ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
School of Information Sciences and Technology
Department of Computer Science

Multicast Forwarding in Future Information-Centric Network Architectures

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Christos Tsilopoulos

Athens
March 2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Completing a Ph.D. dissertation is a long and exciting journey. It often pushes you to your limits, and that can get you exhausted. It is almost impossible to complete a Ph.D. without the valuable support of advising teachers, colleagues, friends and family. This dissertation is no exception to this rule.

First and foremost, I acknowledge the members of my advisory committee: Associate Professor George Xylomenos, Professor George C. Polyzos and Associate Professor Vasilios A. Siris. George Xylomenos, with whom I worked closely, has been a valuable teacher. His mentoring allowed me to improve analytical thinking and – most importantly – he helped me cultivate initiative taking. After completing the dissertation, I consider the latter a skill of the highest value. I am grateful to George C. Polyzos, for he has showed me the *big picture* and taught me the importance of thinking outside of the box. I acknowledge Vasilios A. Siris for his persistence in *always asking the correct question and identifying the real problem.* He has showed me how to make my ideas concrete and comprehensive to others. I am thankful to all three of them for their advice and I feel that they have been true teachers to me.

I also acknowledge my colleagues, the Ph.D. students in the Mobile Multimedia Laboratory at Athens University of Economics and Business. Starting with seniors (at the time) Konstantinos V. Katsaros, who has been a mentor in my first year in graduate school and helped me in my first steps as a researcher, and Pantelis A. Frangoudis, with whom I had long and fruitful discussions on research projects. And then my phd-mates: Vaggelis Douros, Nikos Fotiou and Xenofon Vasilakos with whom we often had long discussions on research problems and politics. Charis Stais, both a friend and a colleague, with whom we played pool and co-developed prototype networking protocols. Yannis Thomas, who started as an M.Sc. student under my co-supervision, before starting on his Ph.D., with whom I have co-authored a few research papers. Finally, Dr. Merkouris Karaliopoulos, the latest addition to the lab, provided valuable senior-level advice at the last stages of this dissertation.

I am grateful to my friends for their love, support and, sometimes, the

patience they have showed with my temperament. They are a great company to hang out with and have fun. Above all, they have been constantly present in times of need. Without them, this journey would be colourless. Thank you George Papaioannou and Maria-Lida Menexis, Yannis Sazonof and Elena Tchekourova, Nikolas Marinakis and Staliana Koskoris.

I want to thank my family. My mother, Eirini Anezakis, and her husband, Lefteris Panou. As parents, they taught me how to pursue high goals, be virtuous and moderate. My grandmother, Antonia Kougioumtziarapis, and my aunt, Paraskevi Kougioumtziarapis, have been always on my side and supported this effort.

Last, but not least, I am thankful to Ms. Danai Argyrakopoulou. She has been a loving and caring partner throughout this journey. Her support seemed endless and her patience ever-lasting. Her contributions to this dissertation are multi-fold. Apart from a loving partner, Danai has been a friend and a great listener. In her efforts to be supportive, she was willing to act as a research-assistant and discuss subjects like multicast, Bloom filters and network architectures. In addition, she has been a thorough editor to my papers, paying attention to every detail. There have been times when she believed in me more than I did myself. There is so much of her in this dissertation.

– Thank you Danai. There are no words to express my gratitude. I cannot wish you anything less than the best.

VITA

2006	B. Sc. in Computer Science, Athens University of Economics and Business, Greece
2009	M. Sc. in Communication Systems and Networks, National and Kapodistrian University of Athens, Greece
2016	Ph. D. in Computer Science, Athens University of Economics and Business, Greece

PUBLICATIONS

Journal Publications

G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros and G. C. Polyzos, "A Survey of Information-Centric Networking Research," in *IEEE Communications Surveys and Tutorials*, vol. 16, no. 2, pp.1024-1039, 2014.

Refereed Conference and Workshop Papers

Y. Thomas, G. Xylomenos, C. Tsilopoulos and G. C. Polyzos, "Multi-Flow Congestion Control with Network Assistance," in *Proc. of IFIP Networking*, 2016.

Y. Thomas, G. Xylomenos, C. Tsilopoulos and G. C. Polyzos, "Object-oriented Packet Caching for ICN,", in *Proc. of ACM SIGCOMM International Conference on Information-Centric Networking (ICN)*, pp. 89-98, 2015.

Y. Thomas, C. Tsilopoulos , G. Xylomenos and G. C. Polyzos, "Accelerating File Downloads in Publish Subscribe Internetworking with Multisource and Multipath Transfers," in *Proc. of World Telecommunications Congress (WTC)*, pp. 1-6, 2014.

C. Tsilopoulos, Y. Thomas and G. Xylomenos, "Reducing Forwarding State in Content-Centric Networks with Semi-Stateless Forwarding," in *Proc. of IEEE INFOCOM*, pp. 2067-2075, 2014.

C. Tsilopoulos, G. Xylomenos and G. C. Polyzos, "Are Information-Centric Networks Video-Ready?," in *Proc. of Packet Video Workshop*, pp. 1-8, 2013.

C. Tsilopoulos, and G. Xylomenos, "Scaling Bloom filter-based multicast via filter switching," in *Proc. of IEEE ISCC*, pp. 548-554, 2013.

C. Stais, Y. Thomas, G. Xylomenos and C. Tsilopoulos, "Networked Music Performance over Information-Centric Networks," in *Proc. of IEEE ICC workshop on Immersive & Interactive Multimedia Communications over the Future Internet (IIMC)*, pp. 647-651, 2013.

C. Tsilopoulos, I. Gasparis G. Xylomenos and G. C. Polyzos, "Efficient Real-time Information Delivery in Future Internet Publish-Subscribe Networks," in *Proc. of IEEE International Conference on Computing, Networking and Communications (ICNC)*, pp. 856-860, 2013.

K. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos and G. C. Polyzos, "On Inter-domain Name Resolution for Information-Centric Networks," in *Proc. of IFIP Networking*, pp. 13-26, 2012.

C. Tsilopoulos, G. Xylomenos, "Supporting Diverse Traffic Types in Information-centric Networks," in *Proc. of ACM SIGCOMM workshop on Information-centric networking (ICN)*, pp. 13-18, 2011.

C. Tsilopoulos, D. Makris and G. Xylomenos, "Bootstrapping a Publish/Subscribe Information Centric Network," in *Proc. of ICT Future Networks and Mobile Summit*, pp. 1-8, 2011.

Refereed Magazine Articles

G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. A. Siris and G. C. Polyzos, "Caching and Mobility Support in a Publish-Subscribe Internet Architecture," in *IEEE Communications Magazine*, vol. 50, no. 7, pp. 52-58, 2012.

Refereed Posters and Demos

G. Xylomenos, C. Tsilopoulos, Y. Thomas and G. C. Polyzos, "Reduced Switching Delay for Networked Music Performance," in *Packet Video Workshop*, 2013.

Y. Thomas, C. Tsilopoulos, G. Xylomenos and G. C. Polyzos, "Multisource and Multipath File Transfers through Publish-Subscribe Internetworking," in *Proc. of ACM SIGCOMM workshop on Information-centric networking (ICN)*, pp. 43-44, 2013.

G. Parisis, B. Tagger, D. Trossen, C. Tsilopoulos, G. Xylomenos and G. C. Polyzos, "Demonstrating Usage Diversity Over an Information-centric Network," in *IEEE INFOCOM*, pp. 27-28, 2013.

G. Parisis, B. Tagger, D. Trossen, D. Syrivelis, P. Flegkas, C. Stais, C. Tsilopoulos and G. Xylomenos, "Demonstrating an Information-Centric Network in an International Testbed," in *International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM)*, pp. 400-402, 2012.

Technical Reports

N. Fotiou, K. V. Katsaros, X. Vasilakos, C. Tsilopoulos, C. N. Ververidis, G. Xylomenos and G. C. Polyzos, "H-Pastry: An Adaptive Multi-level Overlay Inter-Network," MMLab Tech Report 2011-00, 2011.

ABSTRACT OF THE DISSERTATION

Multicast Forwarding in Future Information-Centric Network Architectures

by

Christos Tsilopoulos

Doctor of Philosophy in Computer Science

Athens University of Economics and Business, Athens, 2016

Associate Professor George Xylomenos, Chair

Information-Centric Networking (ICN), an active research area in Computer Networking, has the goal of designing a network architecture that better suits today's networked world: billions of connected devices requesting access to voluminous amounts of digital media. ICN architectures aim to facilitate content distribution by placing self-identified information items at the heart of the networking protocol stack and building routing and transport protocols around them.

Among other design choices, ICN emphasizes support for scalable and efficient multicast delivery. However, practical implementations of ICN architectures have found that achieving this goal is easier said than done. For instance, the

Content-Centric Networking (CCN) architecture realizes single-source multicast delivery through a distributed stateful forwarding scheme, which faces scalability issues with respect to the forwarding state kept at routers. On the other hand, the Publish-Subscribe Internet (PSI) architecture proposes a centrally-controlled fully stateless multicast forwarding scheme with In-packet Bloom filters (IBF), which faces scalability issues with respect to the network and/or multicast group size.

This dissertation addresses the issue of multicast forwarding scalability in two very different ICN architectures, namely CCN and PSI. The common ground in the proposed solutions is the use and extension of Bloom filter-based packet forwarding mechanisms. In CCN, we relax the fully stateful nature of its forwarding scheme, in order to improve the architecture's scalability with respect to the forwarding state kept at routers. We propose a semi-stateless forwarding scheme by incorporating Bloom filter-based forwarding, while respecting the architecture's distributed routing and forwarding operations. Our simulation-based study shows that the scheme can effectively reduce forwarding state at routers by 60%-70% at the cost of 1.5%-12% in bandwidth overhead.

In PSI, we similarly relax the fully stateless nature of its forwarding scheme, proposing a semi-stateless alternative with IBF switching, in order to allow IBFs to efficiently scale to any network and/or multicast group size. We integrate IBF switching in PSI, utilizing the architecture's centralized control-plane operations. Our simulation-based study shows that with IBF switching, forwarding scalability is achieved regardless of network and/or multicast group size, at the cost of placing forwarding state at 0.5%-2.5% of forwarding nodes, which is far less than that of competing technologies.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Clean-slate networking

The Internet architecture and the TCP/IP protocol suite are undoubtedly technological successes. What started out as an academic experiment, has become a global communication infrastructure with billions of connected devices delivering voluminous amounts of digital content. This success was accompanied by a shift in usage: the Internet was initially designed to interconnect hosts for sharing scarce computational resources, while nowadays it is mostly used for ubiquitous content retrieval, accessed by billions of mobile devices, with video dominating Internet traffic [1]. During this transformation in usage and access characteristics, several limitations and constraints arose that stemmed from the Internet architecture itself. To name a few: IP address depletion, lack of inherent support for mobility, spam content and DoS attacks as a result of the network's best effort to carry any traffic sent by hosts. To alleviate these issues, several add-ons and patches have been used, for example Network Address Translation (NAT) for extending the scarce device address space, firewalls for blocking malicious traffic and DNS redirections to serve large traffic volumes from the nearest servers. These solutions however either created their own problems [2] which were themselves solved by yet more ad-hoc solutions [3], or were misused [4]. Overall, this has led to a situation

where network operation and management have become too complicated.

Interestingly, the reason for the persistence of these problems, is largely the exact reason that the Internet architecture has become so successful. The hourglass shape of the Internet architecture had allowed new technologies and protocols to be developed and deployed both below and above the architecture's thin waist, the Internet Protocol. IP serves as the glue that interconnects new transport technologies, from optical to wireless, with new applications, nowadays dominated by HTTP. On the other hand, IP itself proved impossible to evolve. Any significant modification to IP would require tremendous changes in hardware infrastructure deployments all over the world. As a result, for many years, research proposals and experimental prototypes focusing on layer-3 (e.g., [5, 6, 7, 8, 9, 10, 11]) and/or overall architecture enhancements (e.g., [12, 13, 14]) were left untouched behind academic boundaries, without any significant impact on the industry, hence, the real world. Eventually, it was considered that significant innovation could only be achieved by building new *overlay* networks on top of the Internet [15].

In the mid 2000's, academia felt *frustrated* by the so called *ossification of IP* and proposed to start re-thinking our approach to networking, rather than trying to solve the Internet architecture's limitations with additional patches. Researchers aimed to design from scratch a new inter-networking architecture that would better suit current needs [16, 17]. The *movement* was then named *Clean-slate Networking* and set a research agenda for a network architecture with enhanced security, mobility, reliability and availability, scalability, quality of service and economics features, compared to the existing Internet architecture [16]. As a result, several large research projects were initiated, each one proposing to revise inter-networking from entirely different views. Among others, it is worth mentioning the Mobility-First, XIA and NEBULA projects which started in 2010, with funding from the National Science Foundation's (NSF) Future Internet Architecture (FIA) program. MobilityFirst investigated a network architecture where mobility is the norm [18]. XIA, which stands for eXpressive Internet Architecture, proposed a network architecture that has the ability to evolve [19]. NEBULA focused on an evolved version of the current Internet architecture where inter-connected publicly-accessed Data

Centers provide computation and storage and bandwidth resources to users [20].

1.1.2 Information-centric networking

Motivated by the observation that the Internet is mostly used for information delivery and retrieval, researchers have been proposing from the early 2000s modifications and enhancements to the current Internet architecture that would make it more efficient to content delivery (e.g., [5, 7, 8, 21, 22]) or even new entirely new networking architectures designed specifically for content distribution (e.g., [23, 24, 25, 26]). Put into the context of Clean-slate networking, a number of relevant research projects were initiated in the USA and Europe with the goal of adopting the principles of Content Distribution applications, such as Content Distributions Networks (CDN) and peer-to-peer file sharing [27], and use them as the basic abstractions for a new content-oriented network architecture. To name a few, the Content-Centric Networking (CCN) [28] and Named-Data Networking (NDN) [29] projects in the USA, and PSIRP [30] and its successor PURSUIT [31], and 4WARD [32] in Europe.[1] Although some of the proposed solutions and architectures that came out of these projects are completely different, these research efforts shared common starting points and high-level ideas and eventually formed the *Information-Centric Networking* (ICN) research area. ICN holds a prominent position in Clean-slate Networking research and is the primary focus of the ICN Research Group (ICNRG) formed in the Internet Research Task Force (IRTF) [34].

ICN architectures aim to facilitate content distribution by placing self-identified information items at the heart of the protocol stack and building routing and transport protocols around them. To make this more concrete, the basic idea in ICN is that hosts and network routers exchange data packets that contain *named addressable content*. This is in direct contrast to the current Internet architecture in which the basic networking abstraction is to transmit and forward datagram packets destined to addressable hosts. Shifting the main networking abstraction from addressable hosts to named-content, it is argued that it will facilitate the

[1] This is not an exclusive list of relevant projects. More can be found on the survey of Xylomenos et al. [33].

design and implementation of a networking architecture that can better utilize all sorts of networking resources. With name-based abstractions, apart from bandwidth, network operations may directly utilize memory and storage resources (e.g., caching) as well as computation resources (e.g., for data placement and/or media transcoding) in order to efficiently deliver much larger volumes of data [35]. Despite the fact that the main goal of ICN is the specification of an efficient content distribution networking architecture, ICN seeks to remedy all kinds of identified weaknesses of the Internet architecture. ICN-based architectural proposals seek to be more mobile-friendly, reliable, error-resilient and secure compared to TCP/IP. In the past few years, several ICN designs have been proposed with substantial differences in the provided service models and core network functions, including information lookup, routing, forwarding and transport [33].

After a first series of research projects, [2] ICN activities seem to have focused in a few directions. The most prominent is Content-Centric Networking (CCN), [3] originally proposed by Van Jacobson [36]. CCN proposes to use a request-response HTTP-like communication model as the basic layer-3 abstraction, while routers operate in a distributed manner, adopting many ideas from the IP world. Users transmit Interest packets and receive Data packets that contain named-content. Content is named with hierarchical identifiers, e.g., */a/b/c.pdf*, similar to paths in a computer file-system. Routers forward Interests based on *Longest Prefix Matching* that is performed against their *Forwarding Information Base* (FIB), very much like how today's IP routers forward IP packets. FIBs are pre-populated using name-based variants of existing distributed routing algorithms and protocols (e.g., [37, 38]). As Interests are forwarded towards content sources, intermediate routers track each forwarded Interest in the *Pending Interests Table* (PIT). Once an Interest reaches the content source, the requested Data packet is transmitted back to the requestor in a hop-by-hop manner, consuming the respective PIT entries at each router. Essentially, PIT entries are breadcrumbs used to establish data paths on a per-packet basis. In CCN terminology, this is called *stateful data forwarding* and it is claimed to allow the architecture to be more effective with re-

[2] A period that roughly spans from 2007 to 2012.

[3] The Palo Alto Research Center (PARC) runs a project with the same name [28].

spect to caching, multicast delivery and error resilience [39]. On the other hand, it has been identified that keeping forwarding state for each requested packet would have consequences on the architecture's scalability properties [40, 41]. We present further details on the CCN architecture in Section 2.1.

An alternative to CCN, the Publish-Subscribe Internet (PSI) architecture follows a different approach. PSI proposed to use a *publish-subscribe API* as the core network abstraction and instead of a fully distributed operation, it proposed (i) to decouple control-plane from data-plane functionalities and (ii) to logically centralize the control-plane functionalities [35, 42]. The PSI architecture is the outcome of the PSIRP and PURSUIT research projects and its functional model is very much aligned with Software-Defined Networking (SDN), which, at around the same time, proposed to decouple the control-plane from the data-plane in switches and centralize the control-plane functionalities [43]. PSI views the network operation as the synthesis of three district functionalities: (i) the Rendezvous function, used for matching user interests for content and locating the appropriate content sources, (ii) the Topology Management and Path Formation function, which monitors network condition and selects appropriate data paths for content delivery and (iii) the Forwarding function which undertakes the actual packet delivery. The first two functions comprise the control-plane whereas the data-plane is undertaken by the third function. As discussed, PSI decoupled the control-plane from the data-plane and proposed that the Rendezvous and Topology Management function are realized by separate logically centralized systems, e.g., a Hierarchical DHT for the Rendezvous [44] and a centralized Path Computation Element for the Topology Management [45]. We present further details on the PSI architecture in Section 2.2.

1.1.3 Multicast forwarding with In-packet Bloom filters

A key aspect to efficient content delivery is the network's support for multicast delivery, i.e., the network's ability to deliver data to multiple hosts simultaneously. This is realized by replicating data packets at branching points of the delivery tree instead of resorting to multiple unicast transmissions, one for each

recipient host, which induces redundant packet transmissions at the common links of the unicast paths. Even though IP multicast had been an important research topic for a long time, multicast technologies face significant scalability problems with respect to the number of co-existing multicast groups. This technical constraint, as well as business related constraints, have confined IP multicast to closed, centrally controlled, enterprise networks, where it facilitates a limited number of centrally-provided applications (e.g., IPTV) [46]. Multicast scalability constraints are imposed by the hop-by-hop distributed packet forwarding model: multicast forwarding state is distributed among network routers which maintain *Multicast Forwarding Tables* (MFT). Unlike unicast IP addresses, multicast addresses are logical and not topological identifiers; thus routers cannot aggregate MFT entries as in unicast IP forwarding, hence MFT sizes grow proportionally to the number of multicast groups traversing a router.

To alleviate these scalability constraints, Bloom filters, a probabilistic representation of sets, have been proposed as a method for single-source packet forwarding [47]. The idea is quite simple: the links of the delivery path (or tree) are encoded into a Bloom filter which is then placed as source-routing information in packet headers; hence the term In-packet Bloom filter (IBF). At each hop, the router extracts the Bloom filter, checks which of its outgoing links are encoded in the IBF and transmits the packet over those links. This way, routers are fully stateless with respect to forwarding state. Not only routers do not need to maintain multicast forwarding state, they are also stateless regarding unicast forwarding as well. Although IBF forwarding tackles the issue of forwarding state kept at routers, it is susceptible to false positive forwarding decisions (redundant traffic) due to the probabilistic nature of the Bloom filter data structure [48]. The rate of false forwarding decisions increases as more elements are added, i.e., links in our case, which, in the case of relatively static networks, happens as a multicast group grows. Studies have shown that once the false positive probability of an IBF exceeds 0.5%, forwarding anomalies arise, causing a sharp decline in bandwidth utilization [47, 49]. We present further details on Bloom filter-based forwarding in Section 2.3.

1.2 Contributions

ICN places the need for efficient and scalable multicast delivery high in the goals of the network architecture. CCN realizes single-source multicast delivery in a distributed fashion through its stateful data forwarding scheme: when multiple Interests for the same content are transmitted at the same time (e.g., in the case of real-time streaming applications as in [50, 51, 52]), these will be aggregated by the first common router along the data path. When the respective Data packet is returned, the branching router will replicate the packet towards the multiple requestors [36]. In CCN, multicast forwarding trees are formed without the need of a distributed group management protocol and PIT entries live for a short time: until the requested Data packet is returned. Nevertheless, early work on CCN has identified that the increased forwarding state cannot scale without exploding memory requirements at the routers [40]. In PSI on the other hand, group management and data path establishment are handled by the logically centralized Rendezvous and Topology Management and Path Formation functions. On the data plane, PSI proposes to use IBF forwarding which has the advantage of keeping forwarding nodes stateless, i.e., once a multicast group is formed and the respective IBF is constructed, there is no need to explicitly place multicast forwarding state at routers. However, due to the probabilistic nature of IBFs, PSI forwarding faces severe scalability constraints with respect to the network and/or multicast group size.

This dissertation addresses the issue of forwarding scalability, particularly with respect to multicast delivery, in the CCN and PSI architectures, using Bloom filter-based forwarding as a common ground. The contributions of the dissertation are twofold. First, we address the issue of forwarding state scalability in CCN and propose to reduce the size of the PIT using a *semi-stateless forwarding scheme*. We achieve that by incorporating Bloom filter-based forwarding in CCN's distributed forwarding scheme without affecting critical architectural aspects of the system. In our scheme, PIT sizes are reduced by 50%-70% with manageable bandwidth overhead costs of 1.5%-12%. Second, we address the issue of forwarding scalability in PSI with respect to the network/multicast group size. We propose to sacrifice

the fully stateless nature of IBF forwarding and introduce a semi-stateless scheme in which we install multicast forwarding state at few, centrally selected routers that perform *IBF-switching*. In this scheme, the system maintains high forwarding efficiency (redundant traffic is kept below 4%) by installing multicast forwarding state at 0.5%-2.5% of network nodes. More specifically:

1. In **Chapter 3** we present a semi-stateless packet forwarding scheme for Content-Centric Networks (CCN) that reduces the network's forwarding state requirements thus improving the architecture's scalability properties. We achieve that by integrating Bloom filter-based stateless packet forwarding in the CCN architecture. We introduce a distributed Bloom filter source-route construction with which we can mitigate forwarding state requirements without sacrificing crucial qualitative features of the network architecture nor requiring additional control-plane signalling. We evaluate the scheme through simulations using both synthetic and realistic ISP-level topologies. Simulation results show that the scheme is very effective in small-to-medium topologies in which it reduces forwarding state in routers by $60\% - 70\%$ at the cost of $2\% - 16\%$ of additional bandwidth. However, in larger topologies, the bandwidth overheads exceeded 50% (and in some cases reached 170%), thus indicating the limitations of this solution.

2. In **Chapter 4** we further extend the CCN semi-stateless forwarding scheme in order to effectively reduce forwarding state at routers in large topologies, while keeping bandwidth costs at manageable levels. We introduce a dynamic adaptation mechanism in our semi-stateless forwarding scheme with which the system reduces forwarding state until routers detect that excessive overhead thresholds are violated. Routers then instruct hosts to act more conservatively through - what we call - Bloom filter Congestion Notification (BCN) feedbacks. Our simulation-based study shows that, with dynamic adaptation, the scheme is effective in large topologies as well and, in addition, it further improves its efficiency in small-to-medium topologies. Overall, our proposed solution reduces forwarding state in CCN routers by $50\% - 70\%$ with bandwidth costs of $1.5\% - 12\%$ in all tested scenarios.

3. In **Chapter 5** we address IBF forwarding scalability issues in large networks. In addition to the distributed Bloom filter construction presented in Chapters 3 and 4, we present a semi-stateless forwarding solution that utilizes centralized management, suitable for the PSI architecture. We introduce *Bloom filter relay nodes* that perform *IBF switching*. We present a simple post-order tree-traversal algorithm that selects appropriate *IBF switching points*. We evaluate the solution through simulations in large-scale scale-free synthetic graphs and show that, with IBF switching, we can achieve high levels of forwarding efficiency (redundant traffic is kept below 4%) by placing multicast forwarding state at $0.5\% - 2.5\%$ of the network nodes. We compare the scheme's forwarding state requirements with relevant multicast forwarding schemes and show that IBF switching reduces stateful nodes by $42\% - 99.6\%$. Hence, even though the fully stateless operation of Bloom filter based forwarding is sacrificed, the scheme can still get far better scalability compared with competing approaches.

1.3 Dissertation outline

The remainder of the dissertation is organized as follows. Chapter 2 provides background and related work regarding ICN architectures and Bloom filter-based packet forwarding, which comprise the context in which this dissertation was realized. In Chapter 3 we present our semi-stateless forwarding scheme for CCN. We describe how Bloom filter-based forwarding is integrated in the architecture and how to construct Bloom filter source-routes in a distributed manner. In Chapter 4 we present the dynamic adaptation of the semi-stateless forwarding scheme and show its overall effectiveness in all tested scenarios. In Chapter 5 we present the centralized approach to semi-stateless Bloom filter forwarding and present our IBF switching scheme. Finally, we conclude in Chapter 6.

Chapter 2

Information-Centric Networking

In this chapter we provide the basic background which forms the context in which this dissertation was realized. In particular, we describe in detail the Content-Centric Networking architecture in Section 2.1, the PSI architecture in Section 2.2 and Bloom filter-based packet forwarding in Section 2.3.

2.1 Content-Centric Networking

The *Content-Centric Networking* (CCN) architecture places *named content packets* at the thin waist of the protocol stack and provides users with a *request-response* service model in which users *pull* data packets from the network [36]. Users request *named* Data packets via *Interests*. Data packets contain a name that uniquely identifies the carried payload, whereas Interests carry the name of the requested Data; no host addresses are used. For each Interest, a user receives *at most* one Data packet, thus there is a strict one-to-one relation between Interests and received Data packets. The structure of content names is similar to URIs: names are hierarchical with variable-length components, e.g., */a/b/c.mp4*.

Routers propagate Interests towards content sources and return Data packets along the reverse path with the help of three data structures: (i) the *Forwarding Information Base* (FIB), (ii) the *Pending Interests Table* (PIT) and (iii) the *Cache Store* (CS) (Figure 2.1). The FIB serves as a name-based routing table for forwarding Interests. At each arriving Interest, routers perform a *Longest Prefix Match*

on their FIB and push the Interest towards a content source. FIBs are populated through name-based routing protocols, either similar to the ones used in IP [36] or new ones [37, 38, 53]. Before forwarding an Interest, routers insert the Interest in the PIT, noting its incoming interface. If a PIT entry for the same name exists, meaning that the router has already forwarded an Interest for the same packet, the Interest is dropped, otherwise it is forwarded; however, the incoming interface of the Interest is noted in the PIT, thus merging multiple Interests for the same content into a single PIT entry. This operation continues until the Interest reaches the content source which responds with the requested Data packet.

Data packets are delivered by reversing the path taken by the Interest. At each hop, routers check their PIT for a matching entry, transmit the Data packet backwards and delete the PIT entry. If a router had received Interests for the same Data from multiple interfaces, the router duplicates the Data packet, thus realizing *multicast* delivery. Finally, the CS, as it name suggests, is a cache containing Data packets. Intermediate routers may directly serve an Interest if the requested Data is present in the CS. Routers fill their CSes with traversing Data packets, but their detailed operation (e.g., cache replacement policy and interaction with the routing system) is an open research issue [54].

Apart from multicast, CCN inherently supports *anycast*, where the same data is available in multiple locations, and Data *multihoming*, where a content source can be reached through multiple paths. These two features are implemented by having the FIB point to multiple next-hops for forwarding an Interest. The Interest route (and consequently the Data route) is decided by on-path routers through a CCN module called the *strategy layer*. Anycast and/or multipath is transparent to users.

The Interest-Data mechanism essentially constitutes a *request-response* communication model. Data transport is receiver-driven, with stateless senders (content sources), i.e., senders do not maintain any connection state, simply responding to incoming Interests. Transport related issues are handled by the receiver. For error control, the receiver retransmits Interests for missing Data packets. Flow and congestion control are also applied by the receiver by pipelining Interests. For

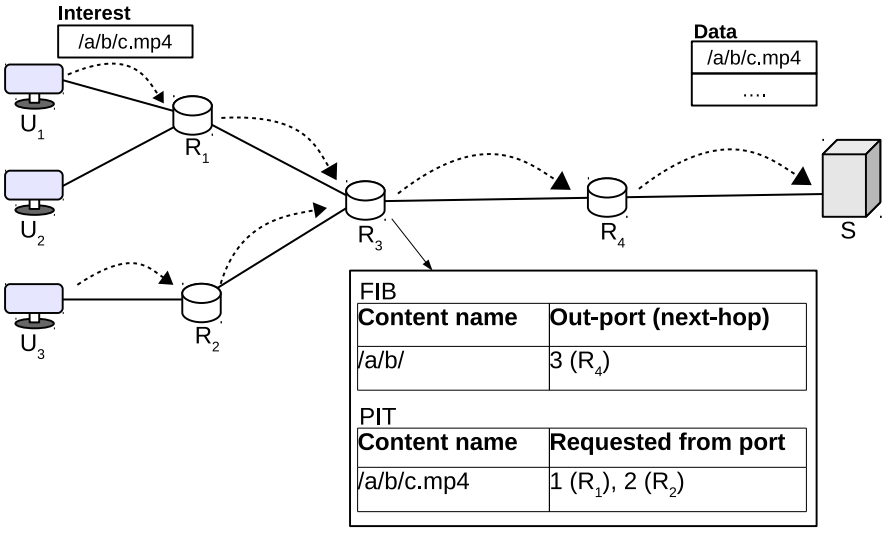


Figure 2.1: A sample CCN interaction. U_1 issues an Interest for */a/b/c.mp4* which is located at S. Arrows show the propagation of the Interest. Data follows the reverse path. The FIB and PIT (but not the CS) for R_3 are shown.

example the amount of transmitted Interests can be controlled with a TCP-like sliding window mechanism [55].

It is easy to see that CCN employs a stateful forwarding plane. For each Data packet to be delivered to the requester, a PIT entry is required in each router along the data path. Data paths are established as Interests are forwarded towards content sources. Essentially, PIT entries act as breadcrumbs produced during Interest forwarding and consumed during Data forwarding. The stateful name-based forwarding of CCN offers four key advantages. First, the network provides native support for multicast delivery. If multiple users request the same content, their Interests are suppressed by common on-path routers, which later duplicate the received Data [36]. Second, hosts are *address-less* which, it is claimed that, will reduce a number of address-related vulnerabilities such as address depletion as well as governance and management costs. Third, security is enhanced by dropping unwanted traffic since each router prevents the delivery of data that has not been explicitly requested (e.g., DoS attacks and *spam* content) [36]. Fourth, maintaining per-packet forwarding state enables routers to realize *adaptive forwarding* functionalities, i.e., routers may actively participate in functions such as link failure

recovery, flow control and detection of malicious user behaviour [39].

Tracking each forwarded Interest, however, raises scalability concerns [40, 56]. Indeed, the CONET project has proposed avoiding per-packet forwarding state by adopting *stateless* forwarding [57]: Interests gather path information on their way to the content source; Data are source-routed by reversing that path information. The removal of forwarding state, however, nullifies the advantages of CCN forwarding: (i) routers cannot aggregate Interests or duplicate Data, thus multicast is not supported, (ii) security is downgraded since host addresses (which were omitted on purpose) are required by some source-routing schemes and routers cannot drop unwanted packets because forwarding state is removed, which is also the reason why (iii) adaptive forwarding is disabled.

2.2 Publish-Subscribe Internetworking

The *Publish Subscribe Internetworking* (PSI) architecture [42] shares many high-level goals with CCN, but it tries to achieve them following an entirely different design approach. PSI differs in two major aspects from CCN. First, the notion of content objects in PSI is more abstract, i.e., a content object is not strictly mapped to a single data packet. Second, the core network operation (routing requests and forwarding data) is split in three distinct functional modules, emphasizing the decoupling of routing control from data packet forwarding.

2.2.1 Functional organisation

PSI models content objects as *publications*, content sources as *publishers* and content consumers as *subscribers*. The architecture provides users with a publish-subscribe API for announcing and requesting data. The architecture organizes the core network operation into three distinct functions, each one implemented as a separate subsystem [35]: (i) the Rendezvous system (RVS), (ii) the Topology Management and Path Formation system (TMPFS) and (iii) the Forwarding system (FS) [42].

The RVS serves as a resolution system; it tracks available publications in

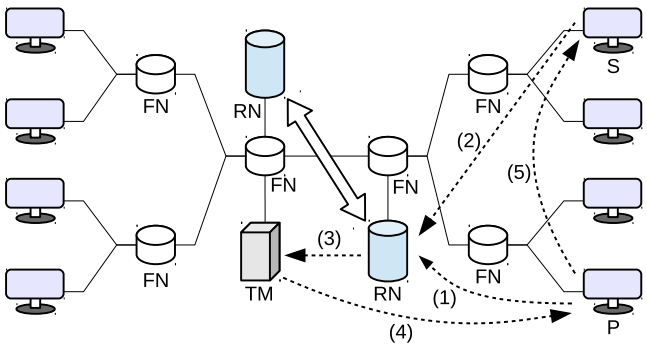


Figure 2.2: A sample PSI interaction. P announces a content object to the network (step 1). S subscribes to it. The RVS handles the subscription (step 2), the TMPFS selects the delivery path (step 3) and notifies P (step 4). P transmits the requested object over the specified path (step 5).

the network and resolves user subscriptions. Nodes that implement RVS functionality are called Rendezvous Nodes (RNs). The Topology Management and Path Formation function serves a dual cause. First, it monitors network topology and link conditions. Second, it computes optimal delivery paths for disseminating publications from publishers to subscribers. The TMPFS functionality resides in Topology Manager nodes (TMs), which may be co-located with RNs or placed in separate physical hosts. The Forwarding System undertakes the actual data transmission, i.e., packet forwarding, through Forwarding Nodes (FNs).

In terms of functional design, content delivery in PSI is a step-wise process in which the three subsystems interact as shown in Figure 2.2. Initially, a publisher announces the availability of a publication to the RVS (step 1). To receive a publication, a user issues a subscription which is handled by the RVS as well (step 2). The RVS locates and selects the *best* publisher and requests the TMPFS to compute the publisher-to-subscriber path (step 3). The TMPFS selects a *suitable* path and instructs the FS to establish the selected path. Specifically, the TMPFS hands a Forwarding Identifier (FID) to the publisher and instructs the publisher to forward the requested item (step 4). Finally, the publisher transmits the requested content object over the specified path (step 5).

PSI does not enforce a particular naming scheme. The architecture is compatible with both flat and hierarchical identifiers. The granularity of publications

is not mandated by the architecture either. Publications may represent files (of arbitrary size), chunks of files or individual network packets. Publications may also represent live streaming flows of undefined size. It is left to the application to decide how to represent a content object, taking into consideration the transport context and that each available publication must be announced separately to the RVS. For example, it is not advisable to represent a live stream as a series of publications, one for each data packet, as in CCN. Doing so in PSI would require that each stream packet is announced to the RVS and users issue subscriptions that are handled by the RVS-TMPFS subsystems. This triangular *resolution-path formation-notification* process may lead to excessive network load and large delays. In addition, since streaming packets have relatively short lifetimes, they would need to be unpublished (withdrawn) shortly after they are published, thus further burdening the network operation. Live streams are better represented with a single publication: stream sources announce the stream once and receivers *tune in* with a single subscription. Once the subscription is resolved, the source transmits the streaming packets to the subscriber until the latter withdraws the subscription. Note that an explicit *unsubscription* message is also required.

Multicast delivery requires some network entity to keep track of users subscribed to information items (e.g., RSS feeds) in order to construct and establish the respective forwarding trees. In PSI, the task of tracking multicast receivers is assigned to the RVS. When a user issues (withdraws) a subscription, the RVS adds (removes) the user from the list of item receivers and requests the TMPFS to create a source-specific forwarding tree, rooted at the publisher and with item subscribers as the tree's leafs. The details of how these mechanisms are implemented vary according to the actual function implementations used.[1] Once multicast forwarding state is established, the respective FID is constructed and communicated to the content source. PSI decouples routing control from FNs and delegates path/tree computation to the TMPFS. This design choice has the advantage of making optimal routing decisions feasible, for example multicasting data over minimum-cost Steiner trees [58, 59]. Such optimizations are too complex to implement in a fully

[1] In IP multicast for instance, all routers implement both the Rendezvous and Topology Management functionalities, as they participate in distributed multicast routing protocols.

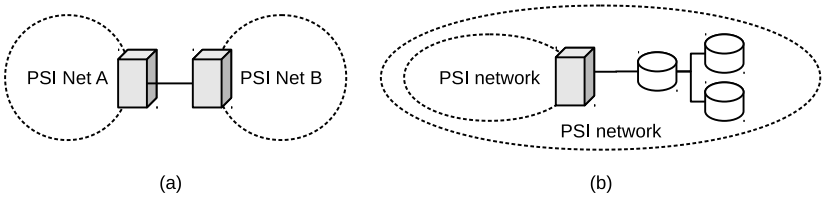


Figure 2.3: Interconnection of PSI networks: (a) neighbouring PSI networks, (b) nested PSI networks.

distributed manner, as in IP and CCN, where routing and forwarding are strongly coupled and routing decisions rely on distributed mechanisms. On the other hand, centralized designs often face scalability issues, e.g., computation delays in the TM nodes.

PSI networks interconnect with each other through border gateways for internetworking purposes. PSI networks may be connected with other PSI networks forming neighboring relations (Figure 2.3(a)) or a network may contain nested PSI networks (Figure 2.3(b)). Complex network organizations may use a mixed combination of the two interconnection types.

Each PSI network runs its own domain-specific implementations of the core functions, carefully selected for the particular networking environment it operates in (e.g., home, enterprise, mobile, wireless ad-hoc, data-center, etc). PSI networks communicate with each other through inter-PSI implementations for RVS and TMPFS, in the same spirit of the inter-domain routing protocols used in the Internet. In inter-PSI communication, each network aggregates subscriptions issued by domain-local users when these need to be resolved by other PSI networks. Upon the arrival of the requested data packets, the network de-multiplexes them and delivers the data to local subscribers. Apart from enhancing system scalability, this inter-PSI communication abstraction allows each network to apply its own domain-specific policies in order to optimize its internal operation, without affecting global communication.

2.2.2 Implementation of PSI functions

Apart from a high-level functional design, specific solutions have been proposed for each one the core PSI functions. The RVS may be implemented as a Distributed Hash Table formed by RNs [44]. The TMPFS can be realized by one or more physical TM nodes that gather link-state information from FNs, compute data paths upon requests from the RVS and send FIDs to publishers instructing them to transmit data [45]. Last, but most important in the context of this dissertation, PSI promotes the use of IBF forwarding as the main forwarding solution. That is, the paths that are computed by the TM nodes are then used to construct IBFs. These IBFs, which can be unicast or multicast trees, are then handed to publishers. Due to the stateless nature of IBF forwarding, the TMPFS does not have to explicitly install state at FNs before handing the IBF to the publisher, hence item resolution delays include the RVS and TMPFS delays only.

2.3 Bloom filter-based packet forwarding

Researchers have been long investigating ways to tackle the scalability issues arising when multicast forwarding state is kept at routers [46]. An initial approach was to reduce the amount of multicast forwarding state by placing forwarding state in few routers only. For example, REUNITE proposed to place multicast forwarding state only at branching points of the multicast tree and utilize unicast forwarding between branching points [60]. This, however, is effective for sparse trees where only a few branching points exist but not for dense ones. An alternative approach is to completely remove multicast forwarding state from routers and resort to source-routing techniques by placing forwarding information inside the packet headers. This way, routers are *stateless* and the network can support an arbitrary number of multicast sessions without worrying about forwarding state explosion. In Explicit Multicast (Xcast), the addresses of the multicast group participants are explicitly inserted into the packet header [61]. At each hop, routers extract the list of recipient addresses from the packet header and forward packets according to their unicast forwarding table. Due to the limited capacity of packet

headers, however, Xcast can accommodate relatively small groups. In order to accommodate larger groups, Xcast needs to install multicast state in some of the multicast tree routers as well [62].

To further improve the scalability properties of multicast forwarding, Bloom filters, a probabilistic representation of sets [63], have been proposed. The basic idea is to reduce the forwarding state either by compressing the Multicast Forwarding Table (MFT) in the case of statefull multicast forwarding [64] or by compressing the source-route header in the case of stateless multicast forwarding [47]. For example, BUFFALO [65] and Application-oriented Multicast (AOM) [66] are Bloom filter-based extensions of Xcast and, essentially, encode the list of the multicast group's host addresses into a Bloom filter. However, these techniques require unicast forwarding state in order to forward packets using the encoded host addresses and, in addition, involve, complex, resource-consuming operations at each hop. The work of LIPSIN [47] introduced a lightweight Bloom filter-based technique in which the source-route contains the links of the delivery path or tree. The technique is fully stateless in the sense that routers need not maintain any forwarding information, i.e., multicast and unicast. The work of this dissertation utilizes and extends this technique.

2.3.1 Packet forwarding

In Bloom filter-based forwarding, the links of the unicast path or multicast delivery tree are encoded into a Bloom filter which is then placed as a source-route in the packet header; hence the term *In-packet Bloom filter* (IBF). In general, the construction of the IBF may occur at the source node [49] or it may be delegated to a separate routing module [47, 59]. In PSI, for instance, this is delegated to TM nodes. To encode path links into Bloom filters, links are assigned with a *Link Identifier* (LID). An LID is an m-bit string with only m bits set to 1 ($k << m$). The positions of the k bits are determined using k hash functions, e.g., by applying the k hash functions on the network adapters MAC address. LIDs are unidirectional (a bi-directional link is assigned two LIDs, one for each direction) and do not need to be unique.

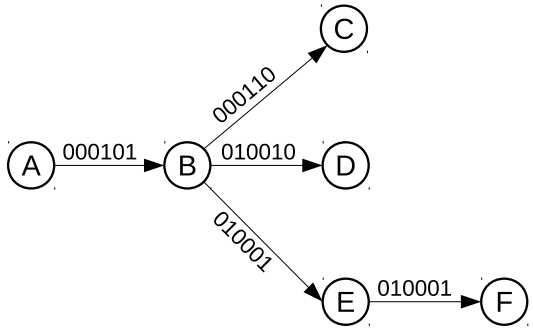


Figure 2.4: A simple network with assigned LIDs, m=6 and k =2. LIDs need not be unique. Only left-to-right LIDs are shown.

A delivery path is encoded into an IBF by *adding* (*OR*ing) the path LIDs. In the example of Figure 2.4, the IBF for transmitting data from A to C is

$$IBF_{A\rightarrow C} = LID_{AB}\ OR\ LID_{BC} = 000101\ OR\ 000110 = 000111$$

which is then placed at the packet header. Forwarding nodes extract the IBF from packet headers, examine which of their outgoing links are part of the IBF and transmit the packet over those links. To examine whether an LID is contained in the IBF, the forwarding evaluates the following expression

$$(IBF\ \&\ LID_i) == LID_i$$

and if it evaluates to true the node assumes that is encoded into the iBF and transmits the packet over link i. Continuing the example of Figure 2.4, when node B receives a packet with the IBF 000111, it will forward it to C since

$$IBF_{A\rightarrow C}\ AND\ LID_{BC} = 000111\ AND\ 000110 = 000110 = LID_{BC}$$

IBFs inherently support multicast delivery. The method is simple: to add (OR) the LIDs for all the tree links comprising the delivery tree. In the example of Figure 2.4, the IBF for multicasting data from node A to nodes C and D is

$$IBF_{A\rightarrow C,D} = LID_{AB}\ OR\ LID_{BC}\ OR\ LID_{BD} = 010111$$

The only difference multicast brings to the forwarding node is that it must be prepared for a packet's IBF to match many of its LIDs, which should cause the packet to be forwarded out of each corresponding link.

2.3.2 Bloom filter construction

Bloom filter-based packet forwarding systems are generally classified in two categories, depending on how the IBF is constructed. In the first category, the data path is constructed in a distributed manner and the IBF is computed at the source node [49, 67]. In these systems, recipient nodes send special packets towards a designated source. As these messages are propagated, they collect reverse path information, i.e., an ID for each traversed node. When the packet reaches its destination, the source node extracts the reverse path, and computes the respective IBF. A basic assumption in these systems is that nodes have sufficient topological information to forward the data path construction messages and compute the IBFs.

In the second category, the data path (or tree) and the respective IBF are centrally computed by a separate routing module that resides in a dedicated server [47, 59]. The PSI architecture belongs to this category. In these systems, path construction messages are delivered directly to the routing module which computes the data path (or multicast tree) based on topological information (e.g., link delays), constructs the IBF and then sends it to the group source. In these systems, the delay for path construction is increased, due to the need to also communicate with the routing module.

2.3.3 Forwarding efficiency

There are two significant advantages in IBF forwarding. First, it is very lightweight in terms of the state required at forwarding nodes: nodes only need to store the LIDs for their outgoing links, which requires space proportional to the node's degree. This also applies to multicast: as forwarding information is kept at packet headers and not at routers, the network can support an arbitrary number of multicast sessions without worrying about MFT explosion. The second advantage of IBF forwarding regards its capability for line-speed operation, in contrast to other source-routing methods. The forwarding logic is based on simple bitwise operations which can be easily implemented in hardware [47].

On the other hand, IBF multicast faces severe scalability issues with respect to multicast group size as a result of the Bloom filters' probabilistic nature. When

a Bloom filter is queried whether an item is contained in the set, it may return a false positive. In a packet forwarding context, the IBF may falsely answer that an LID is present in the path, thus the packet will be falsely forwarded over that link. Continuing the example of Figure 2.4, the IBF for multicasting data from node A to nodes C and D was $IBF_{A \rightarrow C,D} = 010111$. When the multicast packet arrives at node B, the query against LID_{BE} returns a false positive since

$$IBF_{A \rightarrow C,D} \ \& \ LID_{BE} = 010111 \ AND \ 010001 = 010001 = LID_{BE}$$

Thus, node B falsely assumes that the packet is meant to be transmitted over link $B \rightarrow E$. To make matters worse, at node E the IBF also matches LID_{EF}, so the packet is also falsely forwarded from node E to node F. In this example, a multicast delivery intended for 3 links resulted in 5 transmissions. The amount of redundant traffic generated depends on the false positive probability (fpp) of the Bloom filter as given by [48]

$$fpp = (1 - e^{-kn/m})^k \tag{2.1}$$

where m is the size of the Bloom filter, k is the number of hash functions used and n is the number of inserted items. The fpp increases as (i) the size of the Bloom filter decreases or (ii) the number of inserted items increases. In IBF multicast, as group size increases, more links are added, thus the fpp also increases. Previous research in IBF multicast forwarding reported that once the fpp exceeds 0.5% there is a sharp rise in the amount of redundant traffic caused by false positives [47, 49].

To better illustrate the performance degradation of IBF multicast when the group size grows, Figure 2.5 presents simulation results for a synthetic scale-free graph with 500 nodes. The plot shows the forwarding efficiency of multicast delivery for various IBF sizes, defined as

$$forwarding \ efficiency = \frac{\#multicast \ tree \ links}{\#total \ packets \ transmitted} \tag{2.2}$$

If we target a forwarding efficiency of 90% (i.e., 10% of redundant traffic), we see that multicasting with a 256-bit IBF can scale up to roughly 6-7 nodes. A 1024-bit IBF scales up to 60 recipient nodes. These results are consistent with previous research [47, 49, 59].

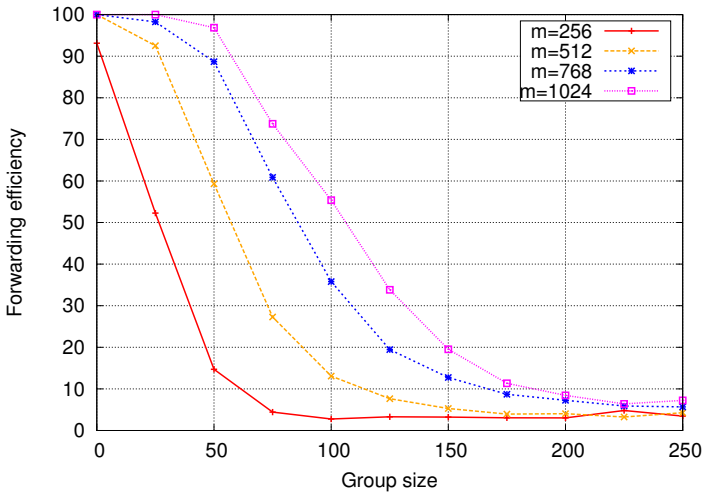


Figure 2.5: IBF forwarding efficiency in a network with 500 nodes.

Several research studies have focused on mitigating IBF forwarding anomalies such as redundant transmissions and, even, routing loops. LIPSIN proposed to install short-lived packet caches in routers in order to eliminate forwarding loops [47]. When a looped packet arrives at a router, the router compares it against this cache and discards it, thus eliminating the loop. This approach, however, introduces extra state at routers and increases per-packet processing overhead. Särelä et al. proposed a varying-k method for computing the LIDs and a bit-permutation scheme during IBF construction and forwarding [49]. These schemes mitigate forwarding anomalies but do not provide significant scalability benefits. Going one step further, Särelä et al. proposed BloomCast, a protocol for inter-domain IBF multicast [49]. In BloomCast, the IBF is computed based on the inter-domain graph and separate IBFs are used at each individual Autonomous System (AS). When packets enter ASes, ingress routers encapsulate packets with a domain-specific IBF and tunnel them towards AS exit points. Although BloomCast shrinks the topology graph by applying the abstraction of inter-domain and intra-domain graphs, it does not solve the scalability problems on a single graph. For instance, according to CAIDA traces, the Internet inter-domain graph is currently reported to contain more than 35000 ASes [68]. Using a single IBF for multicast delivery over the inter-AS graph, BloomCast scales up to 20 AS nodes

with a 1024-bit IBF. In the Efficient and Scalable data center routing scheme (ESM), which focused on IBF multicast for data center topologies, the technique was used only for small (manageable) multicast groups, resorting to hop-by-hop multicast forwarding for bigger group sizes [59]. In Hierarchical Tree Splitting (HST), the multicast tree is split in several sub-trees, all rooted at the source, so that the IBF for each sub-tree has a low fpp [67]. The source node maintains several IBFs (one per sub-tree) and transmits packets over all trees. HST preserves the stateless operation of routers, at the expense of additional multicast state at the source node and redundant traffic: the sub-trees may have overlapping links, thus causing replicas of data packets to traverse the overlapping links more than once. Overall, HST still suffers from excessive redundant traffic in very large trees.

Other studies focus on increasing IBF capacity in order to fit more tree links without significant performance penalties. In general, the approaches for increasing the capacity of a IBFs are (i) to vary the size m of the Bloom filter or (ii) to compute the optimal value for k so that fpp remains low [48]. Varying m can be realized in two ways. First, by using a Bloom filter that grows dynamically as items are inserted [69, 70]. Second, by computing a value for m that minimizes the fpp, provided that we know n beforehand, i.e., we must first determine the multicast tree size and then compute the size of the IBF. In both cases, varying m means using different IBF sizes on a per tree size basis, which is not practical in the context of packet forwarding. First, m cannot grow without bound: there are limits to the size of the IBF imposed by Maximum Transfer Unit (MTU) packet sizes; in addition, IBFs should be a relatively small part of packets as they represent control overhead. Second, using variable length identifiers is unfriendly to line-speed hardware-level implementations of IBF forwarding.

The second approach for increasing the capacity of a Bloom filter is to keep m fixed and vary k [67]. This approach requires first determining the number of multicast tree links and then computing k. The value for k that minimizes fpp is [2]

$$k_{opt} = \frac{9m}{13n} \tag{2.3}$$

Varying k is impractical because it does not guarantee that the fpp will remain

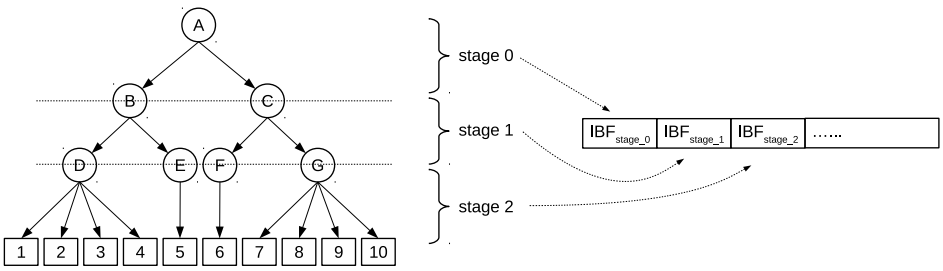


Figure 2.6: Multi-stage Bloom filters.

below a desired threshold. For example, if $m = 256$ bits and $n = 200$ tree links, then $k_{opt} = 0.88$. In practice, however, k cannot be less than 1; at least one bit needs to be set in LIDs, otherwise packet forwarding will fail. In this case, if we set $k = 1$ and apply Equation (2.1) we get $fpp \approx 0.55$, which is far too big compared to the suggested 0.5% [49]. Moreover, varying k is also impractical as it would require computing and setting LIDs in nodes on a per multicast tree size basis. In contrast, having a unique, predefined value for k allows LIDs to be computed and installed once, during network bootstrap.

Multi-stage Bloom filters (MSBF) apply a combination of the techniques described above in order to provide a false positive-free Bloom filter [71]. The basic idea in MSBF is to use multiple IBFs in the packet header, one for each *stage* for the delivery path/tree (see Figure 2.6), where $stage_i$ is the set of links that are i hops away from the source node. For example, $stage_0$ includes all delivery links that are attached to the source node and $stage_1$ includes all links that are 1 hop away from the source node. At each hop of the forwarding phase, routers inspect a TTL in the packet to identify the stage at which the packet is and then forward the packet by inspecting the respective IBF inside the MSBF header. The MSBF is constructed by a centralized routing module. For each stage, the module computes the optimal IBF, i.e., an IBF that produces as few false positives as possible when matched against all LIDs of the specific stage. To achieve that, MSBFs are variable-length: the routing module computes the optimal IBF size in the attempt to produce a false-free IBF for a particular part of the network topology. Overall, MSBF is a forwarding mechanism that breaks the delivery path/tree in multiple variable-length IBFs. MSBF can eliminate redundant traffic to a large extend, however it

requires the presence of a centralized routing module (suitable in the context of PSI but not CCN) and, in addition, the variable-length nature of IBFs nullifies the line-speed forwarding properties of plain fixed-size IBFs.

Chapter 3

Reducing Forwarding State in Content-Centric Networks

3.1 Stateful data forwarding

As discussed in Section 2.1, CCN employs a stateful forwarding logic in which routers maintain forwarding state for each forwarded Interest in their Pending Interests Table (PIT). These entries are consumed as Data packets are propagated from content sources to content requesters in a hop-by-hop manner, following the Interest's reverse path. CCN advocates stateful data forwarding due to four key advantages:

1. It natively supports multicast delivery. The first common router towards a content source will aggregate Interests for the same name and duplicate the Data packets returned. This is particularly useful in real-time streaming applications [50, 51, 72] where users consume the same content in a synchronized manner, i.e., they transmit Interests for the same Data simultaneously.

2. Host addresses are omitted. This may reduce (or, even, eliminate) a number of address-related problems, such as address space depletion, address assignment and governance [36].

3. The network delivers only data that has been requested; routers drop Data

packets that do not have a matching Interest in their PIT, thus unwanted traffic (e.g., *spam*) is discarded near the source and not at the recipient [36].

4. Maintaining per-packet forwarding state is an enabling factor for *adaptive forwarding* [39], in which routers may exploit forwarding state to assist functions such as fast recovery from link failures, congestion avoidance and early detection of malicious users.

The amount of forwarding state kept in routers, however, raises scalability concerns related to the PIT size. The number of Interests that must be stored in the PIT in order to fully utilize the network depends on the link capacity, the size of Data packets and the average *Round-Trip Time* (RTT) which defines the lifetime of Interests inside the PIT. A rough estimate for the required number of PIT entries per link is $bandwidth \times RTT/data\_packet\_size$. For example, to fully utilize a 40 Gbps link with 1000-byte Data packets and an average RTT of 80 ms, the PIT must contain 400K entries; this must be multiplied by the number of links hosted by the router. Furthermore, real-time applications such as multimedia streaming [50, 51, 52, 72] and publish-subscribe applications [73], request Data before they are generated. This leads to an increased effective RTT, as Interests remain longer in the PIT. The work in [74] mapped realistic IP traffic onto CCN and estimated that a 20 Gbps access router would require 1.5M of PIT entries.[1] Furthermore, work in [75] estimates that, in an extreme worst case scenario, the PIT may reach 30M-60M entries. Taking into account that CCN names are variable-length and, in general, much longer than host addresses, the total memory requirements for the PIT grow significantly.

A very large PIT has grave implications for network throughput. Since the PIT is examined for *every* arriving packet, it should *ideally* reside in a line-card's on-chip memory, which is very fast but has very limited capacity [40, 56]. Initial investigations reported that a hash table-based implementation is too big to fit inside today's on-chip memories [40, 75]. The fallback option is to place the PIT

[1] In that study, the authors assumed that an Interest can correspond to multiple Data packets. This radically changes the basic CCN behavior of *one Interest per Data* and may heavily underestimate the amount of PIT entries.

in the router's main memory which is much slower, thus causing a performance degradation. Two proposed PIT implementations, *DiPIT* [76] and *Encode Name Prefix Trie* (ENPT) [74], can, under certain assumptions on the traffic mix and average content-name length, substantially reduce the memory-footprint of the PIT, but not enough to fit it into a line-card's on-chip memory. In addition, DiPIT encodes multiple Interests in Bloom filters, thus losing the information of which particular Interests are stored, which is crucial for dropping stale Interests [75] and adaptive forwarding. ENPT, on the other hand, organizes the PIT in a trie-like structure with a *linear* ($O(N)$) complexity for insert and lookup operations (where N is the number of components in a CCN name), compared to the *constant* ($O(1)$) complexity of hash tables, which severely jeopardizes the ability of CCN to perform packet forwarding at line speed.

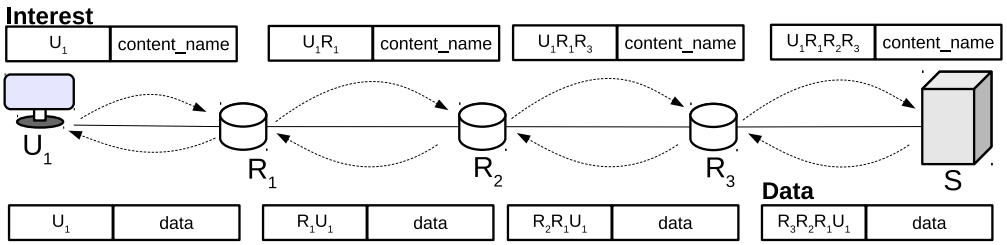


Figure 3.1: The CONET variation of CCN.

The CONET architecture addressed the issue of forwarding state by moving the forwarding information from the routers to the packet headers [57]. In essence, CONET is a fully stateless forwarding variant of CCN. In CONET, all packets carry a path header listing a sequence of node identifiers. During Interest propagation, routers append their identifier in the packet's path header. When the Interest reaches the content source, the header is reversed and placed as a source-route in the Data header (see Figure 3.1). By removing the PIT entirely however, this stateless forwarding approach loses the advantages of CCN's stateful forwarding. First, support for multicast is nullified, as a router no longer has the required information to suppress Interests and duplicate Data. Interests for the same content are individually forwarded to the content source, which then unicasts the corresponding Data to each requester. Second, forwarding relies on

node identifiers (e.g., IP host addresses) that were originally omitted on purpose. Bringing node identifiers back to the network architecture will eventually lead us to the problems that CCN meant to avoid in the first place. Third, nodes cannot drop unwanted traffic since they no longer track which Data have been requested. And fourth, the complete removal of forwarding state from routers does not allow offering adaptive forwarding functionality.

3.2 Persistent Interests

An alternative way to maintain the stateful nature of CCN's forwarding while relaxing the PIT state requirements is through *Persistent Interests* (PIs) [77]. With PIs, users send Interests in order to receive *channels* of streaming Data packets which are grouped by a *prefix* of the content-name. That is, instead of requesting each packet individually, a user may issue a Persistent Interest for receiving all Data packets under a certain content-name prefix. In contrast to plain Interests, CCN routers store PIs in their PIT for a period of time. Since a PI matches a number of Data packets, PIs are not deleted after a matching Data packet is forwarded; instead, they remain in the PIT until users explicitly unsubscribe from the channel or the PI's lifetime expires. Users may issue PIs periodically so that state in routers is refreshed, otherwise PIs are discarded as stale.

In the data plane, each Data packet in a channel still has its own name, in order to distinguish Data packets, but they all share a common prefix which serves as the the channel name. For example, if a channel is named *"SportsTV"*, its Data packets could be called *"SportsTV/Packet1"*, *"SportsTV/Packet2"* and so on. Channel Data packets must be specially marked, so that forwarding can be performed on the channel name, in order to also match PIs. When CCN routers receive Data packets belonging to channels, they extract the channel name from the packet name and search their PIT for a matching PI. Once a match is found, the Data packet is forwarded and the PI is kept in place.

PIs can be particularly helpful in real-time streaming content. An appli-

cation can choose a name for the channel and advertise it to users which issue the respective PIs. In practice, the name must be carefully selected so that CCN routers will propagate the PI to the right content provider. If multiple users send PIs for the same channel, CCN can implicitly group all users into a single multicast tree and forward the same data packets simultaneously to the subscribed users.

With PIs, the amount of Interests transmitted may be significantly reduced. However the lifetime of a PI inside the PIT is much longer than that of a plain Interest, thus it is unclear whether the PIT size is effectively reduced. The amount of PIT state required for PIs is proportional to the number of active channel sessions (regardless of unicast or multicast delivery). Essentially, the channel-based model of PIs is the same as in IP multicast data forwarding, therefore scalability constraints with respect to active sessions are still present [46].

3.3 Semi-stateless packet forwarding

In this chapter we present a forwarding scheme for CCN that combines stateful and stateless forwarding, so as to reduce the resource requirements of routers, without losing its advantages, i.e., multicast delivery, *address-less* hosts, detection of unwanted traffic and support for adaptive forwarding. Instead of storing forwarding state per Interest in either *all* or *none* of the routers, as in plain CCN and CONET, respectively, we propose tracking Interests at *some* of the on-path routers and using a mix of stateful (in-router) and Bloom filter-based stateless (in-packet) forwarding.

During Interest propagation, instead of updating the PIT at each router, the Interest is tracked at every d hops, where d is a predefined system parameter, e.g., $d = 3$ or $d = 4$. We call d the *Forwarding State Reduction Factor*. Intermediate routers add reverse path information inside Interests. When a router tracks an Interest, instead of storing the Interest's incoming interface, the router stores the reverse path (or tree) gathered by the Interest. During Data forwarding, routers that tracked a particular Interest place the source-route for the downstream path (tree) in the Data packet and push it towards the next stateful router(s). Between

stateful points, packets are forwarded according to the in-packet source-route.

Our solution reduces forwarding state requirements, while preserving the desired properties of CCN's forwarding. Specifically, native multicast and host anonymity are preserved due to the adoption of Bloom filter-based forwarding for source-routing, while dismissal of unwanted traffic and adaptive forwarding is still supported for the fraction of Interests that each router is tracking. Though the latter two are supported in a more coarse manner, our approach compares favourably to either fully stateless forwarding solutions [57] or Interest compression schemes that drop fine-grained forwarding information [76]. Our forwarding scheme consists of two logical parts: (i) updating the PIT upon the arrival of an Interest and (ii) tracking reverse path information in Interests in order to construct Bloom filter source routes which are then used in Data forwarding. We elaborate on these below.

3.3.1 Interest tracking

The main idea in our approach is to track an Interest at every d hops. We describe three *Interest tracking policies* that can achieve this.

Probabilistic tracking

In this policy, a router decides to track every incoming Interest with probability $1/d$. For example, when $d = 4$, an Interest is tracked with probability $\frac{1}{4} = 0.25$, thus routers track on average 25% of the received Interests. The probabilistic decision alone is not sufficient for two reasons. First, it obstructs the aggregation of Interests at common on-path routers in multicast applications. When an Interest reaches a router, the probability of *not* storing the Interest is $\left(\frac{d-1}{d}\right)$. Considering that $d \geq 2$, when a new Interest reaches a router where an Interest for the same content has already been stored, it is more probable that the new Interest will not be stored there, therefore it will not be suppressed. To avoid this, upon receiving an Interest, routers first check their PIT and proceed with the probabilistic decision only if no match is found; if a match is found, the Interest is merged into the PIT and suppressed. The second inefficiency regards the prob-

Algorithm 1 Probabilistic Interest tracking policy.

```
1: procedure PROB_TRACKING(interest, incoming_port)
2:     pit_entry := PIT_lookup(interest)
3:     if pit_entry not null then
4:         store_in_PIT(interest, incoming_port)
5:         return                                   ▷ Interest suppressed
6:     end if
7:     rand := random()
8:     if rand ≤ (1/d) then
9:         store_in_PIT(interest, incoming_port)
10:    end if
11:    out_port := FIB_lookup(name)
12:    forward(interest, out_port)
13: end procedure
```

ability of not tracking an Interest at all. In a N-hop path, the probability of *not* storing an Interest at any intermediate hop[2] is $\left(\frac{d-1}{d}\right)^N$. When $d = 4$ and $N = 8$, the probability of not storing an Interest at any router is $0.75^8 \approx 0.1$, i.e., 10% of the Interests issued by that particular application are source-routed on an end-to-end basis. As we will discuss later in Section 3.3.2, this has a negative impact on the stateless forwarding part of our protocol due to *false positives* in Bloom filters: when the amount of source-routing information exceeds a limit, the Bloom filter-based degenerates and causes excessive redundant traffic [47]. Algorithm 1 shows the detailed algorithm for the Probabilistic tracking policy.

Hash-based tracking

This policy tackles the *rendezvous* inefficiency of Probabilistic tracking, i.e., aggregating Interests for the same Data in common on-path routers, with the use of a hash function. When router i receives an Interest, it performs the following

[2]We assume a unicast path, i.e., the same Interest has not previously crossed part (or all) of the path.

computation

$$\nu = hash(content\_name + suffix_i) \bmod d \tag{3.1}$$

If $\nu = 0$, we say that the Interest made a *rendezvous* at this node and the router stores it in the PIT. Note that we hash the content name appended with a router-specific $suffix_i$, in order to produce a different ν for the same Interest at each hop. If we did not use the suffix, hashing the content name alone would produce the same result at all routers, thus the Interest would either be stored in all routers (if $\nu = 0$) or in none (if $\nu \neq 0$). If the selected hash function has good uniformity properties, ν is uniformly distributed in $[0, d-1]$ and Interests are stored at each router with probability $1/d$, as desired. If all routers use the same hash function (e.g., MD5) and each router uses a fixed $suffix_i$ (e.g., a *uuid* initialized during node bootstrap), then multiple Interests for the same Data will *deterministically* rendezvous at the exact same routers.[3] Hence, routers will be able to suppress Interests without performing a PIT lookup. The cost of this policy is the hash computation at each router. In addition, there is a chance that an Interest may not be stored at any router, as in the Probabilistic tracking policy. Algorithm 2 shows the detailed algorithm for the Probabilistic tracking policy.

Hop Counter-based tracking

In this policy, routers track Interests based on the value of a *Hop Counter* (HC) inside the Interest header. The HC is incremented at each hop and when $HC = d$, routers store the Interest in their PIT. Routers always perform an initial PIT lookup in order to suppress multicast Interests regardless of the HC value, as in Probabilistic tracking. The initial value for the HC is set by the issuing host but instead of setting it to 0, the initial HC is randomly selected in the range $[0, d-1]$ so as to distribute forwarding state to all routers. If the initial HC was always set to 0, routers with distance $d-1$ from hosts would be kept stateless. In the example of Figure 2.1, if $d = 3$, all Interests issued by hosts would be tracked by R_4, while R_1 to R_3 would have an empty PIT. PIT state would thus be unevenly distributed: R_4 would be a bottleneck point and R_1 to R_3 would not participate

[3]Note that the suffix is not communicated, thus host anonymity is preserved.

Algorithm 2 Hash-based Interest tracking policy.

```
1: procedure HASH_TRACKING(interest, incoming_port)
2:     suppress_interest := false
3:     ν = hash(content_name + suffix_i) mod d
4:     if ν = 0 then
5:         suppress_interest := (PIT_lookup(interest) ≠ null)
6:         store_in_PIT(interest, incoming_port)
7:     end if
8:     if not suppress_interest then
9:         out_port := FIB_lookup(name)
10:        forward(interest, out_port)
11:    end if
12: end procedure
```

in (say) adaptive forwarding at all. In contrast, with a randomly selected HC, there is a $1/d$ probability for each on-path router to track the Interest. This policy guarantees that an Interest will always be stored at most after d routers, thus avoiding the inflation of false positives due to overflowing iBFs, which may occur in the Probabilistic and Hash-Based policies. Algorithm 3 shows the detailed algorithm for the Hop Counter-based tracking policy.

3.3.2 Semi-stateless data forwarding

We now describe how semi-stateless packet forwarding can be incorporated into CCN, without sacrificing the native multicast and node anonymity of CCN. In our scheme, the network supports multicast efficiently, even though the forwarding state may be stored in non-branching points of the multicast tree. That is, the source-routing scheme duplicates data *only* at branching points, even when the source-route is maintained elsewhere, without resorting to multiple unicast transmissions and without re-introducing host addresses to CCN.

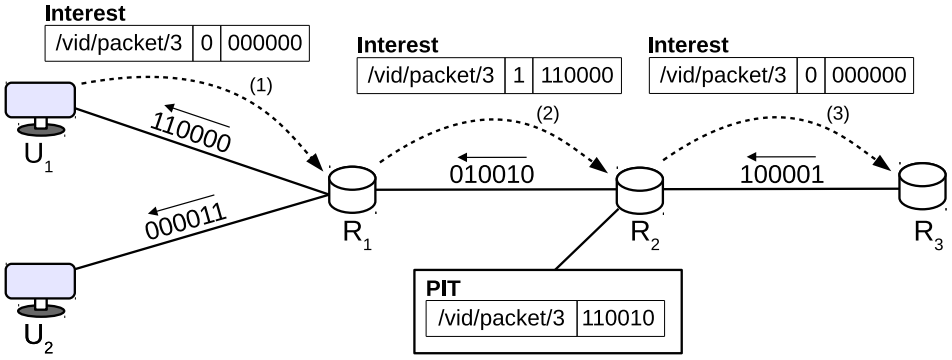
To integrate Bloom filter-based forwarding in CCN, we extend Interest and Data packets to carry an IBF in their headers. Interest packets accumulate the IBF

Algorithm 3 Hop Counter-based Interest tracking policy.

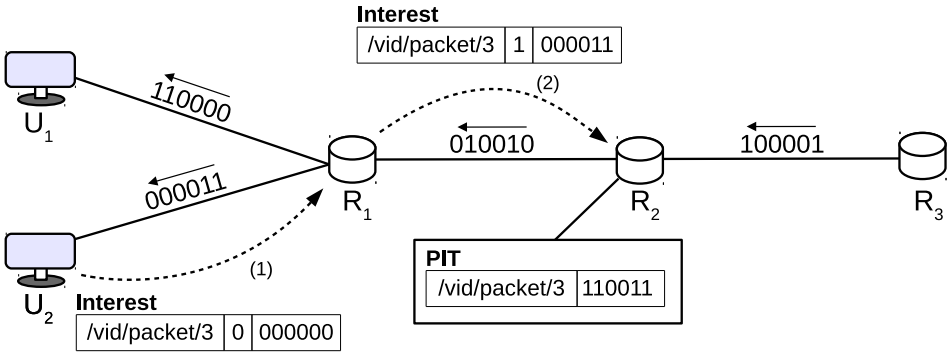
```
1: procedure HOP_COUNTER_TRACKING(interest, incoming_port)
2:     pit_entry := PIT_lookup(interest)
3:     if pit_entry not null then
4:         store_in_PIT(interest, incoming_port)
5:         return                                  ▷ Interest suppressed
6:     end if
7:     hc := increment_hop_counter(interest)
8:     if hc = d then
9:         store_in_PIT(interest, incoming_port)
10:        reset_hop_counter(interest)
11:    end if
12:    out_port := FIB_lookup(name)
13:    forward(interest, out_port)
14: end procedure
```

for the traversed (reverse) path and Data packets carry the IBF for the delivery path (or tree). Specifically, upon receiving an Interest, routers update the Interest's traversed path by adding (ORing) the *outgoing* LID of the packet's incoming link. If a router decides to store an Interest in its PIT, it also stores the IBF and then *resets* the IBF in the Interest before further forwarding it. When the respective Data packet arrives, the router acts as a relay point by inserting the stored IBF in the Data packet and then forwarding it based on the IBF.

Figure 3.2(a) shows an example where $d = 2$, U_1 and U_2 are two multicast users and the network uses Hop Counter-based tracking. At some point, U_1 requests the packet `/vid/packet/3` with initial $HC = 0$. U_1 creates the Interest with an *empty* IBF (i.e., all bits are set to 0) and transmits the packet. R_1 receives the Interest, increases the HC and adds the LID for the reverse direction, i.e., $LID_{R_1 \to U_1}$, to the Interest IBF (step 1). R_1 forwards the Interest to R_2. Node R_2 increases the HC to 2 and adds $LID_{R_2 \to R_1}$ to the Interest IBF (now containing the path $R_2 \to R_1 \to U_1$). Since $HC = 2$ $(= d)$, R_2 stores the Interest along with the IBF in its PIT (step 2). R_2 then resets the Interest's HC and IBF and forwards



(a) Interest from U_1: R_1 updates the Interest iBF. R_2 tracks the Interest in its PIT with the iBF for $R_2 \rightarrow R_1 \rightarrow U_1$. R_2 resets the Interest iBF to 0 and further forwards the Interest.



(b) Interest from U_2: R_1 updates the Interest iBF. R_2 adds the iBF to its *existing* PIT entry and suppresses the Interest. The stored iBF contains the multicast tree to U_1 and U_2.

Figure 3.2: Interest propagation using IBFs to track reverse paths. Only right-to-left LIDs are shown.

the Interest (step 3). This continues until the Interest reaches the data source.

During Interest forwarding, if a router finds a matching PIT entry, it adds (OR) the Interest's IBF to the IBF already stored in the PIT. The resulting IBF is the union of the already stored and the additional path links, which form a multicast tree. This is shown in Figure 3.2(b). U_2 transmits an Interest for the same content as U_1 did, with initial $HC = 0$. The request arrives at R_1 which updates the Interest's HC and IBF (step 1) and forwards the Interest to R_2. At that point, R_2 updates the Interest's IBF, adds it to the IBF already stored in the PIT and suppresses the Interest (step 2). The PIT entry at R_2 now contains the

Algorithm 4 Semi-stateless forwarding.

```
1: procedure SEMI_STATELESS_FORWARDING(data_packet, incoming_port)
2:     pit_entry := PIT_lookup(data_packet.name)
3:     if pit_entry not null then
4:         data_packet.IBF := pit_entry.IBF
5:     end if
6:     IBF_FORWARD(data_packet, incoming_port)
7: end procedure
8: procedure IBF_FORWARD(data_packet, incoming_port)
9:     for port in (ports − incoming_port) do
10:        lid_port := link_id(port)
11:        if data_packet.IBF AND lid_port = lid_port then
12:            transmit(data_packet, port)
13:        end if
14:    end for
15: end procedure
```

IBF for the multicast tree $R_2 \rightarrow R_1 \rightarrow \{U_1, U_2\}$.

Upon the arrival of a Data packet, a router checks its PIT and if a matching entry exists, it replaces the Data IBF with the stored IBF and further forwards the packet. If no PIT entry exists, the router forwards the Data packet according to its IBF. If no LID matches the Data packet's IBF, the router drops the packet. Finishing the example of Figure 3.2, when the Data packet `/vid/packet/3` arrives at R_2 , the router replaces the Data IBF with the one stored in the PIT. The IBF now contains $LID_{R_2 \rightarrow R_1}$, $LID_{R_1 \rightarrow U_1}$ and $LID_{R_1 \rightarrow U_2}$, therefore the packet is delivered to R_1 which then duplicates the Data packet to U_1 and U_2. It is important to note that even though the IBF is stored at a non-branching router (R_2), Bloom filter-based forwarding ensures that the Data packets are only duplicated at branching nodes (R_1). Algorithm 4 shows the router's data plane logic upon the reception of a Data packet.

3.3.3 Architectural considerations

Our semi-stateless forwarding scheme requires slight changes in the CCN architecture. Apart from the modified Interest and Data handling operations, the incorporation of Bloom filter-based forwarding does not affect the architecture's control plane. That is, there is no need for any additional routing information exchange. Routers only need to know their own outgoing LIDs, which can be autonomously computed, e.g., by Double Hashing [48] the MAC address of each network interface during node bootstrap. There is no need to coordinate LID assignment, as LIDs do not need to be globally unique. In addition, source-routes are constructed in a distributed manner and no separate centralized module is required to construct the Bloom filters [78]. Nodes also remain anonymous, as in CCN. Security is not downgraded, as due to the Bloom filter-based and source-specific representation of the source-routes, it is very difficult to perform targeted attacks to nodes. Hosts are unaware of router LIDs and it is highly improbable that a host can *guess* a valid IBF to attack a particular node [47]. Content sources obtain valid IBFs only when Interests arrive at them, but they have no idea where these IBFs lead and they rarely obtain an IBFs for an entire end-to-end path.[4]

3.3.4 Performance trade-offs

The reduction of forwarding state achieved by our semi-stateless approach introduces some performance overheads, thus there are performance trade-offs to be made, which are analyzed in this section.

The first overhead is a possible increase in the Interests transmitted when multicast trees are created. This happens because Interests are not necessarily aggregated at the *first* common router of the multicast tree. For example, assume that U_1 and U_2 in Figure 3.2 consume the same content. If $d = 3$, their Interests will be aggregated at either $R1$, $R2$ or R_3, although R_1 is the nearest common point. When Interests are aggregated at R_2, an additional Interest is transmitted, compared to CCN. For Interests that rendezvous at R_3, two additional Interests are

[4]The path has to be smaller than d hops (minus the initial value of HC in Hop Counter-based tracking).

Content name	Out-ports
/a/b/c/1	1,2
/a/b/c/2	1,2
/a/b/c/3	1,2
/a/b/c/4	1,2

(a) Basic CCN PIT

Content name	Downstream IBF
/a/b/c/1	1000101110100101
/a/b/c/3	1000101110100101

(b) CCN PIT with IBFs

Figure 3.3: In semi-stateless forwarding, the PIT contains fewer entries but each entry occupies more memory due to the IBF size.

transmitted. This is a penalty our scheme pays in order to reduce forwarding state in routers. The amount of additional Interests depends on the *Forwarding State Reduction Factor* d. For larger values of d, forwarding state is further reduced, but Interests may be suppressed further away than the first common on-path router. The overhead caused by additional Interests is also subject to the group size and the density of the multicast tree. When the multicast tree is sparse, Interests are rarely aggregated anyway, thus few additional Interests can be transmitted. Note that this penalty affects multicast delivery only; with unicast, there is no Interest aggregation anyway. We further discuss the impact of increased Interest transmissions in Section 3.4.

The second overhead is that all packets (both Interests and Data) must carry IBFs, therefore bandwidth consumption is increased due to the extra field in packet headers. The third overhead is that the PIT now stores IBFs instead of interface ports, as shown in Figure 3.3. While IBFs are typically 128-256 bits long [47], up to x ports can be encoded with an x-bit mask, e.g., 32-bits for 32 ports.[5] Hence, the actual memory reduction of the PIT is not equivalent to the reduction of PIT entries, e.g., a 50% reduction of the PIT entries does not lead to a

[5]Note that a full port mask, rather than a port number, is needed at the PIT in order to support multicasting.

50% reduction in the actual memory footprint. Finally, the fourth overhead is due to the fact that IBF-based forwarding is susceptible to false forwarding decisions which cause redundant traffic, especially as more LIDs are added to the IBF. The scale of this overhead depends on the size of the multicast group and the value of d.[6] We elaborate on these trade-offs in Section 3.4.

3.4 Evaluation

3.4.1 Simulation setup

We evaluated the effectiveness of our approach through simulations, using synthetic scale-free graphs generated with the Barabási-Albert algorithm [79] and ISP topologies obtained from Rocketfuel [80] and the Internet Topology Zoo [81]. Table 3.1 shows the graph characteristics of the tested topologies.

Table 3.1: Graph characteristics of topologies used in experiments.

Topology	Nodes	Access Nodes	Links	Diameter	Avg (Max) degree
AS-20965	40	8	61	8	3 (10)
AS-224	74	15	101	9	2.7 (8)
AS-3967	79	7	147	10	3.7 (12)
AS-1755	87	10	161	11	3.7 (11)
AS-1221	104	51	151	8	2.9 (18)
AS-6461	138	9	372	8	5.4 (20)
scale-free-50	50	24	62	6	2.5 (18)
scale-free-100	100	46	133	7	2.6 (33)

For each tested topology, we considered the graph to represent the core backbone network of a Content-Delivery Network (CDN) provider. Nodes with a single link ($degree = 1$) are considered to be access routers/gateways providing access to local ISPs. We attached 5 additional graph nodes in each access router, each one representing the aggregate demand generated by the ISP. In our experiments, IBFs are 16 bytes long ($m = 128$). LIDs are computed autonomously by

[6] With unicast paths, the probability of false forwarding decisions is negligible for the values of d considered.

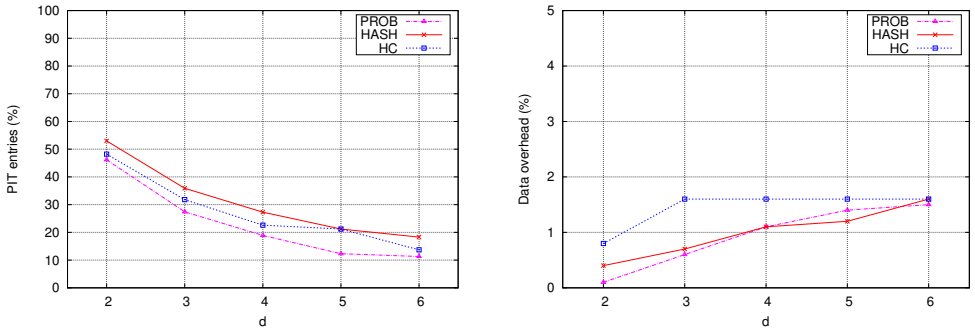
each router: a router i sets $k_i = \lceil log_2(deg_i - 1) \rceil$ as in [49], where deg_i is the degree of node i and then uses Double Hashing [48] for generating the k_i hash functions. In all tests, routing information in routers (FIBs) was pre-populated, allowing Interests to reach content sources over the shortest paths, with hop count as the routing metric.

We tested our scheme with both unicast and multicast deliveries using the same application set-up: an HTTP-like live streaming application [82]. A streaming server generates data chunks at a constant rate and clients request Data packets in a Stop-and-Wait fashion. For unicast scenarios, a single client downloads the content, while for multicast delivery multiple clients download the live stream at the same time. Both server and client(s) are located at the edges of the graph, i.e., inside regional ISPs, and they are randomly selected in each experiment. Experiments last until 1000 Data packets are delivered, which may represent a live video streaming session with duration ranging from 2000 seconds (≈30 minutes) if each chunk has a 2-second duration as in Microsoft Smooth Streaming, to 15000 seconds (≈4 hours) if each chunk has a 15-seconds duration as in Apple HTTP Streaming [83].

For each tested topology t, we ran our application from 1 and up to C_t stream receivers, where C_t is the number of edge nodes (the regional ISPs) that are attached to the core network's access nodes. Each experiment was repeated 20 times, changing the random generator's seed, thus selecting different server and client(s).

3.4.2 Evaluation metrics

Our evaluation focused on the reduction of forwarding state in routers in terms of (a) the number of PIT entries and (b) the actual memory consumed by the PIT. For the memory footprint, we considered a hash table-based implementation which is reported to be the most suitable data structure for the PIT [75]. We assumed that memory pointers are 32 bits and that the interface ports in basic CCN were encoded with 32-bit masks. For the size of content names, we adopted the real-world measurements of [74] which reported two sizes: small content names



(a) PIT entries (b) Data overhead

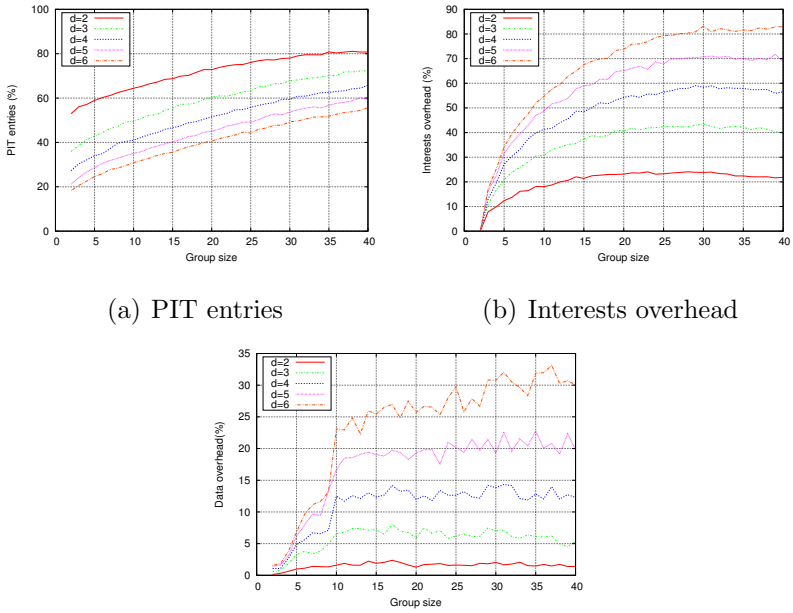
Figure 3.4: Performance results for unicast traffic in topology AS-20965 as a function of d for all Interest tracking policies: (i) Probabilistic (PROB), (ii) Hash-based (HASH) and Hop Counter-based (HC).

that are on average 20 bytes and large content names which are on average 56 bytes. Apart from the reduction of forwarding state, we also measured the bandwidth overhead due to (a) additional Interests caused by not storing PIT entries at the first common router on the multicast tree and (b) redundant Data caused by false positives in the Bloom filters. Note that additional Interests apply in the case of multicast delivery only; there is no Interest aggregation in unicast delivery. On the other hand, redundant Data transmissions are caused by false positives in Bloom filters, which may occur in both unicast and multicast deliveries.

In all results presented in the following sections, results are normalized against the basic CCN behaviour which serves as the performance baseline.

3.4.3 Unicast

Figure 3.4 shows the performance for unicast flows in topology AS-20965 for the three Interest tracking policies. Figure 3.4(a) shows the average PIT state in each router normalized against baseline CCN as a function of d. PIT state is reduced to $\frac{1}{d}$ as expected, with slight differences between the three policies. That is, for $d = 3$, the PIT now stores approximately $1/d = 33\%$ of the entries compared to basic CCN. Figure 3.4(b) shows the amount of additional Data packets transmitted, caused by false positives in the Bloom filters with respect to d. While



(a) PIT entries

(b) Interests overhead

(c) Data overhead

Figure 3.5: Probabilistic-based Interest tracking in topology AS-20965 for various values of d with respect to the multicast group size.

some differences can be observed between the three Interest tracking policies, the amount of Data overhead remains very low, below 1.75% in the worst case. Figures A.1 to A.7 in Appendix A show unicast results for PIT entries and Data overhead for all tested topologies.

3.4.4 Multicast

We now examine the behaviour of the system in multicast scenarios. We first present the performance of the system with respect to the multicast group size for various values of the *Forwarding State Reduction Factor* d and then examine the system behaviour under uniform and Zipf distributions of the multicast group sizes.

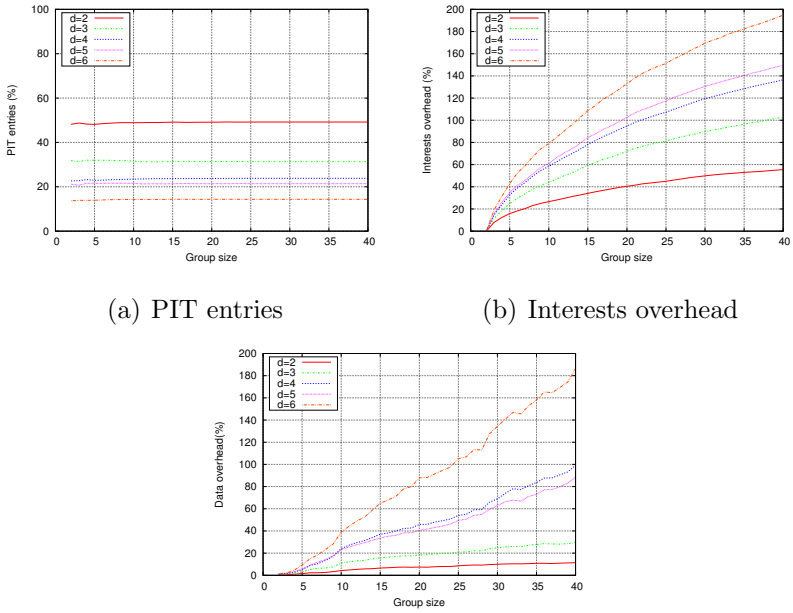
Performance with respect to multicast group size

Figure 3.5 shows system performance with Probabilistic Interest (PROB) tracking in topology AS-20965 with respect to the multicast group size. Fig-

ure 3.5(a) shows that PROB works best with smaller groups; as group size grows, the PIT state grows as well, thus reducing the effectiveness of the scheme. On the other hand, the fraction of additional Interests grows with group size up to a certain threshold, after which it remains stable, as shown in Figure 3.5(b). As described in Section 3.3.4, Interests may not be suppressed at the first common router. This is a minor problem for very small groups where the multicast trees are relatively sparse and there exist only a few aggregation points. As groups grow and the multicast tree becomes denser, the possibility of not storing state at aggregation points increases, and so do the additional Interests. As more users issue Interests for the same Data, the probability of aggregating an Interest closer to the first branching router increases, therefore, once the group size passes a threshold (around 20 receivers in AS-20965), the amount of additional Interests stops increasing. A similar pattern is observed for the redundant Data, as shown in Figure 3.5(c). For very small group sizes, the multicast trees have very few aggregation points that could lead to redundant Data. As the group size grows, larger trees are encoded in IBFs since Interests are not suppressed at the first branching router, increasing the number of false positives. Beyond a group size threshold, the system shows a steady behaviour, as in the Interests case.

Figures 3.6 and 3.7 show the results for Hash-based (HASH) and Hop Counter-based (HC) Interest tracking in topology AS-20965 respectively. The behaviour of these policies is similar, but different than the behaviour of PROB. They both manage to obtain a steady reduction of the PIT size regardless of the group size but show an increasing number of additional Interests and Data packets as the group size grows. HASH and HC offered better PIT reduction compared to PROB, at the cost of increasing the Interests and Data overhead.

Overall, there is a common performance pattern for all Interest tracking polices: semi-stateless forwarding is more effective for small groups. It causes less bandwidth overhead in terms of redundant Interests and Data packets compared to baseline CCN and, in addition, PROB reduces the PIT more effectively for small groups. On the other hand, HASH and HC perform better than PROB, with HC producing less overheads compared to HASH. In the following sections



(a) PIT entries

(b) Interests overhead

(c) Data overhead

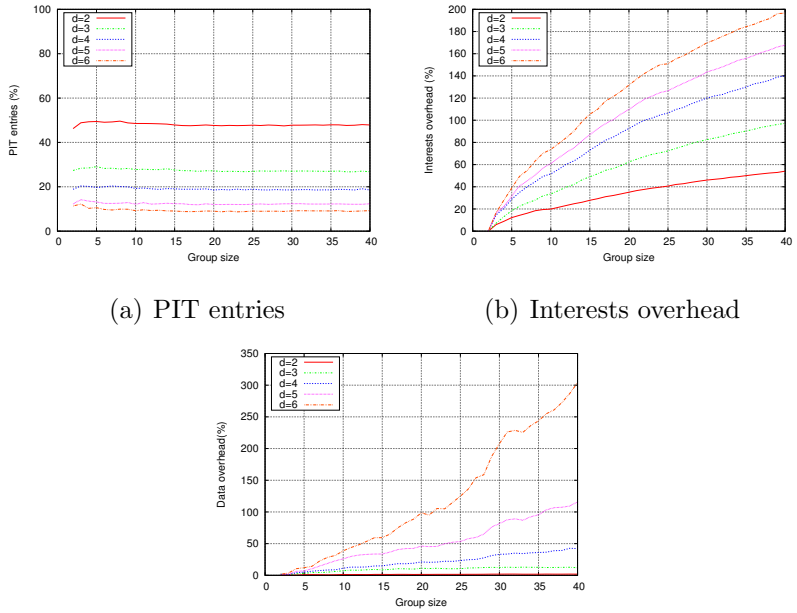
Figure 3.6: Hash-based Interest tracking in topology AS-20965 for various values of d with respect to the multicast group size.

we examine how the distribution of the multicast group sizes affects the overall system performance.

Uniform distribution of group size

In this section we present results for our scheme's performance when multicast group size follows a uniform distribution. In this case, we computed the system's average behaviour for all multicast groups as described in Section 3.4.1: For each topology t, we took into account all experiments of multicast groups with $[3, C_t]$ participants,[7] where C_t is the number of edge nodes (the regional ISPs) that are attached to the core network's access nodes. Recall that, for each group size, each experiment was repeated 20 times changing our random generator's seed, thus each time choosing different sender and receivers. Overall, for each topology t, we take into account the average system behaviour of $20 * (C_t - 2)$ experiments and the group size is uniformly distributed in $[3, C_t]$.

[7]The smallest multicast group consists of 1 sender and 2 receivers while the largest multicast group consists of 1 sender and $C_t - 1$ receivers.



(a) PIT entries

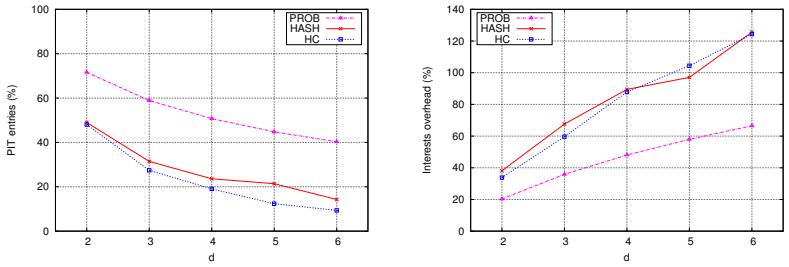
(b) Interests overhead

(c) Data overhead

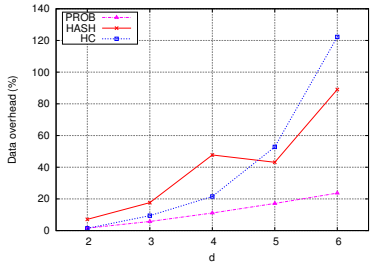
Figure 3.7: Hop Counter-based Interest tracking in topology AS-20965 for various values of d with respect to the multicast group size.

Figure 3.8 shows the performance results for topology AS-20965 for all Interest tracking policies with respect to d normalized against basic CCN. Figure 3.8(a) shows that, on average, the PIT state is reduced as d grows. For HASH and HC, the average PIT state is reduced nearly to $1/d$, although PROB did not manage to achieve this. In terms of PIT reduction, HC performs better. For $d = 3$, HC reduces the average PIT state to 27% compared to basic CCN, at the cost of 60% additional Interests transmitted and 9% additional Data packets. For $d = 4$, the average is reduced to 19% compared to the basic CCN PIT at the high cost of 89% of additional Interests and 22% of additional Data. Our results show that HC is the most effective policy. Together with HASH, it reduces the PIT far more effectively than PROB, but it also produces slightly lower overheads than HASH.

Results for all topologies are presented in Figures A.8 to A.14 in Appendix A. Figure 3.9 shows the results for HC, the best performing policy, in all tested topologies for $d = 3$ and $d = 4$ and Table 3.2 summarizes all the numerical data. In order to effectively reduce the PIT size (setting $d \geq 3$), it is clear



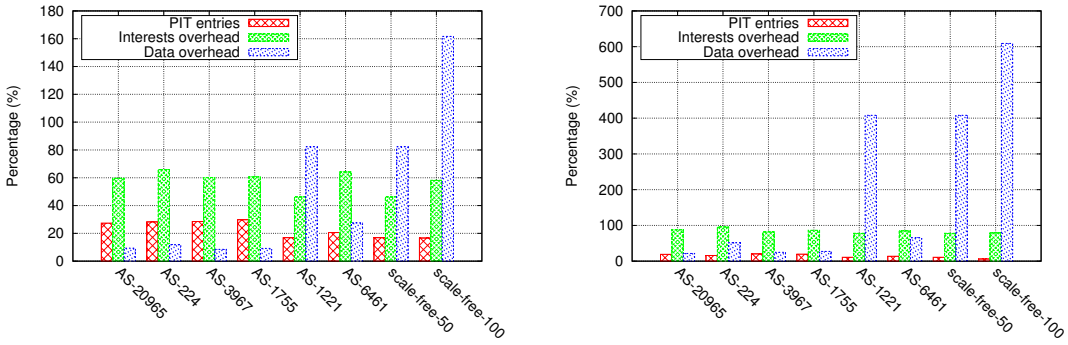
(a) PIT entries (b) Interests overhead



(c) Data overhead

Figure 3.8: Multicast performance results for uniform distribution of group size in topology AS-20965 for all Interest tracking policies with respect to d: (i) Probabilistic (PROB), (ii) Hash-based (HASH) and (iii) Hop Counter-based (HC).

that the scheme induces a performance penalty in terms of bandwidth overhead. When the group size follows a uniform distribution, the costs can be very high. Especially in topologies *AS-1221*, *scale-free-50* and *scale-free-100*, which are the largest topologies tested, particularly regarding the number of access nodes, these costs are prohibitive.



(a) $d = 3$ (b) $d = 4$

Figure 3.9: PIT entries, Interests overhead and Data overhead for all topologies, uniform distribution of group sizes, with Hop Counter-based Interest tracking.

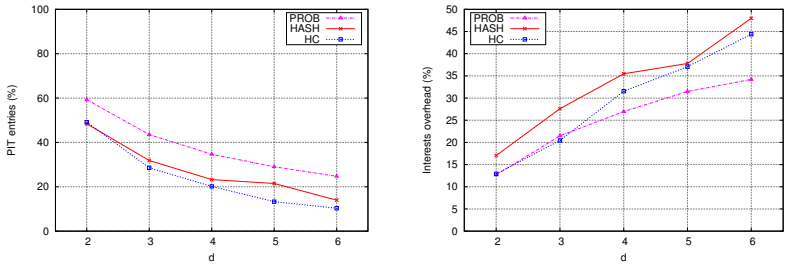
Table 3.2: PIT entries, Interests overhead and Data overhead for all topologies, uniform distribution of group sizes, with Hop Counter-based Interest tracking.

(a) $d = 3$

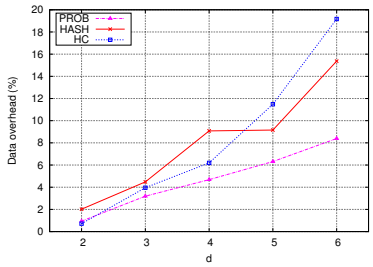
Topology	PIT entries (%)	Interests overhead (%)	Data overhead (%)
AS-20965	27.4	59.7	9.4
AS-224	28.2	65.8	11.8
AS-3967	28.5	60.2	8.5
AS-1755	29.8	60.5	9.2
AS-1221	16.9	46.2	82.3
AS-6461	20.6	64.1	27.5
scale-free-50	16.9	46.2	82.3
scale-free-100	16.6	58.2	161.7

(b) $d = 4$

Topology	PIT entries (%)	Interests overhead (%)	Data overhead (%)
AS-20965	19	87.8	21.5
AS-224	15.7	95.5	51.4
AS-3967	20.3	81.9	24.9
AS-1755	19.7	86.3	27.3
AS-1221	10.8	78	408.1
AS-6461	13.8	84.5	65.4
scale-free-50	10.8	78.0	408.1
scale-free-100	6	78.8	609.2



(a) PIT entries (b) Interests overhead



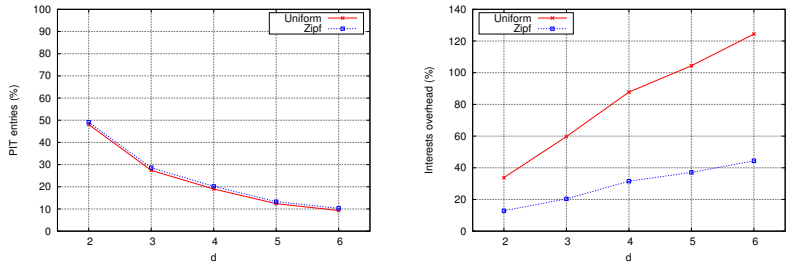
(c) Data overhead

Figure 3.10: Multicast performance results for Zipf distribution of group size in topology AS-20965 for all Interest tracking policies with respect to d: (i) Probabilistic (PROB), (ii) Hash-based (HASH) and (iii) Hop Counter-based (HC).

Zipf distribution of group size

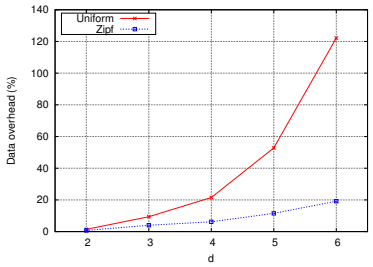
In this section we present performance results when the multicast group size follows a Zipf-like distribution, which is a more realistic scenario [84]. We expect the Zipf distribution to benefit our scheme's performance as the majority of groups will be small-sized. For the Zipf distribution, we follow the methodology of [85]: for each tested topology t, the size of the i^{th} group is $group\_size(t,i) = \lfloor i^{-\alpha_t} * C_t + 0.5 \rfloor$ where C_t is the maximum group size in topology t which depends on the network's number of edge nodes (ISPs), and α_t is selected in each topology so that the smaller multicast group size is 3 (one server and two receivers). As in the previous sections, all results are normalized against baseline CCN.

Figure 3.10 shows the performance results for topology AS-20965 for all Interest tracking policies with respect to d normalized against basic CCN. The performance shows the same pattern as in the uniform distribution (shown in Fig. 3.8) however the results are much more encouraging. Again, HC and HASH



(a) PIT entries

(b) Interests overhead



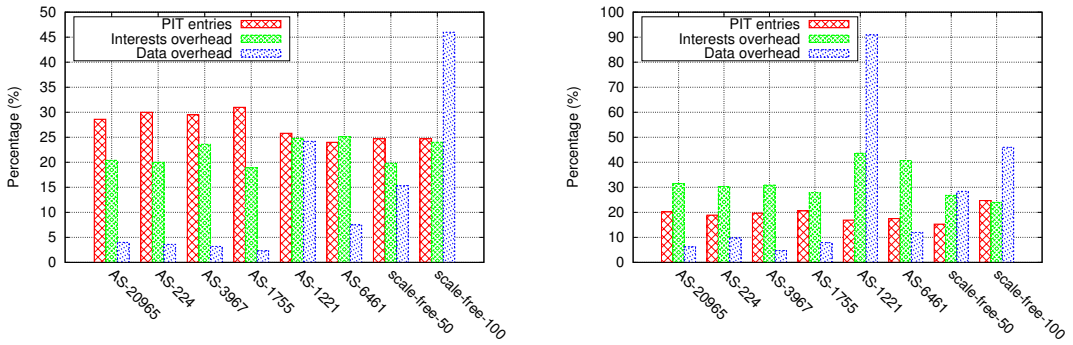
(c) Data overhead

Figure 3.11: Multicast performance results HC policy in AS-20965 for uniform and Zipf group size distributions.

are more effective with regards to PIT state reduction compared to PROB, with HC performing slightly better. Furthermore, HC introduces fewer costs compared to HASH (Interests and Data overhead) specifically when $d = 3$ and $d = 4$, in which overall costs seem manageable.

As in the uniform distribution case, HC is the overall best-performing Interest tracking policy. Apart from its performance, it is the most simple to implement: it only requires a counter in each Interest. With HC we can avoid the expensive hash computations of HASH and reduce the PIT more effectively compared to PROB. We, therefore, choose to focus on HC for remainder of the dissertation. With HC, when $d = 3$, PIT entries are on average reduced to 29% at the cost of 20% additional Interests and 4% additional Data transmitted. For $d = 4$, the PIT is on average reduced to 20% at the cost of 32% additional Interests and 6% additional Data transmitted.

To better understand the effect of group size distribution, Figure 3.11 compares the scheme's performance in the HC policy in the uniform and Zipf distribu-



(a) $d = 3$ (b) $d = 4$

Figure 3.12: PIT entries, Interests overhead and Data overhead for all topologies, Zipf distribution of group sizes, with Hop Counter-based Interest tracking.

tions. Although the PIT is reduced equally in both distributions, it is clear that there are significant performance differences with respect to Interests and Data overheads. The semi-stateless forwarding scheme provides significant PIT reduction gains when multicast group sizes follow a Zipf distribution with relatively manageable bandwidth overheads. If the group size distribution is not skewed, the costs are too expensive for the returned gains.

Results for all topologies are presented in Figures A.15 to A.21 in Appendix A. Figure 3.12 shows the results for the HC policy for all tested topologies for $d = 3$ and $d = 4$ and Table 3.3 summarizes all the numerical data. In order to effectively reduce the PIT size (setting $d \geq 3$), in most topologies the Data overhead is below 10% but the Interests overhead is around 30%. However, the costs in *AS-1221*, *scale-free-50* and *scale-free-100* are still too high.

Table 3.3: PIT entries, Interests overhead and Data overhead for all topologies, Zipf distribution of group sizes, with Hop Counter-based Interest tracking.

(a) $d = 3$

Topology	**PIT entries (%)**	**Interests overhead (%)**	**Data overhead (%)**
AS-20965	28.6	20.4	4
AS-224	30	20	3.6
AS-3967	29.5	23.6	3.2
AS-1755	31	18.9	2.3
AS-1221	25.8	24.8	24.2
AS-6461	24	25.1	7.5
scale-free-50	24.7	19.8	15.3
scale-free-100	24.7	24	46

(b) $d = 4$

Topology	**PIT entries (%)**	**Interests overhead (%)**	**Data overhead (%)**
AS-20965	20.2	31.5	6.2
AS-224	18.9	30.3	9.8
AS-3967	19.7	30.9	4.7
AS-1755	20.6	27.9	7.9
AS-1221	16.9	43.6	90.9
AS-6461	17.5	40.7	12
scale-free-50	15.3	26.8	28.4
scale-free-100	24.7	24	46

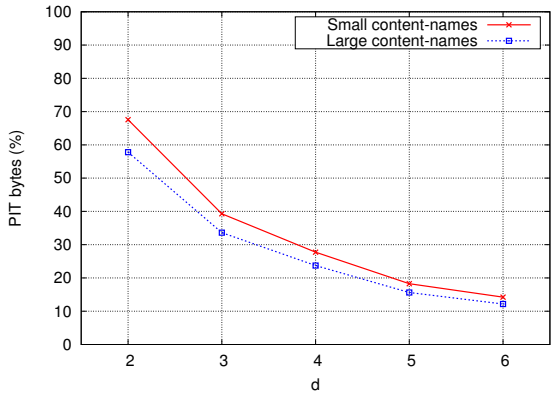
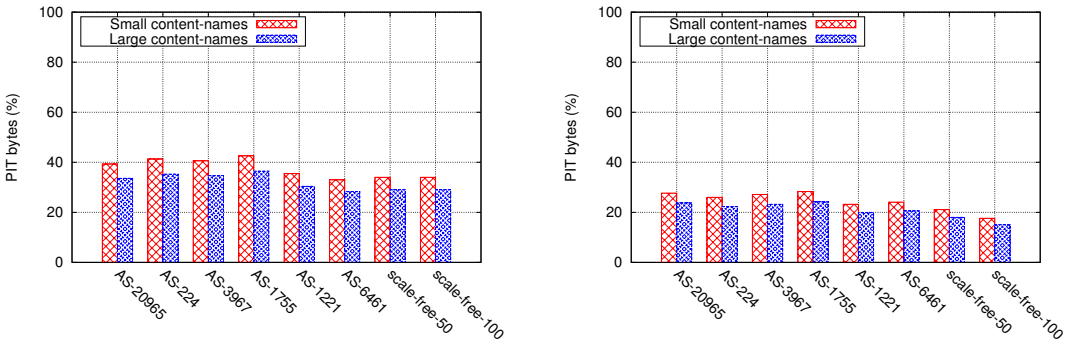


Figure 3.13: Estimated PIT size (in bytes) for small (20-bytes) and large (56-bytes) content names in topology AS-20965, Zipf distribution of group sizes using the Hop Counter-based policy.

PIT memory reduction

We now turn our focus on the actual memory size reduction of the PIT. Recall from Section 3.3.4 that our new PIT may contain fewer rows but each row occupies more memory due to the Bloom filter size. Figure 3.13 shows the size reduction of a hash table-based PIT for the HC policy as a function of d. For $d = 3$, the actual memory footprint for the PIT is reduced to 39% for small content names and 37% for large content names. For $d = 4$, the memory footprint for the PIT is reduced to 28% for small content names and 24% for large content names.

Figure A.22 in Appendix A presents the PIT memory reduction for all tested topologies. Figure 3.14 shows a summary for all topologies for $d = 3$ and $d = 4$ and Table 3.4 contains the respective the numerical data.



(a) $d = 3$ (b) $d = 4$

Figure 3.14: Estimated PIT size (in bytes) for small (20-bytes) and large (56-bytes) content names for all topologies, Zipf distribution of group sizes using the Hop Counter-based policy.

Table 3.4: Estimated PIT size (in bytes) for small (20-bytes) and large (56-bytes) content names for all topologies, Zipf distribution of group sizes using the Hop Counter-based policy.

(a) $d = 3$

Topology	**Small content-names (%)**	**Large content-names (%)**
AS-20965	39.3	33.6
AS-224	41.3	35.3
AS-3967	40.6	34.7
AS-1755	42.6	36.5
AS-1221	35.5	30.4
AS-6461	33	28.3
scale-free-50	34	29.1
scale-free-100	34	29.1

(b) $d = 4$

Topology	**Small content-names (%)**	**Large content-names (%)**
AS-20965	27.7	23.7
AS-224	26.0	22.3
AS-3967	27.1	23.2
AS-1755	28.3	24.2
AS-1221	23.2	19.9
AS-6461	24.1	20.6
scale-free-50	21.1	18.0
scale-free-100	17.6	15.1

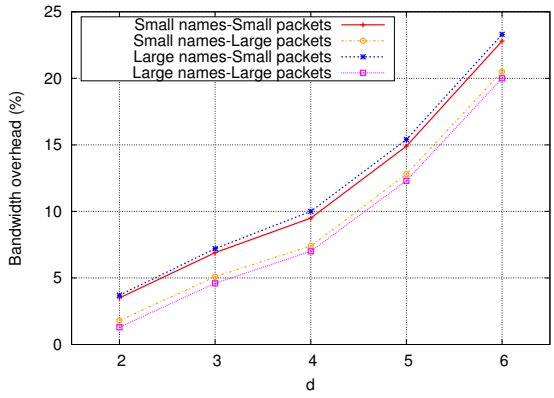
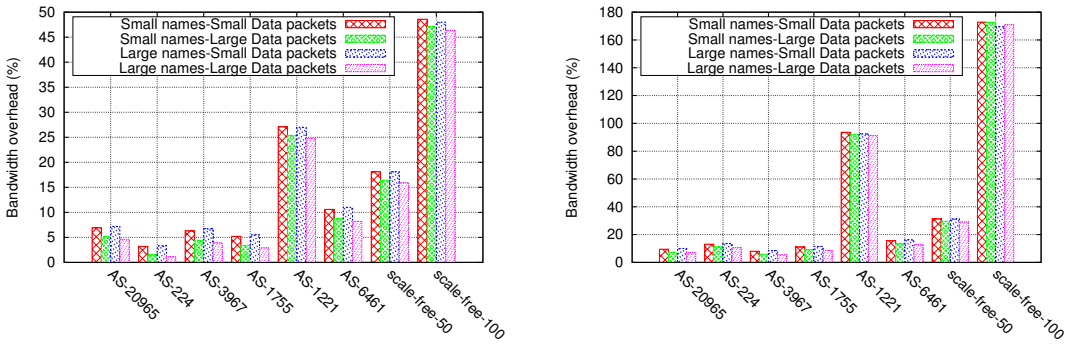


Figure 3.15: Multicast bandwidth overhead with a Zipf distribution of group sizes in the HC policy for topology AS-20965. Small content-names (36-byte Interests) and large-content names (70-byte Interests), small (1500 bytes) or large (jumbo) frames (7500 bytes) Data packets.

Bandwidth overhead

As already discussed, PIT state reductions come at the cost of additional Interest and Data transmissions. When the *Forwarding State Reduction Factor d* is increased in order to reduce the PIT size, these overheads also increase. Among these additional transmissions, most are Interests that are not ideally aggregated at the first on-path branching routers of the multicast tree, as opposed to baseline CCN. In terms of actual bandwidth, however, the more numerous Interests are much smaller than Data packets, thus the overall bandwidth overhead depends on the size of Interests relative to Data.

In Figure 3.15 we present the overall bandwidth overhead of the HC policy in topology AS-20965 in terms of the fraction of additional bytes transmitted compared to basic CCN. We assume small (20 bytes) and large (56 bytes) content names, so with the additional CCN meta-data, Interests are on average 36 and 70 bytes long respectively. We then consider two types of Data packets: a small Data packet that carries 1500 bytes of payload, targeting a CCN deployment over Ethernet, and a large Data packet that carries 7500 bytes of payload, targeting a CCN deployment over either Ethernet with jumbo frames or a UDP-based overlay. We also take into account the extra fields required by our scheme in the Interest and Data packet headers (IBF and HC). When $d = 3$, the additional bandwidth is



(a) $d = 3$ (b) $d = 4$

Figure 3.16: Multicast bandwidth overhead with a Zipf distribution of group sizes in the HC policy for all topologies. Small content-names (36-byte Interests) and large-content names (70-byte Interests), small (1500 bytes) or large (jumbo) frames (7500 bytes) Data packets.

between $4.6\% - 7\%$ and for $d = 4$ the additional bandwidth is between $7\% - 9.5\%$.

To summarize the results for topology AS-20965, for $d = 3$, on average the PIT contains 29% of the baseline entries and occupies 40% of the initial memory footprint (i.e., a reduction of 60%) at the cost of $4.6\% - 7\%$ of additional bandwidth transmitted. When $d = 4$, the PIT entries are reduced to 20% of baseline CCN, reducing the actual memory footprint at 28% (i.e., a reduction of over 70%) at the cost of at most $7\% - 9.5\%$.

Figure A.23 in Appendix A shows the estimated bandwidth overheads for all tested topologies and Figure 3.16 summarizes these results for $d = 3$ and $d = 4$ and Table 3.5 contains the respective numerical data. For $d = 3$, in most of the tested topologies the bandwidth overhead is below 10%. However, the results in AS-1221 and scale-free-100 remain problematic.

Table 3.5: Multicast bandwidth overhead with a Zipf distribution of group sizes in the HC policy for all topologies: (i) S-S: small content-names and small Data packets, (ii) S-L: small content-names and large Data packets, (iii) large content-names and small Data packets and (iv) L-L: large content-names and large Data packets.

(a) $d = 3$

Topology	**S-S (%)**	**S-L (%)**	**L-S (%)**	**L-L (%)**
AS-20965	6.9	5.1	7.2	4.6
AS-224	3.2	1.5	3.3	1.1
AS-3967	6.3	4.4	6.7	3.9
AS-1755	5.2	3.4	5.5	2.9
AS-1221	27.1	25.4	27	24.8
AS-6461	10.6	8.7	11	8.2
scale-free-50	18.1	16.4	18.1	15.9
scale-free-100	48.6	47.1	48	46.4

(b) $d = 4$

Topology	**S-S (%)**	**S-L (%)**	**L-S (%)**	**L-L (%)**
AS-20965	9.5	7.4	10	7
AS-224	13.1	11.1	13.5	10.6
AS-3967	8	5.9	8.5	5.4
AS-1755	11.1	9.1	11.5	8.6
AS-1221	93.6	92.1	92.4	91.2
AS-6461	15.6	13.4	16.2	12.8
scale-free-50	31.3	29.6	31.2	29
scale-free-100	172.8	172.5	169.5	171.2

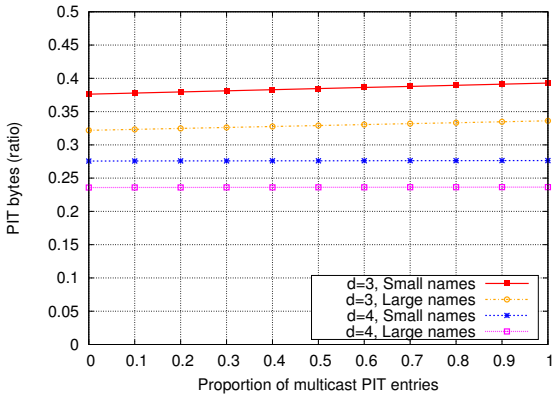
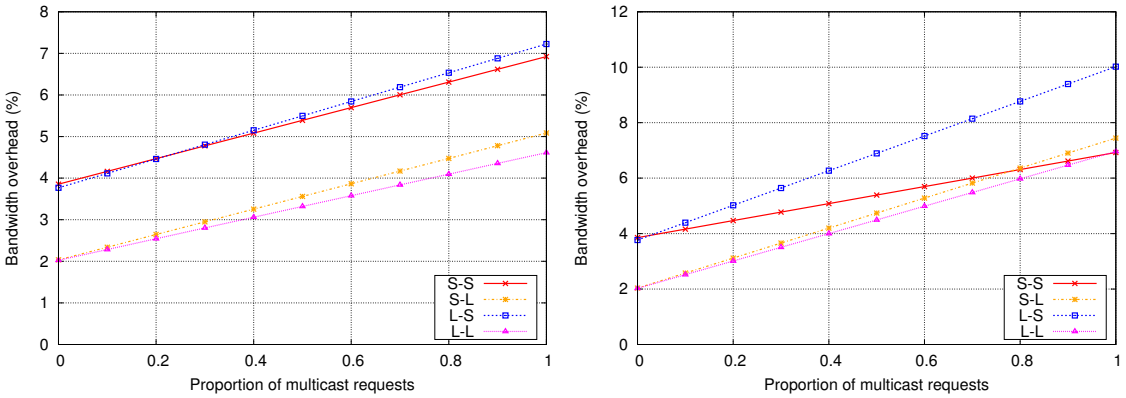


Figure 3.17: PIT size (in bytes) with respect to unicast-multicast proportion of PIT entries, with Hop Counter-based Interest tracking in topology AS-20965.

3.4.5 Unicast - Multicast traffic mix

In Section 3.4.3 we presented results for unicast traffic, observing significant gains in PIT state reduction at a minimal cost of redundant Data transmissions caused by false positives in the Bloom filters. In Section 3.4.4 we presented results for multicast applications and observed the PIT state reduction gains come at the cost of additional bandwidth overheads caused by redundant Interest and Data packets. In order to evaluate the overall benefits and costs of semi-stateless forwarding, we need to consider a mix of unicast and multicast traffic. In this section, we present some numerical calculations based on our previous results with respect to the proportion of unicast vs. multicast load.

Figure 3.17 shows the overall PIT size reduction for small and large names in topology AS-20965. The x-axis in Figure 3.17 represents the proportion of PIT entries created by Interests belonging to multicast applications. More specifically, a value of 0.0 in the x-axis means that 0% of PIT entries in each router are generated by multicast applications (100% of Interests are for unicast) while a value of 1.0 means that 100% of PIT entries correspond to Interests transmitted by multicast applications. Although our method reduces the PIT more effectively for unicast applications, there are not significant differences between the two extreme cases of network usage, i.e., (i) all PIT entries are created by unicast applications and (ii) all PIT entries are created by multicast applications.



(a) $d = 3$ (b) $d = 4$

Figure 3.18: Overall bandwidth overhead with respect to proportion of unicast-multicast traffic mix in topology AS-20965.

In a similar manner, we computed the bandwidth overheads as a function of the traffic mix, shown in Figure 3.18. The x-axis in Figure 3.18 shows the proportion of requests (therefore Interest and Data packets) generated by multicast applications. A value of 0.0 in the x-axis represents the case of 0% packets being generated by multicast applications (100% of traffic belongs to unicast applications) while a value of 1.0 in the x-axis represents the case where 100% of traffic is generated by multicast applications. In contrast to the PIT state reduction, the traffic mix plays a more important role with respect to bandwidth overheads. As shown in Figure 3.18, the overheads introduced by multicast applications are approximately twice the amount of overheads caused by unicast applications.

The results presented in Figures 3.17 and 3.18 give us insight on the minimum and maximum gains and overheads considering the network usage. For $d = 3$, the PIT size in AS-20965 is reduced to $33\% - 40\%$ compared to CCN (i.e., a reduction of $60\% - 67\%$) at a cost of $2\% - 7\%$ additional bandwidth (Fig. 3.18(a)) according to the unicast-multicast traffic mix. For $d = 4$, the PIT requires $24\% - 28\%$ of the memory compared to a basic CCN PIT (i.e., a reduction of $72\% - 76\%$) at the bandwidth cost of $2\% - 10\%$ (Fig. 3.18(b)).

Table 3.6 presents the numerical data for minimum and maximum PIT size and bandwidth overheads for all tested topology with Hop Counter-based Interest

tracking for $d = 3$ and $d = 4$.

Table 3.6: Minimum and maximum PIT memory sizes and bandwidth overheads for all topologies.

(a) $d = 3$

Topology	**PIT size % [min, max]**	**Bandwidth overhead % [min, max]**
AS-20965	[32.2, 39.3]	[2, 7.2]
AS-224	[35.3, 42.5]	[3, 6.8]
AS-3967	[32, 40.6]	[0.4, 6.7]
AS-1755	[36.1, 42.6]	[0.4, 5.5]
AS-1221	[30.4, 38.8]	[1.2, 27.1]
AS-6461	[28.3, 34.8]	[0.4, 11.0]
scale-free-50	[29.1, 38.8]	[1.2, 18.1]
scale-free-100	[28.2, 34]	[3, 48.6]

(b) $d = 4$

Topology	**PIT size % [min, max]**	**Bandwidth overhead % [min, max]**
AS-20965	[23.6, 27.6]	[2, 10]
AS-224	[24.1, 28.2]	[3, 13.5]
AS-3967	[23, 27]	[0.4, 8.5]
AS-1755	[24.6, 28.8]	[0.4, 11.5]
AS-1221	[21.5, 25.4]	[1.2, 92.4]
AS-6461	[23.1, 27]	[0.4, 16.2]
scale-free-50	[21.1, 25]	[1.2, 31.2]
scale-free-100	[24.5, 28.6]	[3.9, 172.5]

3.5 Conclusion

In this chapter we presented a semi-stateless forwarding scheme that mitigates the PIT state requirements of CCN routers by integrating Bloom filter-based packet forwarding. Our scheme maintains the desired qualitative features of CCN's stateful forwarding and requires minimal changes to CCN's basic operation.

The semi-stateless forwarding scheme depends on the *Forwarding State Reduction Factor* d, a crucial parameter for system performance. The factor basically dictates how many *stateless* hops an Interest should perform before being tracked by a CCN router and therefore affects the amount of PIT state reduction and bandwidth overheads. In this chapter we assumed a static value for d and eval-

uated the scheme's performance with respect to various values of d in order to understand the system's behaviour.

With the exceptions of the under-performing topologies *AS-1221*, *scale-free-50* and *scale-free-100*, setting $d = 3$ resulted in a PIT that is approximately $30\% - 40\%$ of the baseline CCN PIT at the cost of $2\% - 11\%$ bandwidth overhead. For $d = 4$, the PIT is approximately reduced to $22\% - 28\%$ compared to baseline CCN PIT the cost of $2\% - 16\%$ of additional bandwidth (see Table 3.6). These results could be a good indicator for choosing an appropriate network-wide static value for d, for example during the dimensioning phase of the network design. Yet, the scheme performed poorly in 3 of our tested topologies, which were the largest ones with respect to the number of access nodes. This suggests that we either need to set a smaller value for d, e.g., no less than $d = 2$ in order to have any effect on PIT state reduction, or consider other alternatives, such as choosing a *correct* d on finer granularity.

In the next chapter we present a scheme that dynamically adapts the *Forwarding State Reduction Factor* on a per-group basis, in order to reduce the amount of false positive Data transmissions. We will show how this effectively adjusts the value of d so that bandwidth overhead is kept low while preserving the benefits of PIT state reduction.

Chapter 4

Adaptive Semi-stateless Forwarding in Content-Centric Networks

4.1 Introduction

In Chapter 3 we presented a semi-stateless forwarding scheme for CCN with the goal of mitigating PIT state requirements in CCN routers, thus improving the architecture's forwarding scalability properties. The basic idea of the semi-stateless forwarding scheme is to track Interests at every d hops and use Bloom filter-based stateless forwarding between routers that stored a particular Interest. We investigated system performance with respect to d, the *Forwarding State Reduction Factor*, and saw that by setting a global value for d, e.g., $d = 3$ or $d = 4$, we obtained significant PIT memory savings at the order of $\approx 70\%$. These gains came at the cost of additional bandwidth that varied between $2\% - 16\%$ in small-to-medium topologies. However, in the largest topologies (with respect to the number of access nodes) these costs were significantly higher and raised doubts regarding the scheme's efficiency.

We discussed in Section 3.4.4, and showed in the results of Figure 3.5, that the performance and overhead trade-offs of the semi-stateless scheme depend on

the multicast-group size and the value of d. Our scheme performs better for small multicast groups; as the multicast group grows, redundant Data packet (primarily) and Interest packet (secondarily) transmissions lead to bandwidth overheads compared to basic CCN stateful forwarding. A *relatively* large value of d, for example $d = 3$ or $d = 4$, favours small groups; there is significant reduction of the PITs with almost negligible costs in bandwidth overhead. For larger groups, a smaller d is more suitable in order to avoid excessive bandwidth overheads, for example $d = 2$. Setting a low value for d, however, means that we obtain fewer gains with respect to PIT reduction.

Ideally, instead of setting network-wide fixed value for d, we should set d on a per-group basis, i.e., decide a suitable value for d based on the size of the multicast group. Our goal is to preserve the desired PIT reduction gains in small-to-medium multicast groups with $d = 3$ or $d = 4$ and set $d \leq 2$ in larger groups in order to keep redundant packet transmissions low. Unfortunately, such an approach is difficult within the architectural context of CCN. Hosts and routers are address-less; routers see only incoming Interest and Data packets that carry content-names. Unless an addressing scheme is used and Interests/Data packets carry these addresses, a CCN router cannot determine the number of downstream receivers for a requested content item. Hence, a suitable d cannot be determined based on the number of receivers without sacrificing crucial aspects of the CCN architecture.

An alternative approach is to constrain the amount of redundant traffic by limiting the number of links inserted into the IBFs. Previous studies have shown that in order to maintain a high *forwarding efficiency* in Bloom filter-based forwarding (defined in Eq. 2.2), the *false positive probability* (fpp) of IBFs must be kept below 0.5% [49]. Recall from Section 2.3.3 that the *false positive probability* is defined in Equation 2.1 as

$$fpp = (1 - e^{-kn/m})^k$$

where m is the Bloom filter size (bits), k is the number of hash functions and n is the number of items inserted (LIDs). Though m and k can be known to each router, n is not since Interests carry the accumulated IBF for the reverse path

and routers store the entire IBF in the PIT. In order for a CCN router to be able to obtain n, Interests would need to carry each traversed LID separately, thus increasing the Interest header size. In addition, routers would need to track each downstream LID separately instead of an accumulated IBF, thus requiring more memory per PIT row.

The fpp, on the other hand, can also by calculated by examining its *fill factor* which is the portion of bits set in the Bloom filter

$$fill\_factor = \frac{\#\ bits\ set}{m} \tag{4.1}$$

The fpp can be then calculated as [48]

$$fpp = (fill\_factor)^k = (\frac{\#\ bits\ set}{m})^k \tag{4.2}$$

With Equation 4.2, a router can estimate an IBF's fpp just by counting the number of bits that are set to 1.[1] Therefore, a router can infer when an IBF is congested, e.g., the IBF's $fill\_factor$ is greater than 0.4,[2] caused by adding too many links to it. A congested IBF is likely to lead to excessive false forwarding decisions. We utilize this information and extend our forwarding scheme with the ability to dynamically adapt d based on the IBFs constructed inside the network.

4.2 Adapting the Forwarding State Reduction Factor

At a high level, our idea is the following: the system tries to effectively reduce the PIT size, until it detects that certain redundant traffic thresholds may be violated. These violations are detected by routers who inspect the $fill\_factor$ of downstream IBFs. Once such a violation is detected, routers inform hosts with – what we call – Bloom filter Congestion Notifications (BCN). More specifically, the system dynamically adapts the value of d in order to keep redundant Data packet transmissions low, at the expense of less-effective PIT reduction as follows:

[1] And having a global agreed value for k.
[2] Assuming $k = 5$ or $k = 6$.

Instead of having a network-wide fixed value for d, d is set on a per-Interest basis, selected by the issuing host. During Data forwarding, routers that perform IBF switching inspect the stored downstream IBFs. If the $fill\_factor$ of an IBF is above a predefined threshold, e.g., $fill\_factor \geq 0.4$, the router assumes that this IBF is *congested* and inserts a BCN feedback inside the Data packet. The BCN is carried downstream to all receivers and instructs them to lower their d. If a host receives no BCN for a number of consecutive Data packets, it attempts to increase its d in order to achieve better PIT reduction. Hosts decide when to increase their d based on an exponential back-off scheme.

4.2.1 Interest tracking

Each time a host issues an Interest, the host selects a desired d for this Interest and places it in the Interest header.[3] During Interest forwarding, routers perform the same operations as before, except that, for each received Interest, d is obtained from the Interest's header, rather than being a fixed system-wide value. In addition, for each stored Interest in the PIT, routers store the *minimum* d among the different d values that have been received from different Interests requesting the same Data packet. For example, a user may request "*/a/b/c.mp4*" with $d = 3$ while another user may request the same Data with $d = 4$. If the two Interests are merged on the same router, the router will store $d_{min} = min(3, 4) = 3$ for "*/a/b/c.mp4*". Algorithm 5 shows the modified Hop Counter-based Interest tracking policy with d carried in the Interest header.

4.2.2 Data forwarding

In the data plane, routers forward Data packets as before, with the addition of the $fill\_factor$ check each time a router inserts an IBF to a Data packet; this takes place only at the routers where the corresponding Interest is tracked. More specifically, upon the reception of a Data packet, a router must check whether there is a correspondent PIT entry for the Data packet's name. If a PIT entry exists, the

[3]Note that d is stored in the packet in addition to the hop counter.

Algorithm 5 Hop Counter-based Interest tracking policy with adaptive d.

```
1: procedure HOP_COUNTER_TRACKING_ADAPTIVE(interest, incoming_port)
2:     d := extract_d(interest)
3:     pit_entry := PIT_lookup(interest)
4:     if pit_entry not null then
5:         d_min := min(d, pit_entry.d)
6:         store_in_PIT(interest, d_min, incoming_port)
7:         return                                   ▷ Interest suppressed
8:     end if
9:     hc := increment_hop_counter(interest)
10:    if hc = d then
11:        store_in_PIT(interest, d, incoming_port)
12:        reset_hop_counter(interest)
13:    end if
14:    out_port := FIB_lookup(name)
15:    forward(interest, out_port)
16: end procedure
```

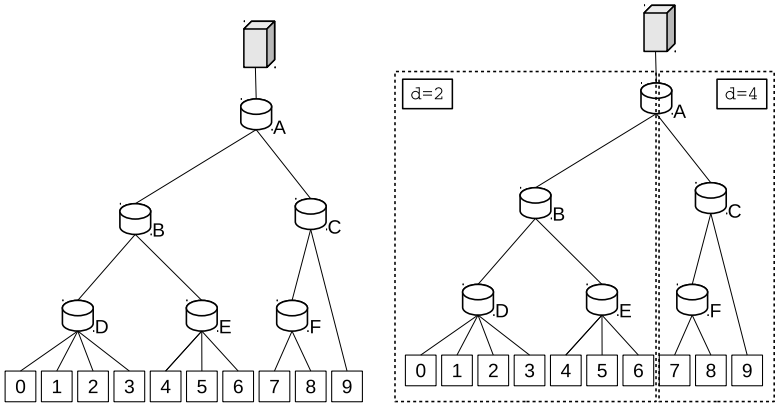
router checks the new IBF's $fill\_factor$ and if it exceeds the maximum $fill\_factor$ threshold, the router sets the BCN feedback field inside the Data packet. The BCN feedback is an integer number set to $d_{BCN} = max(1, d_{min} - 1)$; it is added to the Data packet header. The BCN feedback will instruct all downstream receivers to lower their individual d to d_{BCN}. When the BCN feedback is set in the header of a Data packet, it is transmitted all the way to the receivers. If a Data packet contains BCN feedback, indicated by a non-zero value in the BCN field, and along the data path another router needs to also set the BCN feedback, this is set to the minimum d_{BCN}. For example, if a Data packet has a BCN with $d_{BCN} = 3$ and along the path another router must set BCN to $d_{min} = 2$, then the latter router will update feedback to $d_{BCN} = min(d_{BCN}, d_{min}) = 2$. Algorithm 6 shows the algorithm for semi-stateless forwarding with adaptive d.

Algorithm 6 Semi-Stateless Forwarding with Adaptive d.

```
1: procedure SEMI_STATELESS_FORWARDING(data_packet, incoming_port)
2:     pit_entry := PIT_lookup(data_packet.name)
3:     if pit_entry not null then
4:         data_packet.IBF := pit_entry.IBF
5:         if fill_factor(data_packet.IBF) >= fill_factor_max then
6:             d_BCN := max(1, pit_entry.d_min − 1)
7:             if BCN_is_set(data_packet) then
8:                 d_BCN = min(d_BCN, data_packet.d_BCN)
9:             end if
10:            data_packet.d_BCN = d_BCN
11:        end if
12:    end if
13:    IBF_FORWARD(data_packet, incoming_port)
14: end procedure
15: procedure IBF_FORWARD(data_packet, incoming_port)
16:    for port in ports do
17:        if port = incoming_port then
18:            continue
19:        end if
20:        lid_port := link_id(port)
21:        if data_packet.IBF AND lid_port = lid_port then
22:            transmit(data_packet, port)
23:        end if
24:    end for
25: end procedure
```

4.3 Receiver-side adaptation

Hosts not only update d according to the feedback received by the network, they also try to increase their d during the application session. This is required for two reasons. First, without prior knowledge of the network conditions and/or ap-



(a) Multicast tree. (b) Adaptation of d.

Figure 4.1: Adaptation of d on sub-tree granularity.

plication participants (unicast or multicast), hosts start with a moderate value for d, $d_{default} = 2$ in our experiments, in order to avoid excessive bandwidth overheads in case they join a large multicast group. However, a low value for d is less effective for unicast and small multicast groups, in which cases d can be increased without significant bandwidth penalties. Second, when a router sets the BCN feedback, this affects all of its downstream receivers, even though the decision is made based only on the locally stored IBF. This is a relatively coarse-grained action; selecting a suitable d does not only depend on the total number of multicast receivers but also on the shape of the multicast tree. Consider, for example, the multicast tree of Figure 4.1. The tree is quite imbalanced; the subtree that is formed from receivers 0-6 is more dense compared to the subtree formed by receivers 7-9. Assume that router A sets the BCN feedback to $d_{BCN} = 2$. This will cause all receivers to set their d to 2, as the feedback is propagated all the way to the receivers, but this decision is mostly caused because of the dense subtree's shape. Assigning $d = 2$ to receivers 7-9 will miss the opportunities for further PIT state reduction in routers C and F. For example, in this region of the delivery tree, a $d = 4$ might produce better PIT state reduction results will little overhead. Therefore, after A sends its BCN feedback, causing all receivers to lower their d, there is reasonable interest for receivers 7, 8 and 9 to increase their own particular d.

Hosts attempt to increase their d, but, in order to avoid situations where

all hosts act simultaneously, the time when such a decision is made is chosen individually, based on an exponential back-off scheme. At a high-level, the goal is to avoid a kind of *collision* happening when all hosts increase their d together, e.g., every 10 Interests, as such synchronization might generate congested Bloom filters. In contrast, we want each host to act autonomously and not in coordination with other hosts so that the system adapts gradually to a steady state. We achieve this as follows: When hosts receive Data packets with BCN feedback, they update their d so that subsequent Interests will carry the updated d. Hosts mark when the last BCN-enabled Data packet arrived and after a number of consecutive non-BCN Data packets arrive, they assume that the network is in a steady state with respect to non-congested IBFs. In this case, hosts will increase their own d by 1 in an attempt to achieve better PIT state reductions. The number of consecutive non-BCN Data packets that will trigger an increase in d is calculated as

$$exp\_backoff = random(1,\ slot\_size * 2^{1+d_{BCN}}) \tag{4.3}$$

where the $slot\_size$ is selected by the application and may depend on the application packet transmission rate. In more detail, once a host receives a BCN feedback Data packet, it will set its own d to d_{BCN} and then select a random number in $[1, slot\_size * 2^{1+d_{BCN}}]$, which dictates the number of consecutive non-BCN Data packets required to arrive before attempting to increase its d. If, in the meantime, another BCN-marked Data packet arrives, the whole process is restarted: d is updated and the exponential back-off is reset taking into account the new d. After increasing its d, the entire process restarts. That is, the host will attempt to further increase its d, this time waiting for *statistically* more time, unless of course a BCN feedback arrives. As a result, receivers of the same multicasted BCN-marked Data packet will not act in a synchronized manner, due to the randomization of the backoff interval; in addition, the backoff interval will be doubled for each consecutive increase to d. Algorithm 7 shows a host's operation when a Data packet is received and Algorithm 8 shows the host's operation when transmitting an Interest.

Note that we chose to set an upper bound on d in order to avoid excessive redundant Interests. As we will later show in Section 4.4, the adaptive scheme is able to keep additional Data packets low. However, the amount of additional

Algorithm 7 Data packet reception on the receiver side.

```
1: procedure DATA_RECEIVED(data_packet)
2:     if BCN_is_set(data_packet) then
3:         d_BCN = data_packet.d_BCN
4:         SET_STATELESS_FACTOR(data_packet.name, d_BCN)
5:     end if
6:     pass_data_to_app(data_packet)
7: end procedure
8: procedure SET_STATELESS_FACTOR(content_name, d)
9:     stateless_factor[content_name] := d
10:    exp_backoff = random(1, slot_size * 2^(d+1))
11:    increase_thres[content_name] := exp_backoff
12: end procedure
```

Algorithm 8 Interest transmission on the receiver side.

```
1: procedure SEND_INTEREST(content_name)
2:     if has_feedback(content_name) then
3:         d := stateless_factor[content_name]
4:         increase_thres[content_name] := increase_thres[content_name] − 1
5:         if increase_thres[content_name] == 0 AND d < d_max then
6:             d := d + 1
7:             SET_STATELESS_FACTOR(content_name, d)
8:         end if
9:     else
10:        d := d_default
11:        SET_STATELESS_FACTOR(content_name, d)
12:    end if
13:    hc := random(0, d − 1)
14:    transmit_interest(content_name, hc, d)
15: end procedure
```

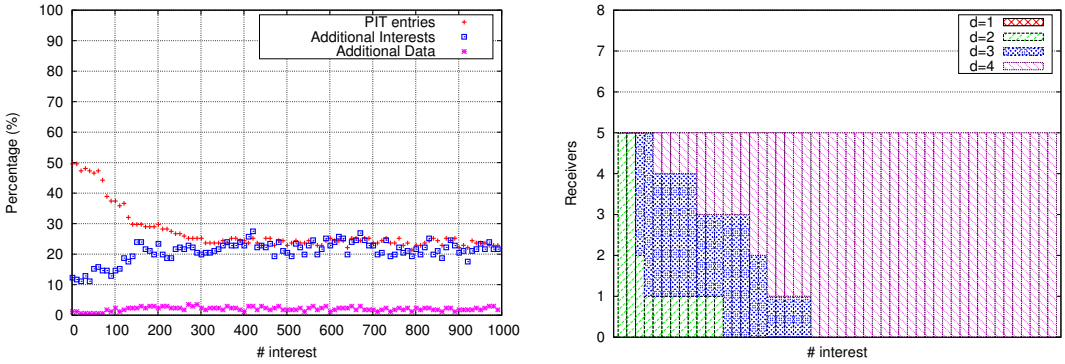
Interests (compared to basic CCN forwarding) is hard to detect in a distributed manner. Based on our experimental results presented in Section 3.4, we decided to set an upper bound on d to $d_{max} = 4$, beyond which we considered that the amount of additional Interests will not be acceptable.

4.4 Evaluation

We repeated the same set of experiments described in Section 3.4 using the dynamic adaptation scheme. Hosts start with $d_{default} = 2$ and the upper bound of d is $d_{max} = 4$. For the HTTP-like streaming application, we set the $slot\_size$ to 20 Interests. Our focus remains on PIT state reduction and bandwidth overhead caused by additional Interest and Data transmissions compared with baseline CCN. Again, all presented results are normalized against the basic CCN scheme.

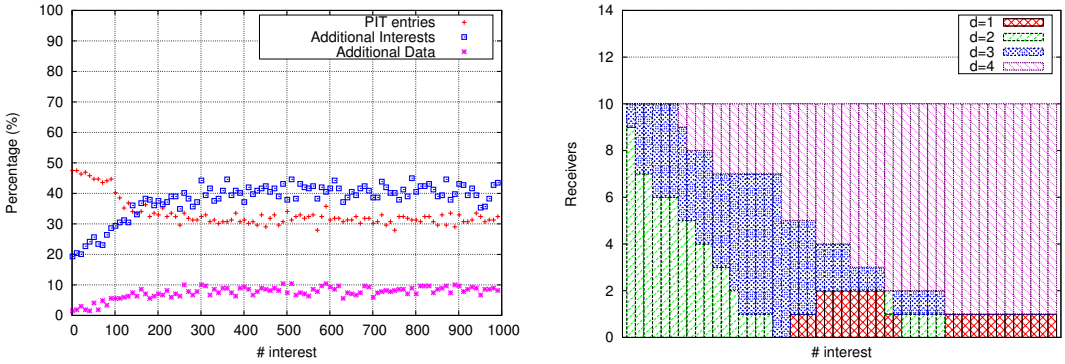
4.4.1 Performance results

Figures 4.2 to 4.4 show the behaviour of the adaptive scheme over time for a multicast group of 5, 10 and 30 receivers, respectively, in topology AS-20965. In each figure, we show the PIT entries, additional Interests and Data packets (sub-figures (a)) accompanied with the distribution of d among receivers of the same multicast group (sub-figures (b)). There is a common pattern in all figures: hosts start with $d_{default}$ and adjust d according to received BCNs. Eventually, the system converges to a steady state. In all figures, Data overhead remains low: less than 5% in Figures 4.2(a) and 4.4(a) (5 and 30 receivers) and approximately 10% in Figure 4.3(a) (10 receivers). Data overhead is kept low at the cost of varying gains in PIT state reduction. For the small group of Figure 4.2, the PIT is reduced to less than 30% of basic CCN, since all hosts were able to increase their d to 4 (Fig. 4.2(b)) without receiving BCN feedbacks from the network. For the group of 10 receivers (Fig. 4.3(a)), the PIT for each router was on average reduced to a little above 30% and 8 out of the 10 receivers gradually set their d to 4 whereas the remaining 2 set their d to 1 (Fig. 4.3(b)). On the large multicast group of Figure 4.4, the PIT was reduced to 75% compared to baseline CCN (Fig. 4.4(a));



(a) Performance. (b) Distribution of d among receivers.

Figure 4.2: Performance results for a multicast group of 5 receivers in AS-20956.

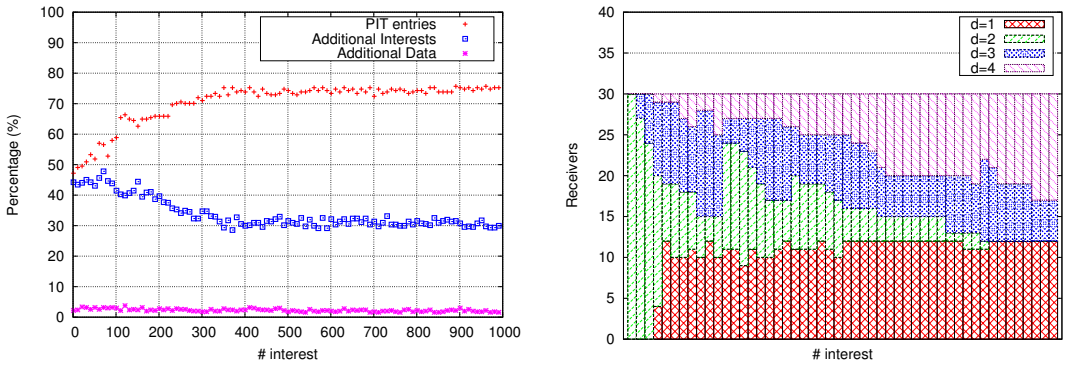


(a) Performance. (b) Distribution of d among receivers.

Figure 4.3: Performance results for a multicast group of 10 receivers in AS-20956.

due to the group's size and shape of the multicast tree, almost half of the receivers were instructed to set their d to 1 (Fig. 4.4(b)), i.e., the system resorted to stateful forwarding for a part of the delivery tree in order to avoid redundant Data packets. The Interests overhead also varies: $\approx 20\%$ for the small multicast group, $\approx 40\%$ for the group of 10 receivers and $\approx 30\%$ in the large multicast group. Recall, however, that Interests, due to their small size, affect the overall bandwidth overhead much less than Data packets.

In Figure 4.5 we present the scheme's performance depending on group size in topology AS-20956. With the dynamic adaptation of the *Forwarding State Reduction Factor*, the Data overhead is kept below 10% for all group sizes at the



(a) Performance.

(b) Distribution of d among receivers.

Figure 4.4: Performance results for a multicast group of 30 receivers in AS-20956.

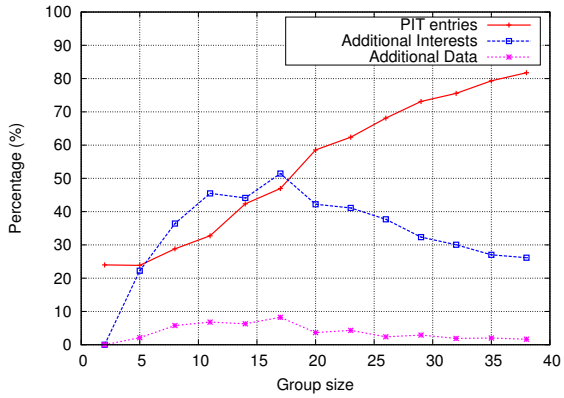
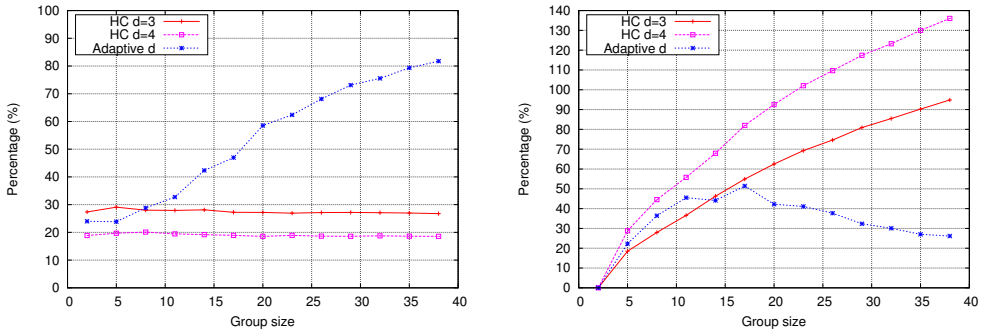


Figure 4.5: Performance results in topology AS-20965 with dynamic adaptation of d with respect to group size.

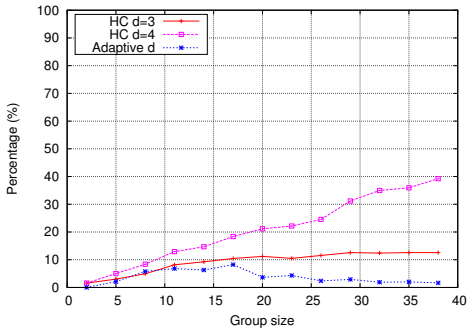
cost of a less effective reduction in the PIT compared to the fixed d schemes. Interests overhead remains high, but as we showed in the previous section, we consider their impact on the overall bandwidth overhead to be manageable. The results for all topologies with respect to group size are presented in Figure A.24 in Appendix A. To better understand the improvements achieved with dynamic d compared to fixed d, we provide detailed comparisons for the three measured metrics in Figure 4.6.

Figure 4.7 shows the overall performance results for 1000 multicast groups, with the group size following a Zipf distribution in all topologies, while Table 4.1 contains the respective numerical data. In topology AS-20965, the PIT is reduced



(a) PIT entries

(b) Interests overhead



(c) Data overhead

Figure 4.6: Comparison of fixed-d with adaptive d with respect to the group size in topology AS-20965.

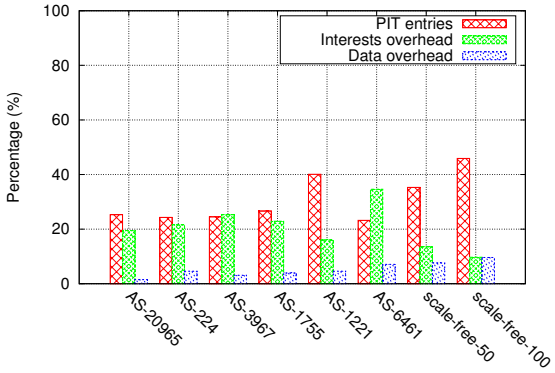


Figure 4.7: PIT entries, Interests overhead and Data overhead for all topologies with dynamic adaptation of D, 1000 multicast groups, group size follows a Zipf distribution.

Table 4.1: PIT entries, Interests overhead and Data overhead for all topologies with dynamic D, Zipf distribution of group sizes.

Topology	PIT entries (%)	Interests overhead (%)	Data overhead (%)
AS-20965	25.3	19.5	1.5
AS-224	24.3	21.6	4.6
AS-3967	24.5	25.3	3.1
AS-1755	26.6	22.9	3.9
AS-1221	40.1	16	4.6
AS-6461	23.2	34.6	7.1
scale-free-50	35.3	13.5	7.6
scale-free-100	45.9	9.7	9.6

to 25% compared to basic CCN, at the cost of $\approx 20\%$ additional Interests and less than 2% of additional Data packets. Notice the improvement in topologies *AS-1221*, *scale-free-50* and *scale-free-100* in which we had observed severe overheads with the fixed d semi-stateless scheme. In all topologies, the PIT was reduced to $24\% - 45\%$ compared to the basic CCN PIT with additional Data packets of $1.5\% - 9.6\%$.

Finally, we compare the overall adaptive scheme's performance when group sizes follow a Zipf distribution against the fixed-d scheme. Figure 4.8 shows the basic performance metrics, i.e., PIT entries, additional Interests and Data packets in topology AS-20965 with $d = 3$, $d = 4$ and adaptive d. With adaptive d, the PIT entries are reduced to $\approx 25\%$, which is between what we achieved with $d = 3$ and $d = 4$. With adaptive d, the Interests overhead is $\approx 20\%$, close to what $d = 3$ achieves. The Data overhead is lower compared to both fixed-d schemes.

For illustration purposes, Figure 4.9 compares the performance results for all tested topologies. In the topologies that performed well with fixed d, the PIT reduction has a similar behaviour: the adaptive scheme reduces the PIT between $d = 3$ and $d = 4$. In the three under-performing topologies, PIT size was reduced less: 40% for *AS-1221*, 35% for *scale-free-50* and 45% for *scale-free-100*. Notice, however, the tremendous improvements on Data overheads (Fig. 4.9(c)) in the under-performing topologies *AS-1221*, *scale-free-50* and *scale-free-100*: Data overhead has dropped below 10% in all cases.

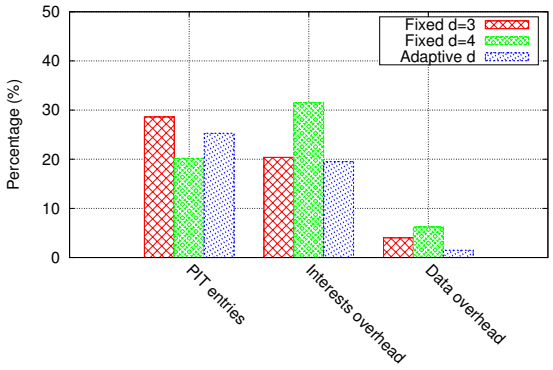
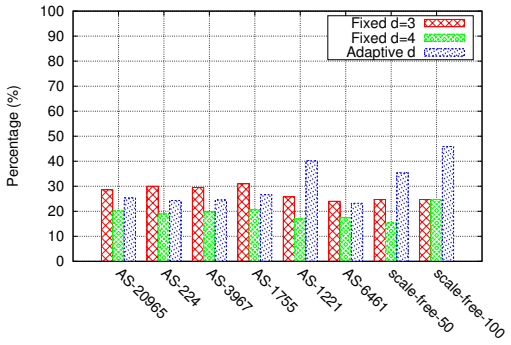
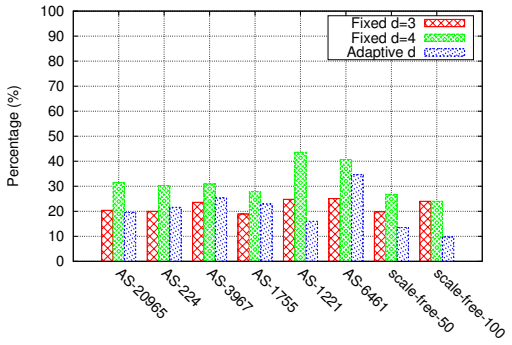


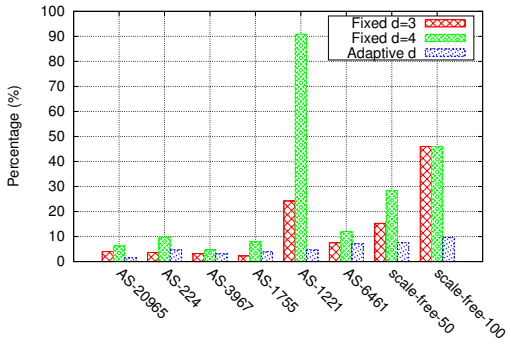
Figure 4.8: Comparison of fixed-d with adaptive d in topology AS-20965.



(a) PIT entries



(b) Interests overhead



(c) Data overhead

Figure 4.9: Comparison of fixed-d with adaptive d for all topologies.

4.4.2 PIT memory reduction

We now present the PIT memory reductions achieved with the adaptive scheme. As we described in Section 3.4.4, we consider a hash table-based im-

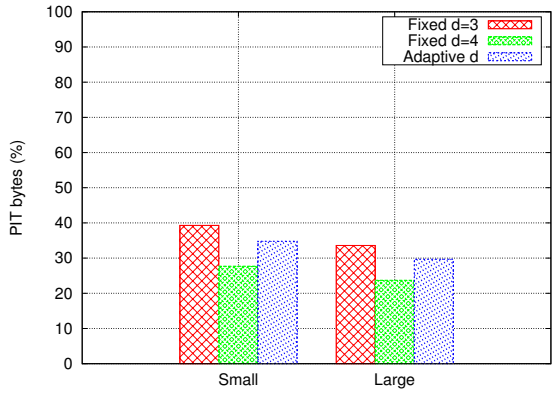
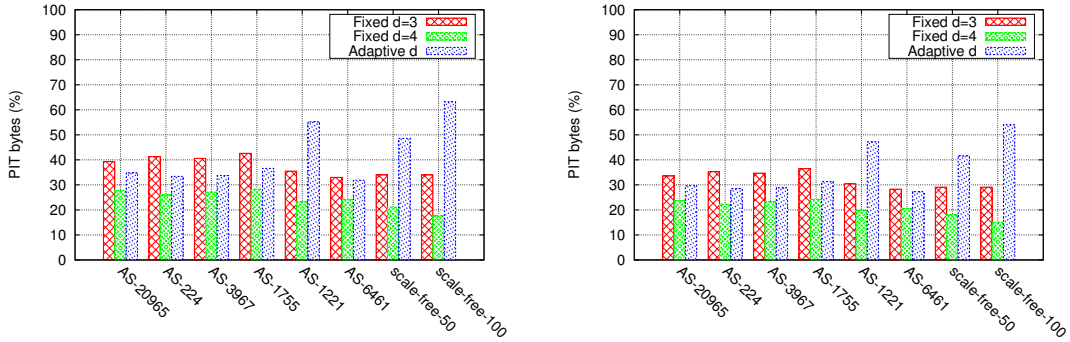


Figure 4.10: Estimated PIT size (in bytes) for small (20-bytes) and large (56-bytes) content names for topology AS-20965.



(a) Small (20 bytes) content-names

(b) Large (56) content-names

Figure 4.11: Estimated PIT size (in bytes) for small (20-bytes) and large (56-bytes) content names for all topologies.

plementation of the PIT and two classes of content-names: small content-names which are on average 20 bytes and large content-names which are on average 56 bytes. Figure 4.10 shows the PIT memory size for the adaptive scheme in topology AS-20965 compared to fixed $d = 3$ and $d = 4$. The adaptive scheme reduces the PIT memory size to 35% for small names and to 30% for large names against the basic CCN PIT, i.e., we achieved a memory reduction of $65\% - 70\%$. Compared to the fixed d schemes, the performance of the adaptive scheme is between $d = 3$ and $d = 4$. Figure 4.11 shows the PIT memory reduction in all tested topologies.

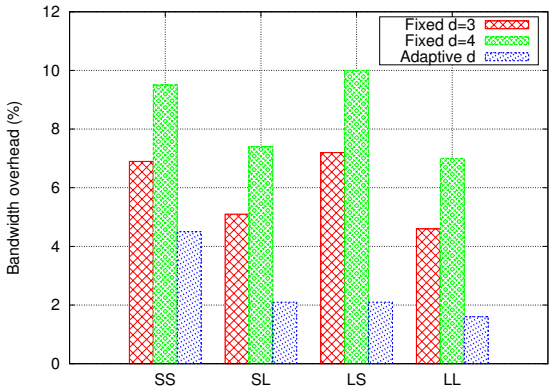


Figure 4.12: Multicast bandwidth overhead for topology AS-20965. Small content-names (36-byte Interests) and large-content names (70-byte Interests), small (1500 bytes) or large (jumbo) frames (7000 bytes) Data packets. Results are normalized against basic CCN.

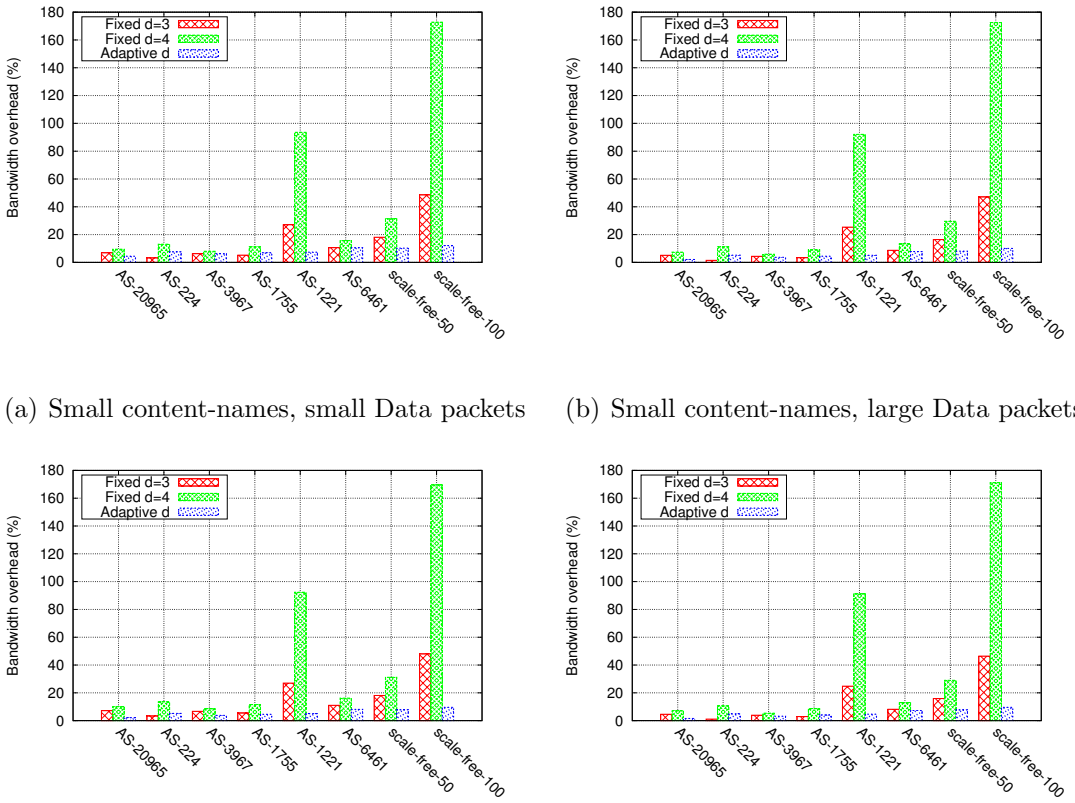
4.4.3 Bandwidth overhead

In this section we quantify the Interests and Data overheads in terms of bandwidth. We follow the same approach as in Section 3.4.4: we consider two classes of content-name lengths, small (36 byte Interests) and large (70 byte Interests) and two types of Data packets, small (1500 byte Data) targeting a CCN deployment directly over Ethernet, and large (7500 byte Data) targeting a deployment with Jumbo Ethernet frames.

Figure 4.12 presents the multicast bandwidth overhead in topology AS-20965 and compares the adaptive scheme with fixed $d = 3$ and $d = 4$. The bandwidth overhead is between 1.6% for large packets carrying large content names and 4.5% for small packets carrying small content names. Figure 4.13 shows the bandwidth overhead results for all tested topologies and Tables 4.2 and 4.3 contain the numerical data.

4.4.4 Overall performance

Table 4.4 summarizes the overall performance results of the adaptive semi-stateless forwarding scheme. With the exception of topologies *AS-1221*, *scale-free-50* and *scale-free-100*, the adaptive scheme reduced the PIT to $\approx$ 30%, i.e., a



(a) Small content-names, small Data packets

(b) Small content-names, large Data packets

(c) Large content-names, small Data packets

(d) Large content-names, large Data packets

Figure 4.13: Multicast bandwidth overhead for all topologies.

reduction of $\approx 70\%$, compared to the basic CCN forwarding scheme, with bandwidth overheads between $1.5\% - 10\%$. For the three aforementioned topologies, in which the fixed d approach was producing unbearable traffic overheads, the PIT was reduced to $47\% - 63\%$, i.e., a reduction of $37\% - 53\%$, at the cost of $4\% - 12\%$ additional bandwidth.

4.5 Conclusions

In this section, we presented an extension to the semi-stateless forwarding scheme for CCN that dynamically adapts the *Forwarding State Reduction Factor* d in order to limit the amount of redundant Data packets caused by IBF false positives. The scheme relies on Bloom filter Congestion Notifications (BCN) set

Table 4.2: Multicast bandwidth overhead for all topologies with small content-names.

(a) Small content-names, small Data packets

Topology	**Fixed** $d = 3$ **(%)**	**Fixed** $d = 4$ **(%)**	**Adaptive** d **(%)**
AS-20965	6.9	9.5	4.5
AS-224	3.2	13.1	7.6
AS-3967	6.3	8	6.2
AS-1755	5.2	11.1	6.9
AS-1221	27.1	93.6	7.4
AS-6461	10.6	15.6	10.5
scale-free-50	18.1	31.3	10.3
scale-free-100	48.6	172.8	12.1

(b) Small content-names, large Data packets

Topology	**Fixed** $d = 3$ **(%)**	**Fixed** $d = 4$ **(%)**	**Adaptive** d **(%)**
AS-20965	5.1	7.4	2.1
AS-224	1.5	11.1	5.2
AS-3967	4.4	5.9	3.7
AS-1755	3.4	9.1	4.5
AS-1221	25.4	92.1	5.1
AS-6461	8.7	13.4	7.8
scale-free-50	16.4	29.6	8.1
scale-free-100	47.1	172.5	10.1

by CCN routers inside Data packets when too many links are added in IBFs. Hosts set d on a per-Interest basis and adapt their d based on received BCNs. The adaptive scheme allowed not only the automated selection of a suitable d per multicast group, but also the variation of d among receivers of the same multicast group. As a result, bandwidth overheads were kept low, while PIT state reductions were between what fixed $d = 3$ and $d = 4$ achieved. Overall, we have been able to reduce the PIT to 50% to 70% of baseline CCN, with bandwidth overheads of $1.5\% - 12\%$.

Table 4.3: Multicast bandwidth overhead for all topologies with large content-names.

(a) Large content-names, small Data packets

Topology	**Fixed** $d = 3$ **(%)**	**Fixed** $d = 4$ **(%)**	**Adaptive** d **(%)**
AS-20965	7.2	10	2.1
AS-224	3.3	13.5	5.2
AS-3967	6.7	8.5	3.8
AS-1755	5.5	11.5	4.6
AS-1221	27	92.4	5
AS-6461	11	16.2	8.1
scale-free-50	18.1	31.2	7.8
scale-free-100	48	169.5	9.6

(b) Large content-names, large Data packets

Topology	**Fixed** $d = 3$ **(%)**	**Fixed** $d = 4$ **(%)**	**Adaptive** d **(%)**
AS-20965	4.6	7	1.6
AS-224	1.1	10.6	4.8
AS-3967	3.9	5.4	3.2
AS-1755	2.9	8.6	4
AS-1221	24.8	91.2	4.7
AS-6461	8.2	12.8	7.3
scale-free-50	15.9	29	7.7
scale-free-100	46.4	171.2	9.6

Table 4.4: Multicast bandwidth overhead for all topologies with large content-names.

Topology	**PIT size % [min, max]**	**Bandwidth overhead % [min, max]**
AS-20965	[29.7, 34.8]	[1.6, 4.5]
AS-224	[28.5, 33.4]	[4.8, 7.6]
AS-3967	[28.8, 33.7]	[3.2, 6.2]
AS-1755	[31.3, 36.6]	[4, 6.9]
AS-1221	[47.2, 55.2]	[4.7, 7.4]
AS-6461	[27.3, 31.9]	[7.3, 10.5]
scale-free-50	[41.6, 48.6]	[7.7, 10.3]
scale-free-100	[54.0, 63.2]	[9.6, 12.1]

Chapter 5

Bloom filter Switching for Publish-Subscribe Internet

5.1 Introduction

In Chapters 3 and 4 we addressed the issue of forwarding scalability, particularly for multicast distribution, with respect to the memory requirements for routers in the Content-Centric Networking architecture. We proposed a semi-stateless forwarding scheme by integrating Bloom filter-based forwarding in the architecture. Our scheme respected the distributed operation of the basic CCN architecture, i.e., IBF source-route construction and forwarding state placement were performed in a distributed manner.

In this Chapter, we address the issue of IBF scalability with respect to the network/multicast group size when a centralized approach is used, as in the context of the Publish-Subscribe Internet (PSI) architecture. The problem we need to address is the increasing number of redundant transmissions due to false positive matches as an IBF is filled up to represent larger trees, which can be due to larger networks, larger groups, or both. We propose to partially sacrifice the fully stateless operation of IBF forwarding and, therefore, its ability to support unlimited numbers of groups. Specifically, we propose to split a multicast delivery tree in a few sub-trees, install multicast forwarding state at few, centrally selected,

sub-tree root-nodes and apply an IBF-switching forwarding logic. By doing so, we aim to minimize redundant traffic regardless of the group size.

The contribution of IBF switching for multicast is twofold. First, we present a simple algorithm for the placement of IBF multicast state at network routers and evaluate the scheme through simulations over large, synthetic scale-free graphs. Evaluation results show that IBF switching can scale to any group size while keeping redundant traffic below $1\% - 4\%$, at the cost of placing state at no more than $0.5\% - 2.5\%$ of network nodes. Second, we compare the forwarding state requirements of IBF switching with IP multicast and other multicast schemes proposed to reduce multicast forwarding state. The results show that IBF switching achieves a tremendous reduction of multicast state in the range of $87\% - 99.6\%$, i.e., nearly two levels of magnitude less. Hence, even though we sacrificed the fully stateless operation of IBF multicast forwarding, we still get far better scalability with respect to the number of groups that can be supported.

5.2 Overview

The basic idea in IBF switching is to break the initial delivery tree into several sub-trees, compute the IBF for each sub-tree and place the respective IBFs as multicast forwarding state at sub-tree roots. At the data plane, multicast packets are forwarded using the same IBF forwarding logic with one differentiation: upon reaching a stateful node, the node switches the packet's IBF with the stored IBF and further relays the packet using the new IBF. Sub-trees are selected so that the resulting IBFs will have a very low false positive probability (fpp). Figure 5.1 shows an example of a multicast tree with two stateful nodes. The source node uses an IBF to multicast data to the next immediate stateful nodes, i.e., B and C, which apply IBF switching and further push packets down to receivers.

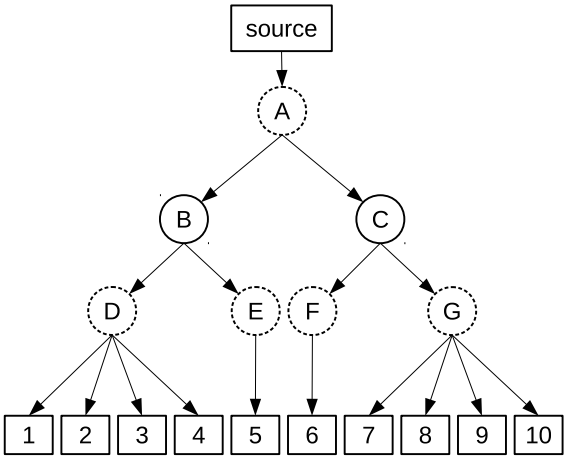


Figure 5.1: An example delivery tree with 10 receivers. Dashed-line nodes are stateless. Solid-line nodes contains sub-tree IBFs.

5.3 Selection of IBF-switching points

To reduce redundant traffic we need to maintain the Bloom filter's fpp below a certain threshold. According to Equation 2.1, fpp depends on m, k and n. The first two parameters are fixed in practice, hence the only tunable parameter is n, i.e., the number of sub-tree links. We define a desired fpp_{max} threshold and use Equation 2.1 to obtain the maximum number of sub-tree links

$$n_{max} = -ln(1 - {fpp_{max}}^{-\frac{1}{k}})(\frac{m}{k}) \tag{5.1}$$

In order to split a multicast tree into subtrees that can fit the constraint of having no more than n_{max} links each, we traverse the multicast tree in a *bottom-up post-order fashion*, breaking it into sub-trees containing n_i links, so that $n_i \rightarrow n_{max}$ as shown in Algorithm 9. In the algorithm we denote n_i as the number of links of the sub-tree rooted at node i, C_i as the set of i's children in the delivery tree and IBF_i as the IBF for the sub-tree rooted at node i. For example, consider the tree of Figure 5.1 and assume $n_{max} = 6$. When we visit nodes D and E we have $n_D = 4$ and $n_E = 1$, so we move to node B where $n_B = 6 = n_{max}$. B becomes an IBF-switching point. We compute IBF_B, install it at B, prune the sub-trees rooted at B and reset n_B to 0. Similarly, node C is selected as an intermediate IBF-switching point. Finally, at the source node we compute the IBF for multicasting data to

Algorithm 9 Sub-tree selection.

1: **function** *tree_traverse*(t: multicast tree, s: tree source, fpp_{thres}: maximum fpp)
2: $n_{max} := -ln(1 - fpp_{thres}{}^{\frac{-1}{k}})(\frac{m}{k})$
3: $n_{source} := sub\_tree\_traverse(t, s, n_{max})$
4: $IBF_s := compute\_IBF(t, s)$
5: $return\ IBF_s$
6: **end function**
7: **function** *sub_tree_traverse*(t: multicast tree, i: current root node, n: maximum number of links)
8: $n_i := 0$
9: **for** $j\ in\ C_i$ **do**
10: $n_i = n_i + 1 + sub\_tree\_traverse(t, j, n)$
11: **end for**
12: **if** $(n_i \geq n_{max})$ **then**
13: $IBF := compute\_IBF(t, i)$
14: $install\_state(IBF_i, i)$
15: $remove\_sub\_tree(t, C_i)$
16: $n_i = 0$
17: **end if**
18: **end function**

nodes B and C.

5.4 Integration with PSI

The selection of the IBF switching points is performed in a centralized fashion, since the algorithm requires full knowledge of the topology graph and multicast tree. This is fully compatible with the PSI architecture, in which multicast group management and data path establishment are handled by the logically centralized Rendezvous (RVS) and Topology Management and Path Formation (TMPFS) subsystems. Whenever a user subscribes to a publication, the RVS adds

the user to the set of subscribed users and requests the TMPFS to install the respective data paths and construct IBFs. With IBF-switching, the TMPFS will select the IBF-switching points and explicitly instruct them to install multicast forwarding state for the specific publication. After the operation is completed, the TMPFS will send the root IBF to the publisher, thus notifying the publisher to start transmitting data to the formed multicast group.

Note, however, that this operation occurs whenever a user subscribes to a publication, as well as when it unsubscribes from it. In both cases, the multicast group is affected and a new multicast tree needs to be computed. Since IBF switching requires centralized computation and installation of multicast forwarding state at routers, this solution is susceptible to some delays and is therefore most suitable for applications with low group dynamics, for example orchestrated data delivery and data backups. In contrast, IBF switching is not suitable for applications with highly dynamic multicast groups (e.g., IPTV); in such cases, a distributed IBF construction and state installation scheme would probably introduce smaller delays. We investigated such distributed IBF construction and installation mechanisms in Chapters 3 and 4.

5.5 Evaluation

We evaluated the scalability properties of IBF switching via extensive simulation tests on a custom-made simulator. We used synthetic scale-free graphs generated with the Barabási-Albert algorithm [79] ranging from 1000 to 5000 nodes. LIDs for links were constructed using *Double Hashing* [48] with SHA-1 and MD5. For each tested graph of size s we generated s different multicast groups, e.g., for a network of 5000 nodes we generated 5000 multicast groups. The size of each multicast group is uniformly distributed in $[10,\ s-10]$, i.e., groups have at least 10 and at most $s-10$ nodes. For each multicast group, group members are randomly selected among all nodes. For multicasting we used the shortest-path trees, i.e., the union of the shortest paths between the source and each receiver.

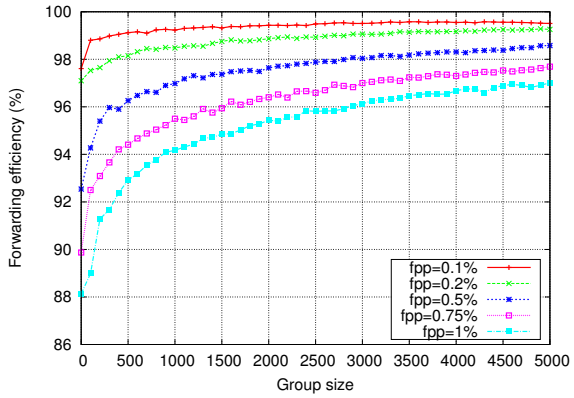


Figure 5.2: Forwarding efficiency of IBF switching. Network size 5000, $m = 256$ bits, $k = 4$, variable fpp_{max}.

5.5.1 Forwarding efficiency and state requirements

Figure 5.2 shows the forwarding efficiency defined in Equation 2.2 as a function of group size in a graph of 5000 nodes with $m = 256$ and $k = 4$. Recall that forwarding efficiency is defined as

$$forwarding\ efficiency = \frac{\#multicast\ tree\ links}{\#total\ packets\ transmitted}$$

When $fpp_{max} = 0.1\%$, forwarding efficiency is kept above 99%, i.e., false positives account for less than 1% of overall traffic, regardless of group size. Forwarding efficiency is not constant: it starts from a lower value and increases quickly. In small groups several false positives occur, but as group size grows and more links are added, fewer links count as false positive. Overall, the forwarding efficiency of IBF switching increases as the group size increases, i.e., IBF switching behaves better in larger multicast groups.

Figure 5.3 shows the state requirements of IBF switching as a function of group size, with the same parameters as above. The Y-axis shows the percentage of nodes where state must be installed. The fraction of stateful nodes grows linearly with multicast group size. As fpp increases, the size of sub-trees also increases; hence the number of stateful nodes decreases. In spite of the linear growth of state requirements, notice that state requirements are overall quite low. For instance, multicasting to a group of 2500 nodes with $fpp = 0.5\%$ (forwarding efficiency at 98%) requires installing multicast state at only 1%, i.e., $5000 \times 1\% = 50$ nodes.

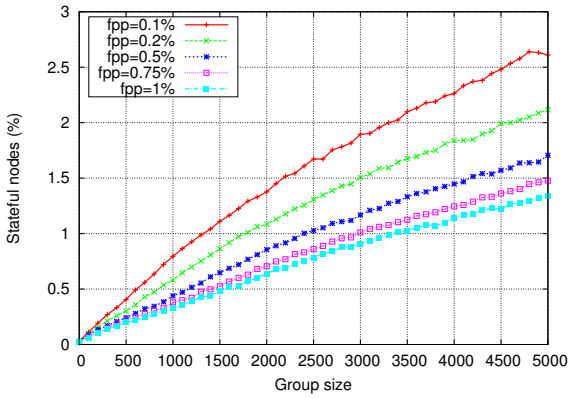


Figure 5.3: State requirements of IBF switching. Network size 5000, $m = 256$ bits, $k = 4$, variable fpp_{max}.

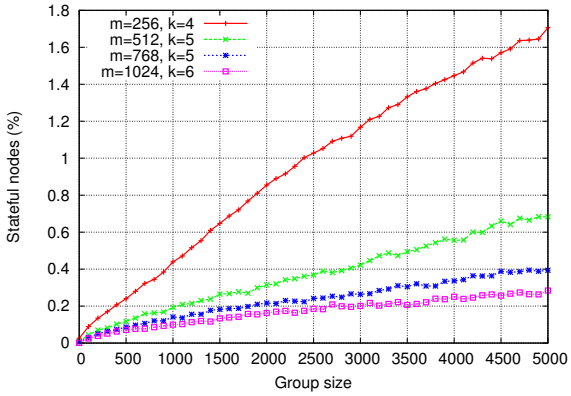


Figure 5.4: State requirements of IBF switching. Network size 5000, $fpp_{max} = 0.5\%$, variable m and k.

In the remainder of this chapter, we present results for $fpp = 0.5\%$ as it provides a good trade-off between forwarding efficiency (quickly passes 96% and converges to 98%) and state requirements (requires half the nodes compared to $fpp = 0.1\%$). In Figure 5.4, we show the state requirements for various values of m and k with $fpp_{max} = 0.05\%$ in a 5000 node graph. As m increases, the fraction of stateful nodes decreases as expected: as IBF size grows, more tree links can be added to the iBF without the fpp passing the selected threshold. Notice that with $m = 1024$ bits, IBF switching for multicast requires state at around 0.4% of the nodes, i.e., we can almost broadcast (e.g., transmit a firmware update to all routers) with multicast forwarding state at only $5000 \times 0.4\% = 20$ nodes.

5.5.2 Comparison with IP multicast forwarding schemes

We now compare the multicast forwarding state requirements of IBF switching and various schemes proposed for reducing IP multicast forwarding state. Specifically, we compare IBF switching for multicast against REUNITE [60], the semi-stateful Xcast [62] and plain IP Multicast. REUNITE reduces the amount of multicast forwarding state by installing forwarding entires only at branching points of the multicast tree; non-branching points forward packets using the existing unicast information [60]. This significantly reduces multicast forwarding state for sparse trees, but its benefits are minor in dense trees. Explicit Multicast (Xcast) adds the addresses of multicast receivers in packet headers [61], with routers forwarding multicast packets based on that information only. Xcast is a stateless scheme and therefore scales with respect to the number of multicast groups. However, due to the limited capacity of the packet header, it is only suitable for small groups. Yang and Liao proposed a semi-stateful version of Xcast [62] where multicast forwarding state is installed in few, carefully selected routers, and packets are forwarded among these routers via Xcast. Finally, plain IP multicast places multicast forwarding state at each node of the multicast tree.

Figure 5.5 presents the fraction of stateful nodes as a function of multicast group size in a 5000-node graph. We chose $m = 1024$ bits and $fpp_{max} = 0.5\%$ for IBF switching for multicast. For semi-stateful Xcast we considered a header of the same size, i.e., 1024 bits, and then considered two variants: (i) in XCAST-8 the header may contain up to 8×128-bit host addresses and (ii) in XCAST-32 the header may contain up to 32×32-bit host addresses. XCAST-8 targets a possible deployment of Xcast in IPv6 networks while XCAST-32 targets a hypothetical deployment in IPv4 networks. XCAST-32 is impossible to realize of course, as IPv4 headers cannot exceed 40 bytes, but we present the results for completeness. The simulation setup is the same as in the previous section. Due to wide difference between the various schemes, the Y-axis in Figure 5.5 is in log scale. In all schemes, multicast forwarding state grows linearly with respect to group size. However, IBF switching achieves a large state reduction compared to the other schemes. For instance, for a 2000 node group, IBF switching reduces the number of stateful

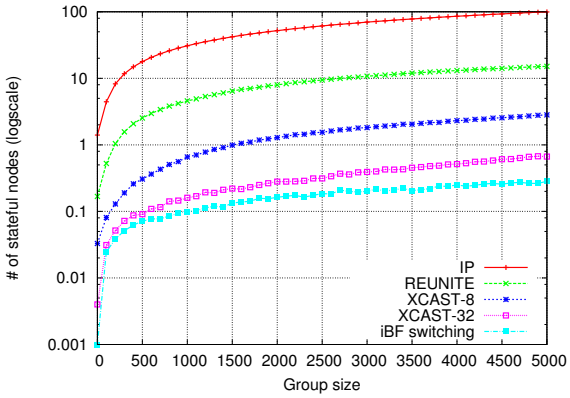


Figure 5.5: Fraction (%) of stateful nodes against group size for IP Multicast, REUNITE, semi-stateful Xcast and IBF switching (log scale)

nodes by 99.6% compared to IP multicast, 98% compared to REUNITE, 87% compared to XCAST-8 and 42% compared to XCAST-32.

Figure 5.6 presents the same data but only for semi-stateful Xcast and IBF switching, with the Y axis in linear scale. Although IBF switching and semi-stateful Xcast both use a combination of source-routing and stateful nodes, IBF switching requires far less state due to the efficient encoding of the Bloom filter, even though we restrict the number of tree links so that $fpp_{max} = 0.5\%$. Even XCAST-32 requires double the state of IBF switching. Taking into account that REUNITE and Xcast also require unicast forwarding state, it is clear that an IBF-based forwarding node requires far less overall state. Hence, although we sacrificed the fully stateless nature of IBF multicast, it both scales with respect to multicast group size and remains a far more scalable solution in terms of the number of multicast groups supported.

5.6 Conclusion

We addressed the scalability issues of IBF multicast with respect to multicast group size. We examined the option of sacrificing the fully stateless operation of Bloom filter-based forwarding in order to minimize the redundant traffic caused by false positives in large multicast groups. We presented a IBF switching scheme and an algorithm for the selection of IBF switching points. Our evaluation through

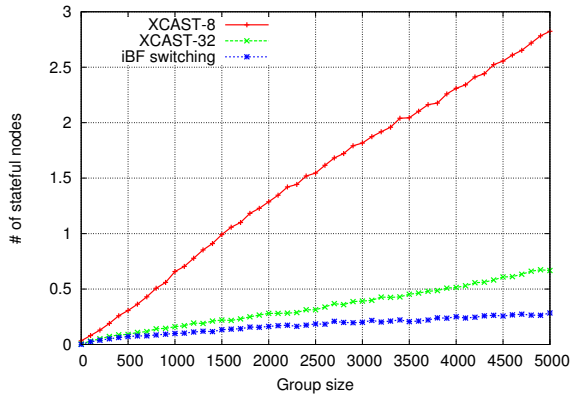


Figure 5.6: Fraction (%) of stateful nodes against group size for semi-stateful Xcast and IBF switching (linear scale)

simulation showed that our solution scales to groups of arbitrary size with a (minimal) cost of placing multicast forwarding state at no more than $0.5\% - 2.5\%$ of network nodes. Moreover, compared with other multicast schemes that reduce forwarding state, IBF switching for multicast achieves state reductions varying from 87% to 99.6%. Hence, even though IBF switching places state inside the network, it provides far better scalability properties with respect to the number of groups supported, compared to alternative solutions.

Chapter 6

Conclusions and future work

This dissertation addresses the issue of multicast forwarding scalability in two widely different Future Information-Centric Network architectures, namely Content-Centric Networking (CCN) and Publish-Subscribe Internet (PSI). The common ground in the proposed solutions is the use and extension of Bloom filter-based packet forwarding mechanisms.

In CCN, we proposed a semi-stateless forwarding scheme that integrates Bloom filter-based forwarding in the architecture, with the aim of mitigating forwarding state requirements at routers. Our proposed scheme, presented in Chapters 3 and 4, can achieve a reduction of the forwarding state at the levels of 50%-70% at the cost of bandwidth overheads of 1.5%-12%. The semi-stateless forwarding scheme respects the core CCN architecture, and maintains its fully distributed operation in routing and forwarding, even though we integrate a source-routing technique for Data forwarding.

In PSI, we proposed an IBF switching scheme in order to address the poor scalability properties of Bloom filter-based forwarding with respect to network/multicast group size. We utilized the centralized operation of control-plane functionalities in PSI, i.e., the Rendezvous and Topology Management and Path Formation functions, and proposed a centralized algorithm for selecting IBF switching points. We described this solution in Chapter 5 and showed that, although we sacrificed the full stateless nature of Bloom filter-based forwarding, the amount of redundant traffic is effectively reduced while the required forwarding state in

routers is by far less than that of similar multicast forwarding technologies.

Ultimately, there is an underlying pattern in the solutions proposed by this dissertation: the use of semi-stateless Bloom filter-based forwarding. Fully stateful forwarding, as in CCN, scales greatly with respect to network and/or mutlicast group size but has scalability constraints with regards to the multicast forwarding state kept at routers. On the other hand, fully stateless forwarding, as in PSI with IBFs, scales greatly with respect to the number of data paths and active multicast groups but scales poorly with respect to the network/multicast group size. In both cases, our approach was to improve each system's scalability constraints by relaxing the forwarding schemes; introducing *stateless-ness* in a fully stateful forwarding architecture and introducing *stateful-ness* in a fully stateless forwarding architecture. In the context of CCN, that required the contrivance of a distributed mechanisms for IBF construction and state placement, whereas in the context of PSI we approached the same ideas in a centralized manner. In both cases, we achieved significant gains towards our target, the improvement of forwarding scalability, at the expense of unavoidable, but low, costs. In the case of CCN, and generally fully distributed multicast forwarding schemes, we paid the penalty of bandwidth overhead for two reasons. First, since we relaxed the model, multicast requests (i.e., Interests) travel more hops compared to the optimal case. In addition, the probabilistic nature of IBFs adds Data overhead. In the case of PSI, we sacrificed the fully stateless nature of multicast forwarding, meaning that now there is an upper bound of the supported active multicast sessions in contrast to the unbounded one before, with the exact amount depending on the capacity of the deployed hardware. Yet, that is the penalty we need to pay in order to scale to large networks/multicast groups without the fear of collapsing from storms of false traffic.

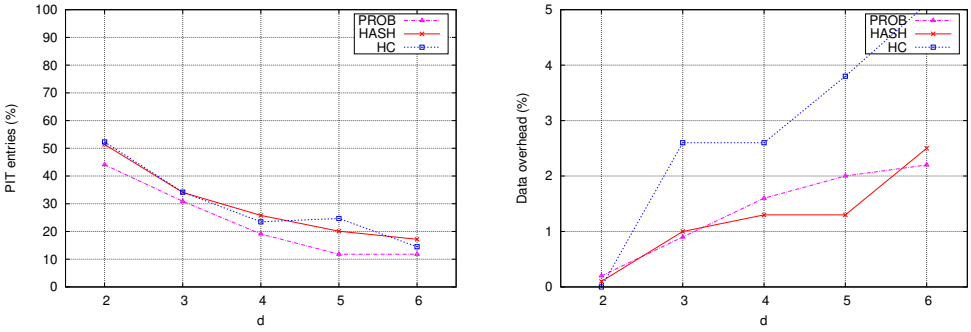
A general question, that possibly sets a possible research agenda, is whether the gains (and their respective trade-offs) offered by the presented solutions are adequate enough for the envisaged Future Information-Centric Internet. For example, is a 70% reduction of the PIT enough? Would placing multicast forwarding state at 0.5% of PSI routers suffice for a Internet-wide multicast distribution? Is

that enough or do we need further improvements? A clear and honest answer to these questions is *we don't really know yet*. Although the proposed solutions do improve the forwarding operation in CCN and PSI, it is not clear whether the remaining 30% of the PIT would actually fit inside a router's memory, or, if the delays for installing IBFs at 0.5% of a network with tens of thousands of nodes would not affect user experience. Although ICN is near its 10th birthday,[1] the research community has still not made concrete conclusions for several critical questions: what are the actual requirements for ICN, how would the applications perform and how would the users behave. The ICNRG has identified the need to concretely define *"metrics that make it possible to evaluate ICN implementations in a consistent manner"* [34]. After completing this dissertation, our opinion is that before investigating further improvements in ICN architectures, these metrics (and accompanied requirements) must be defined, to allow evaluating whether any proposed solution suffices, and if not, how far is it from achieving its goal. Without these metrics, any proposed solution lacks the answer to an obvious question: is it good enough? We therefore believe, that top priority for any future work, would be a in-depth investigation and definition of the required evaluation metrics and acceptable thresholds.

[1] Indeed, seminal research that led to ICN is older than a decade, for example [5, 7, 8, 23]. The survey paper of Xylomenos et al. provides a thorough review [33].

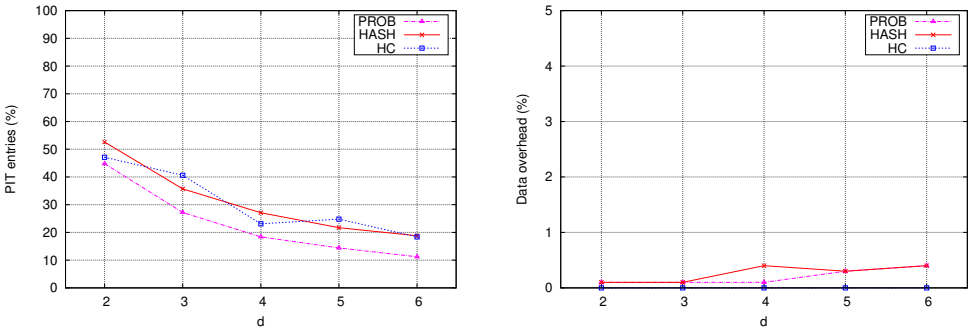
Appendix A

Evaluation Results



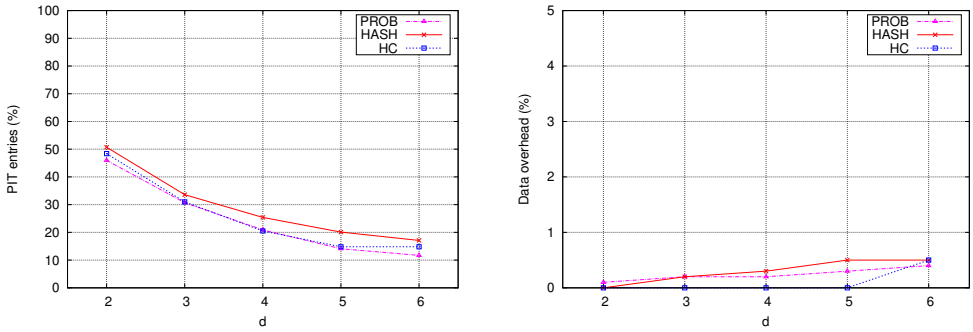
(a) PIT entries (b) Data overhead

Figure A.1: Performance results for unicast traffic in topology AS-224.



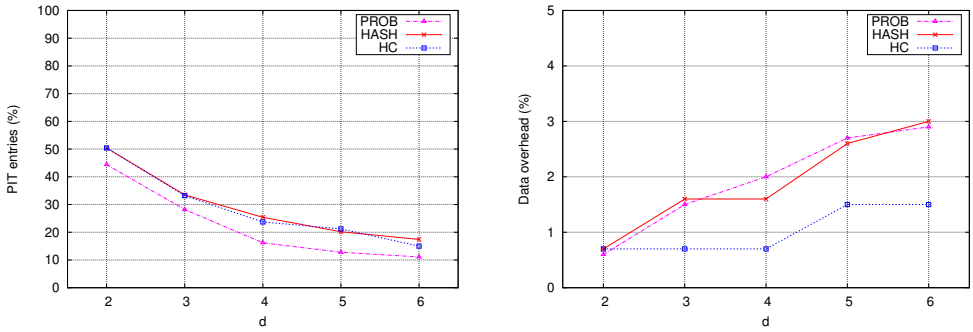
(a) PIT entries (b) Data overhead

Figure A.2: Performance results for unicast traffic in topology AS-3967.



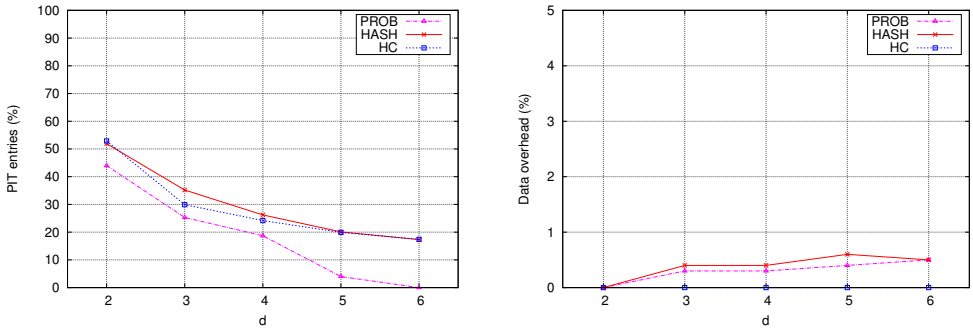
(a) PIT entries (b) Data overhead

Figure A.3: Performance results for unicast traffic in topology AS-1755.



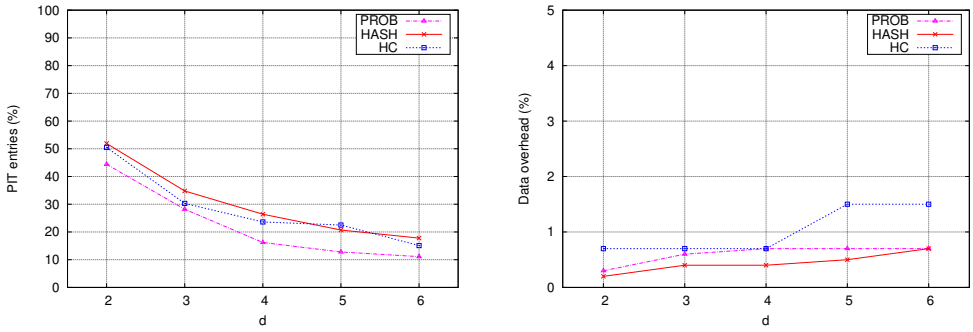
(a) PIT entries (b) Data overhead

Figure A.4: Performance results for unicast traffic in topology AS-1221.



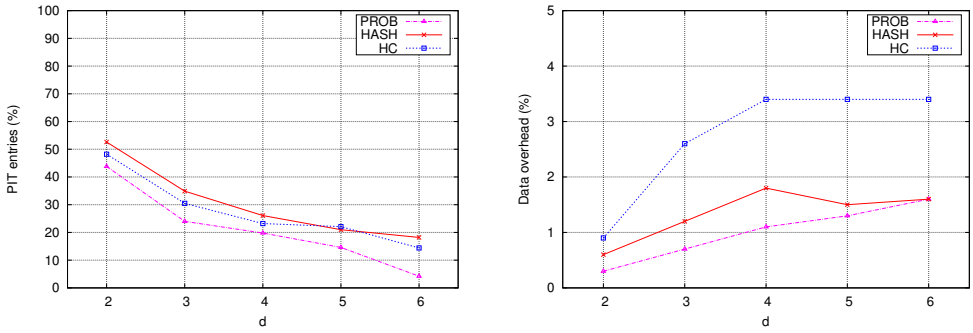
(a) PIT entries (b) Data overhead

Figure A.5: Performance results for unicast traffic in topology AS-6461.



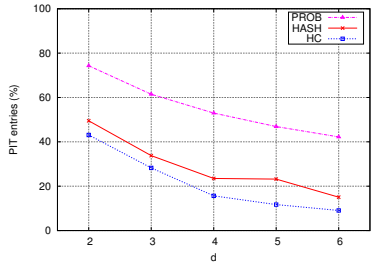
(a) PIT entries (b) Data overhead

Figure A.6: Performance results for unicast traffic in topology scale-free-50.

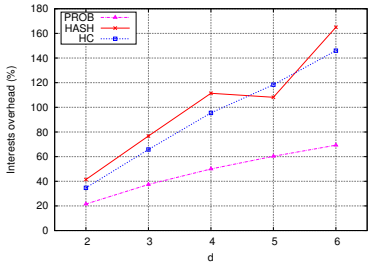


(a) PIT entries (b) Data overhead

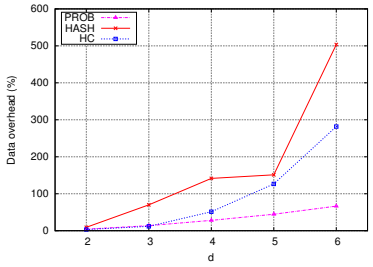
Figure A.7: Performance results for unicast traffic in topology scale-free-100.



(a) PIT entries

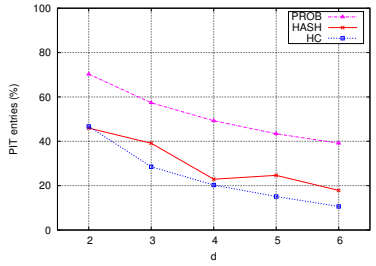


(b) Interests overhead

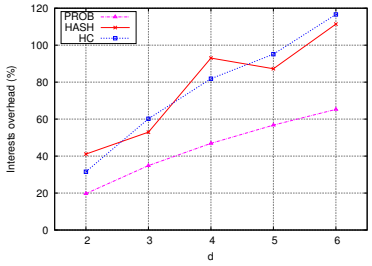


(c) Data overhead

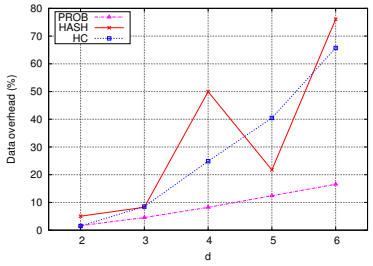
Figure A.8: Multicast performance results for uniform distribution of group size in topology AS-224.



(a) PIT entries

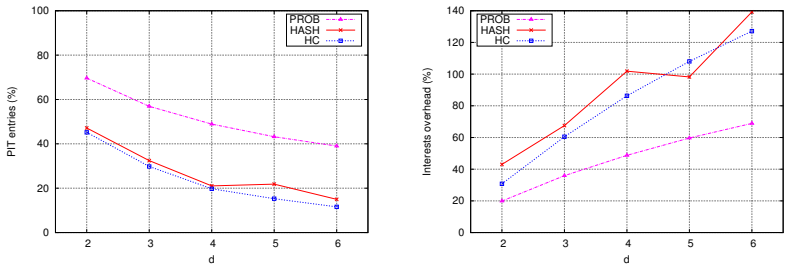


(b) Interests overhead



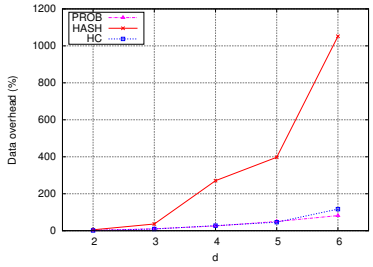
(c) Data overhead

Figure A.9: Multicast performance results for uniform distribution of group size in topology AS-3967.



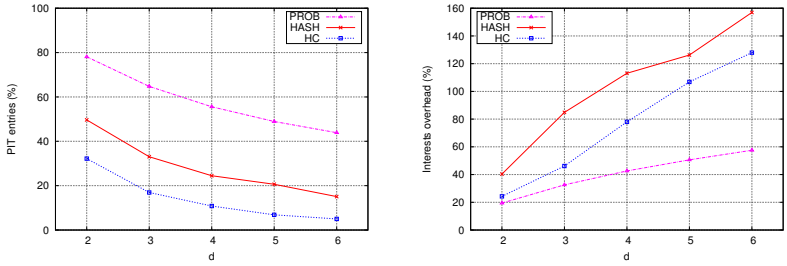
(a) PIT entries

(b) Interests overhead



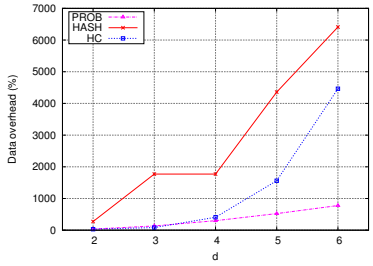
(c) Data overhead

Figure A.10: Multicast performance results for uniform distribution of group size in topology AS-1755.



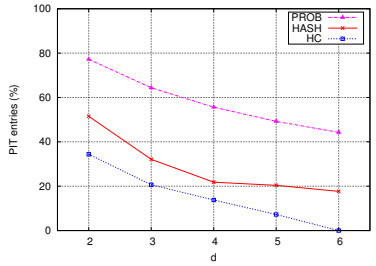
(a) PIT entries

(b) Interests overhead

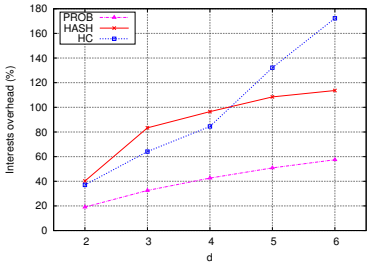


(c) Data overhead

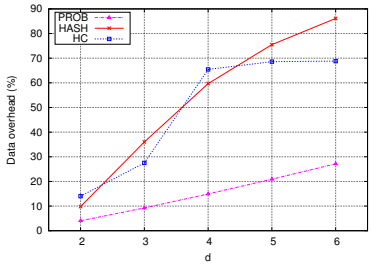
Figure A.11: Multicast performance results for uniform distribution of group size in topology AS-1221.



(a) PIT entries

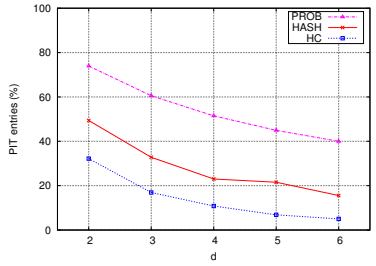


(b) Interests overhead

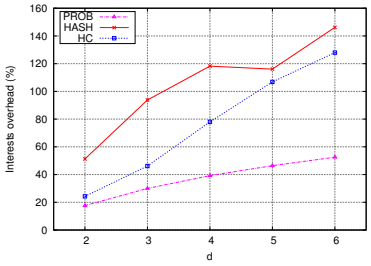


(c) Data overhead

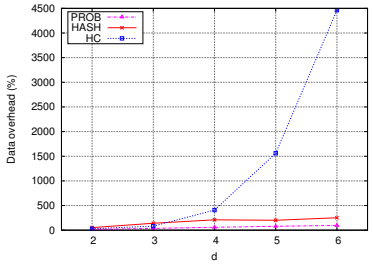
Figure A.12: Multicast performance results for uniform distribution of group size in topology AS-6461.



(a) PIT entries

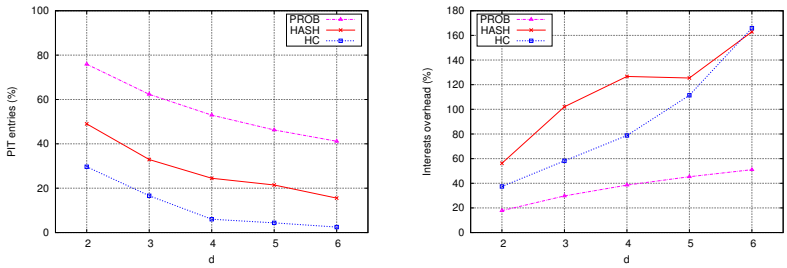


(b) Interests overhead



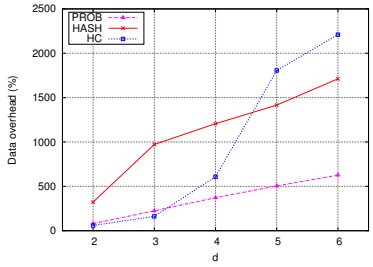
(c) Data overhead

Figure A.13: Multicast performance results for uniform distribution of group size in topology scale-free-50.



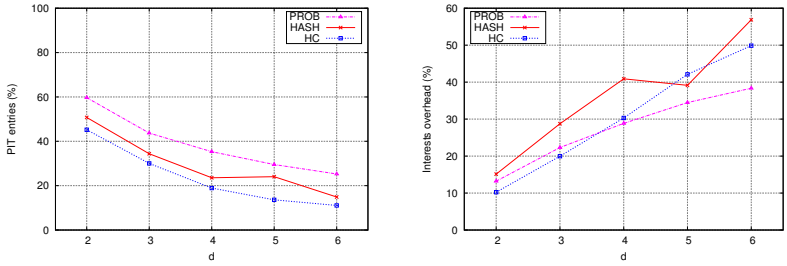
(a) PIT entries

(b) Interests overhead



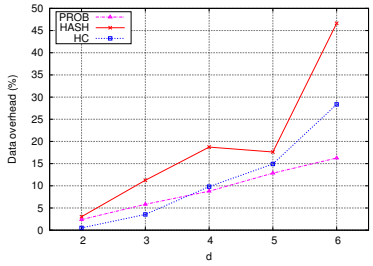
(c) Data overhead

Figure A.14: Multicast performance results for uniform distribution of group size in topology scale-free-100.



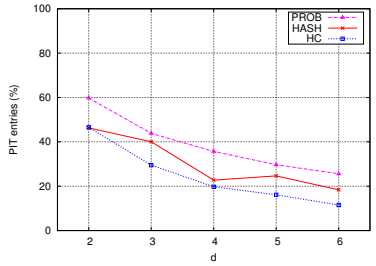
(a) PIT entries

(b) Interests overhead

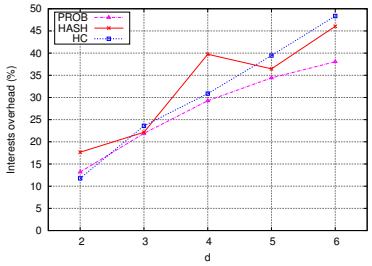


(c) Data overhead

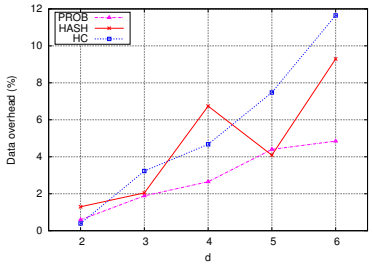
Figure A.15: Multicast performance results for Zipf distribution of group size in topology AS-224.



(a) PIT entries

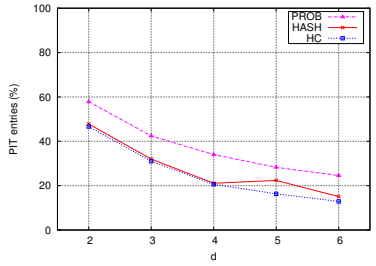


(b) Interests overhead

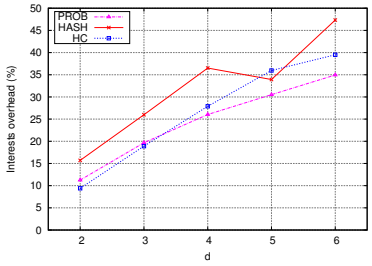


(c) Data overhead

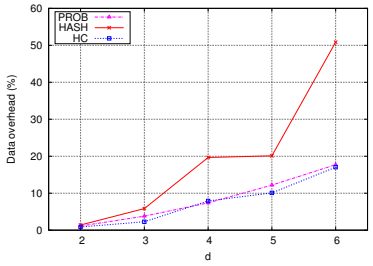
Figure A.16: Multicast performance results for Zipf distribution of group size in topology AS-3967.



(a) PIT entries

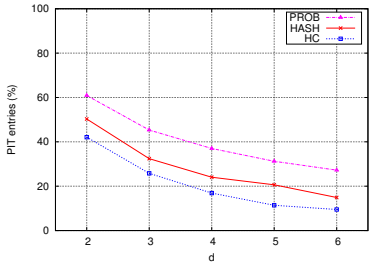


(b) Interests overhead

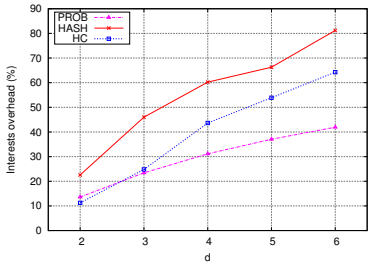


(c) Data overhead

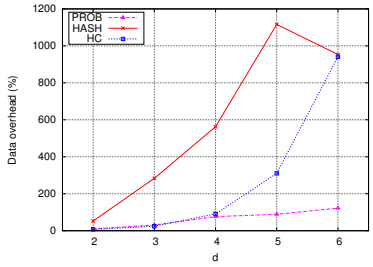
Figure A.17: Multicast performance results for Zipf distribution of group size in topology AS-1755.



(a) PIT entries

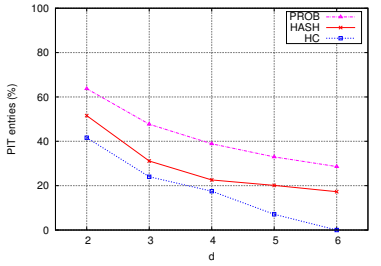


(b) Interests overhead

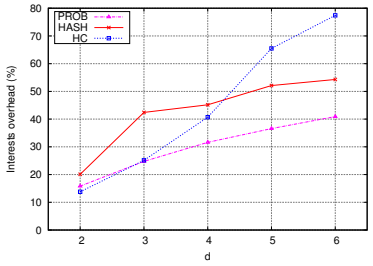


(c) Data overhead

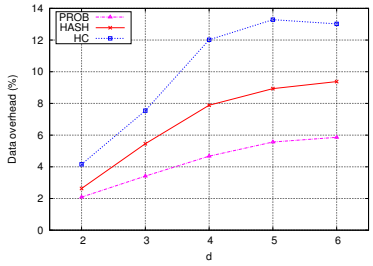
Figure A.18: Multicast performance results for Zipf distribution of group size in topology AS-1221.



(a) PIT entries

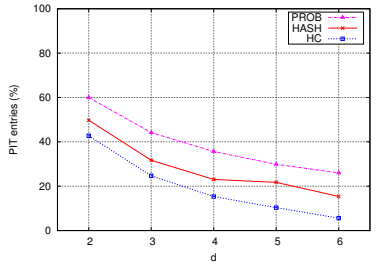


(b) Interests overhead

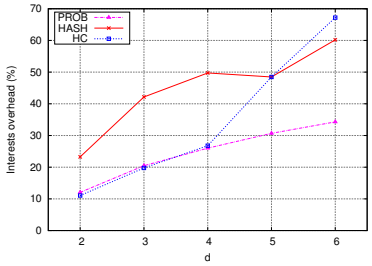


(c) Data overhead

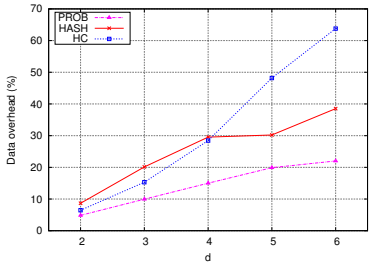
Figure A.19: Multicast performance results for Zipf distribution of group size in topology AS-6461.



(a) PIT entries

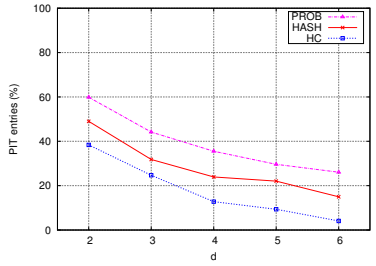


(b) Interests overhead

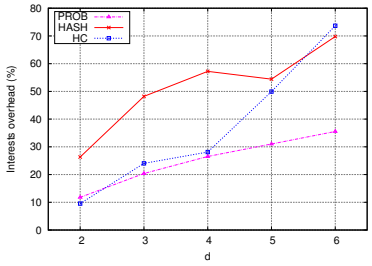


(c) Data overhead

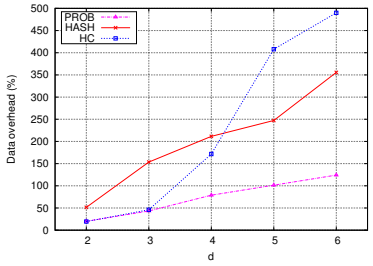
Figure A.20: Multicast performance results for Zipf distribution of group size in topology scale-free-50.



(a) PIT entries

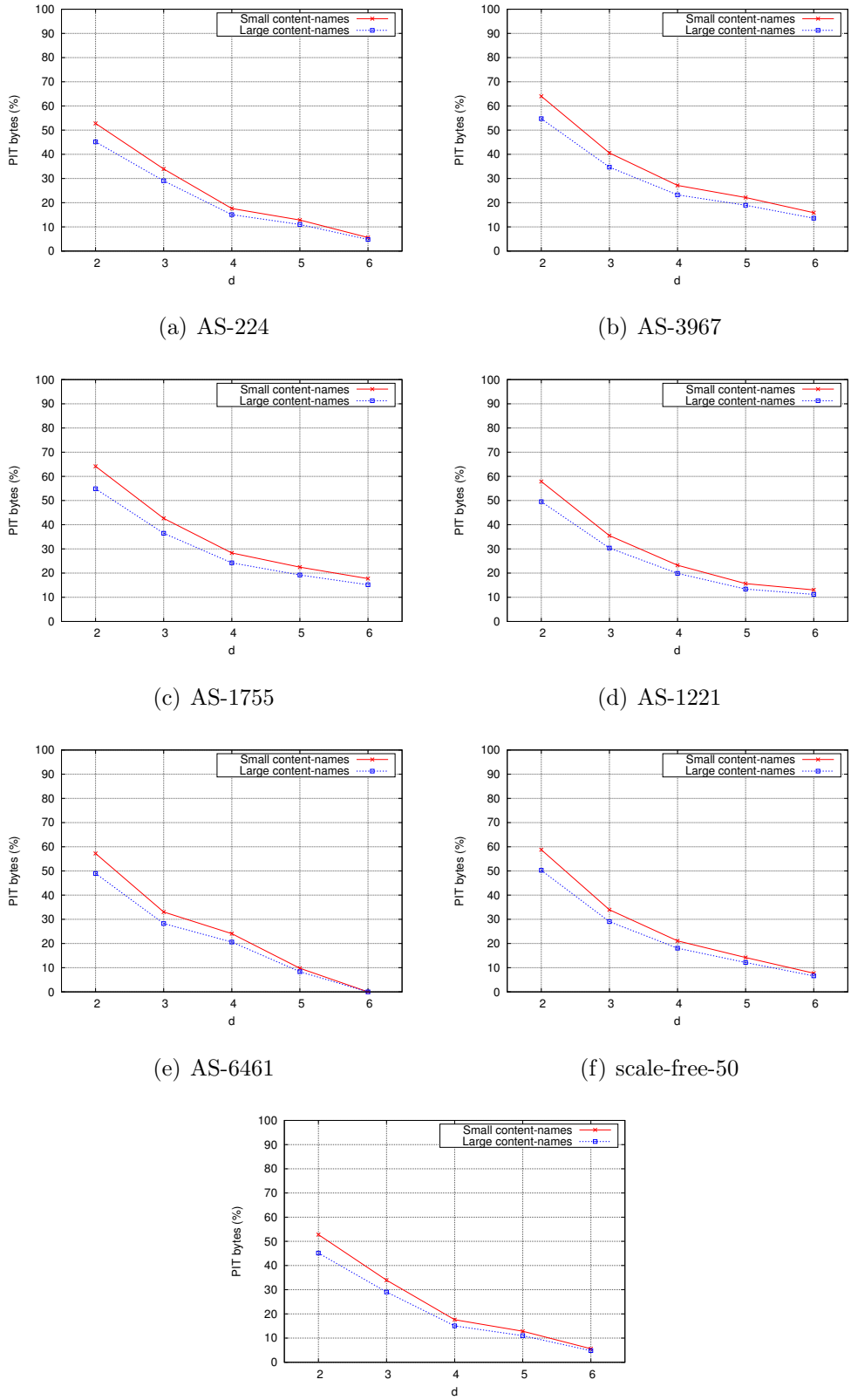


(b) Interests overhead



(c) Data overhead

Figure A.21: Multicast performance results for Zipf distribution of group size in topology scale-free-100.



(a) AS-224

(b) AS-3967

(c) AS-1755

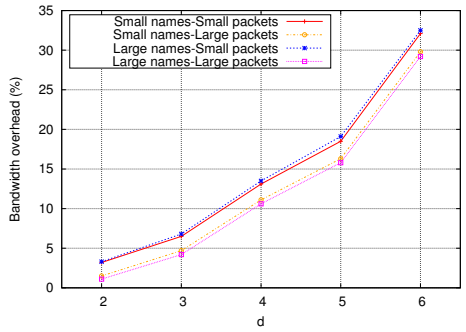
(d) AS-1221

(e) AS-6461

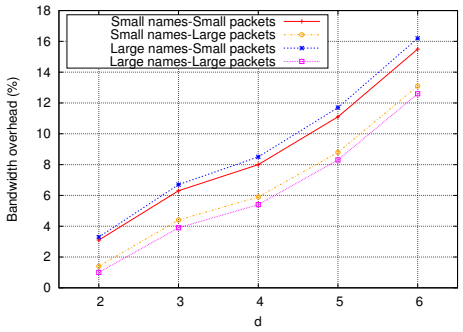
(f) scale-free-50

(g) scale-free-100

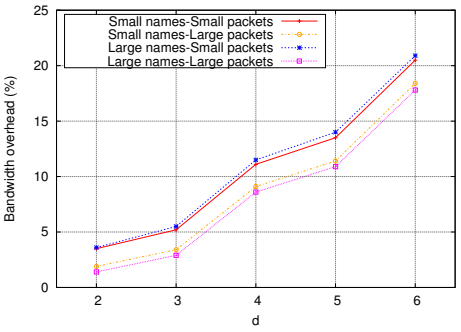
Figure A.22: Estimated PIT size (in bytes) for small (20-bytes) and large (56-bytes), Zipf distribution of group sizes using the Hop Counter-based policy.



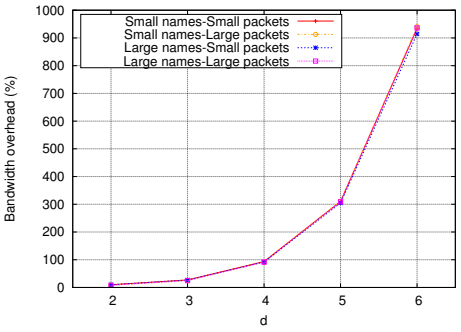
(a) AS-224



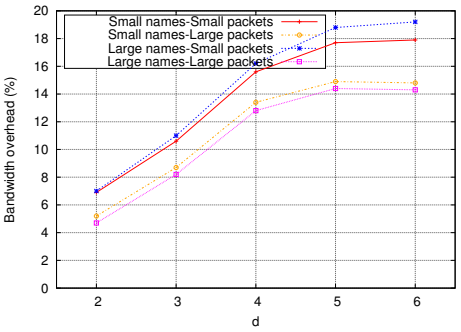
(b) AS-3967



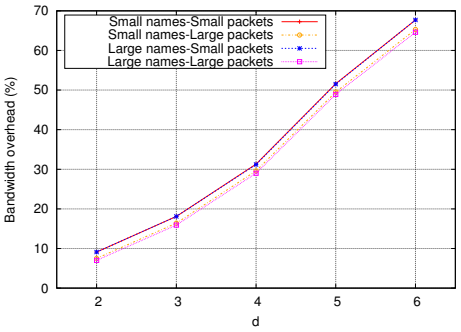
(c) AS-1755



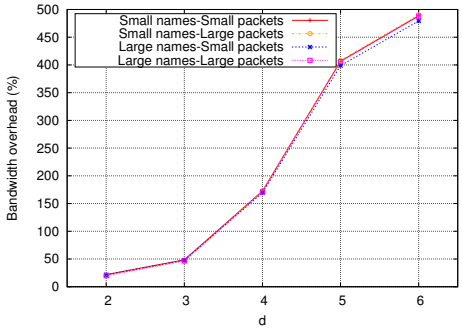
(d) AS-1221



(e) AS-6461

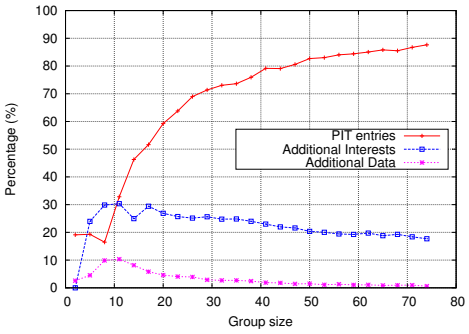


(f) scale-free-50

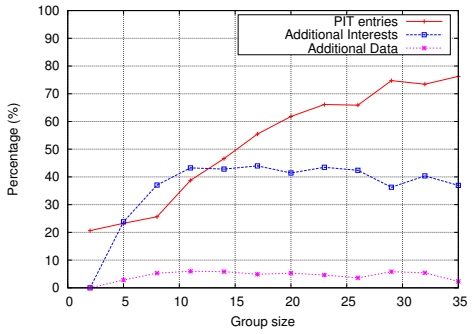


(g) scale-free-100

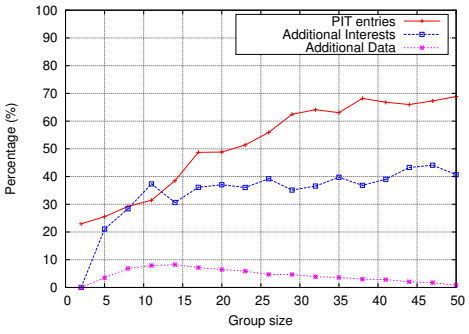
Figure A.23: Multicast bandwidth overhead with a Zipf distribution of group sizes in the HC policy. Small content-names (36-byte Interests) and large-content names (70-byte Interests), small (1500 bytes) or large (jumbo) frames (7500 bytes) Data packets.



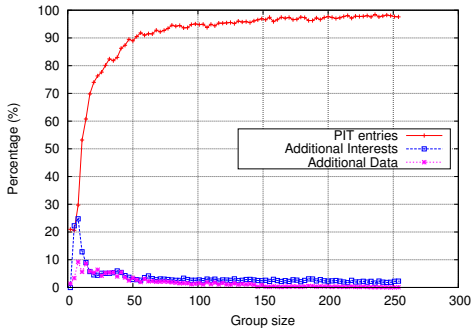
(a) AS-224



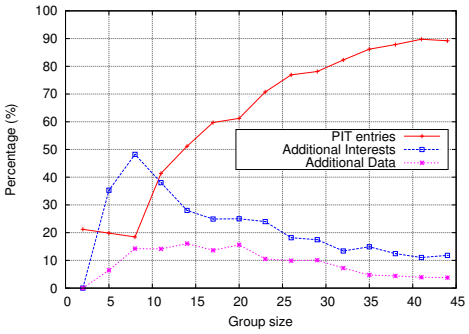
(b) AS-3967



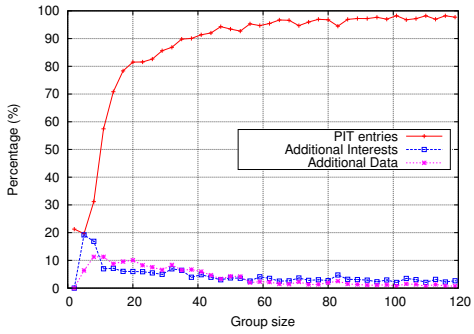
(c) AS-1755



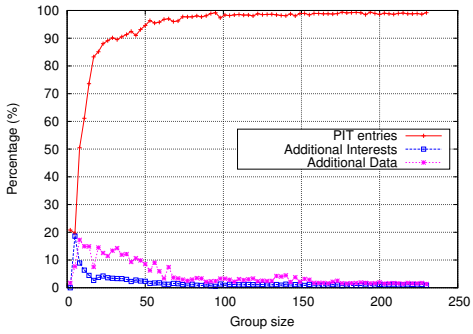
(d) AS-1221



(e) AS-6461



(f) scale-free-50



(g) scale-free-100

Figure A.24: Performance results with dynamic adaptation of d with respect to group size.

Appendix B

Acronyms

AS Autonomous System
CCN Content Centric Networking
CDN Content Distribution Networking
CONET Content Network
CS Content Store
DHT Distributed Hash Table
FIA Future Internet Architecture
FIB Forwarding Information Base
FID Forwarding Identifier
FN Forwarding Nodes
FS Forwarding System
IBF In-packet Bloom Filter
ICN Information Centric Network
ICNRG Information Centric Network Research Group
IRTF Internet Research Task Force
LID Link Identifier
MFT Multicast Forwarding Table
MTU Maximum Transfer Unit
NAT Network Address Translation
NDN Named Data Networking

NSF National Science Foundation

PCE Path Computation Element

PI Persistent Interest

PSI Publish Subscribe Internet

RN Rendezvous Node

RVP Rendezvous Point

RVS Rendezvous System

RTT Round Trip Time

SDN Software Defined Networking

TM Topology Manager

TMPFS Topology Manager and Path Formation System

XIA eXpressive Internet Architecture

Bibliography

[1] "CISCO Visual Networking Index: Forecast and Methodology, 20122017," 2016. [Online]. Available: http://www.cisco.com

[2] K. Moore, "Things that nats break," 2004. [Online]. Available: http://web.mit.edu/6.033/2002/wwwdocs/papers/what-nats-break.html

[3] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "Stun-simple traversal of user datagram protocol (udp) through network address translators (nats)," RFC 3489, IETF, Mar, Tech. Rep., 2003.

[4] P. Vixie, "What dns is not," *Communications of the ACM*, vol. 52, no. 12, pp. 53–47, 2009.

[5] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet." in *USITS*, vol. 1, 2001, pp. 4–4.

[6] P. Francis and R. Gummadi, "IPNL: A NAT-extended internet architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, 2001, pp. 69–80.

[7] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4. ACM, 2002, pp. 73–86.

[8] M. Walfisha, H. Balakrishnana, and S. Shenkerb, "Untangling the web from dns," *Proc. of NSDI*, 2004.

[9] M. Walfish, J. Stribling, M. N. Krohn, H. Balakrishnan, R. Morris, and S. Shenker, "Middleboxes no longer considered harmful." in *Proc. of OSDI*, vol. 4, 2004, pp. 15–15.

[10] S. Zhuang, K. Lai, I. Stoica, R. Katz, and S. Shenker, "Host mobility using an internet indirection infrastructure," *Wireless Networks*, vol. 11, no. 6, pp. 741–756, 2005.

[11] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host identity protocol," RFC 5201, 2008.

[12] R. Braden, T. Faber, and M. Handley, "From protocol stack to protocol heap: role-based architecture," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 17–22, 2003.

[13] D. Clark, R. Braden, A. Falk, and V. Pingali, "Fara: Reorganizing the addressing architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 4, 2003, pp. 313–321.

[14] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, "A layered naming architecture for the internet," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, 2004, pp. 343–352.

[15] D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawski, "Overlay networks and the future of the Internet," *Communications and Strategies*, vol. 63, p. 109, 2006.

[16] A. Feldmann, "Internet clean-slate design: what and why?" *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, 2007.

[17] J. Rexford and C. Dovrolis, "Future internet architecture: clean-slate versus evolutionary research," *Communications of the ACM*, vol. 53, no. 9, pp. 36–40, 2010.

[18] "MobilityFirst Future Internet Architecture Project," 2016. [Online]. Available: http://mobilityfirst.winlab.rutgers.edu/

[19] "eXpressive Internet Architecture," 2016. [Online]. Available: https://www.cs.cmu.edu/ xia/

[20] "NEBULA: Future Internet Architecture," 2016. [Online]. Available: http://nebula-fia.org/

[21] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil, "An architecture for internet data transfer." in *Proc. of NSDI*, 2006.

[22] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: Provider portal for applications," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, 2008, pp. 351–362.

[23] A. Carzaniga and A. L. Wolf, "Content-based networking: A new communication infrastructure," in *Developing an Infrastructure for Mobile and Wireless Systems*, 2001, pp. 59–68.

[24] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: network routing inspired by dhts," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 351–362, 2006.

[25] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "Rofl: routing on flat labels," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 363–374, 2006.

[26] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, 2007, pp. 181–192.

[27] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[28] "Content-Centric Networking project," 2016. [Online]. Available: https://www.parc.com/work/focus-area/content-centric-networking/

[29] "Named-Data Networking project," 2016. [Online]. Available: http://named-data.net/

[30] "PSIRP: Publish-Subscribe Internet Routing Paradigm," 2016. [Online]. Available: http://www.psirp.org/

[31] "PURSUIT: Pursuing a Pub/Sub Internet," 2016. [Online]. Available: http://www.fp7-pursuit.eu/PursuitWeb/

[32] "4WARD project," 2016. [Online]. Available: http://www.4ward-project.eu/

[33] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[34] "Information-Centric Networking Research Group," 2016. [Online]. Available: https://irtf.org/icnrg

[35] D. Trossen, M. Sarela, and K. Sollins, "Arguments for an information-centric internetworking architecture," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 26–33, 2010.

[36] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of ACM CoNEXT*, 2009, pp. 1–12.

[37] J. J. Garcia-Luna-Aceves, "Name-based content routing in information centric networks using distance information," in *Proc. of ACM SIGCOMM ICN*, 2014, pp. 7–16.

[38] C. Yi, J. Abraham, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "On the role of routing in named data networking," in *Proc. of ACM SIGCOMM ICN*, 2014, pp. 27–36.

[39] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Elsevier Computer Communications*, vol. 36, no. 7, pp. 779 – 791, 2013.

[40] D. Perino and M. Varvello, "A reality check for content centric networking," in *Proc. of ACM SIGCOMM ICN workshop*, 2011, pp. 44–49.

[41] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Computer Communication Revieww*, vol. 42, no. 3, pp. 62–67, 2012.

[42] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. Siris, and G. Polyzos, "Caching and mobility support in a publish-subscribe internet architecture," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 52–58, 2012.

[43] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[44] K. V. Katsaros, N. Fotiou, X. Vasilakos, C. N. Ververidis, C. Tsilopoulos, G. Xylomenos, and G. C. Polyzos, "On inter-domain name resolution for information-centric networks," in *Proc. of IFIP NETWORKING 2012*, 2012, pp. 13–26.

[45] B. A. Alzahrani, M. J. Reed, J. Riihijärvi, and V. G. Vassilakis, "Scalability of information centric networking using mediated topology management," *Journal of Network and Computer Applications*, vol. 50, pp. 126–133, 2015.

[46] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the ip multicast service and architecture," *IEEE Network*, vol. 14, no. 1, pp. 78–88, 2000.

[47] P. Jokela, A. Zahemszky, C. Esteve-Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking," in *Proc. of ACM SIGCOMM*, 2009.

[48] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, pp. 131–155, 2012.

[49] M. Särelä, C. E. Rothenberg, T. Aura, A. Zahemszky, P. Nikander, and J. Ott, "Forwarding anomalies in bloom filter-based multicast," in *Proc. of IEEE INFOCOM*, 2011, pp. 2399–2407.

[50] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard, "VoCCN: voice-over content-centric networks," in *Proc. of ACM ReArch workshop*, 2009, pp. 1–6.

[51] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "ACT: audio conference tool over named data networking," in *Proc. of ACM SIGCOMM ICN workshop*, 2011, pp. 68–73.

[52] P. Gusev and J. Burke, "Ndn-rtc: Real-time videoconferencing over named data networking," in *Proc. of ACM SIGCOMM ICN*, 2015, pp. 117–126.

[53] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, and D. Perino, "Exploit the known or explore the unknown?: hamlet-like doubts in icn," in *Proc. of ACM SIGCOMM ICN workshop*, 2012, pp. 7–12.

[54] W. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks," in *Proc. of IFIP NETWORKING*, 2012, pp. 27–40.

[55] G. Carofiglio, M. Gallo, and L. Muscariello, "ICP: Design and evaluation of an interest control protocol for content-centric networking," in *Proc. of IEEE INFOCOM NOMEN workshop*, 2012.

[56] H. Yuan, T. Song, and P. Crowley, "Scalable NDN forwarding: Concepts, issues and principles," in *Proc. of IEEE ICCCN*, 2012, pp. 1–9.

[57] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini, "CONET: a content centric inter-networking architecture," in *Proc. of ACM SIGCOMM ICN workshop*, 2011, pp. 50–55.

[58] C. Tsilopoulos, I. Gasparis, G. Xylomenos, and G. Polyzos, "Efficient real-time information delivery in future internet publish-subscribe networks," in *Proc. of IEEE ICNC*, 2013, pp. 856–860.

[59] D. Li, Y. Li, J. Wu, S. Su, and J. Yu, "Esm: Efficient and scalable data center multicast routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 944–955, 2012.

[60] I. Stoica, T. E. Ng, and H. Zhang, "REUNITE: A recursive unicast approach to multicast," in *Proc. of IEEE INFOCOM*, vol. 3, 2000, pp. 1644–1653.

[61] D. Ooms, O. Paridaens, R. Boivie, N. F. Y. Imai, and W. Livens, "Explicit multicast (xcast) basic specification," Internet Draft, June, Tech. Rep., 2004.

[62] D.-N. Yang and W. Liao, "Protocol design for scalable and adaptive multicast for group communications," in *Proc. of IEEE ICNP*, 2008, pp. 33–42.

[63] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[64] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting ip multicast," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, 2006, pp. 15–26.

[65] M. Yu, A. Fabrikant, and J. Rexford, "Buffalo: Bloom filter forwarding architecture for large organizations," in *Proc. of ACM CoNEXT*, 2009, pp. 313–324.

[66] X. Tian, Y. Cheng, and B. Liu, "Design of a scalable multicast scheme with an application-network cross-layer approach," *IEEE Transactions on Multimedia*, vol. 11, no. 6, pp. 1160–1169, 2009.

[67] S. Rizvi, A. Zahemszky, and T. Aura, "Scaling bloom filter based multicast with hierarchical tree splitting," in *Proc. of IEEE ICC*, 2012, pp. 2790–2795.

[68] Caida. [Online]. Available: http://caida.org

[69] P. S. Almeida, C. Baquero, N. Preguiça, and D. Hutchison, "Scalable bloom filters," *Information Processing Letters*, vol. 101, no. 6, pp. 255–261, 2007.

[70] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo, "The dynamic bloom filters," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 120–133, 2010.

[71] J. Tapolcai, J. Biro, P. Babarczi, A. Gulyas, Z. Heszberger, and D. Trossen, "Optimal false-positive-free bloom filter design for scalable multicast forwarding," *IEEE/ACM Transactions on Networking*, vol. 23, no. 6, pp. 1832–1845, 2015.

[72] C. Stais, Y. Thomas, G. Xylomenos, and C. Tsilopoulos, "Networked music performance over information-centric networks," in *Proc. of IEEE ICC IIMC workshop*, 2013, pp. 647–651.

[73] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. Ramakrishnan, "COPSS: An efficient content oriented publish/subscribe system," in *Proc. of ACM/IEEE ANCS*, 2011, pp. 99–110.

[74] H. Dai, B. Liu, Y. Chen, and Y. Wang, "On pending interest table in named data networking," in *Proc. of ACM/IEEE ANCS*, 2012, pp. 211–222.

[75] M. Varvello, D. Perino, and L. Linguaglossa, "On the design and implementation of a wire-speed pending interest table," in *Proc. of IEEE INFOCOM NOMEN workshop*, 2013.

[76] W. You, B. Mathieu, P. Truong, J. Peltier, and G. Simon, "DiPIT: A distributed bloom-filter based PIT table for CCN nodes," in *Proc. of IEEE ICCCN*, 2012, pp. 1–7.

[77] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *Proc. of ACM SIGCOMM ICN workshop*, 2011, pp. 13–18.

[78] ——, "Scaling bloom filter-based multicast via filter switching," in *Proc. of IEEE ISCC*, 2013.

[79] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[80] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *Proc. of ACM SIGCOMM*, 2002, pp. 133–145.

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

[82] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive http streaming of live content," in *Proc. of IEEE WoWMoM*, 2011, pp. 1–8.

[83] Optimal segment length for adaptive streaming formats like MPEG-DASH & HLS. [Online]. Available: http://www.dash-player.com/blog/2015/04/using-the-optimal-segment-length-for-adaptive-streaming-formats-like-mpeg-dash-hls/

[84] T. W. Cho, M. Rabinovich, K. Ramakrishnan, D. Srivastava, and Y. Zhang, "Enabling content dissemination using efficient and scalable multicast," in *Proc. of IEEE INFOCOM*, 2009, pp. 1980–1988.

[85] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1489–1499, 2002.