Reference [35] analyzes in-network caching in CCN under various conditions, such as multiple topologies, content popularity, cache replacement policies, and traffic patterns. In [36], the authors argue that from a network operator perspective, the success of content-oriented networks hinges on the performance of in-network caching. References [37, 38] propose cache replacement policies which are based on content popularity. Reference [37] analyzes and addresses the limitations of a Most Popular Content cache replacement strategy. Reference [38] proposes a new replacement strategy that achieves higher cache hit rates and less traffic compared to the traditional LRU and LFU policies. Some authors put the efficiency of in-network caching – a fundamental component of CCN, and of ICNs in general – into question. Reference [39] concludes that cache storage should be placed only at the network edge. Reference [40] questions the efficiency of universal in-network caching. It argues for an approach that focuses on a strategic placement of caches in the network, exploiting the concept of Betweenness Centrality of nodes in order to achieve better gains. On the other hand, Reference [41] presents a case for larger cache storage at nodes with higher degree (essentially, the core nodes). Reference [42] concludes that a one-size-fits-all solution for cache storage allocation will never be optimal for CCN as there are too many variables, such as network topology and content request patterns, which have an impact on performance.

2.4 Research in Forwarding Strategy

Research in forwarding strategies is not always an exception to the trend that privileges caches and content placement described in Section 2.3 above. The research can be classified according to two broad approaches. In the first approach, the primary focus is, again, on optimizing content placement in the caches. Forwarding strategy is also a concern, but follows as a consequence of cache content placement and distribution, and aims at better leveraging the latter. In [43, 44] content placement is dependent on the popularity of an object. In [44] popular content is pushed towards the end users so that the latency and traffic generated to retrieve such content is reduced. References [43, 44] focus on content placement first, and then simply adopt CCN-Publication as a forwarding strategy, and thus leverage on-path caches only. On the other hand, [45-47] leverage off-path caches. They use hashing schemes, both to distribute cached content between caches within the network, and to retrieve that content. The goal is to maximize utilization of available cache storage capacity by eliminating redundant content, which consequently reduces traffic as more requests will be resolved using in-network caching.

The second approach takes the design of the forwarding strategy as its starting point and attempts to enhance the delivery of content to the user, leveraging in-network cache content along the way as best it can. In this approach, the placement and distribution of cache content becomes a consequence of the way the forwarding strategy operates. Caches are populated according to the content retrieved by the forwarding strategy as that content weaves its way back to the requesting node. Our work in this thesis falls squarely within this second approach.

Reference [48] explores the design space between an “exploration” approach (flooding) on the one hand, and an “exploitation” approach (pre-populated FIBs) on the other. The authors point out that having infinitely large FIBs is not possible in real-world situations, and thus, flooding, despite its heavy cost, is unavoidable in situations where no FIB entry is found for an object. They explore the exploration-exploitation design space to analyze the tradeoffs between FIB size and the frequency of flooding required.

Reference [49], by the same authors, on the other hand, also explores the exploitation-exploration design space. The goal here, however, is to design a dynamic forwarding strategy that leverages off-path caching. This is similar to what we do in Chapter 4 where we also propose a forwarding strategy that explores the potential afforded by off-path caching. Reference [49] implements a reinforcement learning algorithm which is used to rank interfaces associated with “volatile” FIB entries, *i.e.*, FIB routing information pointing to content that was located in caches. “Non-volatile” (regular) FIB entries typically point to the repositories that “own” that content. Initially, requests for the first few chunks of an object are sent not only to the repository (in order to guarantee data delivery), but are also flooded on other interfaces in the hope of locating off-path content. This state of the forwarding strategy is called exploration. If off-path cache content is found, the forwarding strategy goes into the exploitation state, in which it uses the interface ranked best amongst those pointing to off-path content, to forward subsequent interest requests for the remaining chunks of the object. The exploration phase of the forwarding strategy is thus critically dependent on objects being divided into chunks. The strategy we present and analyze in Chapter 4 has different operational mechanisms and does not impose this restriction. Furthermore, while [49] achieves its maximum gain for larger cache sizes, our strategy works well also for under-provisioned caches.

Reference [50] analyzes multipath forwarding strategies under various conditions, such as network topology, cache size, cache replacement policy, content popularity, *etc*. It assumes that an object can exist in multiple repositories. The FIBs are pre-populated, and, for each repository holding a copy of the object, a FIB entry may point to multiple shortest paths to that repository. The authors reach several conclusions. For a given object, making use of multiple shortest paths to one single repository yields better performance than using the available paths to all of the multiple repositories. Consequently, if repository load reduction is not an issue, then a single repository would suffice for the object. The only advantage, then, of multipath forwarding to that repository is robustness in case of link failure. Finally, they conclude that multipath forwarding strategies that complement single-path routing with an added component of opportunistic exploration of a CCN neighborhood is worth investigating. This is akin to what we do in Chapter 4.

Reference [51, 52] proposes an adaptive forwarding mechanism. The paper highlights the benefits of maintaining in-network state information in CCN. This information is utilized to make instantaneous decisions, thereby making the adaptive forwarding mechanism resilient to changes in the state of the network, such as congestion, packet loss, link failures, *etc*. The study assumes pre-populated FIBs that maintain information on multiple interfaces towards a repository hosting a given object. The interfaces are ranked according to selected criteria, such as shortest path, fastest route, *etc*. While comparative evaluation of which criterion works best is outside the purview of the study, the results show that the adaptive forwarding mechanism can provide excellent performance in handling blackhole hijacks, link failures, and network congestion.

2.5 Dissertation Outline

This dissertation deals with forwarding strategies in the specific context of CCN. The outline of the thesis is as follows:

- In Chapter 3, we present a simulation-based performance study of bandwidth consumption for CCN-Publication and a version of CCN-Flooding that is used to populate the FIBs. The chapter is divided into two sections.

  In the first section, we study the bandwidth consumption of the CCN-Flooding strategy. We also identify and analyze various issues pertaining to its behavior, such as calibration of PIT timeouts; a PIT-induced isolation effect that negatively impacts bandwidth consumption and system response time; and the effects of adopting FIB routes based on volatile in-network cache entries.

  Due to the well-known drawbacks of a flooding-based approach in computer networks, preference in the research community soon shifted to CCN-Publication. However, performance studies in the relevant literature do not take into account the bandwidth costs associated with pre-populating the FIBs that a CCN-Publication approach entails. We analyze CCN-Publication from a perspective that does take into account these bandwidth costs, and compare that to CCN-Flooding.

  The second section of the chapter focuses on the interplay between two important components of CCN, caches and the Pending Interest Table (PIT), under CCN-Publication. We show that the PIT aggregation effect increases commensurately with both network load and the Betweenness Centrality attribute of the network topology. While both PIT aggregation and in-network caching contribute significantly to achieving savings in network bandwidth consumption, they do not do so in an additive fashion. As network load increases, gains from cache hits diminish and are replaced by gains from PIT aggregation, without dramatically impacting the overall level of savings achieved.

– In Chapter 4, we propose a new lightweight forwarding strategy for CCN. One of the primary objectives of any forwarding strategy, on the one hand, is to improve user experience by exploiting in-network caching, while minimizing the associated costs on the other. To this effect, we compare the new forwarding strategy against the well-established CCN-Publication and CCN-Flooding strategies. The results show that the new strategy, which we call CCN-FOH, provides for a better user experience while keeping its overheads comparable to, if not better than, the less costly CCN-Publication.

– In Chapter 5, we present future directions for our research and offer some concluding remarks.

Chapter 3

Performance Study of CCN Forwarding Strategies: CCN-Flooding and CCN-Publication

3.1 Introduction

In this chapter, we present simulation-derived results and observations on bandwidth consumption in CCN networks with respect to a disparate set of issues that have not received much attention in the research literature. We examine CCN under the two well-known forwarding strategies, CCN-Flooding and CCN-Publication, using a combination of several topologies and workload sets with differing characteristics. As outlined in Section 2.2 of Chapter 2, there are two distinct versions of CCN-Flooding presented in the research literature. One version – the version we deal with in this chapter – defines CCN-Flooding essentially as a mechanism by which to populate the FIBs. An interest packet that finds forwarding information in a FIB follows the path designated by the FIB entry. In the absence of such a FIB entry, the interest packet is instead flooded. FIBs are initially empty and interest packets are therefore flooded. Over time, however, as data packets are returned from repository nodes, they populate the FIBs along the return shortest paths to the requesting nodes with forward routing information. Thus, this version of CCN-Flooding gradually converges to CCN-Publication. The second version of CCN-Flooding, which we adopt and further discuss in Chapter 4, uses flooding essentially to leverage off-path content.

Not much attention has been given to overall network behavior under the CCN-Flooding approach. Nevertheless, as a practical matter, a finite-sized FIB cannot maintain entries for all objects in the system. When we do not have a “hit” in the FIB for a given object request, the corresponding interest packet still needs to be forwarded somehow. Flooding is therefore still maintained as a possible option to handle such situations [48, 51]. Hence, we consider investigating CCN performance under flooding to be worthy of some interest despite the generally deprecated status associated with this approach. The first major theme of this chapter, presented in Section 3.3, therefore will be to highlight and characterize some issues that arise under the CCN-Flooding approach. But first, we present our experiment design space and methodology in Section 3.2 with respect to the simulator we developed; the topologies and workloads implemented, and their characteristics; cache configurations; and performance metrics.

Under CCN-Flooding, populating the FIBs and making use of FIB entries to forward interest packets are two aspects of a single, integrated dynamic. Studying bandwidth consumption under CCN-Flooding consequently inherently accounts for the bandwidth costs of populating the FIBs. This is not the case under CCN-Publication, in which populating the FIBs via an OSPF-like mechanism on the one hand, and forwarding interest packets by means of such pre-populated FIBs, on the other, are two distinct stages. All studies we are aware of, to the extent that they report on bandwidth consumption under CCN-Publication (*e.g.*, [35, 48]), take a pre-populated FIB as their starting point, ignoring the costs incurred by the OSPF-like mechanism that provisions the FIB content in the first place. A second theme of this chapter, therefore, is to present in Section 3.4 a comparison of the bandwidth consumption associated with CCN-Flooding *vs*. CCN-Publication, taking into account of the bandwidth cost of pre-populating the FIBs under the latter approach so as to provide a meaningful comparison with the former.

The Pending Interest Table (PIT) is a vital component of the CCN architecture (Subsection 1.2.2, Chapter 1), whereby second and subsequent interest packets reaching a node at which there is a pending (*i.e.*, still unsatisfied) request for the same object are suppressed and the data packet for the object is then multicast back to the requesting nodes when it arrives. This is the so-called PIT suppression or PIT aggregation effect. However, the approach taken by the CCN-specific studies [34, 35, 41], all of which assume CCN-Publication, tends to isolate the in-network caching behavior from the PIT aggregation effect. In [35], for example, simulations are run in a congestion-free network; and in [34] the simulations are driven using traditional IP-network packet-level traces and their results are post-processed to take into account in-network caching. The PIT aggregation effect is not considered nor taken into account in these studies. The third major theme of this chapter, presented in Section 3.5, therefore is to investigate the effect of PIT aggregation and of Cache-PIT dynamics under the CCN-Publication approach.

3.2 Experiment Design Space and Methodology

3.2.1 Simulator

We developed a simulator in JavaSim [53] to perform our experiments. The following are some of the more salient features of our simulator.

- CCN data structures and operations are supported as described in [22] and Subsection 1.2.2 of Chapter 1, unless otherwise noted below.
- Generation of new requests is Poisson. The generated interest packet is then randomly assigned to an emitting node with equal probability. The object associated with the interest packet is also randomly chosen with equal probability.

- The specific data entity requested by an interest packet is either a single, complete object or one chunk/segment of an object, depending on whether we are simulating object-mode or chunk-mode operation.
- In the CCN-Publication approach, the forwarding strategy is to follow the shortest path to the object. In the case of CCN-Flooding, the forwarding strategy is to flood on all interfaces (excluding the interface on which the interest packet was received) if there is no FIB entry for the object.
- In the CCN-Publication approach, FIBs are pre-populated based on the shortest path to the owning node using Dijkstra's algorithm. In the CCN-Flooding approach FIBs are only populated when an interest packet is satisfied. More to the point, the interest packet must be satisfied from a node that "owns" the requested object *(i.e.*, repository). No FIB entry is made if the object is fetched from an intermediate cache because: (a) cache content is volatile; and (b) we discovered that FIB entries based on cache content can lead to anomalous circuitous paths that are much longer than the shortest path between the node requesting the data object and the owning node.
- There is no support for FIB entries directed towards multiple outgoing interfaces.
- We do not implement CCN's hierarchical naming conventions and security aspects. Objects are given flat id numbers and FIB entries are based on these object ids. It should be noted, however, that our results are not severely impacted by the inability to aggregate names that a hierarchical naming scheme would have allowed, and this for two reasons. Firstly, our study does not focus on FIB size and we permit our FIBs to be provisioned with as many object id entries as the CCN-Flooding and CCN-Publication approaches supply. Secondly, under chunk-mode operation, a chunk is identified by the

2-tuple <object id, segment number>. In this case, a PIT entry for the chunk consists of the full 2-tuple, but a FIB entry is based only on the object id.

– Cache replacement policy is LRU.

The architecture of a node in our simulator consists of a centralized processing queue in which all arriving interest and data packets are placed in a first-come-first-served order, including interest packets generated at that node. The processing delay for each packet is set to a constant 0.1 time units. We assume that transmission links have infinite bandwidth. Nevertheless, we have a link propagation delay, also of 0.1 time units. This was introduced so that the flow of packets in the network is temporally staggered, in order to avoid bulk point arrivals of packets at a node. It should anyway be noted that our simulation methodology is set up to focus on investigating bandwidth usage in a manner that is not coupled to delays in the network.

3.2.2 Topologies

We consider four networks of which two are synthetically generated (using BRITE [54]) and two are real-world. Their main topological characteristics are summarized in Table II. The networks themselves are shown in Figure 3-1.

Table II. Characteristics of the Network Topologies

Nodes	***Type***	***Structure***	***|E|/|V|***	***|D|***	***cDC***	***cSC***	***cBC***
39	Synthetic	Core & Edge	1.05	5	0.21	0.63	0.41
51	Real-world	Core	3.15	7	0.34	0.79	0.29
70	Real-world	Core	5.071	3	0.78	0.89	0.22
100	Synthetic	Core & Edge	2	6	0.08	0.24	0.10

|E|/|V| is the ratio of vertices to edges; |D| is the diameter; cDC, cSC & cBC are centralized graph metrics.

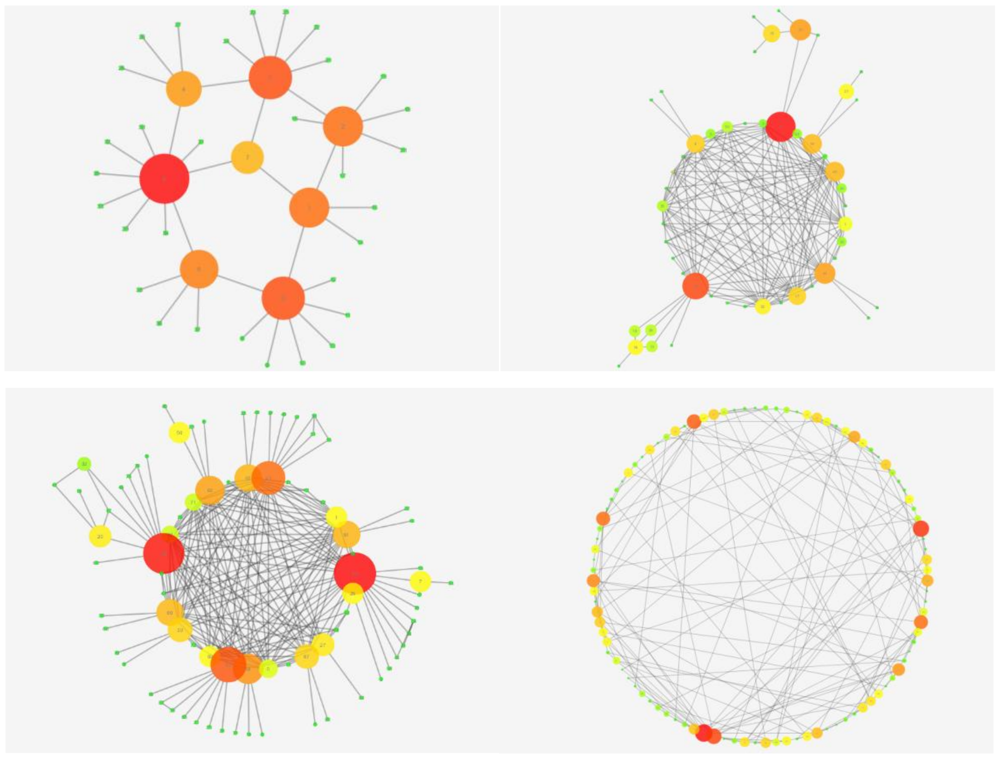


Figure 3-1. 39-node (*top-left*), 51-node (*top-right*), 70-node (*bottom-left*), and 100-node (*bottom-right*).

The illustrations are based on the Betweenness Centrality of nodes. The size of a node is directly proportional to its BC value: nodes with larger sizes and brighter colors (reddish) have higher Betweenness Centrality values, while ones with smaller size and lighter colors (greenish) have the lowest.

The synthetic networks are the 39-node and 100-node. Both topologies comprise a set of core CCN router nodes surrounded by a set of edge nodes that act as gateway routers of stub user networks. Interest packets are generated only at the edge nodes which are also the only nodes that can "own" data objects (*i.e.*, an edge node acts as the repository for the data objects residing at the user networks behind it). The 39-node network comprises 8 core nodes that are almost like a ring, with 31 edge nodes hanging off it. For the 100-node topology we arbitrary designated nodes of degree ≤ 2 as edge nodes and the rest as core nodes. The network thus comprises 68 core nodes of degree ≥ 3 and 32 edge nodes which all happen to have exactly degree 2 (there are no degree 1 nodes).

The real-world topologies are selected from the well-known collection of PoP-level ISP maps generated using the Rocketfuel ISP topology mapping engine [55]. At this level, the topologies of the ISPs show only the gateway routers. Hence we did not make a core-edge distinction between the nodes in these two networks. All nodes in 51-node and 70-node act both as CCN routers and as “repository” gateways to networks that generate user interest packets and own data objects. 51-node is the ISP map of TW Telecom (AS 4323), located in the US; 70-node is the ISP map of Deutsche Telekom (AS 3320) in Germany.

Figure 3-2 illustrates the key characteristics of the four networks in terms of centralized metrics based on the following standard graph topology measures of vertex centrality [56]: Degree Centrality (DC), Stress Centrality, and (Normalized) Betweenness Centrality (nBC)[1]. These standard measures are calculated on a per node basis. A single “centralization” metric that characterizes the network overall can be derived from them using the approach proposed by Freeman [57]:

Centralized Degree Centrality (cDC). cDC is close to 0 for a network in which each node has the same degree. It gets closer to 1 the more “imbalance” there is, with fewer nodes of high degree and more with low degree, a star topology being the extreme such case.

Centralized Stress Centrality (cSC). cSC is close to 0 for a network in which each node has the same Stress Centrality. It gets closer to 1 the more imbalance that there is, with fewer nodes having a high number of shortest paths going through them

[1] nBC is a normalization of Betweenness Centrality (BC) across networks with different numbers of nodes. It is obtained by uniformly dividing BC_i (the BC for node *i*) by the number of pairs of nodes in the topology, not including node *i* itself. This yields the rescaled measure, nBC_i, for node *i*.

and more nodes having fewer or none; again, a star topology is the extreme such example. Thus, cSC measures the degree to which the totality of shortest paths in the network pass through a subset of “choke-point” intermediate nodes, without distinction as to the individual pairs of nodes connected by these paths.

Centralized Betweenness Centrality (cBC). cBC is close to 0 for a network in which each node has the same Betweenness Centrality (*i.e.*, shortest paths for pairs of endpoint nodes are evenly spread, with no overlap between intermediate nodes on the paths connecting the endpoint pair). It gets closer to 1 the more imbalance there is, with increasingly fewer nodes lying along an increasingly higher proportion of these pair-wise shortest paths; a star topology is again the extreme such example. In distinction to cSC, cBC measures the degree to which shortest paths for each pair of end nodes themselves pass through a subset of “choke point” nodes.

The cDC of network 70-node is almost 0.8, implying that this topology has a nexus of disproportionately high-degree nodes as compared to, for example, the network 100-node for which the cDC value is close to 0, indicating that its nodes have evenly distributed connectivity. This aspect of 70-node is further confirmed by the high cSC value of 0.89. As shown by Figures 3-1 and 3-2, in 70-node four nodes out of the total provide a disproportionate amount of the shortest-path connectivity in the network. However, though a small subset of nodes are thus acting as relays for a disproportionately large number of shortest paths, the relatively low cBC value of 0.23 indicates that these paths are well-distributed amongst this subset. Network 51-node displays some of the same general characteristics as 70-node, though in a less sharply accentuated fashion, with a subset of nine such relay nodes instead of just four.

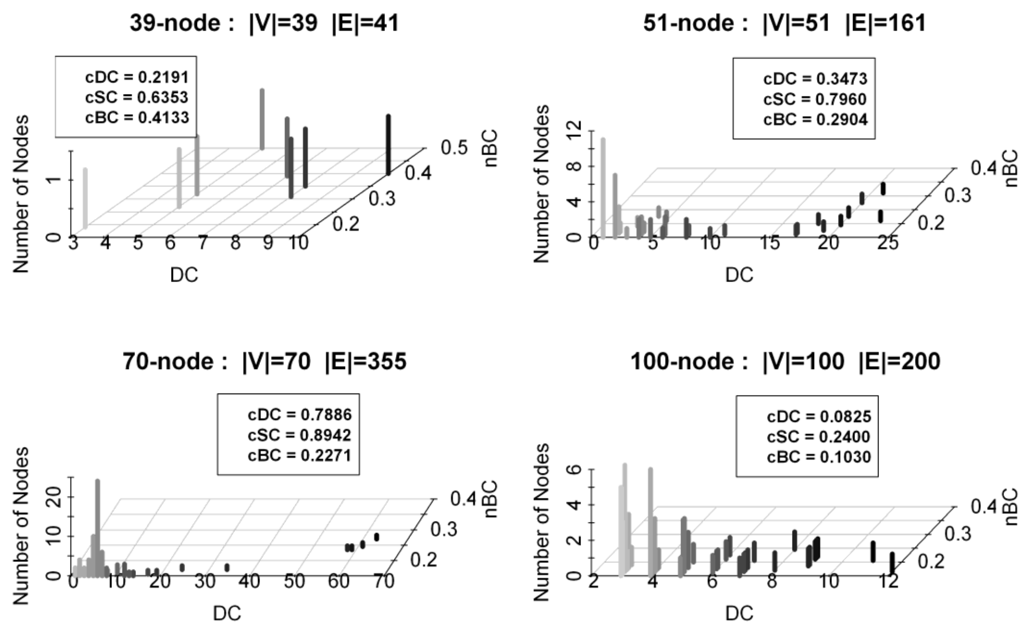


Figure 3-2. Key Characteristics of the Network Topologies

The figure plots the Degree Centrality (DC) along the x-axis, and the normalized Betweeness Centrality (nBC) along the y-axis. The z-axis gives the count of the number of nodes.

Figure 3-2 shows only the core nodes for networks 39-node and 100-node. Both networks are servicing essentially the same number of edge nodes (31 and 32, respectively). However, 39-node has only eight core nodes whereas 100-node has several times that number. Not surprisingly, therefore, the core nodes of 39-node are more central to network connectivity, as can be seen from its cSC and cBC values. Overall, 100-node is the best balanced of our networks, which visually can be directly perceived from Figures 3-1 and 3-2, and from its cSC and cBC values which are the lowest across the four networks.

3.2.3 Traffic Generated

We use the GlobeTraff tool [58] to create a synthetic stream of traffic workload. GlobeTraff is a derivative of ProWGen [59], which itself is a tool for generating

Web traffic. Essentially, GlobeTraff builds on ProWGen's functionality, providing additional traffic types such as P2P, Video, and "Other". We used GlobeTraff to generate four workload sets with different characteristics, as shown in Table III.

Table III. Workload Characteristics

Parameters	*Set 1*	*Set 2*	*Set 3*	*Set 4*
Distribution	**α** = 0.75 Zipf	**α** = 1.5 Zipf	**α** = 0.75 Zipf	k=0.513, λ = 6010 Weibull
Number of Objects	54273	54273	18091	101
Object Median Size	10KB	10KB	10KB	5MB
Object Mean Size	10KB	10KB	10KB	10MB
Number of Requests	183682	177355	175696	174
Requests/Object	3.4	3.3	9.7	1.7

The traffic workload parameters are classified with respect to two criteria.

Content Popularity:

Object popularity is modeled by a Zipf distribution with parameter α. There is general agreement that α has significant impact on overall system performance. But there is no consensus as to appropriate α values for modeling various types of traffic, such as web objects, UGC (YouTube-like User-Generated Content), P2P, and VoD (Video on Demand). A wide range of values, α ∊ [0.6 , 2.5], are discussed in [50]. For web-like objects, an α value in the range [0.64 , 0.83] is accepted as a good approximation [60, 61]. The α values for our workload Sets 1 & 3 are selected from within this range (α = 0.75). For Set 2, we double the α value to 1.5, keeping the number of requests and objects approximately the same as in Set 1 in order to investigate the effect of further biasing requests in favor of the more popular objects of Set1. Set 3, on the other hand, is used to investigate a more intensive pattern of requests over a smaller number of objects, by reducing the number of objects to a third (from 54,273 to 18,091) as compared to Sets 1 & 2, but keeping the number of requests approximately the same (175,696 ~ 183,682). Effectively, Set 3 issues the same number of requests as Sets 1 & 2, but only over the subset constituting the most popular, top third objects of Set 1.

As reported in [62], UGC traffic does not have a heavy tail, hence the Zipf distribution only partly captures its popularity characteristics. Reference [62] proposes either the Weibull or Gamma distributions. GlobeTraff implements both. We chose the Weibull distribution with parameters $k = 0.513$ and $\lambda = 6010$ to generate the UGC-traffic workload Set 4.

Content Characteristics:

We have configured the traffic workload to be of fixed size ~1.8GB to facilitate performance comparison across the four sets. The sizes of the web-like objects in Sets 1, 2 & 3 is modeled by the concatenation of the Lognormal (body) and Pareto (tail) distributions [58] with a mean of 10KB [58, 63]. For the UGC workload Set 4, object sizes are modeled by a concatenated normal distribution [58] with mean size 10MB [64]. We therefore reduce the number of objects (to 101) and requests (to 174) in order to keep the overall workload size fixed at ~1.8GB. Furthermore, objects in Set 4 are broken into chunks of size 10KB [35, 48], which enables us to investigate the effects of chunk-mode operation in comparison to the object-mode of Sets 1, 2 & 3. In studies comparable to our work (e.g., [48]) an object space of 10,000 objects is taken as adequate. As such, the ~54,000 objects for Sets 1 & 2, and ~18,000 for Set 3 are well within precedent.

3.2.4 Cache Sizes and FIBs

Cache sizes are dimensioned as a percentage of the object space. As shown in Table III, workload Sets 1 & 2 have 54,273 objects each of mean size 10KB, giving an object space of 542 MB. For Set 3, the corresponding values are 18,091 objects, 10KB, and 180MB, respectively. For Set 4: 101 objects, 10MB, and 1GB. Whether we are operating in object-mode or chunk-mode, a cache of size *n* represents a per-node cache capacity of *n*×10KB, which we assume capable of caching *n* objects (chunks) of average size 10KB each. For Sets 1 & 2, therefore, experiments conducted with cache size 1,000 represent a per-node cache capacity of ~1.8% of the object space (this quantity is also called the Cache/Catalog size). Similarly, for Sets 3 & 4, cache size 1,000 represents Cache/Catalogue sizes of ~5% and ~1%, respectively. Hence, Cache/Catalogue sizes in our study vary in the range [1.8% , 5%]. Previous ICN studies used values of 0.25% [65], [0.5% , 20%] [66], and [0.000001% , 1%] [67]. Our values are compatible with these.

3.2.5 Performance Metrics

In order to uniformly compare bandwidth consumption across topologies with different characteristics, we adopt Path Stretch as our basic unit of performance. Path Stretch is defined as:

Path Stretch	= Hopstotal / Hopsshortest , where
Hopstotal	= total number of hops that a packet travels in the network,
Hopsshortest	= number of hops along the shortest path route between the node originating an interest packet and the node which owns the data object being requested.

Interest and data packets sizes vary by orders of magnitude the one from the other, with a commensurate difference in the raw bandwidth each type consumes per hop in the network. In order to keep this difference in view at all times, we use Path Stretch to define distinct performance metrics for each of interest and data packets as follows:

Expended Effort. Measures the Path Stretch for an interest packet (inclusive of retransmissions and flooded “clones”). If the packet is following a FIB-defined route right from the source node and does not need to be retransmitted, then its Expended Effort will be 1 (as in an IP network). Values greater than 1 reflect extra bandwidth consumed, expressed in terms of multiples of the bandwidth expenditure for the end-to-end, shortest path.

Path Cost. Measures the Path Stretch for a data packet. A data packet originating at the node which owns the data object will have Path Cost value 1. Values less than 1 reflect situations in which the data object is fetched from a nearer cache. Values greater than 1 imply that the object is satisfied from multiple locations (caches) in response to a flooded request.

3.3 Results — CCN-Flooding

3.3.1 Timeouts and Interest Packet Generation

The CCN architecture assumes that interest packets will be backed by timeouts so that requests that remain unsatisfied in a lossy environment are retransmitted. PIT timeouts are also necessary so that unsatisfied PIT entries are purged from the PIT tables.

Preliminary experiments demonstrated that setting appropriate timeouts is a particularly difficult and troublesome challenge, especially under the CCN-Flooding approach. Clearly, an interest packet’s timeout needs to be set somewhat larger than the timeouts for the PIT entries made at intermediate nodes by this interest packet. On the one hand, this is so that retrieved data has a chance of reaching the requesting node without getting suppressed along the way due to PIT entries prematurely timing out. On the other hand, we do not want an interest packet to timeout before its intermediate-nodes PIT entries have done so, otherwise the retransmitted interest will simply be suppressed due to these original, unexpired PIT entries. Our preliminary experiments made it clear that determining appropriate timeout values is very topology-dependent. Different networks experiencing roughly the same overall traffic load display wide variation in their RTT times. For example, shortest paths in network 70-node, which has a diameter of only 3, are disproportionately dependent on a very few number of central, high-degree nodes, as has been mentioned above. These nodes experience rapid processing queue buildups compared, for example, to the more "balanced" network 100-node. An appropriate timeout value for the one network is by no means adequate for the other. Furthermore, unlike TCP in the traditional Internet, CCN operation (at least in object-mode, but less so in chunk-mode) does not provide relatively stable end-to-end paths with a stream of outgoing and returning traffic by means of which we may constantly evaluate RTTs for the path, dynamically adjusting timeout values accordingly.

Since, as already mentioned, our simulation methodology is set up to focus on investigating bandwidth usage in a manner that is not coupled to delays in the network, we are able to bypass these difficulties in a manner which, while presupposing some "oracular" knowledge of the state of the network, nevertheless serves to uphold the purposes of our study. First, we dynamically adjust the generation rate for new interest packets so as to keep the network load low (see below). Timeouts for interest packets and PIT entries are then calculated dynamically as follows. Define:

Q_i = queue size at node i,

Q_{max} = $\max_{\forall \text{ nodes } i} (Q_i)$, the queue size at the maximally congested node,

T_{proc} = packet processing time at a node,

T_{prop} = one-hop link propagation delay,

$T_{max} = ((Q_{max} + 1) \times T_{proc}) + T_{prop}$, the maximum one-hop delay.

The PIT timeout at a node is then defined as:

$PIT_{TO} = 2 \times D \times T_{max}$, where D is the network diameter.

The PIT entry timeout for a given interest packet is thus based on the RTT along the network diameter path. This RTT is computed in terms of the queuing delay at the maximally-congested node in the network at the instant when the interest packet is generated. The timeout value loosely assumes that interest packets and their corresponding data packets travel the full diameter of the network, through maximally congested nodes at each hop. It does not adjust for the diminishing remaining path distance as each of these packets progresses towards its destination.

The interest packet timeout is defined as:

$iPKT_{TO} = PIT_{TO} + \eta$,

where η is a small “safety margin” value added to make the interest packet timeout larger than its PIT entry timeouts. These timeout values are quite conservative and tend to maximize the chances that interest packets are satisfied by their corresponding data packets without triggering off an undue number of spurious timeouts and interest retransmissions.

Note that when interest packets are flooded, they create PIT entries all over the network. In particular, these entries are created also in sectors of the network through which encountered data would not be flowing back to satisfy the interest packets. Yet if a node in such a sector happens to issue a request for the same object, its interest packet would be immediately suppressed for the duration of the timeout of the PIT entries made by the flooding packet, even though there is no chance that data which consumes these PIT entries is going to flow through that sector of the network. Thus, in the context of CCN-Flooding, an unintended consequence of

CCN's PIT timeout mechanism is, in effect, to temporarily isolate sectors of the network from receiving requested data in a timely manner when requests for this data are already pending from other, non-intersecting sectors of the network.

Finally, the mean of the exponential inter-generation time distribution for new interest packets is set to 1.0 time units (EM=1.0). The packet is then randomly assigned to an emitting node. Initial experiments showed we had 10%-15% retransmissions in the network due to the "isolation" effect of PIT timeouts described above. In order to mitigate the spurious impact this has on bandwidth consumption, we dynamically adjust the generation rate such that no packet is generated if $Q_{max} > 1$ at the instant of generation.

3.3.2 Effect of Topology on Expended Effort

Figure 3-3 shows the Expended Effort for the four network topologies under the Set 1 workload. A configuration of Cn_FIB / Cn_noFIB denotes a network with a cache capacity of *n* objects (in object-mode; *n* chunks in chunk-mode) with FIBs and with no FIBs implemented, respectively. C0_FIB / C0_noFIB denotes no caching. Cinf represents the hypothetical case where we have caches of infinite size[2].

For each topology, we observe an expected general trend of improving Expended Effort as we traverse through the configurations in each individual box plot, from C0_noFIB (pure flooding) to Cinf. On the other hand, there is a measurable increase in overall Expended Effort as we go from network 39-node, to

[2] Under the CCN-Flooding approach for this hypothetical case of infinite-sized caches, an object will have an entry in a node's FIB if and only if a copy of that object also resides in the node's cache. Thus we talk of configuration "Cinf" with no distinction as to whether FIBs are implemented or not since the presence of FIBs has no impact on performance. As will be seen below, the exception to this is Set 4, for which infinite-sized caches with FIBs yield significantly lower Path Stretch than without FIBs. The Cinf results given throughout the paper for Set 4 are therefore those for which there is a FIB.

100-node and 51-node, and on to 70-node. As may have been anticipated, flooding costs increase – in fact, disproportionately so – with the number of edges in the network.

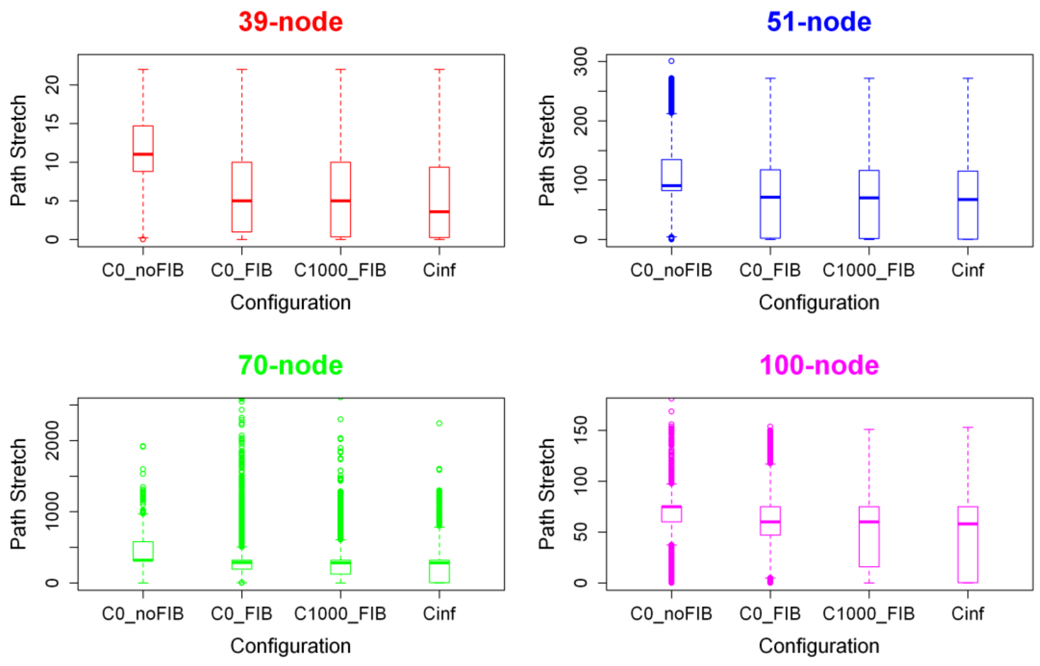


Figure 3-3. Box Plots for Expended Effort under Workload Set 1

The whiskers stretch to a maximum of 1.5 times the interquartile range (IQR), which is represented by the height of the box.

A more detailed examination of the variation in Expended Effort seen in Figure 3-3 serves to highlight the contributing effect of secondary topological differences, other than the number of edges. Consider the fact, for example, that even though 100-node has more edges than 51-node, its overall performance is somewhat better. The average Expended Effort for the configurations of 100-node range from a high of 76.57 to a low of 47.52; for 51-node, the corresponding values are 117.06 and 69.65. 51-node is a less "well-balanced" network than 100-node to the extent that it has a smaller number of central nodes that support a higher proportion of shortest paths, and which consequently are relatively more congested. The difficulty of correctly calibrating PIT timeout values has been mentioned above. PIT timeouts are more likely to be underestimated the more a node is congested, thereby triggering interest packet retransmissions. We conjecture that the poorer performance of 51-node compared to 100-node is due to this effect and the consequent flooding cost of the interest packet retransmissions. This conjecture is further buttressed when we consider 70-node, which has four highly congested central nodes. Interest packet retransmissions yield the extremely high outlier values seen in Figure 3-3, the most extreme case of which occurs for C0_FIB whose mean and median Expended Effort are 293.91 and 286.5, respectively, but whose outlier values (truncated at 2,500 in the figure) stretch to 3,899.

3.3.3 Effects of Non-volatile Entries in FIBs: Expended Effort

As mentioned before, volatile FIB entries can lead to routing anomalies. On the other hand, the absence of volatile entries also causes anomalous behavior, with respect to bandwidth consumption for certain types of workloads. To investigate this anomalous behavior we limit our discussion in this subsection to 39-node and 51-node (whose topological characteristics are perhaps somewhat more representative of networks in general than 70-node and 100-node).

Set 4 is a chunk-mode UGC traffic workload. This type of workload derives maximum benefit from the FIBs: the retrieval of the first chunk of an object populates the FIBs and, within one RTT, subsequent chunks benefit from this.

Intuitively, increasing the cache size should, in principle, have a positive impact on performance. Figure 3-4, however, shows not only that this is not the case, but that bandwidth consumption degrades with increasing cache size. The figure gives the quantile-quantile (Q-Q) plots for the Expended Effort distributions under configurations C100_FIB *vs.* C1000_FIB for each of 39-node and 51-node. It clearly shows that bandwidth consumption increases with increased cache size. The reason for this is straightforward. The larger the in-network caching capacity, the more chance that a node requesting an object will find that object without having to flood all the way to the owning node. Because we do not make volatile FIB entries, the FIBs are consequently not populated by the returning data packets, and all chunks of the object end up being retrieved by flooding. Analogous Q-Q plots (not shown) for configurations C0_FIB *vs*. C100_FIB establish that the corresponding Expended Effort CDFs are virtually identical, highlighting the fact that no benefit was derived from adding in-network caching (representing ~0.1% of the object space) to FIBs. On the other hand, Q-Q plots for C1000_FIB *vs*. Cinf (also not shown) demonstrate a continuing degradation in performance of the type observed in Figure 3-4 as the cache size is increased further.

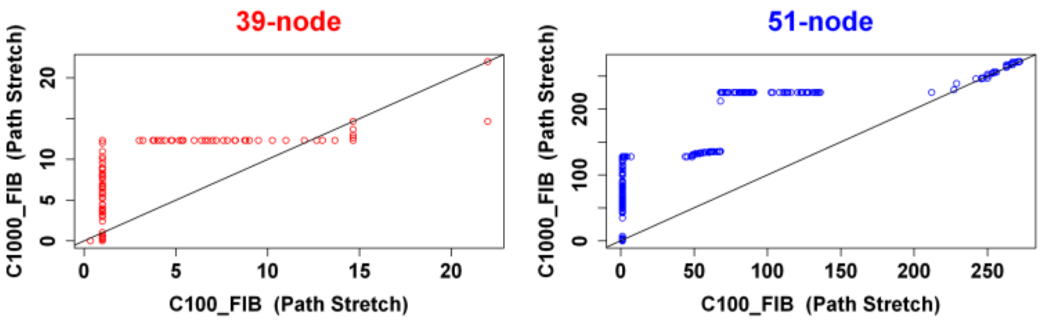


Figure 3-4. Q-Q Plots for C1000_FIB *vs.* C100_FIB under Workload Set 4

3.3.4 In-network Caching and FIBs under CCN-Flooding: Expended Effort

We first consider the worst case scenario for bandwidth consumption, given by the C0_noFIB configuration (pure flooding). As is evident from Figures 3-5 and 3-6, C0_noFIB has the worst Path Stretch across workload sets. Next, consider configuration C0_FIB to observe the effects of introducing FIBs in the network. Our experiments start with empty FIBs, which are then populated as requests are issued using CCN-Flooding. The improvement in performance is quite apparent irrespective of topology and workload.

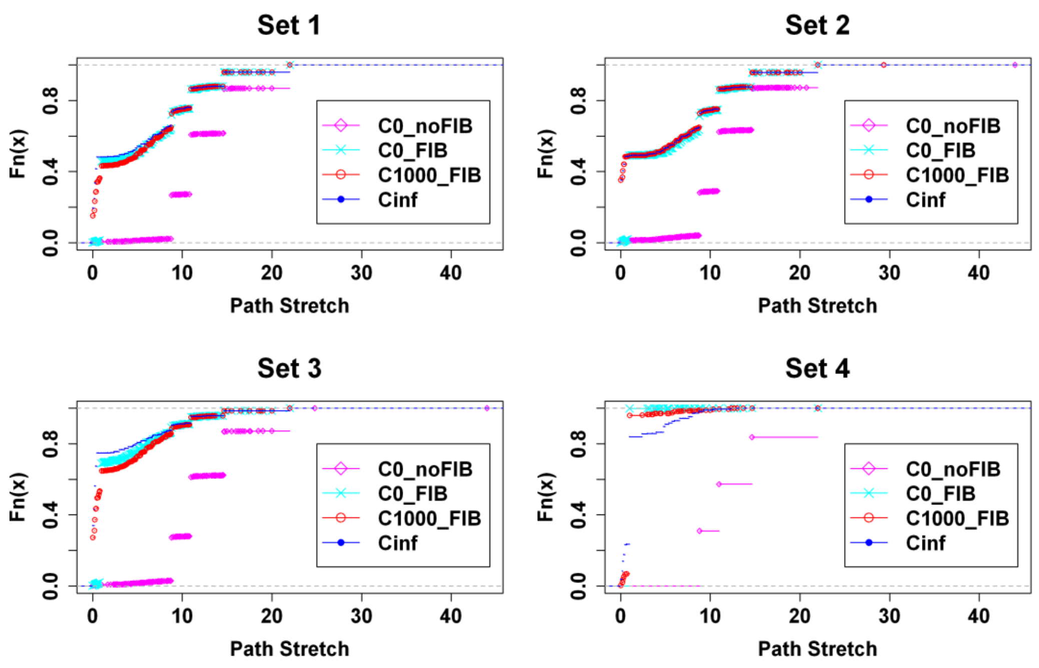


Figure 3-5. CDFs for Expended Effort in Network 39-node

Under CCN-Flooding, caching, on the other hand, has limited impact on Expended Effort. To illustrate this point, consider the extreme example of Cinf. Trivially, an infinite-sized cache will provide the best cache hit ratio irrespective of network topology or workload characteristics (for the case of Sets 1, 2 & 3; the anomalous impact of caching on Set 4 has been dealt with in the preceding subsection and is excluded from our discussions for the remainder of this chapter). However, a request that does not locate its data object directly in its own source node's cache will cause an interest packet to be flooded, thereby, by and large, entailing the full bandwidth cost of this flooding even though the data will be encountered in the caches of very nearby routers.

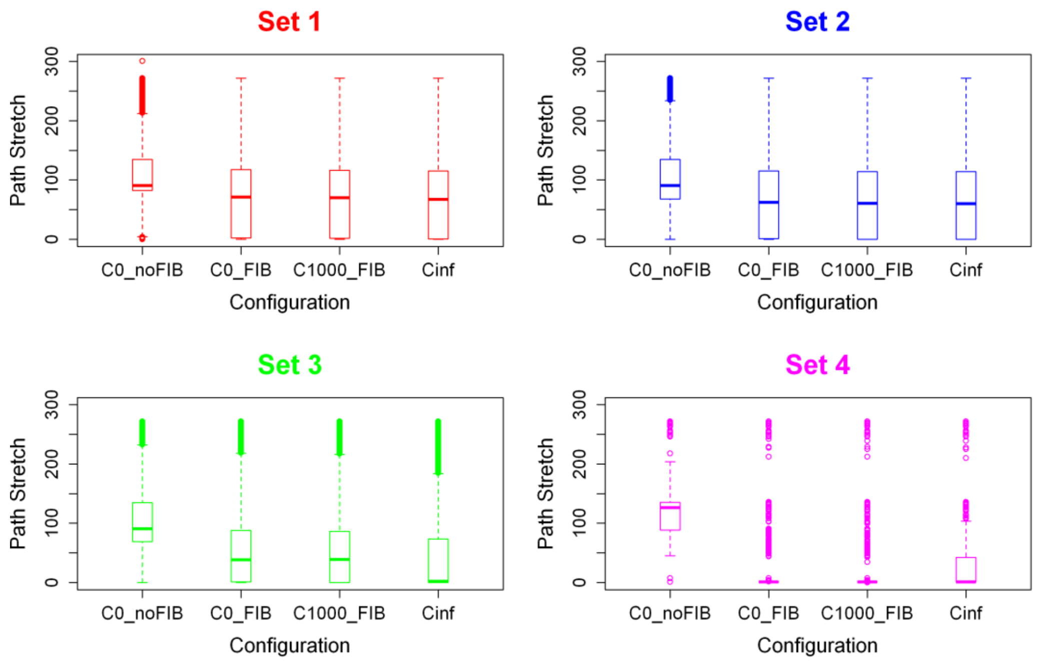


Figure 3-6. Box Plots for Expended Effort in Network 51-node

This point can be seen from Figures 3-5 and 3-6 when we move from C0_FIB to C1000_FIB, adding in-network caching to the system. While some improvement in performance for Sets 1, 2 and 3 occurs, it is nowhere as marked as the improvement achieved when FIBs were introduced. Lest it be thought that the modest contribution of caching to Expended Effort performance is due to inadequate cache size, consider the hypothetical Cinf case, which yields the theoretical upper bound of system performance for Sets 1, 2 & 3. In going from C1000_FIB to Cinf, we note relatively little improvement, though the specific amount of benefit derived varies somewhat with topology and workload. From this we may conclude that cache size 1,000 is broadly adequate from the perspective of Expended Effort. Indeed, for Set 2 under both 39-node and 51-node, and for Set 1 under 51-node, one may even say that the networks are well-provisioned cache-wise.

3.3.5 In-network Caching: Path Cost

The effect of in-network caching on retrieving data objects from caches closer to the requesting node is well-attested in the literature. We call the Path Stretch thereby achieved "Path Deflation" in order to distinguish it from our own Path Cost metric. Path Deflation is shown by the red boxes in Figure 3-7, which presents the case of C1000_FIB. In the context of CCN-Flooding, Path Deflation reflects the Path Stretch for the copy of the data object retrieved from the nearest cache, which could be an "off-path" cache, in the sense that it need not lie along the shortest path route to the owning node. Because of flooding, multiple, replicate copies of the object could be fetched from caches along other paths in the network. Path Cost (blue boxes in Figure 3-7) is the overall bandwidth consumed in retrieving all copies of the object, and therefore reflects the cost overhead for data packets entailed by CCN-Flooding. As seen from the figure, Path Cost > Path Deflation. This is true for all workloads, in all four topologies, under all configurations.

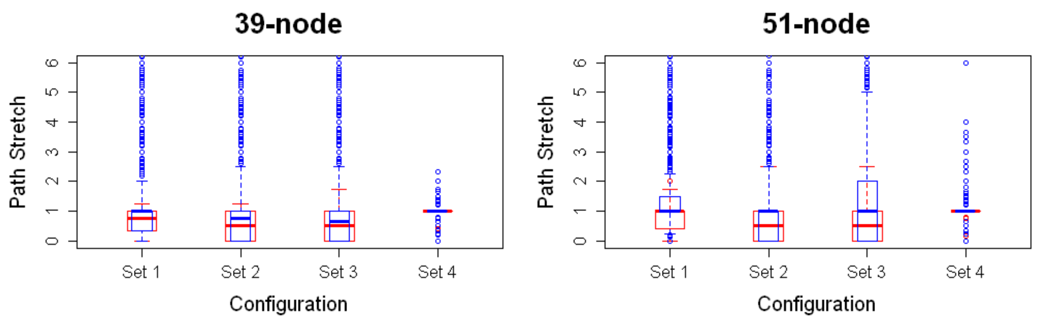


Figure 3-7. Box Plots for C1000_FIB: Path Cost (blue) *vs.* Path Deflation (red)

As expected, Path Deflation improves the more "cache-friendly" the workload (compare the red boxes for Set 1 *vs*. Sets 2 & 3, for each of 39-node & 51-node). The same cannot be said in general for Path Cost, for which topological traits come into play. There is not much overall difference, for example, between Sets 1 & 3 (blue boxes) for 51-node despite their distinct distributional characteristics (averages: 1.42 *vs*. 1.54, respectively; medians: 1 *vs*. 1; 95th quantiles: 4 *vs*. 5.34; standard deviations: 1.83 *vs*. 2.32).

3.4 Results — CCN-Flooding *vs.* CCN-Publication

In this section, we briefly compare some aspects of bandwidth consumption under CCN-Flooding and CCN-Publication. As previously mentioned, there is no FIB information at the start of the simulation for CCN-Flooding. During the simulation, the FIBs are populated over time as data packets are satisfied by "owning" nodes. Thus, CCN-Flooding accounts for the bandwidth costs incurred in populating the FIBs. A direct comparison with CCN-Publication therefore necessitates that we also account for the cost of pre-populating FIBs in the latter. We do so by calculating that cost and amortizing it over interest packets. In doing so, we assume that only one named object is advertized per "publication" packet and that these publication packets are roughly of the same size as interest packets. The amortization is done on a per-object basis: in other words, the cost of publicizing a given object is evenly amortized across the requests (interest packets) for that particular object. This makes the overhead OSPF costs cheaper for a request targeting a more popular object than for a request targeting a less popular one. Our results are presented for the 39-node and 51-node networks under the C1000 configuration in Figures. 3-8 and 3-9.

Figure 3-8 gives the Expended Effort for CCN-Flooding *vs*. CCN-Publication. The results could well have been anticipated: by and large, interest packets expend less bandwidth in locating data objects under CCN-Publication than CCN-Flooding, even when the OSPF "set-up" costs of the former are taken into account. The interesting point, however, is that for 39-node in particular, the cost differential is perhaps less than one might have expected. Indeed, for the more cache-friendly workload Sets 2 and 3, the median cost under CCN-Flooding is actually less than under CCN-Publication. Furthermore, of the four topologies, 39-node, though synthetically derived, perhaps comes closest to what a “typical” small ISP/backbone might look like, having core-edge separation in which 31 repository/gateway edge nodes are evenly distributed around a core of 8 non-repository router nodes.

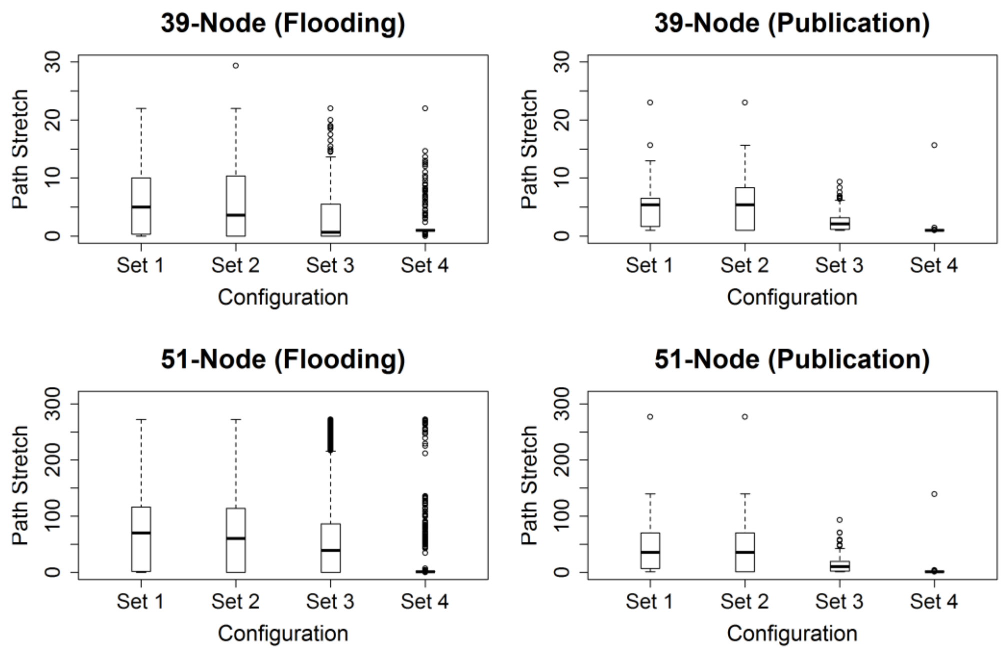


Figure 3-8. Box Plots for Expended Effort: CCN-Flooding *vs*. CCN-Publication under C1000 Configuration

These results, of course, have to be treated with a certain amount of caution. They are dominated by the amortized cost of populating the FIBs under CCN-Publication. The specific number of requests for a given object therefore significantly affects the bandwidth cost entailed in requesting and retrieving that object. Smaller or larger workloads (*i.e.*, with fewer or more requests over the same number of objects) will have significant bearing on the results obtained, as well as the other assumptions underlying these OSPF set-up costs (*e.g*, one named object is advertized per publication packet; publication packets are roughly the same size as interest packets; *etc*.). Nevertheless, our results at least serve to indicate that the issue perhaps could merit further investigation.

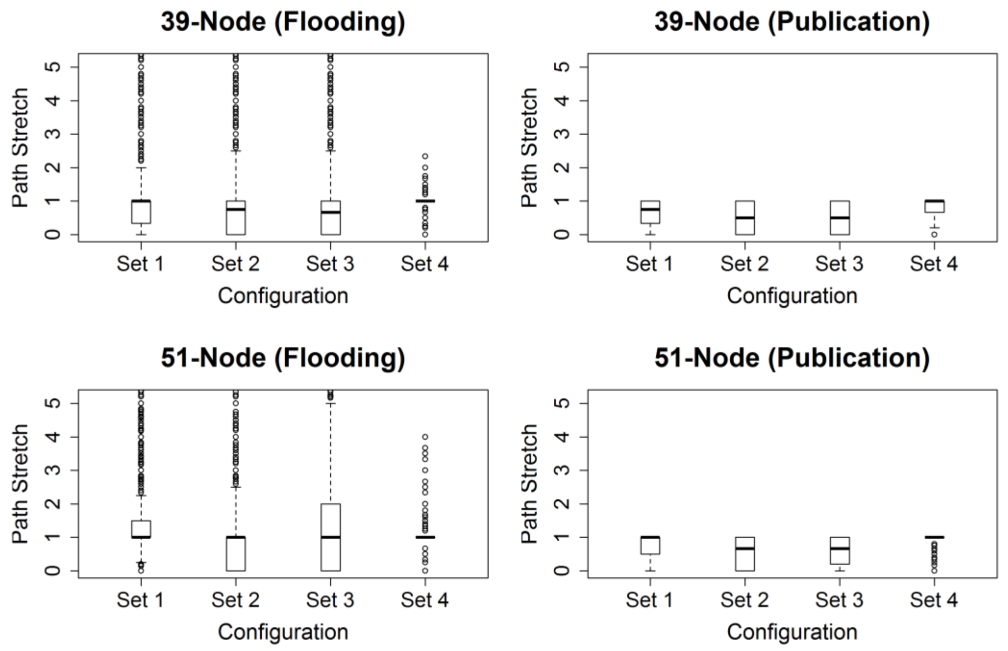


Figure 3-9. Box Plots for Path Cost: CCN-Flooding *vs*. CCN-Publication under C1000 Configuration

Figure 3-9 gives the Path Cost for CCN-Flooding *vs*. CCN-Publication and clearly shows that less bandwidth is expended in fetching data objects under CCN-Publication than under CCN-Flooding. One of the advantages of CCN-Flooding cited by its proponents is that it is able to locate copies of the data in caches that are off the shortest path and potentially closer to the requesting node, unlike CCN-Publication under which objects are satisfied from caches along the shortest path route only. While this might well be the case and of advantage in improving delivery performance [48], it is also clear that because CCN-Flooding fetches multiple copies of the data from wherever it finds them, off-path caching does not yield any bandwidth savings benefit. As seen from Figure 3-9, Path Cost values greater than 1, indicating that multiple copies of the object were received from multiple locations along various paths, are not uncommon. Finally, it is perhaps worth noting from the figure that the bandwidth cost differential between CCN-Flooding and CCN-Publication for 39-node, while clearly in favor of the latter, is less than the corresponding cost differential for the two approaches in 51-node.

3.5 Results — CCN-Publication

The focus of our attention in this section will be Cache-PIT dynamics under CCN-Publication. For simplicity, we assume a loss-free environment which obviates the need for interest packet and PIT timeouts, with all the inherent calibration issues that these entail.

PIT aggregation occurs when an interest packet requesting an object is suppressed at an intermediate node because there is already a pending PIT entry generated by a previous interest packet for that same object. More to the point, PIT aggregation is only possible if second and subsequent interest packets arrive within an RTT of the one that created the PIT entry. Thus, to achieve a discernible PIT aggregation effect, we need to generate interest packets at a high rate so that more interest packets overlap within a single RTT. On the other hand, increasing the generation rate increases queue buildup and congestion at the nodes, which will increase network RTT and thereby accentuate the PIT aggregation effect.

To observe the PIT aggregation effect, we set the mean of the exponential inter-generation time distribution to 0.025 time units (EM=0.025). The networks are significantly congested at this generation rate, with traffic intensities for the most central, congested nodes reaching 0.99 in all four topologies. We call this the "congested network" scenario, as compared to the "non-congested" scenario for which EM=1.0, yielding traffic intensities at the nodes under 0.1 for all topologies and workload sets.

3.5.1 PIT Aggregation: Topological Considerations

Figure 3-10 shows the Expended Effort for the four network topologies under workload Set 1 with EM=0.025, and a PIT but no caching, so as to further highlight the PIT aggregation effect.

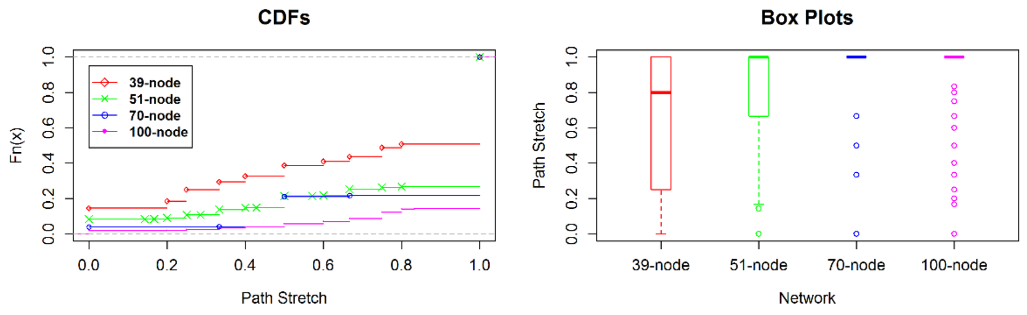


Figure 3-10. CDFs and Box Plots for the Congested Network Scenario with a PIT but no Caching: Workload Set 1

A pattern of decreasing benefit derived from PIT aggregation may be observed as we move from the 39-node topology, to 51-node, 70-node, and 100-node, in that order. As may be seen from the CDF plots of Figure 3-10, the proportion of requests that retrieve their objects from owning nodes (i.e., Path Stretch = 1) are: 0.50 (39-node), 0.72 (51-node), 0.78 (70-node), and 0.83 (100-node). Average Expended Effort, of course, follows the same pattern, increasing monotonically from a value of 0.66 for 39-node, which is leveraging the PIT the most effectively, to 0.93 for 100-node, which leverages it the least. While Figure 3-10 shows the results only for workload Set 1, this pattern is consistent across workload Sets 2 and 3 as well.

This trend relates directly to the cBC values in Table II. 39-node has the highest cBC value (0.41) amongst the four networks. This cBC value reflects the relative imbalance in 39-node of having a higher proportion of shortest paths passing through a smaller subset of central nodes. There is a further aspect of 39-node which enables it to better leverage PIT aggregation benefits. 39-node has core-edge separation, where 31 edge nodes are connected to a small core of 8 nodes. Thus, when the network is congested, these core nodes act as effective points for PIT aggregation. For 51-node and 70-node, the smaller cBC values of 0.29 and 0.22 indicate that shortest path are more evenly distributed amongst intermediate central nodes. Even though only one path is selected to populate the FIBs out of possibly multiple shortest paths connecting a given pairs of nodes, all things being equal FIB shortest paths will nevertheless be more dispersed amongst the central nodes. This will essentially attenuate the potential for PIT aggregation. Of the four networks, 100-node has the most evenly distributed shortest paths over central nodes, and consequently leverages the least from PIT aggregation.

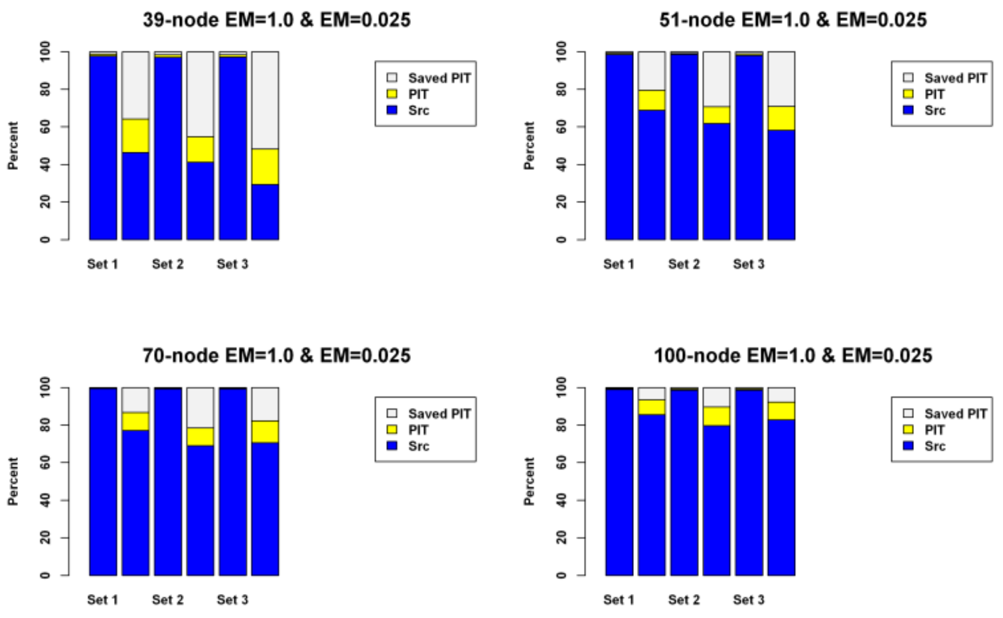


Figure 3-11. Bar Charts for the Number of Hops Expended with PIT but no Caching

The figure gives a set of bar charts for each of 39-node, 51-node, 70-node & 100-node. There are two bar plots for each of workload Sets 1, 2 & 3. The first is for a generation rate EM=1.0 and the second for EM=0.025. An individual bar is calculated based on the total number of hops expended in an IP network for the corresponding workload set, and gives the percentage of this base number that is expended in the CCN-Publication network due to: (i) going the full path to the owning node to fetch the requested object (Repository); and (ii) fetching the object by means of PIT aggregation. Also shown is the percentage of hops thus saved as compared to the IP network base value due to: (iii) PIT aggregation (Saved PIT).

Lower Expended Effort, used as a direct measure of bandwidth expenditure and savings, can be misleading – the more so for networks with small diameters. For example, 70-node has a diameter of only 3; it also has many nodes with significant degree, as reflected by its cDC value of 0.7886. Requests satisfied from owning nodes only 1-hop away from the requesting nodes nevertheless contribute a Path Stretch of 1. In Figure 3-11, we compare the number of hops saved over an entire workload set as a percentage of the hops expended in a shortest-path IP network (*i.e.*, with no PITs). The bars in the figure corresponding to EM=0.025 confirm that 39-node saves the most hops, followed by 51-node, 70-node, and 100-node, in that order, for all three workload sets. Note, incidentally, that the bars corresponding to EM=1.0 (non-congested scenario) show virtually no savings from PIT aggregation.

3.5.2 Cache-PIT Interplay

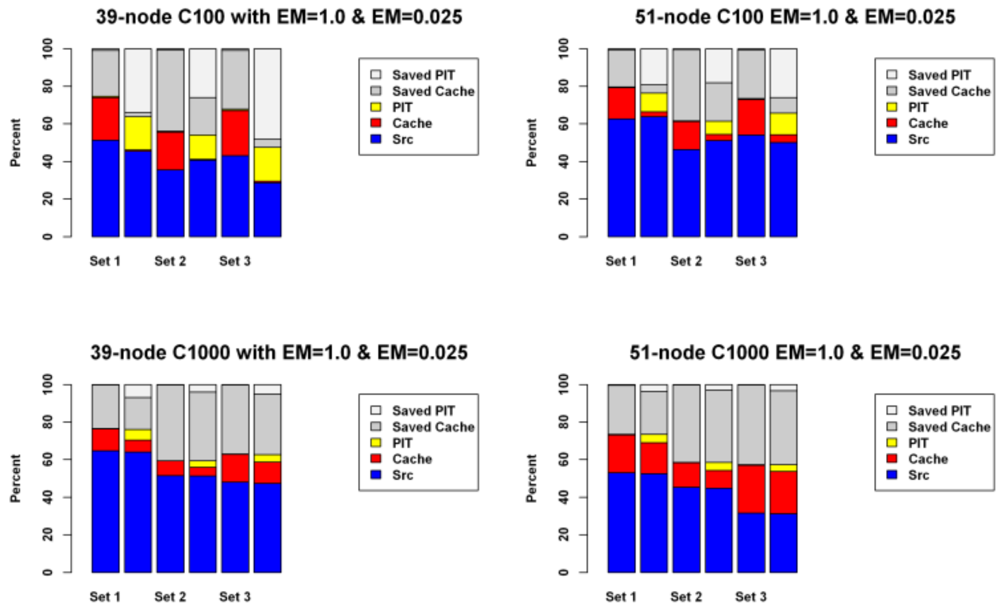


Figure 3-12. Bar Charts for the Number of Hops Expended under C100 and C1000 for 39-node and 51-node

This figure is similar to Figure 3-11 but also gives the percentages of the number of hops: (a) expended by requests satisfied due to cache hits under CCN-Publication; and thus, (b) saved due to these cache hits, as compared to the IP network base value.

To elucidate the Cache-PIT relationship, we consider configurations with cache sizes equal to 100 (C100) and 1000 (C1000) objects. Figure 3-12 compares 39-node and 51-node under congested and non-congested scenarios across the workload sets, for configurations C100 and C1000.

Focusing on the lower two sets of bar charts in Figure 3-12, which are for C1000, we compare the pair of bars for EM=1.0 and EM=0.025 for each workload set in turn, for both 39-node and 51-node. There is no significant change in the overall savings achieved between the non-congested and congested network scenarios for a given workload set in a given topology. Rather, what happens in going from the one scenario to the other is that gain achieved by cache hits in the non-congested case are substituted for by a roughly equal amount of gain achieved by PIT aggregation in the congested case. It thus becomes quite apparent that gains from PIT aggregation and cache hits are not additive but rather somewhat alternates for each other. In the upper two sets of bar charts, for C100, while some extra hop savings are achieved in going from the non-congested to the congested scenarios, nevertheless the same phenomenon prevails overall: gains from PIT aggregation by and large replace gains from cache hits, though not in a proportionate fashion, thus leading to some extra savings. For a given network serving the pattern of requests defined by a given workload under non-congested conditions (*i.e.*, these requests are issued at a low generation rate), RTTs are shorter and subsequent requests for the same object are more likely to find earlier requests for that object already satisfied, thereby both nullifying the opportunity for PIT aggregation and increasing the chances of locating the object in a cache. Under congested conditions, RTTs are longer, earlier requests for the object are more likely still to be pending, thereby both decreasing the probability of having the caches populated with that object and increasing the opportunity for PIT aggregation. Thus, in moving from non-congested to congested, cache hits become less and PIT aggregation becomes more likely, with the latter acting as a sort of "compensatory" mechanism for "gains" deficiencies caused by "inadequately" populated caches.

When we compare C100 to C1000 in Figure 3-12 for a given workload in a given topology under the congested network scenario, we note that the amount of hops saved by PIT aggregation decreases, while hops saved due to cache hits increases. This hold true across all workload sets for both topologies. It becomes apparent that increasing cache size does not dramatically improve overall savings: hops saved by PITs are replaced by hops saved by caches. Better-provisioned caches increase the chances of locating the desired object through a cache hit, which simultaneously obviates the possibility of having a pending PIT request at that node for the object.

3.6 Summary

In this chapter we presented simulation-derived results and observations on a disparate set of issues related to CCN networks. These issues come together to the extent that they all relate to bandwidth consumption on the one hand, and have not received much attention in the current literature, on the other. We examined both the CCN-Flooding approach and the more current CCN-Publication approach.

For CCN-Flooding we identified and reported on the following issues:

– Proper calibration of PIT timeouts is sensitive to topology and inherently difficult. This observation also holds for CCN-Publication, but has graver consequences for CCN-Flooding.

– Flooded interest packets create PIT entries all over the network causing sectors of the network that do not lie along the paths of the returning data objects to become isolated for the duration of the PIT entries with respect to new requests for these same objects, due to the PIT suppression effect. This triggers off spurious interest packet timeouts and bandwidth wastage.

– FIB routes based on volatile cache content can create circuitous paths. However, if they are disallowed then performance degrades in a different way: an object satisfied from a cache does not create a FIB entry and subsequent requests for the object will flood, consuming extra bandwidth.

It is worth pointing out that the issues identified with CCN-Flooding in this chapter are applicable to the version of CCN-Flooding used in Chapter 4. The latter is a far more aggressive version of the strategy since it does not maintain any FIB tables. All requests are therefore flooded in the network. All issues identified here are further magnified for the CCN-Flooding version used in Chapter 4.

For CCN-Publication, we focused on PIT-cache dynamics and showed that the PIT aggregation effect increases commensurately with both network load and the Betweenness Centrality attribute of the topology. While both PIT aggregation and in-network caching contribute significantly to achieving bandwidth savings, they do not do so in an additive fashion. As traffic load increases, enhanced gains from PIT aggregation tend to, by and large, replace diminishing gains from cache hits.

We also examined bandwidth consumption in CCN-Flooding *vs*. CCN-Publication with the overhead bandwidth costs of pre-populating FIBs using OSPF-like mechanisms in the latter approach taken into account. We find that, whatever advantages off-path caching might offer for improved system delivery performance under CCN-Flooding, these come at the expense of increased bandwidth consumption as compared to CCN-Publication.

Chapter 4

CCN-FOH: A Lightweight Forwarding Strategy

4.1 Introduction

A primary objective of any CCN forwarding strategy is to improve user experience by exploiting in-network caching on the one hand, while minimizing the associated costs on the other. To this effect, CCN-Publication implies an extremely conservative approach to in-network caching. Essentially, it performs like IP forwarding, but with on-path caching along the way. It does not attempt to seek the closest available in-network cached copy of the content. CCN-Flooding, on the other hand, has the potential to do so. Because the focus here is on enhancing the user experience by leveraging in-network cached content, the version of CCN-Flooding we adopt is different from that of Chapter 3. The Chapter 3 version uses flooding as a mechanism only to locate content and populate the FIBs with routing information when such routing information happens to be unavailable. The version of CCN-Flooding used here does not maintain or use FIB entries. It uses flooding for every request. Thus, with every request, it carries out an exhaustive search for cached and repository content, and ubiquitously populates the in-network caches with the content it finds. In this respect, CCN-Flooding leverages off-path caches and pulls content closer to requesting nodes very effectively, and will always locate and retrieve the closest copy of the content.

It might seem upon superficial consideration that an aggressive strategy such as CCN-Flooding should prove more beneficial to user experience than the minimalist approach of CCN-Publication. However, on careful analysis, it becomes apparent that this is not necessarily the case. We briefly present results in Table IV that compare CCN-Publication and CCN-Flooding in this context. They are derived from simulations of a 154-node network[3] using a cache-friendly workload[4] with c0.1p[5] cache size.

Table IV: CCN-Publication *vs*. CCN-Flooding

Metrics	Cache Size (c0.1p)	
	CCN-Publication	CCN-Flooding
AvgHits/Req	0.74	9.35
Nearness	100%	77.88%
AvgTIHops/Req	100%	5329.45%
AvgTDHops/Req	100%	541.68%
AvgRT/Req	100%	215.6%

Initially, let us examine Average Cache Hits per Request (AvgHits/Req) and Nearness. AvgHits/Req provides a metric by which we can measure both: (a) how widespread the search for cache content is; and (b) how pervasively we populate the caches themselves[6]. Table IV shows that AvgHit/Req for CCN-Flooding is more than twelve times (9.35/0.74) greater than CCN-Publication. Typically, the more copies located per request, the better the chances of finding content closer to the requesting node, which is measured using Nearness. The Nearness value for CCN-Flooding in Table IV indicates that data packets of the nearest-located object

[3] It contains 154 nodes of which 123 are edge nodes and 31 are core nodes. The edge nodes perform the function of consumers (requesting nodes) and producers (repositories). See Subsection 4.3.2 for further details.
[4] Interest requests are generated using a Zipf-Mandlebrot distribution with $\alpha = 1.5$ for the object requested, from a total object space of 10,000. See Subsection 4.3.3 below for details.
[5] A cache size of "c*X*p" indicates that the cache can hold up to *X*% of the objects in the total object space.
[6] (a) & (b) are tightly coupled given that the search strategy tends to directly impact how caches are populated with content.

satisfying an interest request travelled, on average, 77.88% of the number of hops traversed under CCN-Publication. Thus, for c0.1p caches, which we consider to be under-provisioned cache sizes in this context, CCN-Flooding does succeed in bringing content closer compared to CCN-Publication. In terms of Nearness, CCN-Flooding clearly out performed CCN-Publication.

From the perspective of user experience and bandwidth costs, however, the situation may be entirely different. We measure user experience in terms of the Average Response Time per Request (AvgRT/Req). Bandwidth costs for interest packets are measured as the Average Total Interest Hops per Request (AvgTIHops/Req); for data packets we use the Average Total Data Hops per Request (AvgTDHops/Req)[7]. The costs of retrieving multiple copies from caches and having content closer in CCN-Flooding does not come cheaply. Nearness does not necessarily improve the user experience given the network overheads entailed. For example, Table IV shows that even though CCN-Flooding retrieves content closer, its bandwidth costs, 5,329.45% and 541.68% for AvgTIHops/Req and AvgTDHops/Req, respectively, are significantly higher than CCN-Publication. The AvgRT/Req for CCN-Flooding is also 215.6% that of CCN-Publication. Therefore, CCN-Flooding's success in bringing content closer comes at such exorbitant cost in terms of network resources and congestion that overall user experience decidedly degrades in this instance. We may conclude that, if user experience is the determinant criterion for a strategy's success, getting content closer is not necessarily advantageous. From this perspective, CCN-Publication significantly outperformed CCN-Flooding.

[7] Interest and data packets vary by an order of magnitude the one from the other, with a commensurate difference in the raw bandwidth each type consumes per hop in the network, so we use a separate metric for each.

CCN-Publication and CCN-Flooding may be viewed as two ends of a spectrum. CCN-Publication has low bandwidth costs but is not very effective in provisioning and leveraging cache content. CCN-Flooding, on the other hand, provisions and locates cache content in a pervasive manner, but at high bandwidth cost. In this chapter we present a new lightweight forwarding strategy, and compare it to CCN-Publication and CCN-Flooding. The new strategy is an extension of CCN-Publication that assimilates some of the positive attributes of CCN-Flooding, but in a balanced manner. It improves on CCN-Publication by leveraging the presence of in-network cache content more effectively, without entailing the high costs associated with CCN-Flooding's all-pervasive approach. The chapter is structured as follows. Section 4.2 introduces the new CCN forwarding strategy. Section 4.3, describes our experiment design space and methodology. Section 4.4 presents our results and discussion.

4.2 New Forwarding Strategy: CCN-FOH

The design of the new forwarding strategy is grounded in two principles. The first, which follows from CCN-Flooding, is to enhance the exploration of caches, at least as compared to CCN-Publication. The more extensive the search space, the higher the chances of locating content closer, especially when we have small-sized caches. The second principle is to keep the costs entailed by the first principle manageable, so that user experience response time is improved, or, at the very least, does not degrade.

The new forwarding strategy is built as an extension to CCN-Publication. In addition to forwarding an interest packet along the shortest path to the repository of the requested object, the new strategy also explores the immediate, 1-hop neighbors of each node along that path. At each hop along this shortest path, we “flood” a special “exploration” interest packet[8] on all interfaces excluding: (a) the interface on which the original interest packet was received; and (b) the interface on which the original interest packet will be relayed forward along the shortest path. The exploration packet, when received at the neighboring node, will experience one of three possible outcomes: (1) it will find the content, in which case a data packet will be sent back, and this cache hit will be accounted for as an off-path cache hit; (2) it will find a PIT entry for this object, in which case, we will update the PIT entry with the incoming interface of the exploration packet, and when the PIT entry is satisfied a data packet will be created, and this request will be recorded as satisfied by the PIT; or (3) it will not find the content in the cache or a PIT entry for the object, in which case the exploration packet is terminated. It is worth stressing that in outcome (3) the exploration packet does not create new PIT entries (this is a point we shall return to in Subsection 4.4.4 below). However, as mentioned in outcome (2), exploration packets do take advantage of the PIT mechanism if a PIT entry is already present.

In case of a cache hit from an exploration packet, the corresponding data packet returning to the requesting node will consume all the PIT entries created by the original interest packet at nodes along the shortest path that it had traversed up to that point. This interest packet, on the other hand, continues its journey to the repository and will ultimately cause a corresponding data packet to be generated. This data packet will eventually be suppressed downstream as all the PIT entries beyond a certain point on the return path will have been consumed by the data packet created by the exploration packet. If exploration packets do not find any

[8] No such packet is flooded from the last node, one hop away from the repository, which would be pointless.

cache hits then the new forwarding strategy behaves exactly the same as CCN-Publication. We call our new forwarding strategy CCN-FOH ("Flood One Hop").

4.3 Experiment Design Space and Methodology

This is a simulation-based study in which, unlike Chapter 3, we use the CCN simulator ndnSIM [68] to perform our experiments. Similar to Chapter 3, however, the experiments implement a varied set of topologies (both synthetic and real-world) with distinct topological attributes (Subsection 4.3.2), running two workload sets of different and contrasting characteristics (Subsection 4.3.3), using different cache sizes (Subsection 4.3.4).

4.3.1 Simulator

CCN-Flooding and CCN-Publication are implemented in the pre-packaged version of ndnSIM [68], where they are referred to as "flooding" and "best route", respectively. To conduct our experiments, we have extended ndnSIM's functionality. We briefly highlight a few of these extended features:

- CCN-FOH and its variants are implemented as separate forwarding strategies.
- The "NDN-Consumer-Zipf-Mandelbrot" application is modified:
    - We create prefix strings, denoting the name of the requesting object along with the name of the repository node at which it resides, by mapping the object id generated by the application to that of the repository. For example, if we request object id "5" which resides on repository node "60" then the prefix string will be "60/5".

  - Objects are randomly distributed across requesting/repository (edge) nodes.
  - The retransmission mechanism has been modified to take account of these changes.
- We have made the following modifications to collect more refined statistics:
  - PIT data structure is extended. In addition to the incoming interface of an interest packet, the PIT also stores the requesting node, nonce value, and introduces a strategy bit.
  - We implemented two new tracers. The first tracer, called the "main-tracer", keeps track of the entire history of all Interest and data packets at each hop. The second tracer, called the "cs-eviction-tracer", counts the number of times an object has been evicted from a cache.
  - We created several packet tags which aid in statistics collection.

4.3.2 Topologies

We use four network topologies which overlap with those of Chapter 3 (Subsection 3.2.2), but here we maintain core-edge separation for all four. Thus, the topologies comprise a set of core CCN router nodes surrounded by a set of edge nodes that act as gateway routers of stub user networks. Interest packets are generated only at the edge nodes which are also the only nodes that can "own" data objects (*i.e.*, an edge node acts as the repository for the data objects residing at the user network(s) behind it). The four topologies consist of the synthetically generated 39-node and 100-node networks (using BRITE [54]), and the real-world 51-node network. In place of the real-world 70-node network of Chapter 3, we introduce a new (real-world) 154-node network. We summarize the main topological characteristics of the four networks in Table V, and show the networks themselves in Figure 4-1.

Table V. Characteristics of the Network Topologies

Nodes	Type	|E|/|V|	|D|	cDC	cSC	cBC
39	Synthetic	1.05	5	0.21	0.63	0.41
51	Real-world	3.15	7	0.34	0.79	0.29
100	Synthetic	2	6	0.08	0.24	0.10
154	Real-world	1.182	6	0.18	0.82	0.42

|E|/|V| is the ratio of edges to vertices; |D| is the diameter; cDC, cSC & cBC are centralized graph metrics.

The 39-node and 100-node synthetic networks are in all respects exactly the same as in Chapter 3. We repeat their specifics here for convenience. The 39-node network comprises 8 core nodes that are almost like a ring, with 31 edge nodes attached. For the 100-node topology, we arbitrary designated nodes of degree ≤ 2 as edge nodes and the rest as core nodes. The network thus comprises 68 core nodes of degree ≥ 3 and 32 edge nodes which all happen to have exactly degree 2 (there are no degree 1 nodes).

The real-world topologies were, as in Chapter 3, selected from the well-known collection of PoP-level ISP maps generated using the Rocketfuel ISP topology mapping engine [55]. Although the topologies of the ISPs at this level show only the gateway routers, nevertheless, we take this as the initial point from which (unlike in Chapter 3) we partition the networks into core and edge nodes based on the Betweenness Centrality (BC) value of each node. All nodes with a BC of 0 serve as edge nodes, while the nodes with BC greater than 0 act as CCN routers. The 51-node network is formed of 29 such core and 22 such edge nodes. 154-node comprises 31 core nodes with 123 edge nodes. 51-node is the ISP map of TW Telecom (AS 4323), located in the US; 154-node is the ISP map of AT&T (AS 7018), also in the USA.

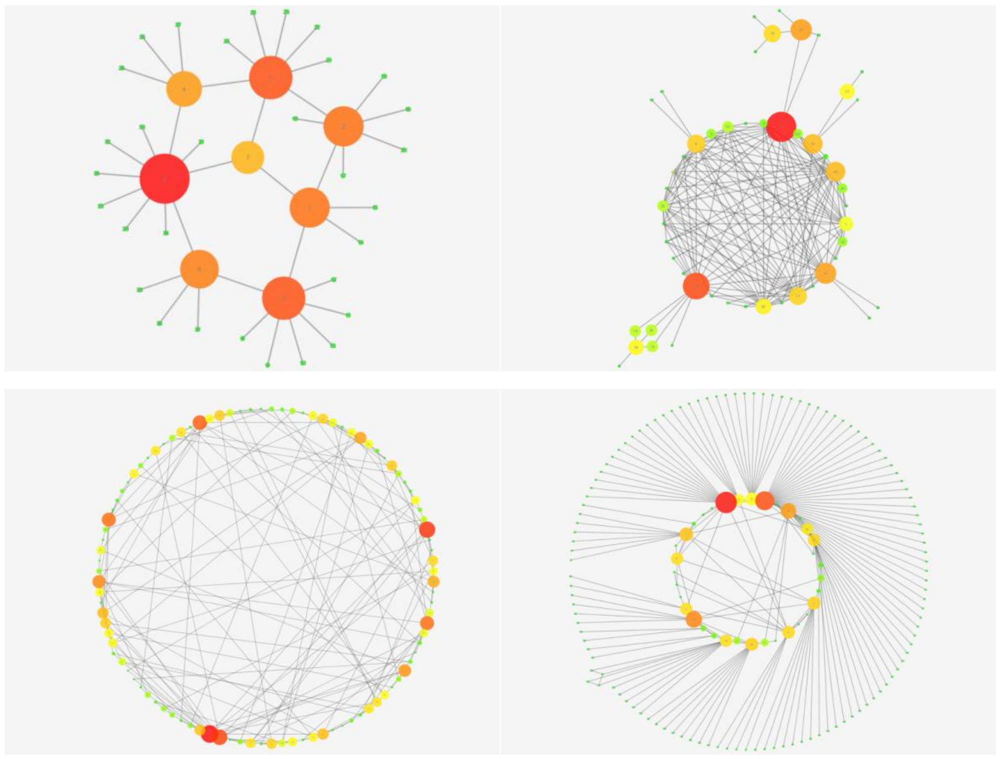


Figure 4-1. 39-node (*top-left*), 51-node (*top-right*), 100-node (*bottom-left*), and 154-node (*bottom-right*)

The illustrations are based on the Betweenness Centrality of nodes. The size of a node is directly proportional to its BC value: nodes with larger sizes and brighter colors (reddish) have higher Betweenness Centrality values, while ones with smaller size and lighter colors (greenish) have the lowest.

Table V gives the centralized topology metrics proposed by Freeman [57] and first introduced in Subsection 3.2.2 of Chapter 3. These depict the overall characteristics of a network, and we briefly restate their salient features here for convenience. The table gives the centralized Degree Centrality (cDC), centralized Stress Centrality (cSC), and centralized Betweenness Centrality (cBC). These metrics are derived from the corresponding standard graph topology measures for vertex centrality [56] for the nodes of the network, namely Degree Centrality (DC), Stress Centrality (SC), and Betweenness Centrality (BC), respectively. Centralized metric values (cDC, cSC, cBC) close to 0 imply that the nodes of the network have similar values for the corresponding vertex centrality measures (DC, SC, BC). On the other hand, values closer to 1 imply that there is a significant imbalance, with a few nodes having significantly higher corresponding vertex centrality values than others.

The cSC of network 154-node is 0.82, implying that this topology has a nexus of nodes with a disproportionately large number of shortest-path connectivity running through them. This may be compared to, for example, 100-node for which the cSC value is close 0.25, indicating that its nodes have relatively fewer such potential “choke” points. Networks 39-node and 51-node exhibit characteristics similar to 154-node, with cSC values of 0.61 and 0.79, respectively. On the other hand, 39-node, 51-node and 154-node, however, have relatively low cBC values (0.41, 0.29 and 0.42, respectively). Thus, even though each network has a small subset of potential choke point nodes through which a disproportionately large number of shortest paths run, these paths are evenly distributed amongst the nodes of the subset, thereby mitigating the potential choking effect. This can be corroborated in Figure 4-1 by the fact that 39-node, 51-node, and 154-node have quite a few nodes represented as large, bright points. In contrast, network 100-node, has a lot of nodes with similar Betweenness Centrality, and thus its overall cBC (0.1) is much lower. As a final note, networks 39-node and 154-node, although the one is synthetic while the other is real-world derived, tend to display similar characteristics notwithstanding the major size difference between them.

4.3.3 Traffic Generated

We deployed the NDN-Consumer-Zipf-Mandelbrot application of ndnSIM at the requesting nodes to create the traffic workload. We used two workload sets with different characteristics, as shown in Table VI.

Table VI. Workload Characteristics

Parameters	*Set 1*	*Set 2*
Distribution	**α** = 0.7 Zipf-Mandelbrot	**α** = 1.5 Zipf-Mandelbrot
Number of Objects	10,000	10,000
Object Median Size	10KB	10KB
Object Mean Size	10KB	10KB
Number of Requests	~100,000	~100,000
Requests/Object	~10	~10

These traffic workloads are similar in their broad characteristics to the Sets 1 and 2 workloads of Chapter 3, but differ in their specific details. As explained there (Subsection 3.2.3, Chapter 3), traffic workload parameters are categorized with respect to two criteria: (1) Content Popularity, and (2) Object Size.

Content Popularity. The Zipf-Mandelbrot α parameter value for workload Set 1 (α = 0.7) is selected from within the accepted range that represents the characteristics of web-like objects; the value for workload Set 2 (α = 1.5) from within the range for UGC (YouTube-like User-Generated Content), P2P, and VoD (Video on Demand) content. While the α value for Set 2 is almost double that of Set 1, the number of requests and objects is kept approximately the same as in Set 1 in order to more uniformly compare the effect of further biasing requests in favor of the more popular objects. We conducted initial experiments where we set the number of requests to 10 and 100 times the number of objects in order to gauge the degree of simulation start-up bias on the performance results collected. Runs of 100 times the number of objects were split up into ten consecutive, equal-sized batches and analyzed accordingly. It was determined that a run comprising requests 10 times the number of objects in the network space (i.e., 100,000 requests) did not suffer significantly from start-up bias. We therefore conducted our experiments using this run size.

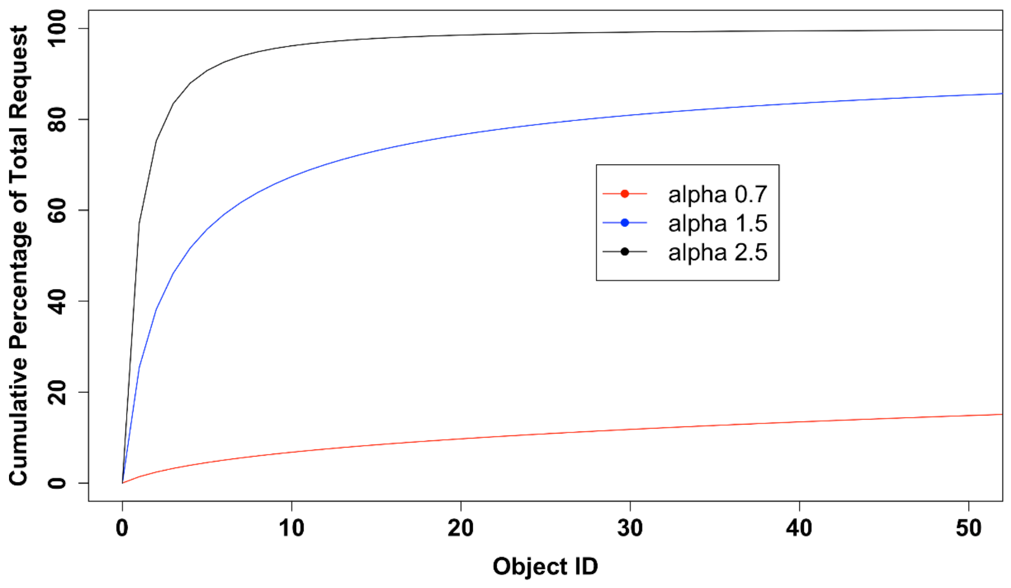


Figure 4-2. Cumulative Frequency of Requests for the Top 50 most Popular Objects generated using Zipf-Mandelbrot Distribution with α = {0.7, 1.5, 2.5} and N=10,000 objects.

Figure 4-2 shows the Zipf-Mandelbrot distribution for our chosen α values 0.7 and 1.5 (we also include α value 2.5 for comparison). It is quite evident from Figure 2 that there is a stark difference between the two workloads with respect to the percentage of requests issued for a subset of objects. With α value 0.7, the first 5 objects account for ~7% of the total 100,000 requests, and the next 15 objects for ~3% more; the remaining 9,980 objects account for 90%. It is clear that, for α value 0.7, there no significantly pronounced bias towards any particular object other than maybe the first few. On the other hand, the situation is quite different for α value 1.5 where there is a very pronounced bias towards the first 10 objects, which account for ~70% of requests. The next 30 objects amount to an additional ~10%, while the remaining 9,960 objects account for only ~20% of the requests.

Consider a cache that can hold, say, *n* objects and uses the traditional LRU or LFU cache replacement policies. By the nature of things, cache contents will be heavily biased towards the most popular objects. It should be evident then that the cache will prove much more effective when dealing with workload Set 2 ($\alpha = 1.5$) than workload Set 1 ($\alpha = 0.7$), the more so if the cache size *n* may be said to be under-provisioned, in the sense of being “too small”. As such, we may characterize workload Set 2 as being “cache-friendly”, especially when contrasted with the more “cache-unfriendly” workload Set 1.

Object size. We have configured the object size with a mean of 10KB [58, 63]. In comparable studies (e.g., [48]) an object space of 10,000 objects is taken as adequate. Again, in the initial phase, we experimented with object spaces of 100,000 objects but we did not observe significant differences in the results. It is worth noting that, when we increased the object space by an order of magnitude in this way, we also increased the cache sizes by an order of magnitude. Thus the ratio of object space and cache sizes remained the same during these initial configuration experiments.

4.3.4 Cache Sizes and Packet Generation Rates

Cache size. Cache sizes are dimensioned as a percentage of the object space. As shown in Table VI, workload Sets 1 and 2 have 10,000 objects each, of mean size 10KB, giving an object space of 100 MB. A cache of size *n* represents a per-node cache capacity of $n \times 10$KB, which we take to be capable of caching *n* objects of average size 10KB each. Experiments conducted with cache size 100, for example, therefore represent a per-node cache capacity (Cache/Catalog size) that is 1% of the object space. We refer to such a cache size as c1p, where 1p denotes 1%.

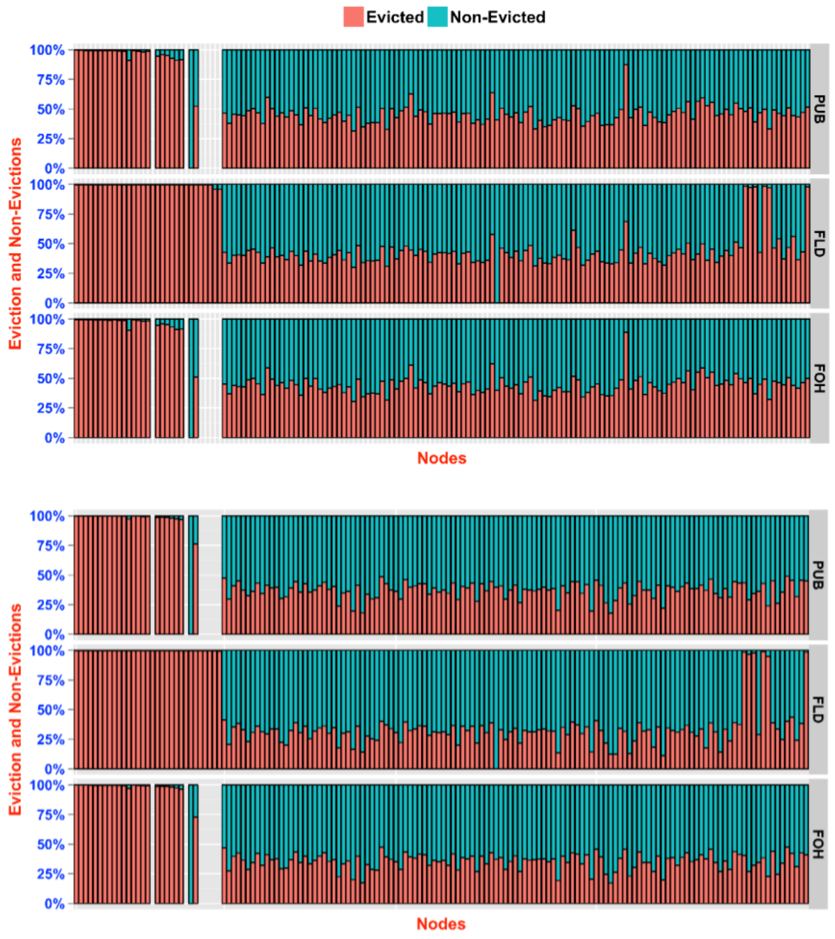


Figure 4-3. Percentage of Cache Evictions and Non-Evictions at Nodes for CCN-Publication (PUB), CCN-Flooding (FLD), and CCN-FOH (FOH) for 154-node under Workload: (a) Set 1 (*top*); (b) Set 2 (*bottom*)

In Figures 4-3(a) & (b) we show the percentage of cache evictions and non-evictions in the network 154-node, for all three forwarding strategies, with the smallest cache-size configuration of c1p and c0.1p for Set 1 and Set 2, respectively. Cache eviction at a node occurs whenever the node is processing a data packet whose object is copied into the cache, thereby evicting some data object resident there, as selected by the cache replacement policy. All data packets processed at the node which do not lead to such an outcome are counted as non-evictions. The small caches sizes were selected in order to experiment with under-provisioned caches. Nodes in the figure are presented in order of decreasing Betweenness Centrality. The first 31 nodes are the core nodes, and the rest are the edge nodes. In CCN-Flooding, all the core nodes have ~100% eviction rate. This is also the case for the core nodes that are heavily used in CCN-Publication and in CCN-FOH (thereby confirming that the selected cache sizes do indeed represent under-provisioned caches)[9].

In order to observe the effects of increasing cache size, we also use cache sizes c10p, c20p, and c32p for Set 1; and c0.5p, c1p, and c10p for Set 2. Overall, the Cache/Catalogue sizes in this chapter vary in the range [0.1% , 32%]. Previous ICN studies used values of 0.25% [65], [0.5% , 20%] [66], and [0.000001% , 1%] [67]. Our values are compatible with these.

Interest packet generation rate. We generate significant levels of traffic in the networks by fine-tuning the (Poisson) interest packet generation rate at edge nodes. For a given network configuration, this rate needs to be consistent across forwarding strategies, irrespective of workload type, so that we may compare between the strategies. The rate is determined by saturating the network with

[9] It is interesting to note that, in a network with reasonably well-distributed shortest paths such as the 154-node, some core nodes might not be utilized at all for forwarding under CCN-Publication and CCN-FOH. Such core nodes show absent bars that appear as white space in the figure.

interest packets to the point where it is on the brink of dropping packets at the router queues. We calibrate the rate in this way under CCN-Flooding, which is the most traffic intensive of the strategies and so is the first to incur packet drops. Thus, if we do not experience packets drops with CCN-Flooding for a given generation rate, none will occur under the other two strategies either. The interest packet generation rates we determined for each topology based on the criterion above is as follows: 60, 55, 40, and 10 interest packets per second at each requesting/edge node, for the 39-node, 51-node, 100-node, and 154-node networks, respectively.

4.4 Results and Discussion

In this section we present our results and analysis of the performance of CCN-FOH and the other forwarding strategies. First, in Subsection 4.4.1, we demonstrate the potential effectiveness of CCN-FOH by taking up and further examining the example we presented in Section 4.1 for the 154-node network, c0.1p cache size, under cache-friendly workload. Subsection 4.4.2 expands the focus to all topologies, all cache sizes and both workload sets and examines some of the topological effects that impact forwarding strategy performance. Subsection 4.4.3, briefly compares the bandwidth costs between the strategies. In Subsection 4.4.4 we examine the performance of CCN-FOH against, on the one hand, CCN-Publication, and on the other, CCN-Flooding from the perspective of user experience. We identify and analyze some of the factors that impact this user experience. Finally, Subsection 4.4.5 reports on our experiments with various variants of CCN-FOH. Note that we consistently report our results for response times (AvgRT/Req), Nearness, and bandwidth consumption (AvgTIHops/Req & AvgTDHops/Req) in terms of percentage differentials rather than absolute values in order to make it easier to assimilate comparative performance results across myriad combinations of forwarding strategy, network topology, and cache size configurations.

4.4.1 Potential Effectiveness of CCN-FOH

We expand upon the results for CCN-FOH for the 154-node network presented in Table IV, Section 4.1, where we have under-provisioned caches (c0.1p) and a cache-friendly workload, to demonstrate CCN-FOH performing effectively with respect to its two main design principles.

The first design principle of CCN-FOH, as described in Section 4.3, is to enhance upon the exploration of caches, at least as compared to CCN-Publication. Figure 4-4(a) shows that CCN-FOH has achieved this inasmuch as it has ~3 times more Average Cache Hits per Request compared to CCN-Publication (2.2 vs. 0.74 AvgHits/Req, respectively). As a direct consequence of its enhanced cache exploration, CCN-FOH successfully locates content closer compared to CCN-Publication; indeed so much so that in this instance it is comparable to CCN-Flooding (Figure 4-4(c)), even though CCN-FOH is far less aggressive and does not explore caches as extensively as CCN-Flooding (9.36 AvgHits/Req). The second design principle, and a vital component of CCN-FOH, is improving user experience, measured in terms of response time. In Figure 4-4(b), we see that CCN-FOH has an AvgRT/Req (Average Response Time per Request) that is 79.1% that of CCN-publication. CCN-Flooding’s AvgRT/Req is 215.6% that of CCN-Publication.

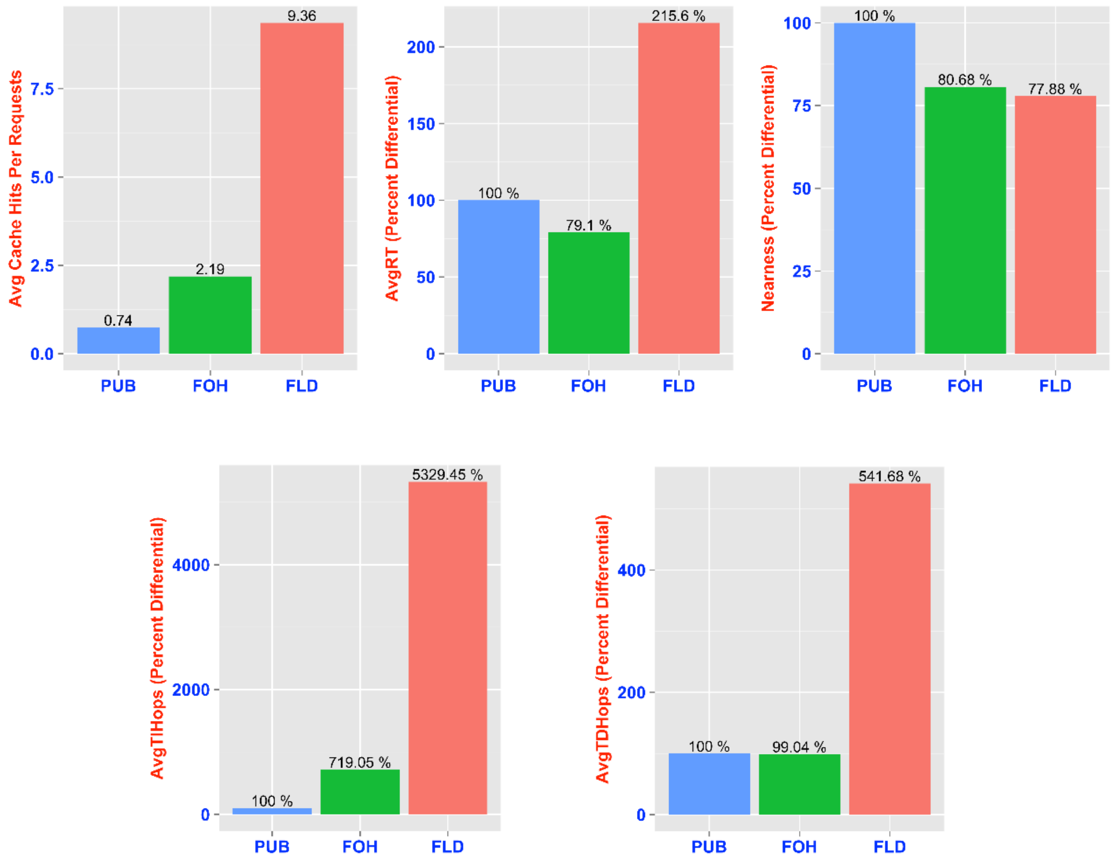


Figure 4-4. (a) Average Cache Hits per Request (*top-left*);
(b) Average-Response-Time-per-Request Differential (*top-center*);
(c) Nearness Percent Differential (*top-right*);
(d) Average-Total-Interest-Hops-per-Request Percent Differential (*bottom-left*);
(e) Average-Total-Data-Hops-per-Request Percent Differential (*bottom-right*)

Figures 4-4(d) & (e) show cost comparisons between the strategies for bandwidth consumed by interest and data packets, respectively. Since CCN-FOH is more aggressive in exploring caches, it exhibits a cost of ~720% more than CCN-Publication in terms of AvgTIHops/Req (Average Total Interest Hops per Request). However, in terms of the cost for AvgDTHops/Req (Average Total Data Hops per Request), CCN-FOH (99.04%) is virtually identical to CCN-Publication (Figure 4-4(e)). It is worth noting that the sizes of interest and data packets vary by an order of magnitude the one from the other (256 *vs.* 10,000 bytes), with a commensurate difference in the raw bandwidth each type consumes per hop in the network. Thus, even though CCN-FOH suffers from higher bandwidth costs for interest packets compared to CCN-Publication, this does not amplify to significantly larger raw bandwidth wastage because of the small size of these packets. Overall, the benefits of CCN-FOH do not come at an exorbitant cost. More to the point, whatever this cost, it does not negatively impact the user experience as in the case of CCN-Flooding.

4.4.2 Effect of Topology

We now expand our view to encompass all network topologies, all cache sizes, and both workloads in order to examine the effect that network topology has on forwarding strategy performance. Figures 4-5(a) & (b) show the ratio of satisfied requests for each workload, distributed between caches, repositories, and PIT[10]. Figures 4-6(a) & (b) show the relative Nearness for the different strategies. For each instance of a particular network, cache size and workload, there is a distinct positive correlation between the relative performance of the three forwarding strategies with respect to the requests satisfied at caches, on the one hand, and Nearness, on the

[10] Note that PITs have virtually no impact on overall performance – an interesting point in its own right which we set aside for now.

other. In each instance, a given strategy's performance relative to the other two is always the same for the proportion of requests satisfied at caches as it is for Nearness.

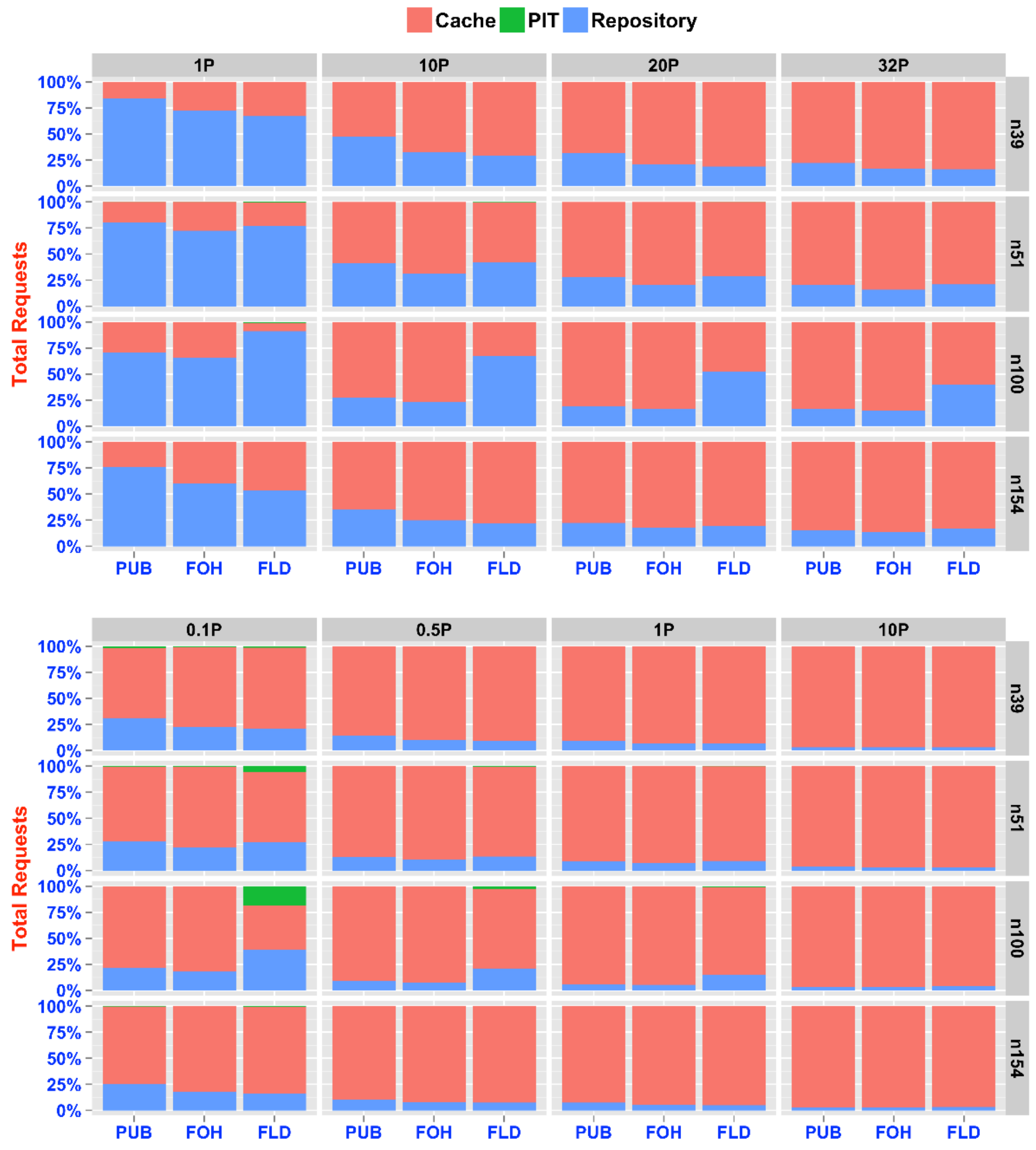


Figure 4-5. Requests satisfied between Caches, PIT, and Repository for: (a) Cache-Unfriendly Workload (*top*); (b) Cache-Friendly Workload (*bottom*)

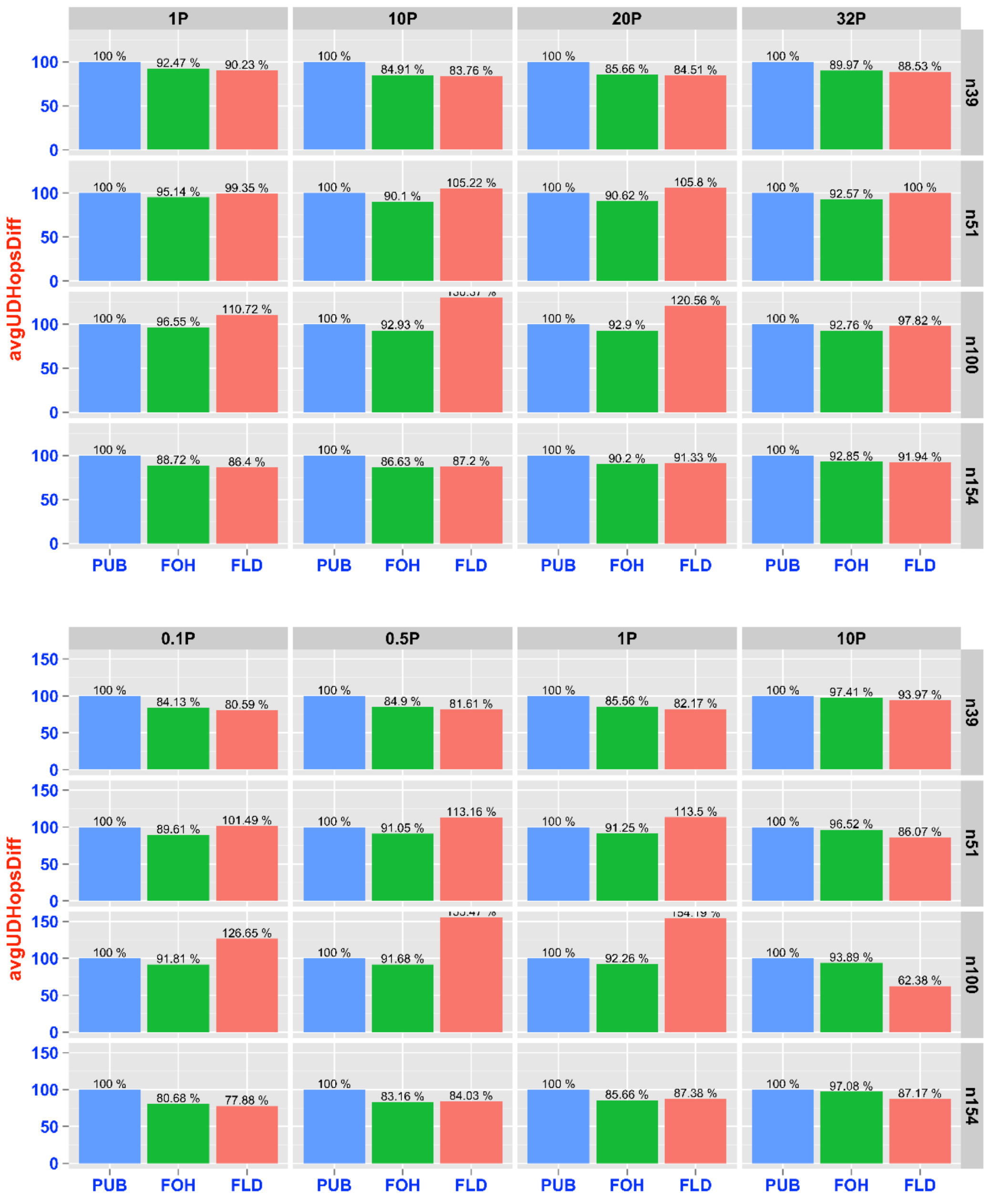
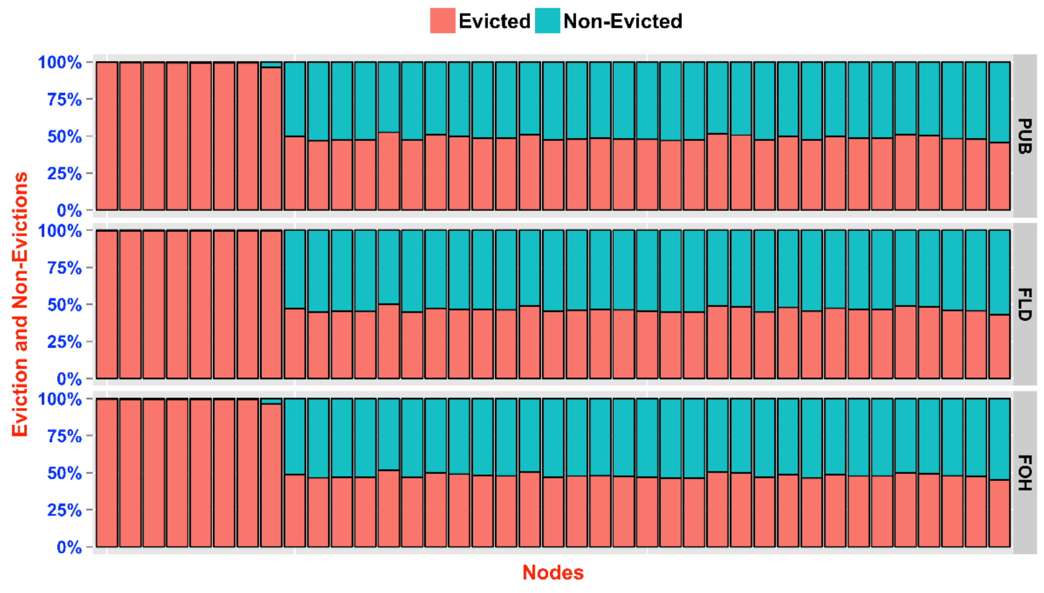


Figure 4-6. Nearness Percent Differential for: (a) Cache-Unfriendly Workload (*top*); (b) Cache-Friendly Workload (*bottom*)



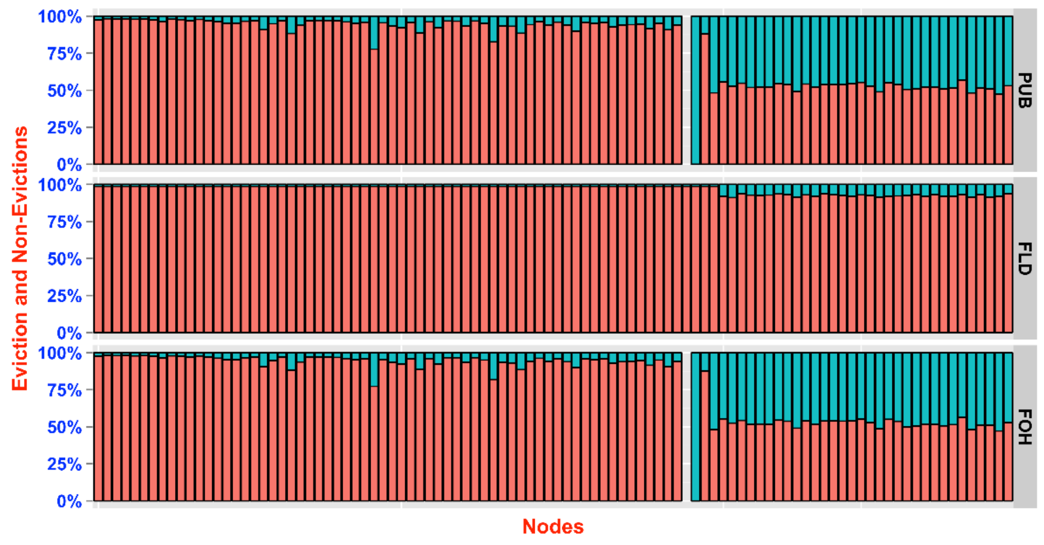


Figure 4-7. Eviction and Non-Evictions in Caches of: (a) 39-node with cache configuration c1p and Cache-Unfriendly Workload (*top*); (b) 100-node with cache configuration c1p and Cache-Unfriendly Workload (*bottom)*

If we focus on CCN-Flooding in particular, it is clear that it is the more successful strategy for requests satisfied at caches in the 39-node network, under both workloads. In contrast, it is the worst for 100-node. This contrast in behavior between 39-node and 100-node for CCN-Flooding extends (as will be seen below) to the AvgRT metric as well. In the remainder of this subsection, we analyze this contrasting behavior of CCN-Flooding in order to gain a better understanding of the impact that network topology has on forwarding strategy performance.

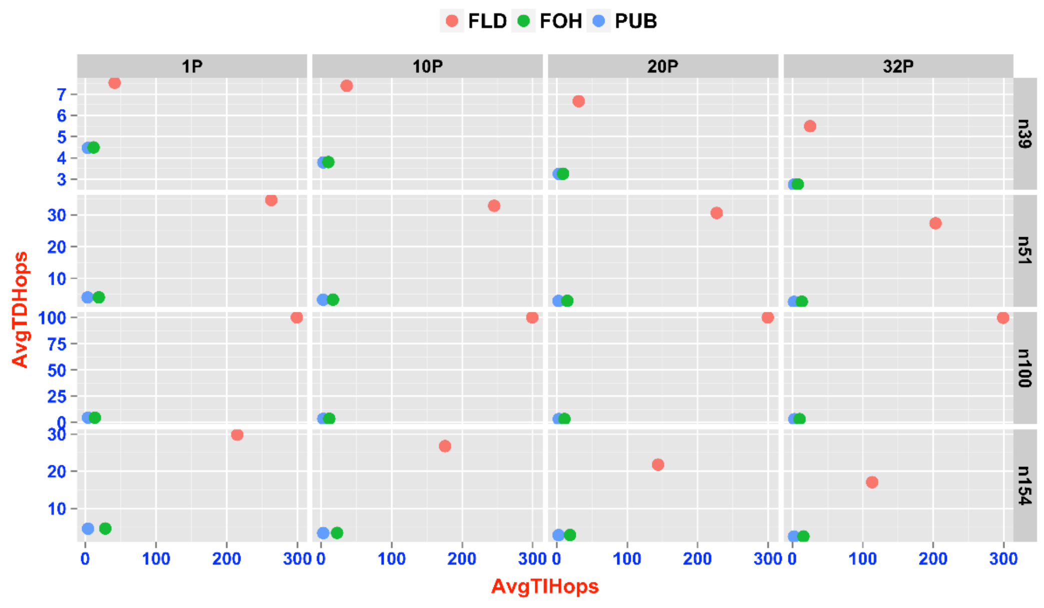
Figures 4-7(a) & (b) show the proportions of cache evictions to cache non-evictions, under CCN-Publication, CCN-FOH and CCN-Flooding, for the 39-node and 100-node networks, with cache-unfriendly workload. Each bar in Figure 4-7 represents the proportion of evictions to non-evictions at a particular node. The nodes are ordered by decreasing Betweenness Centrality. Thus, core nodes, of which there are 8 in 39-node and 68 in 100-node, are towards the left-hand side of the figures, followed by the edge nodes. It is evident that caches at core nodes, which are subjected to through traffic, have churn and are unstable for both networks and all three forwarding strategies. On the other hand, the edge nodes are much more stable, with the exception of 100-node under CCN-Flooding where we have cache churn at these nodes as well.

Edge nodes in the 100-node network all have degree two or more. Thus, with CCN-Flooding in particular, when requests are flooded and replies are generated, data packets may use edge nodes as intermediate nodes, similar to the way core nodes are utilized. Consequently, under CCN-Flooding, these edge nodes do not have stable caches as can be seen in Figure 4-7(b). In the case of 39-node, all edge nodes are of degree one. Edge nodes are thus not used as intermediate nodes, which causes them to have stable caches. For 51-node, 11 of the 22 edge nodes have degree more than one, the highest being ten. In 154-node, 108 of the 123 edge nodes have degree one, and the remaining 15 all have degree two. Concomitantly, the performance of CCN-Flooding improves monotonically as we move from 100-node to 51-node to 154-node, and finally to 39-node (Figure 4-4 above; and Figure 4-11 which is presented and discussed below), a proximate factor being the increasing stability at edge-node caches. Thus, the degree of edge nodes is an important characteristic that effects overall performance. The cache-unfriendly workload highlights this, and is consistent with what can also be observed under the cache-friendly workload.

The second topological factor that effects CCN-Flooding is Centralized Betweenness Centrality (cBC). Consider, on the one hand, a network with a very high cBC value, such as a linear network, and, on the other hand, a well-connected network which will have a low cBC value. The effect of flooding a request in these two networks is quite different. Due to the operation of the PIT-suppression mechanism in a topology in which shortest paths overlap, flooding in the linear network would be similar to "constrained" flooding, returning a strictly limited number of duplicate data packets. In the well-connected network, potentially more duplicate data copies from various locations with non-overlapping shortest paths could be returned for a given request, which also creates unnecessary traffic in the network. More to the point, these duplicate copies would make the caches unstable due to evictions. 39-node and 154-node have higher cBC values than 51-node and 100-node (0.41 and 0.42, vs. 0.29 and 0.10, respectively). Thus, 39-node and 154-node incline more towards a "constrained" flooding dynamic, especially 39-node which has a core connected in a ring-like topology (Figure 4-1(a)). 100-node has a very low cBC, and consequently flooding has a severe impact in terms of bandwidth usage and cache churn. 51-node is effected more than 39-node and 154-node are, but is better than 100-node.

In conclusion, in terms of topological characteristics, the degree of edge nodes and the cBC play a vital role. CCN-Flooding, an aggressive strategy, performs well when used in 39-node because the network's properties with respect to these two factors are such that flooding is relatively constrained. In contrast, flooding is detrimental in the 100-node network. Thus, paradoxically, the success of an aggressive strategy like CCN-Flooding is not due to the pervasiveness with which it searches for content, but rather to the degree to which this pervasiveness happens to be constrained by network topological characteristics. This is where CCN-FOH works well. It seeks to locate content in a more pervasive manner than does CCN-Publication, but uses an approach that is far more constrained than CCN-Flooding's aggressive pervasiveness. As such, it reaps all the benefit that CCN-Flooding is able to when it operates under topologically constraining conditions (*e.g.*, 39-node), while averting the detrimental effects when such constraining conditions do not apply (*e.g.*, 100-node). As will be seen below in Figure 4-11, the performance of CCN-FOH in 39-node and 154-node is comparable to, if not sometimes better than, that of CCN-Flooding in terms of AvgRT, but with a much lower bandwidth-consumption footprint (see Figure 4-8 below). In the case of the 100-node and 51-node networks, CCN-FOH performs better than CCN-Publication, and outclasses CCN-Flooding in particular, on all counts.

4.4.3 Comparison of Bandwidth Costs



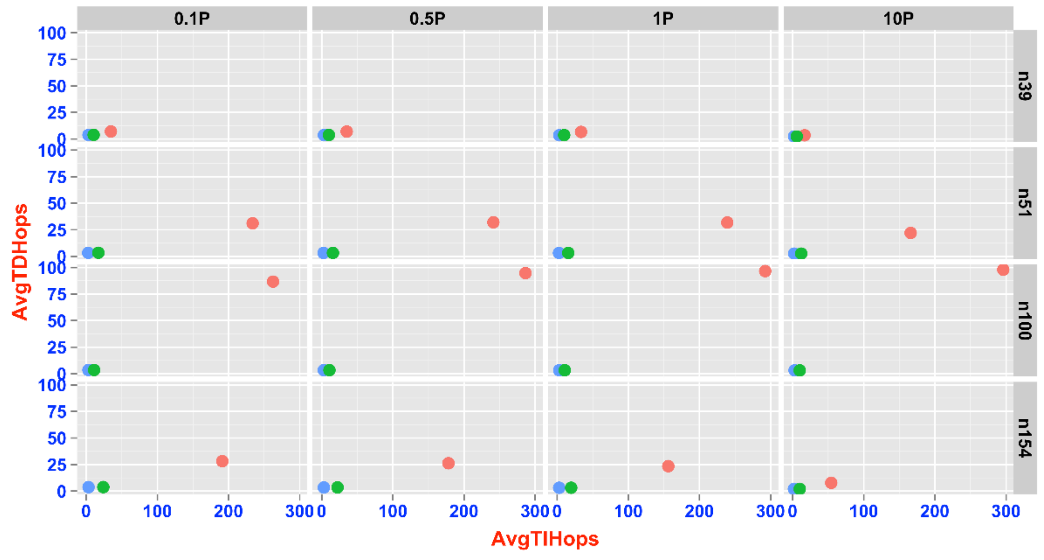


Figure 4-8. Bandwidth costs of CCN-Flooding (FLD - red), CCN-FOH (FOH - green), and CCN-Publication (PUB - blue) for: (a) Cache-Unfriendly Workload (*top*); (b) Cache-Friendly Workload (*bottom)*

Figures 4-8(a) & (b) show the comparison of bandwidth costs for CCN-Publication, CCN-Flooding, and CCN-FOH. As explained in Section 4.1, the bandwidth costs are measured in terms of AvgTIHops/Req (*x-axis*) for interest packets and AvgTDHops/Req (*y-axis*) for data packets. It is abundantly clear from the figures that CCN-Flooding costs are very high, in particular with respect to the bandwidth costs incurred by the interest packets. A similar situation was observed in Chapter 3 (Subsection 3.3.4). Here, however, it is more severe because this version of CCN-Flooding uses flooding for each and every request. This aggressive approach of CCN-Flooding also influences AvgTDHops/Req because it increases the potential for retrieving multiple data copies, which leads to higher AvgTDHops/Req compared to the other strategies. On the other hand, CC-FOH does not suffer from such heavy costs. When compared to CCN-Publication, CCN-FOH being the more aggressive of the two, incurs a higher cost in terms of the AvgTIHops/Req, but this cost is far less than that incurred by CCN-Flooding. With respect to AvgTDHops/Reqs, however, it performs at par with CCN-Publication since CCN-FOH retrieves a single copy of the content, suppressing additional, duplicate copies[11].

4.4.4 Comparisons of Strategies from the Perspective of User Experience

We now shift focus and compare the performance of CCN-FOH, first with CCN-Publication, then with CCN-Flooding, from the key perspective of user experience (primarily AvgRT/Req), and examine factors that influence that experience.

CCN-FOH *vs.* CCN-Publication. CCN-FOH is not only successful in locating content closer (Figures 4-6(a) & (b) above), as discussed in Subsection 4.4.2 above, but also does so with a better response time. Overall, improvements in relative

[11] This point will be dealt with explicitly towards the end of Subsection 4.4.4 which follows.

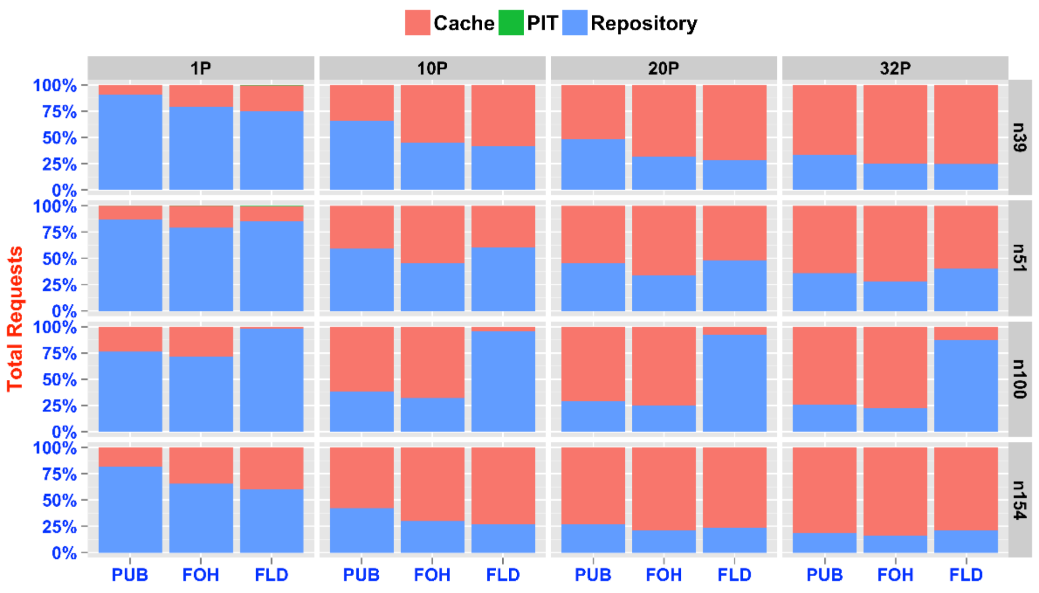
AvgRT/Req are achieved for all four topologies, under both workloads, as can be seen in Table VII. In order to better understand these improvements, we further explore the effect of caching in CCN-FOH.

Table VII: AvgRT/Req Percent Differential between CCN-Publication (100%) and CCN-FOH

Topologies	**Cache-unfriendly (α=0.7)**				**Cache-friendly (α=1.5)**			
	1P	**10P**	**20P**	**32P**	**0.1P**	**0.5P**	**1P**	**10P**
39-node	92.2%	84.6%	85.6%	88.9%	79.6%	84.5%	85.2%	97.5%
51-node	95%	90%	90.5%	92.5%	84.9%	90.7%	91.4%	96.6%
100-node	96.5%	93%	91.9%	92.8%	91.1%	91.4%	92.4%	93.9%
154-node	88.5%	86.8%	90.4%	92.7%	79.1%	83.2%	85.4%	97%

First, note that cache hits may occur in one of two ways: (a) requests that are satisfied at the requesting node itself ("satisfied inside"); and (b) requests satisfied outside the requesting node ("satisfied outside"). The impact that a given strategy has on requests "satisfied inside" is relatively minor. Cache content at a requesting node is, by and large, a function of: (1) the pattern of requests issued by the node, and (2) the caching policy, both of which are independent of forwarding strategy[12]. Therefore, we focus on requests "satisfied outside", for which cache hits are influenced by the forwarding strategies.

[12] This assertion requires some qualification. To the extent that a given strategy (e.g., CCN-Flooding) causes the node's cache to be examined by requests from other edge nodes, cache content *is* impacted by strategy. However, our data shows the effect of this to be relatively minor.



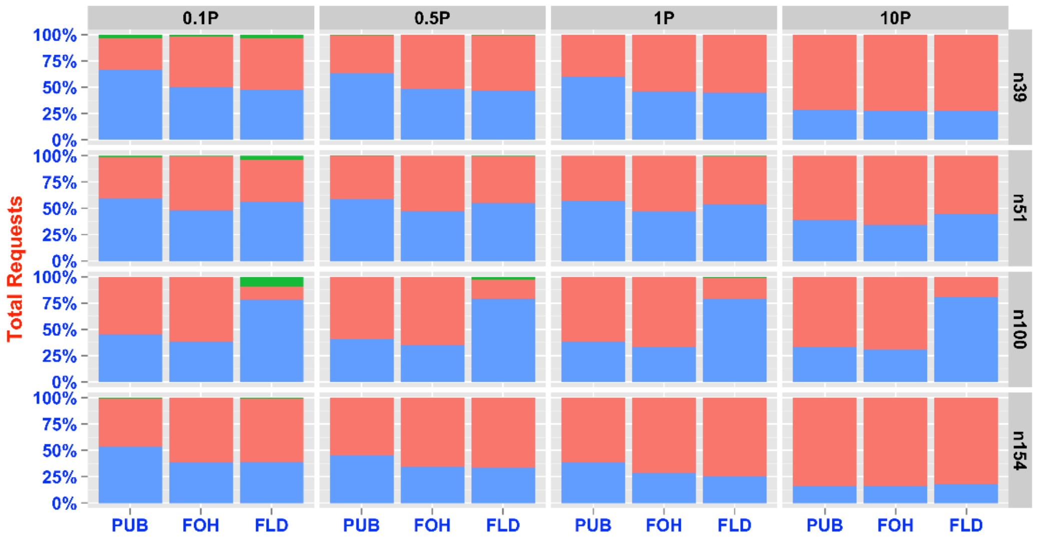


Figure 4-9. Requests satisfied outside the requesting node between Caches, PIT, and Repository for: (a) Cache-Unfriendly Workload (*top*); (b) Cache-Friendly Workload (*bottom*)

Figure 4-9(a) & (b) show the proportion of requests satisfied by cache hits *vs*. repositories *vs*. PIT. These plots are similar to Figures 4-5(a) & (b), the difference being that here we are dealing only with the requests that are satisfied outside. However, the conclusion to be drawn from both Figures 4-5 and 4-9 is the same: CCN-FOH leverages the caches more effectively compared to CCN-Publication. Focusing on the subset of outside requests that are satisfied from caches in Figures 4-9, we next examine their distribution between off-path and on-path caches under CCN-FOH.

In Figures 4-10(a) & (b), we observe that CCN-FOH leverages off-path caching quite effectively, thereby achieving one of its overall design objectives. It is important to highlight, however, that this does not necessarily imply that it achieves more cache hits overall in each and every instance than does CCN-Publication. For example, consider the 51-node and 100-node networks with cache-unfriendly workload and c1p cache configuration. Here, CCN-FOH retrieves 57.5% and 37.8% of the content from off-path caches, respectively (Figure 4-10(a)). However, the differences between CCN-Publication and CCN-FOH for the overall number of requests satisfied by cache hits are not substantial (Figure 4-9(a)), nor are the AvgRT differentials (Table VII). Nevertheless, these contrary scenarios notwithstanding, off-path caching in CCN-FOH plays a significant role overall in utilizing caches more effectively, thereby yielding more cache hits at achieving better Nearness and improved AvgRT/Req than CCN-Publication. It is worth noting that our data shows that this is accomplished at low cost in terms of bandwidth consumed by data packets, comparable to, if at times not better than, the cost entailed by CCN-Publication. In summary, CCN-FOH is quite effective in enhancing user experience (AvgRT/Req) overall, by leveraging cache content at reasonable cost. On the one hand, it does this by being more aggressively pervasive than CCN-Publication. On the other, it ensures that this pervasiveness does not overreach, remaining sufficiently restrained so as not to entail the exorbitant bandwidth costs, and consequent detrimental effect on user experience, of CCN-Flooding.

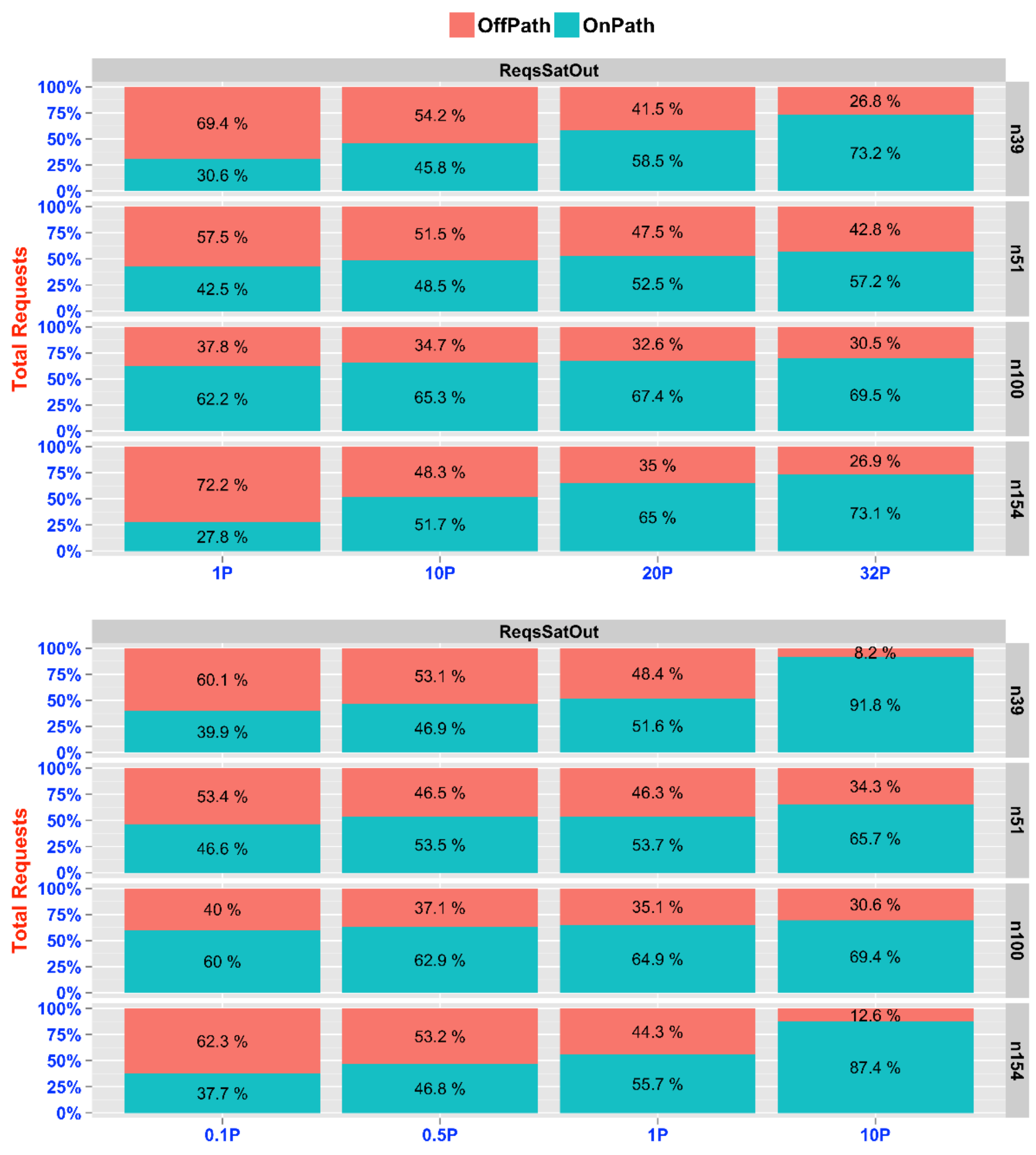


Figure 4-10. Proportion of Requests satisfied between On-Path and Off-Path Caches under CCN-FOH with: (a) Cache-Unfriendly Workload (*top*); (b) Cache-Friendly Workload (*bottom*)

Finally, as an interesting aside on the relative dynamics of CCN-FOH and CCN-Publication performance, note from Figures 4-10 that, as cache sizes increase from under-provisioned to over-provisioned, the majority of requests satisfied through cache hits shifts from off-path to on-path caches in CCN-FOH. The 100-node network is an exception to this: the majority of cache hits are on-path for all cache sizes. A distinctive trend can be seen for all four networks, where the proportion of on-path cache hits increases with increasing cache size. As cache sizes get larger, a forwarding strategy gains increasing benefit from on-path caching. For CCN-Publication, this in effect causes its performance to “catch up” with that of CCN-FOH. This can be seen in Figures 4-9, in which CCN-Publication’s level of cache hits improves with increasing cache size, thereby approaching those of CCN-FOH, as do its relative AvgRT differentials in Table VII.

CCN-FOH *vs*. CCN-Flooding. We now shift our attention to CCN-Flooding and demonstrate that, overall, it is not a viable contender to CCN-FOH from the perspective of user experience. We analyze the dynamics of CCN-Flooding to better understand how its behavior negatively impacts AvgRT/Req.

In Subsection 4.4.2, we established that both CCN-FOH and CCN-Flooding not only have a higher proportion of requests satisfied from caches, but are also efficient in locating content closer to the requesting node, compared to CCN-Publication. As can be seen in Figures 4-11(a) & (b), however, CCN-Flooding is not a serious candidate from the perspective of AvgRT/Req when compared to CCN-FOH. CCN-FOH performs equally well, if not better, particularly for under-provisioned caches. The sole exception to this is the case of the (over-provisioned) c10p cache-size configuration under cache-friendly workload. The performance of CCN-Flooding is mainly dependent on two factors: (a) the “isolation effect”; and (b) the effect of retrieving multiple duplicate copies of data objects (AvgTDHops/Req; Section 4.1 above). These two factors are a direct consequence of CCN-Flooding's operational mechanism. Their effect is particularly pronounced for configurations with under-provisioned cache-size and cache-friendly workloads, and is significantly mitigated with over-provisioned caches.

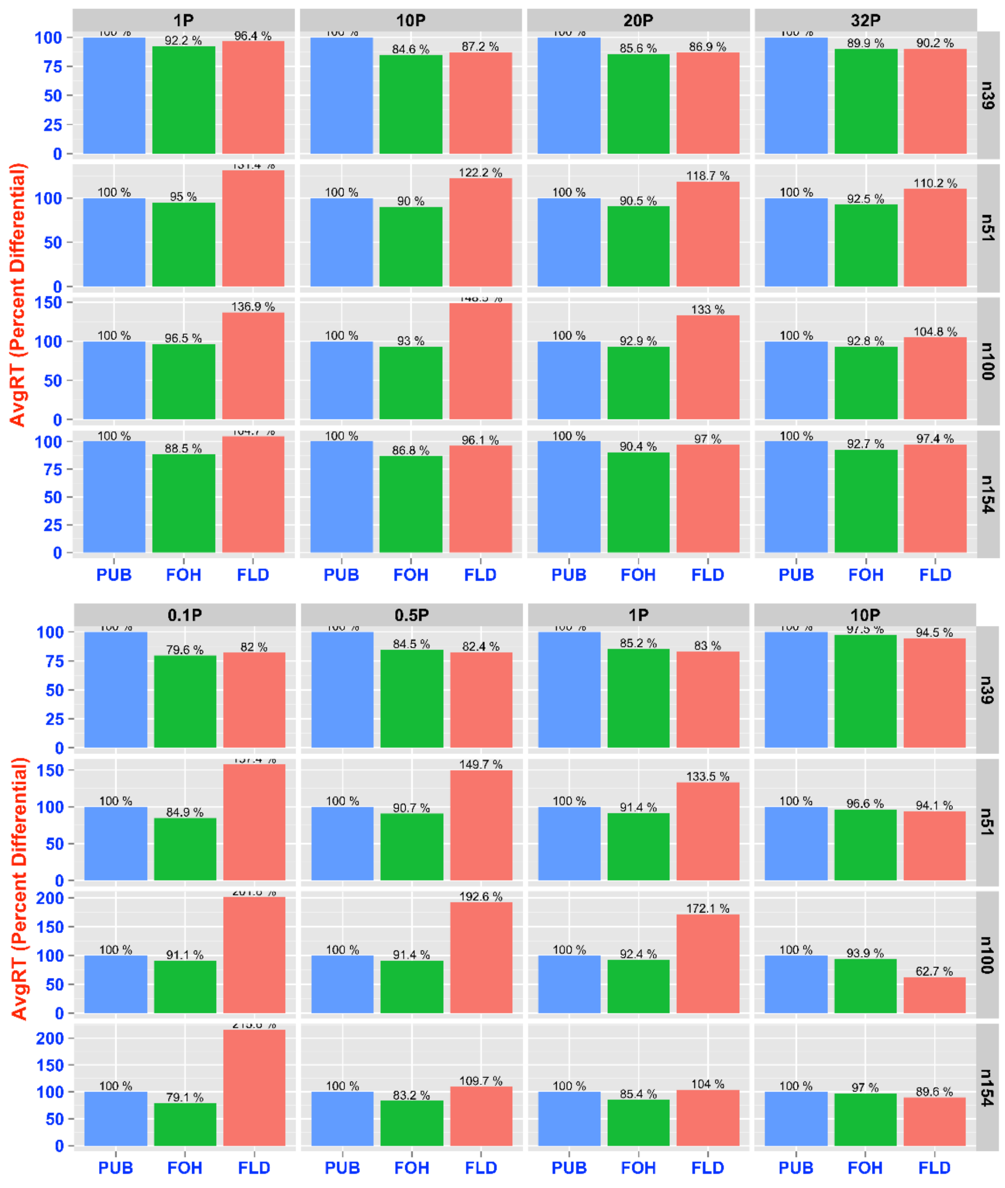


Figure 4-11. AvgRT Percent Differential across Topologies and Cache-Configurations for: (a) Cache-friendly workload (*top*); (b) Cache-unfriendly workload (*bottom*).

Flooded requests retransmitted due to the isolation effect, though constituting only a small proportion of the total number of requests issued, nevertheless have a pronounced negative effect on average Response Time (RT). This is reflected in the data by the fact there are many instances where the median RT for CCN-Flooding, and at times even the 75th percentile value, is significantly lower than the average RT. A particularly egregious example occurs in the 154-node network with cache-friendly workload and cache size c0.1p: average RT is 11.012 ms, while the 75th percentile RT value is 5.610 ms. Further examination of this particular simulation run reveals that one particular interest packet request, for the 7th most popular object, had the worst response time (11,248 ms) and was retransmitted 5 times. Another request, for the 3rd most popular object, took 9,064 ms to complete and was retransmitted 6 times. It is requests such as these that heavily skew the RT distribution to the right and "distort" its mean.

The isolation effect is strongly mitigated when we have over-provisioned caches and a cache-friendly workload because most requests for popular objects are then satisfied at the requesting nodes' own caches. Requests injected into the network core therefore tend to be for less popular objects. Thus, the number of multiple requests for the same objects simultaneously present in the network, and the chances of consequent PIT suppression, are low. With a cache-unfriendly workload, the isolation effect is mitigated for even smaller cache sizes since the popularity profile of the data objects is such that requests simultaneously present in the network are inherently more evenly spread across the object space.

The second factor that has an adverse effect on AvgRT/Req under CCN-Flooding is the retrieval of multiple duplicate copies of objects. This phenomenon is a consequence of CCN-Flooding's aggressively pervasive approach to locating content in caches. While this aggressiveness plays a role in improving Nearness, it does so at heavy bandwidth cost, as discussed in Subsection 4.4.3 above, which negatively impacts AvgRT/Req, especially in bandwidth-constrained networks, due to the congestion it causes in the network. The effect is more prominent with requests for the more popular objects, copies of which would be present at a larger number of node caches.

CCN-FOH does not suffer from either of these two issues. With respect to bandwidth consumed by the data packets, CCN-FOH performance is at par with CCN-Publication, and even slightly better under certain scenarios. Though we retrieve objects from off-path caches, however, unlike CCN-Flooding, these data packets consume the PIT entries along the return shortest path. In the case of multiple cache hits, the data packet generated by the closest copy will consume all such PIT entries. Duplicate data packets (generated further upstream) will therefore be suppressed at the first intermediate shortest-path node at which the closest-copy data packet had already consumed the PIT entry. Thus, although multiple data copies are generated, they are soon suppressed and do not significantly impact bandwidth cost. Furthermore, CCN-FOH does not induce an isolation-effect in the manner of CCN-Flooding because it does not create a PIT entry at an off-path node if none is already present. Such a PIT entry, if present, would have been created by some other interest packet for which the node is on its shortest path. In this case, the off-path interest packet simply extends this pre-existing PIT entry – which does not cause an isolation effect – so as to benefit from retrieved data that flows back through the node.

The performance of CCN-Flooding might be quite competitive if it were not for the isolation effect, especially in a network with ample bandwidth that would not suffer undue congestion due to the retrieval of multiple duplicate data copies. This argues for an adaptive approach to CCN-FOH, which dynamically adjusts along a range from one-hop through to all-pervasive flooding as a function of the popularity of the object sought. It is worth noting that the bandwidth cost would be higher for such an adaptive strategy, compared to CCN-FOH. Thus, an effective adaptive strategy would also have to dynamically adjust its behavior as a function of available network bandwidth if it is to avoid inducing RT degradation due to network congestion. We set these considerations aside for the present, and leave them for future work.

4.5 Variants of CCN-FOH

We investigated several non-adaptive forwarding strategy variants derived from CCN-FOH in a generally unsuccessful attempt to "fine tune" its performance for the better. These schemes differ from CCN-FOH in one of the following two ways: (a) the degree of aggressiveness in locating content in the network; and (b) the degree of aggressiveness in populating caches by the data located. The results were generally disappointing. Nevertheless, we briefly describe the strategy variants for the record.

In the first three variants, we flood interest packets for only one hop (as in CCN-FOH), but based upon probabilities generated according to geometric sequences that are functions of the number of hops an interest packet has traveled along the shortest path from the requesting node (however, we do not flood when we are only one hop away from the repository node). Specifically,

– **Linear-Probability (LP) scheme:** The probability for flooding the interest packet is determined using the following linear geometric sequence:

$$1, \quad \frac{1}{2}, \quad \frac{1}{4}, \quad \frac{1}{8} \quad \ldots$$

  Thus, we will always flood on the first hop. At each subsequent hop the probability will drop by a factor of 1/2.

- **Quadratic-Probability (QP) scheme:** This scheme is less aggressive as it uses a quadratic geometric sequence for the probability of flooding an interest packet:

  $$1, \quad \frac{1}{4}, \quad \frac{1}{9}, \quad \frac{1}{16} \quad \ldots$$

  Again, we will always flood on the first hop. At each subsequent hop the probability will drop by a factor of 1/(number of hops traveled)^2.

- **Reverse-Linear-Probability (RLP) scheme:** This scheme is the opposite of the Linear-Probability scheme. The probability for flooding interest packets is based on the following linear geometric sequence:

  $$\frac{1}{(numHopsToRepository - 1)}, \frac{1}{(numHopsToRepository - 2)}, \ldots\ 1$$

  We flood with higher probability the closer the interest packet is to the repository, on the theory that "neighborhoods" closer to the repository of a given object are more likely to have that object in the node caches.

In contrast to these three schemes, the probability of flooding under CCN-FOH is always 1. LP, QP and RLP are thus less aggressive in locating content compared to CCN-FOH. Their performance with respect to AvgRT/Req, Nearness, and efficient utilization of off-path caches was poor when compared to CCN-FOH.

The next two variants determine how data packets resulting from off-path cache hits by CCN-FOH populate the caches on the return shortest path back to the requesting node (CCN-FOH and the three variants above always populate these caches). The idea was to investigate whether reducing churn in the in-network node caches would improve performance.

– **Off-Path Disabled (OPD) scheme:** In this scheme, data packets resulting from off-path cache hits do not populate caches on the return path to the requesting node (including the cache of the requesting node itself).

– **Off-Path Probability (OPP) scheme:** Data packets resulting from an off-path cache hit populate caches on return path with 50% probability at each hop. This also applies to the cache of the requesting node.

In the OPD scheme, data packets generated at off-path nodes do not populate caches along the return path. Since CCN-FOH always populates these caches, OPD and CCN-FOH constitute opposite ends of a spectrum in this respect. In contrast, in the OPP scheme, the caches are populated based on a 50%-probability Bernoulli trial. The OPP scheme, therefore, lies intermediately along the spectrum between OPD and CCN-FOH. Overall, taking into account all requests, including the ones satisfied at the requesting node itself, we observed that the performance differences between OPD, OPP and CCN-FOH were very minor. CCN-FOH's performance is virtually never worse than that of OPD and OPP. When we consider only the subset of requests that are satisfied outside the requesting nodes' own caches, the situation is slightly different, particularly in the case of under-provisioned caches. As previously noted, such caches are prone to churn. Since both the OPD and OPP schemes are less aggressive in populating caches, they tend to mitigate churn somewhat. This, in turn, leads to slightly better performance in terms of AvgRT/Req and Nearness as compared to CCN-FOH. However, their overall performance across all requests is not better than that of CCN-FOH.

The last variant of CCN-FOH that we explored expands the extent of the flooding:

– ***N*-Hop Neighborhood scheme:** We extend flooding from a neighborhood of fixed radius 1-hop to a neighborhood of radius (*N*-1) hops around each node along the shortest path, where *N* is the residual distance from that node to the repository. This scheme is comparable to an unabated CCN-Flooding in which there is no PIT suppression.

The scheme was implemented for completeness' sake rather than in the expectation that it would yield any performance improvement. As anticipated, it performed considerably worse than CCN-FOH.

Chapter 5

Conclusion and Future Work

The goal of the ICN architecture approach is to provide support for a broad range of robust network services that are resilient to disruption and failure. These services include in-network caching, inbuilt security, mobility, multiparty communication via replication, and so on. In-network caching is correlative to a forwarding strategy which leverages the presence of this cached content in the network. Our work deals with forwarding strategy in, specifically, the CCN ICN architecture.

Research in the field of in-network caching and forwarding strategy has tended to take one of two distinct broad approaches. In the first approach, which came as a natural outgrowth of research into Content Distribution Networks, the primary focus is on optimizing content placement in the in-network caches so that caches are effectively utilized. Forwarding strategy then follows as a consequence of cache content placement and distribution, aimed at leveraging the latter. The second approach takes the design of the forwarding strategy as its starting point and attempts to enhance the delivery of content to the user, leveraging in-network cache content along the way as best it can. In this approach, the placement and distribution of cache content becomes a consequence of the way the forwarding strategy operates: caches are populated with content retrieved by the forwarding strategy as that content weaves its way back to the requesting node. Our work in this dissertation falls within this second approach.

From early on, two basic forwarding strategies were proposed for CCN: CCN-Flooding came first, followed shortly by CCN-Publication. These two basic strategies can be viewed as opposite ends of a spectrum from the perspective of how aggressively a forwarding strategy in CCN might search for network cached content. CCN-Publication takes a minimalist approach. Essentially, it is similar to IP forwarding, but with in-network caching along the path. No attempt is made to explore for network cached content. Such content is made use of only if it happens to be encountered in the caches of intermediate nodes along the shortest path to the repository. It therefore has low bandwidth costs but is not very effective in provisioning and leveraging cached content. CCN-Flooding, on the other hand, both carries out an exhaustive search for cached and repository content, and ubiquitously populates the in-network caches with the content it finds. It therefore provisions and locates cached content in a pervasive manner, but at high bandwidth cost.

In Chapter 3 we investigated the performance dynamics of CCN-Publication and CCN-Flooding, especially from the perspective of bandwidth consumption. Based on the insights gained and presented there, we proposed and analyzed a new lightweight forwarding strategy, CCN-FOH, in Chapter 4. CCN-FOH is based on two design principles: (a) enhancing user experience by more effectively exploring for in-network cached content, as compared to CCN-Publication; and (b) avoiding the costs that induce a deterioration in user experience which an overly aggressive approach to cache exploration is prone to, as exemplified by CCN-Flooding. We demonstrated that CCN-FOH is quite effective in realizing its design goals and delivering an enhanced user experience across a variety of differing network topologies and workload characteristics. As such, it constitutes a better choice for a (non-adaptive) forwarding strategy than both CCN-Publication and CCN-Flooding.

CCN-FOH leverages off-path caching, which plays a significant role overall in utilizing caches more effectively, yielding more cache hits, which in turn improves locating content closer than CCN-Publication, and achieving an improved user experience. It does so without entailing significant additional bandwidth costs. This is because in both CCN-Publication and CCN-FOH the requesting node does not receive duplicate copies of the object requested. CCN-FOH, therefore, does not incur significant additional and potentially damaging costs as is the case with CCN-Flooding, where multiple duplicate copies of the requested object are not suppressed in the network and are delivered to the requesting node. In CCN-Flooding, two main factors, which are both avoided by CCN-FOH, influence user experience, namely: high bandwidth cost and the isolation effect. These two factors combined play a significant role in negatively impacting the user experience. The bandwidth costs entailed by CCN-Flooding's aggressively pervasive approach to locating content are very high, which leads to network congestion. The isolation effect unnecessarily delays the servicing of user requests.

The analysis of performance dynamics carried out in Chapters 3 and 4 demonstrates that the success of a forwarding strategy, and the appropriate degree of pervasiveness with which it should seek in-network cached content, is a function of several factors that mutually effect and impinge on one another in complex ways: network topology, workload characteristics, and cache size. By the nature of things, a forwarding strategy attempts to service only those requests that fail to be satisfied by a cache hit at the requesting node's own cache, and are therefore injected into the network. Since edge nodes tend to pull the most popular objects into their caches, requests that are dealt with by the forwarding strategy will overwhelmingly tend to be for other than these most popular objects. For object requests that are injected into the network, their precise "standing" in the overall popularity profile of the workload is governed by a complicated interplay between: the object popularity characteristics of the workload itself; the degree to which cache sizes constitute a situation of relative under- or over-provisioning (this itself is also a function of the workload popularity profile and not just of the cache size); and the nature of the network topology with respect to whether edge nodes handle through traffic, which induces more churn in their caches, or not. Requests injected into the network therefore can run the gamut from relatively highly popular objects (but not necessarily the most popular), to rarely requested and quite unpopular objects. The degree of pervasiveness that a forwarding strategy seeking to enhance user experience should exercise in satisfying a requested object from in-network caches cannot be determined without taking these considerations into account. Expending effort to pervasively seek an unpopular and rarely requested object, for example, would probably not pay off – it would be more efficient simply to fetch it from its repository. By the same token, expending some appropriate amount of effort for a more popular object is more likely to lead to a payoff. The situation is further complicated by the fact that an unduly pervasive attempt at seeking in-network content could be benignly constrained by the nature of the network topology itself. This strongly argues for the development of forwarding strategies that take these and similar considerations into account in a dynamically adaptive manner, and defines an exciting and challenging agenda for our future work.

Bibliography

1. Clark, D., *The design philosophy of the DARPA internet protocols.* SIGCOMM Comput. Commun. Rev., 1988. **18**(4): p. 106-114.

2. Cerf, V. and R.E. Kahn, *A Protocol for Packet Network Intercommunication.* Communications, IEEE Transactions on, 1974. **22**(5): p. 637-648.

3. Dovrolis, C. and J.T. Streelman, *Evolvable network architectures: what can we learn from biology?* SIGCOMM Comput. Commun. Rev., 2010. **40**(2): p. 72-77.

4. Dierks, T. and C. Allen, *The TLS Protocol.* RFC 2246, 1999.

5. Blake, S., et al., *An Architecture for Differentiated Services.* RFC 2475, 1998.

6. Clark, D.D., S. Shenker, and L. Zhang, *Supporting real-time applications in an Integrated Services Packet Network: architecture and mechanism.* SIGCOMM Comput. Commun. Rev., 1992. **22**(4): p. 14-26.

7. Saroiu, S., et al., *An analysis of Internet content delivery systems.* SIGOPS Oper. Syst. Rev., 2002. **36**(SI): p. 315-327.

8. Su, A.-J., et al., *Drafting behind Akamai: inferring network conditions based on CDN redirections.* IEEE/ACM Trans. Netw., 2009. **17**(6): p. 1752-1765.

9. Srisuresh, P. and M. Holdrege, *IP Network Address Translator (NAT) Terminology and Considerations.* RFC 2663, 1999.

10. Patil, B., P. Roberts, and C. Perkins, *IP Mobility Support for IPv4.* RFC 3344, 2002.

11. Feldmann, A., *Internet Clean Slate*, 2008.

12. Dovrolis, C., *What would Darwin think about clean-slate architectures?* SIGCOMM Comput. Commun. Rev., 2008. **38**(1): p. 29-34.

13. Rexford, J. and C. Dovrolis, *Future Internet architecture: clean-slate versus evolutionary research.* Commun. ACM, 2010. **53**(9): p. 36-40.

14. Feldmann, A., *Internet clean-slate design: what and why?* SIGCOMM Comput. Commun. Rev., 2007. **37**(3): p. 59-64.

15. Roberts, J., *The clean-slate approach to future Internet design: a survey of research initiatives.* Annals of Telecommunications, 2009. **64**(5): p. 271-276.

16. Ahlgren, B., et al., *A survey of information-centric networking.* Communications Magazine, IEEE, 2012. **50**(7): p. 26-36.

17. Trossen, D., M. Sarela, and K. Sollins, *Arguments for an information-centric internetworking architecture.* SIGCOMM Comput. Commun. Rev., 2010. **40**(2): p. 26-33.

18. *Information-Centric Networking Research Group (ICNRG)*. 2012; Available from: http://tools.ietf.org/group/irtf/trac/wiki/icnrg.

19. Xylomenos, G., et al., *A Survey of Information-Centric Networking Research.* Communications Surveys & Tutorials, IEEE, 2014. **16**(2): p. 1024-1049.

20. Ghodsi, A., et al., *Information-centric networking: seeing the forest for the trees*, in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*2011, ACM: Cambridge, Massachusetts. p. 1-6.

21. Cheriton, D., et al. *Translating Relaying Internet Architecture integrating Active Directories*. 2001; Available from: http://www-dsg.stanford.edu/triad/.

22. Jacobson, V., et al., *Networking named content*, in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*2009, ACM: Rome, Italy. p. 1-12.

23. Koponen, T., et al., *A data-oriented (and beyond) network architecture.* Computer Communication Review, 2007. **37**(4): p. 181-192.

24. Nikos Fotiou, D.T., George C.Polyzos, *Illustrating a Publish-Subscribe Internet Architecture*, 2010. p. 22.

25. Bengt Ahlgren, M.D.A., Christian Dannewitz, Anders Eriksson, Jovan Golić, Björn Grönvall, Daniel Horne, Anders Lindgren, Olli Mämmelä, Marco Marchisio, Jukka Mäkelä, Septimiu Nechifor, Börje Ohlman, Kostas Pentikousis, Sabine Randriamasy, Teemu Rautio, Eric Renault, Pasi Seittenranta, Ove Strandberg, Bogdan Tarnauca, Vinicio Vercellone, Djamal Zeghlache, *D-6.2 Second NetInf Architecture Description*, 2010.

26. Bari, M.F., et al., *A survey of naming and routing in information-centric networks.* Communications Magazine, IEEE, 2012. **50**(12): p. 44-53.

27. Ghodsi, A., et al., *Naming in content-oriented architectures*, in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*2011, ACM: Toronto, Ontario, Canada. p. 1-6.

28. Mark Ain, S.T., Dirk Trossen, Pekka Nikander, Trevor Burbridge, András Zahemszky, Jarno Rajahalme, Dmitrij Lagutin, Mikko Särelä, Janne

Riihijärvi, Teemu Rinta-aho, *Conceptual Architecture of PSIRP Including Subcomponent Descriptions*, 2008. p. 85.

29. D'Ambrosio, M., et al., *MDHT: a hierarchical name resolution service for information-centric networks*, in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*2011, ACM: Toronto, Ontario, Canada. p. 7-12.

30. Pentikousis, K., *Distributed Information Object Resolution*, in *Proceedings of the 2009 Eighth International Conference on Networks*2009, IEEE Computer Society. p. 360-366.

31. Christian Esteve, F.L.V., Maurício F. Magalhães, *Towards a new generation of information-oriented internetworking architectures*, in *Proceedings of the 2008 ACM CoNEXT Conference*2008, ACM: Madrid, Spain. p. 1-6.

32. Kazi, A.W., *Prefetching Bloom filters to control flooding in content-centric networks*, in *Proceedings of the ACM CoNEXT Student Workshop*2010, ACM: Philadelphia, Pennsylvania. p. 1-2.

33. Zhang, G., Y. Li, and T. Lin, *Caching in information centric networking: A survey.* Comput. Netw., 2013. **57**(16): p. 3128-3141.

34. Tyson, G., et al. *A Trace-Driven Analysis of Caching in Content-Centric Networks*. in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*. 2012.

35. Rossini, G. and D. Rossi. *A dive into the caching performance of Content Centric Networking*. in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2012 IEEE 17th International Workshop on*. 2012.

36. Ardelius, J., et al., *On the effects of caching in access aggregation networks*, in *Proceedings of the second edition of the ICN workshop on Information-centric networking*2012, ACM: Helsinki, Finland. p. 67-72.

37. Ong, M.D., et al., *FGPC: fine-grained popularity-based caching design for content centric networking*, in *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*2014, ACM: Montreal, QC, Canada. p. 295-302.

38. Han, B., et al., *PPP: prefix-based popularity prediction for effective caching in content-centric networking*, in *Proceedings of The Ninth International Conference on Future Internet Technologies*2014, ACM: Tokyo, Japan. p. 1-6.

39. Fayazbakhsh, S.K., et al., *Less pain, most of the gain: incrementally deployable ICN*, in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*2013, ACM: Hong Kong, China. p. 147-158.

40. Chai, W.K., et al., *Cache "less for more" in information-centric networks*, in *Proceedings of the 11th international IFIP TC 6 conference on Networking - Volume Part I*2012, Springer-Verlag: Prague, Czech Republic. p. 27-40.

41. Rossi, D. and G. Rossini. *On sizing CCN content stores by exploiting topological information*. in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. 2012.

42. Yonggong, W., et al. *Optimal cache allocation for Content-Centric Networking*. in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*. 2013.

43. Li, J., et al., *Popularity-driven coordinated caching in named data networking*, in *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*2012, ACM: Austin, Texas, USA. p. 15-26.

44. Kideok, C., et al. *WAVE: Popularity-based and collaborative in-network caching for content-oriented networks*. in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. 2012.

45. Wang, J.M., J. Zhang, and B. Bensaou, *Intra-AS cooperative caching for content-centric networks*, in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*2013, ACM: Hong Kong, China. p. 61-66.

46. Barakat, C., et al. *Minimizing bandwidth on peering links with deflection in named data networking*. in *Communications and Information Technology (ICCIT), 2013 Third International Conference on*. 2013.

47. Saino, L., I. Psaras, and G. Pavlou, *Hash-routing schemes for information centric networking*, in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*2013, ACM: Hong Kong, China. p. 27-32.

48. Chiocchetti, R., et al., *Exploit the known or explore the unknown?: hamlet-like doubts in ICN*, in *Proceedings of the second edition of the ICN workshop on Information-centric networking*2012, ACM: Helsinki, Finland. p. 7-12.

49. Chiocchetti, R., et al., *INFORM: a dynamic interest forwarding mechanism for information centric networking*, in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*2013, ACM: Hong Kong, China. p. 9-14.

50. Rossini, G. and D. Rossi, *Evaluating CCN multi-path interest forwarding strategies.* Computer Communications, 2013. **36**(7): p. 771-778.

51. Yi, C., et al., *A case for stateful forwarding plane.* Computer Communications, 2013. **36**(7): p. 779-791.

52. Yi, C., et al., *Adaptive forwarding in named data networking.* SIGCOMM Comput. Commun. Rev., 2012. **42**(3): p. 62-67.

53. *JavaSim*. 1997; Available from: http://javasim.codehaus.org/.

54. Medina, A., I. Matta, and J. Byers, *BRITE: A Flexible Generator of Internet Topologies*, 2000, Boston University.

55. Spring, N., et al., *Measuring ISP topologies with rocketfuel.* IEEE/ACM Trans. Netw., 2004. **12**(1): p. 2-16.

56. Newman, M.E.J., *Networks : an introduction*. 2010, Oxford: Oxford University Press. xi, 772 p., [4] p. of plates.

57. Freeman, L.C., *Centrality in social networks conceptual clarification.* Social Networks, 1978. **1**(3): p. 215-239.

58. Katsaros, K.V., G. Xylomenos, and G.C. Polyzos. *GlobeTraff: A Traffic Workload Generator for the Performance Evaluation of Future Internet Architectures*. in *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*. 2012.

59. Busari, M. and C. Williamson, *ProWGen: a synthetic workload generation tool for simulation evaluation of web proxy caches.* Comput. Netw., 2002. **38**(6): p. 779-794.

60. Mahanti, A., C. Williamson, and D. Eager, *Traffic analysis of a Web proxy caching hierarchy*. Network, IEEE, 2000. **14**(3): p. 16-23.

61. Breslau, L., et al. *Web caching and Zipf-like distributions: evidence and implications*. in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. 1999.

62. Xu, C., C. Dale, and L. Jiangchuan. *Statistics and Social Network of YouTube Videos*. in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*. 2008.

63. Fricker, C., et al. *Impact of traffic mix on caching performance in a content-centric network*. in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. 2012.

64. Gill, P., et al., *Youtube traffic characterization: a view from the edge*, in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*2007, ACM: San Diego, California, USA. p. 15-28.

65. Carofiglio, G., et al. *Modeling data transfer in content-centric networking*. in *Teletraffic Congress (ITC), 2011 23rd International*. 2011.

66. Katsaros, K., G. Xylomenos, and G.C. Polyzos, *MultiCache: An overlay architecture for information-centric networking.* Comput. Netw., 2011. **55**(4): p. 936-947.

67. Rossini, G. and D. Rossi, *Caching performance of content centric networks under multi-path routing (and more)*, 2011, Telecom ParisTech.

68. Afanasyev, A., I. Moiseenko, and L. Zhang, *ndnSIM: NDN simulator for NS-3.* Technical Report, NDN-0005, 2012.