ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
Department of Computer Science

An information-centric overlay network architecture for content distribution and mobility support

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

in

Computer Science

by

Konstantinos V. Katsaros

Athens
July 2010

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My advisor, Prof. George C. Polyzos, taught me to ask the right questions and focus on the essence of things; an invaluable lesson in research, but in life too. Always supportive and with a good mood, he created an excellent environment that any other PhD student would envy. I deeply thank him for all that he gave me.

It has been also an honor to work in the same environment with Prof. Costas Courcoubetis and Prof. George Stamoulis. Their influence on my thinking has been substantial.

Assistant Professor George Xylomenos has always been a source of rich feedback and ideas. His essential comments on both high level and technical issues always helped me improve. I owe him a lot.

During these years I met a lot of interesting people. Pantelis Frangoudis has always been supportive and a good companion in difficult times. He is an excellent researcher, hard worker and a good friend. Nikos Fotiou, has also been a great co-worker and friend. Together we had endless hours of joint brainstorming in front of the lab's blackboard. Always discerning, always funny. Also, Vaggelis Douros with its constant "mordant" view of things, triggered numerous delightful conversations in the lab. Moreover, I would also like to thank Vasilis Kemerlis and Charis Stais for the excellent collaboration we had.

Furthermore, I would also like to thank both the *old* and the *new* generation in the Mobile Multimedia Laboratory. The *old* generation: Elias Efstathiou, Christofer Ververidis, Thanasis Papaioannou, Manos Dramitinos, Sergios Soursos and Costas Kalogiros; I thank them for embracing me at the beginning, as well as for the interesting discussions and the great time we had in the lab. The *new* generation: Christos Tsilopoulos and Xenofon Vasilakos, as well as Alexandros Kostopoulos, Ioanna Papafili and Eleni Agiatzidou; I wish them all the best.

I would also like to thank Nausika Kokkini, Lina Kanellopoulou and Kostas Makedos for their support all these years.

Last, but surely not least, I would like to deeply thank my family and especially my parents for their support, as well as Evi for her invaluable patience.

VITA

2003	B. S. in Informatics, Athens University of Economics and Business, Greece
2005	M. Sc. in Computer Science, Athens University of Economics and Business, Greece
2010	Ph. D. in Computer Science, Athens University of Economics and Business, Greece

PUBLICATIONS

G. Xylomenos, K. Katsaros and V. Tsakanikas, "Support of Multiple Content Variants in the Multimedia Broadcast / Multicast Service," in *Wiley International Journal of Communication Systems*, 2010 (to appear).

K. Katsaros, C. Stais, G. Xylomenos and G. C. Polyzos, "On the incremental deployment of overlay information centric networks," in *Proceedings of the Future Network and Mobile Summit*, Florence, Italy, June 2010.

A. Zahemsky, D. Lagutin, B. Gajic, C. Reason, D. Trossen, C. E. Rothenberg, and K. Katsaros, "Experimentally-driven research in publish/subscribe information-centric inter-networking," in *Proceedings of the 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, TridentCom*, Berlin, Germany, May 2010.

K. Katsaros, G. Xylomenos, and G. C. Polyzos, "A hybrid overlay multicast and caching scheme for information-centric networking," in *Proceedings of the 13th IEEE Global Internet Symposium*, San Diego, CA, USA, March 2010.

——, "MultiCache: an incrementally deployable overlay architecture for information-centric networking," in *INFOCOM Work-in-Progress (WiP)*, San Diego, CA, USA, March 2010.

K. Katsaros, V. Kemerlis, C. Stais, and G. Xylomenos, "A BitTorrent Module for the OMNeT++ Simulator," in *MASCOTS '09: Proceedings of the 17th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, London, UK, September 2009, pp. 361–370.

K. Katsaros, N. Bartsotas, and G. Xylomenos, "Router assisted overlay multicast," in *NGI'09: Proceedings of the 5th Euro-NGI conference on Next Generation Internet networks.* Piscataway, NJ, USA: IEEE Press, July 2009, pp. 106–113.

K. Katsaros, N. Fotiou, G. C. Polyzos, and G. Xylomenos, “Overlay multicast assisted mobility for future publish/subscribe networks,” in *Proceedings of the 18th ICT Mobile Summit*, Santander, Spain, June 2009.

——, “Supporting mobile streaming services in future publish/subscribe networks,” in *Proceedings of the Wireless Telecommunications Symposium (WTS 2009)*, April 2009.

G. Xylomenos, K. Katsaros, and V. P. Kemerlis, “Peer assisted content distribution over router assisted overlay multicast,” in *Proceedings of the Euro-NF: Future Internet Architecture - New Trends in Service and Networking Architectures, 1st workshop*, November 2008.

K. V. Katsaros, P. A. Frangoudis, G. C. Polyzos, and G. Karlsson, “Design challenges of an open spectrum access scheme,” in *Proceedings of the 1st IEEE International Workshop on Cognitive Radio and Networks (CRNETS 2008 - In conjunction with IEEE PIMRC 2008)*, September 2008.

K. K. Vaggelis G. Douros, Pantelis A. Frangoudis and G. C. Polyzos, “Power control in WLANs for optimization of social fairness,” in *Proceedings of the 12th Pan-Hellenic Conference on Informatics (PCI 2008)*, August2008.

K. Katsaros and G. C. Polyzos, “Evaluation of scheduling policies in a mobile grid architecture,” in *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2008)*, Edinburgh, UK, June 2008.

——, *The Encyclopedia of Electrical and Electronics Engineering.* IGI Group, 2007, ch. Mobility-aware Grid Computing.

——, “Towards the realization of a mobile grid,” in *CoNEXT ’07: Proceedings of the 2007 ACM CoNEXT conference, Student Workshop.* New York, NY, USA: ACM, December 2007, pp. 1–2.

——, “Optimizing operation of a hierarchical campus-wide mobile grid,” in *Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007. IEEE 18th International Symposium on*, Athesn, Greece, September 2007, pp. 1–5.

——, “Optimizing operation of a hierarchical campus-wide mobile grid for intermittent wireless connectivity,” in *Local & Metropolitan Area Networks, 2007. LANMAN 2007. 15th IEEE Workshop on*, Princeton, NJ, USA, June 2007, pp. 111–116.

G. C. Polyzos and K. Katsaros, *The Encyclopedia of Electrical and Electronics Engineering.* John Wiley & Sons, 2007, ch. Multicast.

G. Xylomenos and K. Katsaros, “A Multiple Content Variant Extension of the Multimedia Broadcast/Multicast Service,” in *Proceedings of the 15th IST Mobile & Wireless Communications Summit*, Mykonos, Greece, June 2006.

——, “Supporting multiple content variants in the multimedia broadcast/multicast service,” in *Proceedings of the World Telecommunications Congress (WTC’ 06*, Budapest, Hungary, May 2006.

G. Xylomenos, K. Katsaros, and G. C. Polyzos, “Analysis of a multiple content variant extension of the multimedia broadcast/multicast service,” in *Proceedings of the 12th European Wireless Conference*, Athens, Greece, April 2006.

C. V. Katsaros, I. L. Niarhos, and V. Vassalos, “DAIMON: Data Integration for a Mobile Network,” in *MobiDE ’05: Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access.* New York, NY, USA: ACM, June 2005, pp. 57–64.

ABSTRACT OF THE DISSERTATION

An information-centric overlay network architecture for content distribution and mobility support

by

Konstantinos V. Katsaros

Doctor of Philosophy in Computer Science

Athens University of Economics and Business, Athens, 2010

Professor George C. Polyzos, Chair

It has been long realized that the Internet has evolved from a network enabling the communication between pairs of stationary end hosts, to a network for the dissemination of information to stationary or mobile end hosts. Users increasingly focus on the desired information, rather the specific end host providing it. However, this shift in communication patterns has not been followed by a corresponding shift in network architecture. As a result, data delivery is often based on end-point centric overlay data delivery structures that neglect network topology, data location and data popularity, ultimately over-consuming network resources. At the same time, mobility has been supported via ”add-on” protocol enhancements, that do not address the root of the problem. In this context, we identify information-awareness as the key enabler for the efficient utilization of network resources through resource sharing and the support of mobility. We propose an information-centric overlay network architecture, named MultiCache, that inherently builds on the knowledge of what is being delivered by the network, in addition to where it originates from or is destined to, in order to allow for the joint operation of multicast and caching resource sharing mechanisms. We demonstrate the benefits of our approach with respect to both resource utilization and end user experience. Particular attention is paid to the overlay character of the proposed architecture that enables the progressive deployment of the proposed functionality over the existing Internet. At the same time, we revisit multicast assisted mobility, in the context of the proposed architecture, as a means for the localization of routing updates and investigate the corresponding benefits with respect to handoff efficiency.

Chapter 1

Introduction

This chapter is a general introduction to the dissertation. Section 1.1 describes the context of our work, identifying the major issues motivating us, with Section 1.2 focusing on content distribution. Based on these observations, Section 1.3 proceeds in setting the design objectives of the proposed architecture, the main subject of this dissertation. Section 1.4 highlights the original contributions of this work and Section 1.5 provides a description of the dissertation's structure.

1.1 Motivation

1.1.1 The Internet architecture

The Internet was originally designed as a communication substrate enabling the delivery of data between pairs of end-hosts. The main purpose of network based communication at that time was to provide users access to mainfraims located at different locations [1]. This model of communication was based on two fundamental assumptions. First, the purpose of communication was inherently correlated with the identity of the communicating entities. For example, thin end-host terminals were communicating with *specific* supercomputers in order to take advantage of the remotely located processing power. Second, the identity of the communicating entities was inherently correlated with their location in the inter-network. The original Internet architecture reflected these assumptions by coupling a host's identity with its location in the network, incorporating both in the network address of the host. In the resulting communication model an end node must know the network address of the selected destination node in order to communicate with it.

At the same time, the end-to-end argument served the need for a neutral network core able to accommodate diverse application needs, by placing the intelligence at the edge of the network [2]. The role of the network was to provide the minimum common functionality required for communication between edge nodes, leaving application specific needs to be served by functionality placed at the end-hosts. This principle yielded the so called "hour-glass" model, in which the IP protocol simply provides a best effort service in delivering packets to destination hosts. Again, the resulting design was inevitably based on the anticipation at the time regarding the minimum set of required core functionality, reflecting the once prevailing communication needs.

The growth of the Internet during the last decades demonstrated the success of this design, confirming the validity of the initial design goals and assumptions, which allowed for the interoperability between thousands of networks and millions of end-hosts, across a wide-range of applications. However, as the Internet evolved to a global communication substrate, so did the needs of its users. This has resulted in cases where the assumptions underlying the Internet architecture no longer adhere to the reality, and in situations where the capabilities of the network architecture are not sufficient to accommodate new end user needs.

In the following sections we identify these cases and discuss how the end-point centric model of the Internet architecture fails to accommodate some of the current end user needs. Our investigation focuses on the cases of *content distribution* and *mobility*, illustrating the consequences of the anticipated mismatch between the Internet architecture and the evolved application communication patterns.

1.1.2 The shift towards information-centric networking

It has become apparent for quite some time that the Internet has evolved from a network connecting pairs of end-hosts to a substrate for information dissemination. The proliferation of *peer-to-peer* (P2P) applications, *Content Delivery Network* (CDN) services, cloud computing applications, etc., demonstrates this radical shift towards information centrism and the departure from the end-host centric model. In the case of CDN services, large scale overlay networks use their resources to redirect end-host requests to nearby replica/caching locations of the content, most of the time *transparently* to the users (see Section 1.2.2). In P2P applications, end users collaborate in order to share information in a scalable manner avoiding the reliance on content providers. In most cases, the exchange of data between end-hosts again takes place transparently to the users who discount the identity of the content providing peer(s) (see Section 1.2.1). In cloud computing environments, users retrieve the desired content from large server farms comprised of thousands of servers residing at some point of attachment to the Internet, which is of no particular importance to them, provided the eventual, timely delivery of the desired content.(e.g. YouTube [3], Google [4]). End users are now primarily concerned with accessing a desired piece of information rather than the specific end point(s) providing it.

However, this shift has not been reflected in a corresponding adjustment of the underlying network model. Though suitable for the once prevailing end-to-end communication patterns, the end point centric model seems to no longer cater to current communications needs. This has been evidently demonstrated by the parallel departure from the initial concept of the end-to-end, client-server model in which the creator/owner of the information serves the incoming requests, transparently to the network. In most cases, alternative distribution models have emerged in order to cope with the apparent model mismatch between end user needs and the Internet architecture. As discussed in Section 1.2.2, CDNs emerged as an alternative, commercial solution to the scalability problems observed in regular Web content retrieval, based on the wide scale, distributed deployment of content servers. In a broader context, P2P applications constitute a highly scalable alternative to the classic client-server model, however not relying on the deployment of third party infrastructure (as in the case of CDNs), but rather on the completely symmetric role of end-hosts which act both as clients and servers. The proliferation of these applications as a solution for massive content distribution has been demonstrated in Internet traffic measurement studies: in 2008 and 2009 the traffic generated by P2P file sharing application constituted 53% of the entire Internet traffic [5]. Evidently, the basic functionality of the Internet has proved insufficient to support the information-centric needs of users resulting in the creation of overlay structures [6].

In this overlay creation process, a translation between the information domain and the networking domain takes place, typically consisting of the establishment of delivery paths between data providers and consumers. However, this translation is usually performed inefficiently, as the created overlay data delivery structures are inherently end-point centric, focusing on the retrieval of the desired information and application level performance metrics (e.g., download times), while neglecting network topology, data location and data popularity. This has ultimately led to the over-consumption of network resources, and is posing significant challenges to network operators in controlling the traffic traversing their networks [7]. In the context of the P2P file sharing applications, and the dominant BitTorrent application specifically (see Section 1.2.1), it has been shown that a major part of the resulting traffic crosses *Internet Service Provider* (ISP) boundaries, even though the desired information could have been retrieved locally [8, 9, 10]. Considering the fact that BitTorrent generates 34% of the entire traffic in the Internet [5], the over-consumption of network resources becomes immediately apparent. As a result some ISP's have made attempts to violate network neutrality by throttling P2P traffic (e.g. [11]).

1.1.3 Mobility

In the early days of the Internet mobility was not an issue, as the Internet was designed to resemble a telephone network where end-points are fixed and uniquely addressed. As the Internet evolved, the need to support mobility appeared and the coupling of identity and location raised a significant obstacle to the seamless support of moving end-hosts. As end-hosts change their point of attachment to the network, their network address changes in contrast to their identity.

Therefore, their ongoing communication sessions are deemed to fail because the underlying communication model assumes that the destination network address is known to the sender.

A substantial body of research has been devoted to circumvent this situation yielding a variety of approaches, spanning across the TCP/IP protocol stack. The *Mobile IP* (MIP) protocol [12], along with its enhancements *Hierarchical MIPv6* [13] and *Fast Handover for MIPv6* scheme [14], aims at solving the problem at the network layer, by propagating the changes of a Mobile Node's (MN's) network address either to a gateway at the MN's home network or the node(s) currently communicating with the MN, i.e. the Correspondent Nodes (CNs). The *Host Identity Protocol* (HIP) [15] introduces an additional layer between the IP and the Transport layers [16]. The purpose of this layer is to decouple identity from location enabling the maintenance of active sessions during handoffs. The *Session Initiation Protocol* (SIP) [17] has also been proposed for the support of mobility at the application layer [18]. In all these approaches, the target is to propagate the network address changes to appropriate nodes in the network (usually the CNs). However, this approach only provides an "add-on" solution that actually attempts to circumvent the problem rather than solve it. An alternative approach, multicast assisted mobility, has been also studied in the context of IP multicast [19, 20, 21, 22]. This approach is based on the creation of multicast trees that enable the wide distribution of data around the (current) area of a mobile user. Multicast based mobility solutions offer the advantage of fast local handoff as well, as they build on the multicast tree hierarchy to localize routing updates. However, these approaches suffer from the limited deployment of IP multicast [23] (see also Section 1.3).

In the context of information-centric networking, multicast assisted mobility re-appears as a promising direction for the effective support of mobility. In principle, multicast is correlated with the *group* notion, defined as the set of nodes interested in the same piece of information. As such, it constitutes a basic building block of our architecture (see also Section 1.3 and Chapters 3, 4 and 5), motivating the investigation of multicast assisted mobility in the proposed networking environment.

1.2 Content distribution

As discussed above, peer-to-peer applications and content delivery network appeared as alternative solutions to the scalability problems faced by the basic Internet architecture. In the following we describe these approaches presenting and discussing representative examples.

1.2.1 Peer-to-Peer file sharing

Peer-to-peer applications base their success on their scalability properties. Departing from the traditional client-server model, end-hosts present symmetric functionality, acting both as service providers and consumers. Thus, by distributing the serving workload among the participating entities, the scalability of these applications is tightly correlated with the number of the participating end-hosts. In the following we describe some P2P file-sharing applications that introduced significant innovations to the peer-to-peer paradigm.

Napster

Napster was the first attempt to take advantage of the serving capabilities of ordinary end-hosts, in the context of file sharing [24]. Each peer in *Napster* uploaded a listing of the files it was willing to share, to a centralized index. Based on this index, each *Napster* client was then able to search and discover the peers sharing the desired file(s). *Napster* was successful in first distributing the file-sharing workload, while centralizing only the search process. However, the centralized character of the indexing service introduced a single point of failure in the system. At the same time *free-riding* issues emerged by peers refusing to offer service. The centralized character of the indexing service eventually allowed the closing down of Napster due to the violation of Intellectual Property Rights (IPR) laws. However, *Napster* demonstrated the scalability of the peer-to-peer paradigm in the context of file-sharing.

eDonkey

EDonkey [25][1] is one of *Napster*'s successor applications in peer-to-peer file-sharing. In *eDonkey*, the indexing of shared content gains a distributed character, in that each peer uploads its shared directory listing to one of the several indexing servers of the system. Each peer is also able to act as an index server, apart from sharing its content, resulting in a more scalable content discovery platform. Moreover, *eDonkey* introduced the notion of content fragmentation, in which the content (i.e., a file) is split into several smaller size pieces, named *chunks* in *eDonkey*. This new feature allowed for both the parallel download of a single item from more than one peers, and the sharing of partial downloaded files, decreasing the download times perceived by the end-hosts.

Gnutella

Gnutella is another P2P application that enables the discovery and sharing of files [27]. *Gnutella* introduced a completely distributed mechanism for the discovery of content. Each peer (called *servent*) is connected to a limited number of other peers. In order to discover a certain file, a servent sends a QUERY message to its neighbors containing keywords describing the desired content, a Time To Live (TTL) value and its network address and listening port. Each neighbor servent checks its local files against the keywords and further propagates the message to its own neighbors. Each servent holding a matching file responds with a QUERYHIT message that follows the reverse path of the QUERY message. The QUERYHIT message contains the Uniform Resource Name (URN) of the matching file as well as the necessary information for the eventual establishment of a connection with the querying node. The querying process results in a flooding process, limited by the TTL field of the QUERY message.

Gnutella's success was first based on its ability to search for desired data in a distributed way, without relying on a centralized indexing service, as the case of *Napster* [24]. However, the flooding character of the search process incurred considerable scalability problems that triggered the design of variations such as

[1]We use this term to refer to eDonkey2000 [25] as well as other variations (e.g. eMule [26])

the *Gia* P2P file-sharing application which employees a two-tier architecture of peers [28]. Moreover, *free-riding* issues have been also identified as a source of significant performance degradation [29].

BitTorrent

BitTorrent is a P2P content distribution system, comprised of a set of network protocols for realizing communication between the participating entities. It utilizes a simple bartering scheme for reducing *free-riding*, where peers only download and never upload content. Each time a host wants to distribute a set of files through *BitTorrent*, it organizes all of them as a sequence of bytes, it logically splits the sequence into equal size *pieces* and calculates a hash value for each piece. Then, a server that is willing to host the file exchange (not the file itself) is located; this is the *tracker*. The content metadata and supporting information such as hash values, piece size, file size, tracker address and so forth, are recorded in a file with an arbitrary name that acts as a description and summary of the content. This *metafile* can then be distributed over the Web so that search engines can match user queries with metafile data. In this way, *BitTorrent* decouples content discovery from content sharing.

When presented with a metafile, a peer connects to the indicated tracker and asks for a list of other hosts currently participating in that particular exchange; all these hosts comprise the *swarm*. Note that the tracker does not itself participate in the swarm. Subsequently, each peer constructs and maintains a bitmap with the pieces that it has (a new peer initially has nothing, while the original distributor has everything). Then, each peer randomly contacts other peers and exchanges bitmaps with them. Based on the bitmaps and other available data, such as path delay or bandwidth, each peer can freely select the peers it will exchange pieces with.In general, pieces are exchanged in a *tit-for-tat* fashion, but for bootstrapping purposes, peers occasionally give pieces for free.

BitTorrent constitutes the dominant P2P application, generating 60% of P2P traffic in the Internet [5]. This success stems from two key characteristics. First, to its ability to distribute resource consumption among the participating entities, thus avoiding the bottlenecks of centralized distribution. Second, to its aptitude to avoid performance deterioration and service unavailability by enforcing cooperation. However, as mentioned earlier, BitTorrent has resulted in the over-consumption of network resources, incurring unecessary costs for ISPs (see Section 1.1.2). The problem stems from the randomized peer selection process. In *BitTorrent*, a peer connects and downloads content from a randomly chosen subset of the peers in the entire swarm, created by the tracker. This randomness results in neglecting the underlying network topology by ignoring peers that could also provide the same content but are located in the networking vicinity of the peer.

1.2.2 Content delivery networks

Content delivery networks emerged as an alternative solution for the scalable delivery of (usually Web or streaming) content across the Internet. Based on the wide-scale deployment of servers across the Internet, CDNs aim at the replication and/or caching of the content at multiple network sites. Their purpose is to intelligently distribute the content distribution workload across multiple network sites by redirecting end-host requests to a carefully selected replica/caching server, usually via DNS redirection [30, 31]. In effect, they remove the performance bottlenecks from ordinary content providers, while enhancing the end user experience. Based on the wide scale deployment of their servers, CDNs manage to serve client requests from nearby network locations thus achieving lower delays compared to servicing the recipients from the original servers.

Unlike the P2P approach, CDNs base their functionality on dedicated servers deployed at strategically selected locations of the network. As such, they are centrally administered yielding a commercial solution for large scale content distribution. Though the proprietary character of the employed technical solutions hardens the investigation of their subtleties, extensive measurement-based studies have shed some light on the operation of CDNs (e.g. [30, 32, 31]). In the following we examine three major CDNs, namely *Akamai* [33], *Limelight* [34] and *Velocix* (previously known as *CacheLogic*) [35].

Akamai

Akamai currently constitutes the largest Content Distribution Network in the world. Its infrastructure includes more than 61,000 servers deployed at approximately 1000 networks in 70 countries [33]. *Akamai*'s overlay network brings content close to end users by deploying its servers inside ISPs Points-Of-Presence (PoPs). On the one hand, this results in the content being replicated/cached closed to end-hosts, yielding low response times. On the other hand, this highly distributed design has incurred the need for large scale deployment in order to achieve wide-area service coverage, also increasing the maintenance and management overhead.

Limelight

Limelight [34] follows a more centralized approach by relying on the deployment of high capacity service centers close to the backbone network. These service centers are typically comprised of thousands of servers and are interconnected via a private fiber-optic network, currently reaching 800 ISPs around the world. The service centers are connected to ISP PoP with high speed connections (typically 10 Gb Ethernet) aiming at the reduction of the response times due to the greater network distance between the service centers and the end users.

Velocix

Velocix follows a hybrid approach based on a multi-tier architecture [35]. At the higher layer, Service Nodes are deployed in backbone networks and are responsible for the management of the CDN, taking decisions related to routing and cache selection. Content is then handled in two layers below. At the highest layer, massive storage is provided by Storage Nodes, which are responsible for the proactive creation of replicas of the content. At the lowest layer Delivery Nodes are deployed at the network edge providing caching services and interfacing with end-hosts.

Velocix also targets at the distribution of large files, in addition to Web content. To this end, it has developed a multi-source delivery method based on file *chunking* i.e., the fragmentation of large files. For each large file, Delivery Nodes collaborate to concentrate the content from multiple caching locations (other Delivery Nodes) or Storage Nodes.

1.3 Thesis

We believe that at the heart of the problems presented in Section 1.1 lies the lack of information awareness inside the network, that is, the fact that only the end-points are aware of *what* is being delivered. Due to this deficiency, attempts to more efficiently use network resources, such as resource sharing via caching and multicast, are hard to succeed, as the corresponding decisions are made at the end-points of the network, based on coarse grained information. Therefore, an information-centric model should be employed in order to enable the efficient use of network resources and better reflect user needs. In this context, the network, by gaining knowledge on *what* is being delivered, in addition to *where* it originates from, or is destined to, becomes inherently capable of forming targeted and efficient delivery structures. At the same time, by departing from the end-to-end argument, the information-centric model is expected to lead to the decoupling of location and identity that severely impacts the ability of the network to seamlessly support mobility.

Motivated by the observations presented in Section 1.1, we identify a set of design goals for our architecture.

Resource sharing through information awareness

As already discussed, the current mismatch between the Internet's host-centric model and the information-centric model that prevails in the usage of the Internet, has resulted in the excessive waste of network resources. Based on this observation, our first design goal in this work is to improve the utilization of network resources by enabling *resource sharing.* Typical examples of resource sharing mechanisms, which we also consider in our architecture, are multicast and caching.[2]

[2]We further discuss the specifics of the employed mechanisms in Chapters 3, 4 and 5

In this context, *request aggregation* has been identified as the enabling mechanism for resource sharing, which in turn is enabled by the ability to identify request similarity. For instance, in multicast, request similarity is identified by the notion of *groups* and the respective group identifiers (either, for example, expressed as a Class D IP address in IP Multicast or a hash-based key in several application-layer multicast schemes) that represent a specific information item and the set of prospective end-host recipients of it correspondingly. In the same vein, similar information identifiers are used to declare interest in a specific cached information item; for example, Web page Uniform Resource Identifiers (URIs). Therefore, it becomes apparent that the representation of information is essential for the operation of both resource sharing mechanisms considered. In this sense, introducing *information-awareness* into the network architecture becomes a corner-stone of this work.

Scalability

Having considered the need for an information-centric architectural design, the emerging design objective concerns the representation and handling of information. By introducing-information awareness in the architecture and focusing on a network-wide deployment of the aforementioned resource sharing mechanisms, it follows that the involved network entities are expected to refer to information items and handle the required signaling and control state for the operation of the respective protocols. For instance, most multicast schemes require the establishment of per group forwarding state at the involved network entities, as well as the respective signaling for its establishment (see Section 2.2). Considering the vast and constantly increasing amount of the available content (reaching 800.000 Pettabytes in 2009, and expected to reach 1,2 Zettabytes by the end of 2010, [36]), *scalability* becomes a major design goal.

Mobility support

At the same time, the proliferation of mobile devices and the aforementioned deficiencies of the current Internet model, neccecitate the capability of the envisioned architecture to inherently support mobility. In this work, this primarily translates to the reduction of the service disruption perceived during handoffs. To this end, we identify the departure from the end-to-end argument as an inextricable part of the solution, served by the adoption of the information-centric paradigm and the intended support for multicast.

Enhancing end user experience

Another major design objective we set for the proposed architecture is to enable the simplification of the current usage model, by focusing on the desired piece of information rather the specific end-host(s) providing/consuming it. To this end, end-hosts should only indicate the desired piece of information to the network, without engaging in the aforementioned translation of *what* to *where*. At the same time, a challenging goal is to improve end-user experience while satisfying all set requirements. In terms of content distribution, this is primarily expressed by improved download times, while in terms of mobility support, it is expressed as the reduced time of service disruption due to mobility. From the content provider side, we also consider the scalability problems faced by the traditional client-server model (see Section 1.1.2). Our target at this point is to circumvent the tradeoff that appeared in the case of P2P applications (see Section 1.3), by both offloading content providers and improving the utilization of network resources.

Facilitating deployment

An important aspect of the envisioned network architecture concerns its applicability with respect to the deployment process. Our objective is to facilitate the adoption of the proposed architecture by targeting reduced investment costs and complexity of the deployment process. To this end, we first revisit available solutions and consider new approaches emerging in this research area.

IP Multicast. The unsuitability of the Internet for information centric applications is evident in its lack of a multicast facility: multicast is inherently information-centric, it decouples the producers and consumers of information, and it promotes the efficient use of network resources. However, while IP multicast [37] has been available for more than two decades, it has not been widely adopted for various reasons. First, there are no gains to be made by supporting IP multicast, unless all routers support it. Therefore, there are no incentives for individual routers to start doing so. The cumbersome, manual configuration of tunnels between multicast-enabled *islands* in the experimental MBone [38] proved the difficulty of partial deployment of IP multicast. Moreover, the lack of group management facilities for access control and billing, have also contributed to the limited deployment of IP multicast [23]. This is largely due to the fact that group management is inherently correlated with application logic, which cannot be incorporated into the network layer IP multicast model. Similarly, higher layer functions such as error recovery, congestion and flow control have not been properly dealt with in IP multicast.

Clean-slate approaches. The need for information-centric networking has triggered a considerable volume of research initiatives, including *clean-slate* approaches, which revisit the fundamental assumptions and design decisions underlying the current architecture (e.g. [39]). However, though targeting at the root of the problem(s), these approaches inherently disrupt the already established network model requiring the replacement of the existing functionality. In effect, their deployment becomes a non trivial process considering the scale of the Internet and the need for interoperability [40, 41].

Content Delivery Networks. As already discussed (see Section 1.2.2), the emergence of CDNs came as a response to the limitations of the Internet with respect to large scale content distribution. Their operation as an overlay on top of the Internet architecture has facilitated their wide-spread deployment. Indeed, their overlay character has disengaged the deployment process from the underlying network functionality allowing in some cases their growth to sizes reaching several thousands of dedicated servers across the Internet [33].

Based on the above observations, we consider the *overlay* character of the deployment as of particular importance in the context of our work. Practically this involves the implementation of the desired functionality at layers above the network layer, so as to not obstruct the already established routing and forwarding functionality. At the same time, this also entails the deployment of additional infrastructure in the form of dedicated servers that, in contrast to regular routers, can support the operation of higher layers of the protocol stack.

1.4 Contributions

With this work we make the following specific original contributions:

- We design a specific information-centric network architecture for content distribution, named *MultiCache*. Our design is based on a hybrid multicast-caching protocol that enables the joint provision of both mechanisms, exploiting multicast forwarding state as a highly scalable distributed cache indexing mechanism.

- We provide a head to head comparison of an information centric architecture against the current Internet model in the context of content distribution, with respect to network resource utilization and end user experience. Our findings show that the proposed architecture dramatically reduces both inter-domain and intra-domain network traffic while yielding substantially reduced download times.

- We evaluate, model and analytically study the process of incremental deployment of the *Scribe* overlay multicast scheme [42] and *MultiCache*. Our investigation reveals the scalability properties of *Scribe*/*MultiCache* trees with respect to the resulting forwarding state, as a function of the overlay deployment density and the imposed workload.

- We design the *Canonical* [43] version of the *Pastry* DHT [44], named *H-Pastry*. Targeting at the adaptation of the selected DHT scheme to the underlying network structure, we provide a detailed design of a two-level, hierarchical *Pastry* version. We analytically demonstrate the scaling properties of the resulting DHT scheme.

- We design an overlay multicast based mobility support scheme. Our design builds on the ease of overlay multicast deployment and the specific locality properties of the selected *Pastry* DHT scheme in order to localize routing updates incurred by mobility, significantly reducing service disruption during handoffs.

- We contribute a full-fledged implementation of the BitTorrent application model for the *OMNeT++* Simulator [45]. Our implementation supports all *BitTorrent* protocol features and further includes a scheduler for the entrance of *BitTorrent* nodes in a swarm, according to an arrival process derived from the analysis of actual *BitTorrent* traces [46].

- We contribute a topology conversion tool for the support of the *Georgia Tech Internet Topology Model* (GT-ITM) [47] by the *OMNeT++* Simulator [45]. We also provide an evaluation study of the scalability properties of the resulting simulation environment with respect to the size of the simulated topologies, highlighting misconceptions of previous attempts.

1.5 Dissertation outline

The remainder of the dissertation is organized as follows. Chapter 2 provides a review of the related literature, focusing on the areas of information-centric networking, content distribution and mobility management. Chapter 3 provides an overview of the proposed architecture. Chapter 4 focuses on the overlay multicast employed by the architecture, paying particular attention to the deployment process. Chapter 5 proceeds to the description of the combined overlay multicast and caching protocols developed for the support of content distribution. Chapter 6 discusses the need for adaptation of the overlay routing mechanisms to the underlying network structure and presents the *Canonical* version of the Pastry routing scheme we developed. Chapter 7 presents the overlay multicast assisted mobility scheme. Chapter 8 then presents our work in the area of simulation, describing our contributions with respect to the simulation-based evaluation of content distribution schemes. Chapter 9 provides a discussion on the relation of our work with related approaches in the literature and further identifies open issues as future research targets. Finally, the conclusions of this dissertation are presented in Chapter 10.

Chapter 2

Literature review

This chapter provides an overview of the related literature. It refers to a variety of research areas ranging from *clean-slate* approaches for the design of Future Internet architectures, through overlay and application layer multicast schemes, to P2P file sharing approaches and optimizations. We believe this diversity demonstrates the breadth of approaches towards overcoming the limitations of the current Internet architecture, all however underlined by the observed shift towards information-centric networking. Chapter 9 refers back to these approaches in an attempt to highlight the similarities and differences with the proposed architecture.

2.1 Information-centric networking

Information-centric networking has attracted the attention of the research community during the last few years (e.g. [48, 49, 50, 39, 51]). In the following we present the major approaches in this field.

Internet Indirection Infrastructure

The *Internet Indirection Infrastructure* ($i3$) [48] is an overlay architecture aiming at the provision of communication services missing from the basic Internet architecture, namely: multicast, anycast, service composition and mobility. The functionality of $i3$ is based on the deployment of infrastructure nodes and the notion of *triggers* which provide a rendez-vous primitive that decouples senders from receivers. Communication is achieved by all communicating parties refering to a common identifier, which is randomly chosen and unique for each communication session. A trigger is composed by a unique identifier as well as the IP address and the listening port of the issuing node. To receive the desired data a receiver issues a trigger towards the *i*3, which is eventually handled by one of the infrastructure nodes. A sender inserts its data to the *i*3 using the same mechanism. As a result, the data reach the node handling the issued triggers, which then forwards it to the appropriate receivers. The rendez-vous functionality is accomplished by the underlying operation of the *Chord* routing framework [52], which enables the distribution of the identifier space to the participating infrastructure nodes. Multicast forwarding is enabled by recipient nodes that issue appropriately selected triggers in order to guide the forwarding data via common paths.

Mobility support has been considered in the context of the *i*3 architecture [53]. Mobile nodes use *i*3 triggers to enable the redirection of traffic to their location in the network. In order to avoid highly stretched paths, mobile nodes employ two complementary mechanisms. In the first one, mobile nodes cache the IP address of the node handling their triggers, so that a single overlay search for it is necessary; all subsequent packets are routed using standard IP routing. The second mechanism uses periodic probes to enable mobile nodes to locate *i*3 nodes residing at nearby networking locations. This allows mobile nodes to select proper identifiers for their triggers, so that they are eventually placed in neighboring *i*3 nodes. A handoff is performed by a mobile node updating its trigger(s) to point to its new location. Multiple triggers may also be used by mobile nodes able to simultaneously receive data from two wireless access points.

Routing on Flat Labels

The *Routing on Flat Labels* (ROFL) [49] approach aims at the establishment of a routing infrastructure based solely on host identifiers, eliminating any notion of location incorporated in the respective routing protocol. In ROFL, routing resembles the routing operation of DHT schemes (e.g. *Chord* [52]), however not relying on the existence of the IP layer underneath. To this end, a set of algorithms is provided, aiming at the establishment of the routing information. In the intra-domain level, a scheme similar to *Chord* is proposed, incrementally building the routing table entries: each ROFL node establishes links to its predecessor and successor nodes in the circular identifier space. Apart from these entries, nodes also cache AS-level routing information carried by incoming packets. In the inter-domain level, the Canon scheme [43] is employed to allow the establishment of inter-domain paths, also adhering to the established routing-policies. Extensive simulations revealed that ROFL incurs highly stretched routes in comparison with IP, requiring a dramatic increase of routing state size to improve routing.

DONA

The *Data-Oriented (and beyond) Network Architecture* (DONA) [50] builds an overlay information-centric layer on the Internet, based on a network of Resolution Handlers (RHs), which follows the AS-level structure of the Internet. In DONA flat[1], self-certifying names consisting of the cryptographic digest of the content provider and an item specific label, are used for the representation of information. Content providers advertise their content to the closest RH which then propagates this information to the rest of the RHs. In this process, the inter-AS forwarding policies are taken into account so that content delivery does not violate any of the established relationships. End hosts also denote their interest on some content to their closest RH. The network of RHs allows then the discovery of the source of the content and data is finally delivered using standard IP routing. In cases of multiple data sources, including also the deployment of caches, DONA also allows the discovery of the nearest of them in an anycast fashion.

Publish/Subscribe Internet Routing Paradigm

The ICT *PSIRP* Project [39] follows a clean-slate approach by redesigning the Internet architecture based on the Publish/Subscribe paradigm [54]. In *PSIRP*,

[1]This term has been used in literature to describe identifiers that contain no semantic content (e.g., [49, 50]).

information is labeled with flat identifiers and organized into access *scopes*. The scopes may be used to define the access rights of users or the context of the information. Based on this framework, a flexible and expressive rendez-vous system acts as a mediator in locating the desired information on behalf of the subscriber(s), allowing the transition from the information domain to the forwarding domain. At the same time, enabled by the introduced notion of scopes, the publish/subscribe nature of the architecture allows the provision of previously unavailable services such as anycast. Data is delivered to the subscriber(s) with a source routing scheme based on *zFilters* i.e., an application of Bloom filters [55] used to compress the delivery path(s) inside the packet header [56]. zFilters are constructed by new network entities, the Topology Managers (TMs) which have complete topological information about the AS they serve. On the inter-domain level, the Inter-domain Topology Formation (ITF) function is responsible to collect routing information from the TMs for the construction of the forwarding paths, taking into account the existence of forwarding policies and the established inter-AS peering and transit relationships. Based on this mechanism, the resulting forwarding plane supports multicast, enabling thus the efficient use of network resources.

Content Centric Networking

The *Content Centric Networking* (CCN) architecture [51] also aims at breaking the end-to-end, conversational model of the current Internet architecture, and focuses on information. In CCN, content is requested with `Interest` packets that flood the network leaving reverse path information at crossed routers. `Data` packets follow these trails in order to reach the origin of the request. In case of multiple recipients, this mechanism ensures the creation of a multicast forwarding tree. On the transport layer, receivers of information are responsible for flow control and error recovery i.e., senders are considered stateless and the receivers actively control the rate of incoming data via their `Interest` packets. In case of a failed packet transmission, an `Interest` packet is also issued to trigger a retransmission. A hierarchical structure is used to represent the content namespace, enabling the aggregation of content names and the corresponding reduction of forwarding state at the CCN routers. The CCN architecture can be incrementally deployed on top of IP by taking advantage of unused features of widely deployed routing protocols such as OSPF and IS-IS [57, 58]. A similar approach is envisioned in the case of inter-domain routing.

2.2 Overlay and application-layer multicast

The lack of native support for multicast practically in the Internet (see Section 1.3) has motivated a substantial body of research in alternative deployment scenarios. Their common target is to allow multicast-based resource sharing, while overcoming the deployment difficulties faced by IP Multicast. To this end, multicast functionality was moved to the application layer of the protocol stack. Two approaches emerged with respect to the deployment of the proposed multicast schemes. On the one hand, *overlay* multicast schemes are deployed inside the network but at higher layers of the protocol stack - compared with IP multicast which operates at the network layer - requiring the deployment of additional infrastructure for the support of the higher layer functionality. On the other hand, *application-layer* multicast schemes are deployed at the end-points of the network, basing their operation on end-hosts. In both cases, the multicast functionality is implemented at the application-layer. However, the term *application-layer* multicast has prevailed for schemes operated by end-hosts, while the term *overlay* multicast refers to deployments on top of the existing network functionality, usually operated by third party service providers (e.g., [59]). In most cases, the basic protocol functionality does not rely on the exact deployment scenario. In the following, we describe most of the key research work in this field.

Narada was one of the first approaches on end-host based multicast [60]. In this work, source-specific multicast trees are built in a two step process. In the first step nodes are organized in an overlay mesh topology with a low out-degree for each node (reducing the signaling overhead of the second step) and edges comprised by low delay and/or high bandwidth underlay paths. In the second step, a DVMRP-like protocol [61] is run on top of the created mesh topology to construct the multicast trees. Both steps rely on the maintenance of group membership state both at a bootstrap node and at each member of a group, limiting the applicability of the protocol to small and medium size groups.

The *NICE* protocol was proposed as a highly scalable alternative solution for application-layer multicast [62]. In *NICE*, group members are organized in a multi-level hierarchical structure. At each level of the hierarchy end-hosts are organized into clusters of size k. For each cluster the node with the minimum maximum distance to all other cluster members is selected as the cluster leader. All cluster leaders at level L_i of the hierarchy are recursively organized into clusters at level L_{i+1}. Source specific trees are built based on the created hierarchy by allowing a source to send data to the members of its cluster and each cluster head to forward the data to the members of the cluster it serves. The proposed hierarchical structure yields a highly scalable multicast scheme with the control overhead and the length of the delivery paths growing logarithmically with the size of the group.

The *Bayeux* overlay multicast scheme [63] is largely based on the underlying operation of the *Tapestry* routing scheme [64]. In Tapestry, each node is assigned a unique fixed-length identifier (usually retrieved via a hashing algorithm, e.g. SHA-1) and maintains routing information allowing the suffix-based forwarding of messages. More specifically, at each routing step towards a target identifier, a neighbor node is located that shares a longer suffix with the target identifier. *Bayeux* employs this routing mechanism to create multicast groups. In order to join a multicast group, a node issues a message towards the source of the multicast session. The message is forwarded by the *Tapestry* substrate until it reaches the root node, which responds to the request, recording at the same time the identity of the joining node. The response message is forwarded towards the joining node again via *Tapestry*, establishing at each encountered node the required multicast forwarding state.

Overcast [59] is a single-source overlay multicast scheme focusing on the construction of high bandwidth multicast trees. Nodes follow a recursive process to join a multicast tree: starting from the root of the targeted multicast tree, a node samples the bandwidth provided by each of the root's children. If in any of these measurements the perceived bandwidth does not deteriorate in comparison to a direct connection to the root, the node continues the procedure probing the nodes under the selected root's child. The process stops when the measured bandwidth degrades or there are no children to probe, and the joining node becomes a child of the last non deteriorating probed node. In case of equally suitable candidate parents, network proximity (hop count) becomes the selection criterion. In order to compensate for node failures and graceful departures, the employed tree reconfiguration mechanism relies on a complete ancestor list available at each node. Based on its contents, a node re-evaluates its position in the tree by probing its ancestors.

Scattercast [65] is another overlay multicast scheme that follows an approach similar to *Narada*. Its operation is based on the *ScatterCast proXies* (SCXs) which are new infrastructure network agents deployed throughout the network, acting as proxies for the attached end-hosts. Similar to *Narada*, SCXs organize themselves in a mesh network on top of which a distance-vector routing algorithm is employed for the construction of single-source, reverse shortest-path multicast trees. Nodes periodically probe randomly selected nodes in order to select better neighbors with respect to a certain cost function (i.e. latency). Each node must maintain a full list of all SCXs in the system and use a flooding mechanism in order to compensate for node failures and departures, raising serious scalability concerns.

2.3 Peer-to-Peer approaches

As discussed in Chapter 1, P2P applications tend to generate excessive traffic due to the establishment of data flows with randomly selected peers. The randomness is introduced by allowing the arbitrary selection of a peer among the set of peers offering the desired content. Hence, the location of peers in the network is neglected, resulting in the content often being uneccessarily transfered over cross-ISP links. This form of inefficiency has attracted the attention of researchers, resulting in a variety of approaches most of which are targeting at improving the peer selection process by incorporating information related to network topology and conditions.

In [66], Bindal *et al.* proposed the consideration of topological information during the peer selection process in the *BitTorrent* application. In the proposed scheme, the tracker collects information regarding ISP locality for each peer i.e., it identifies the ISP of each peer in the swarm. Based on this information it makes a biased selection of peers when responding to a client request: among the entire subset of peers returned, only k peers are chosen from outside the requesting peer's ISP. An alternative implementation is also proposed, based on the deployment of P2P traffic shaping devices using deep packet inspection to intervene in the peer selection process. This approach allows the enforcement of the proposed scheme by ISPs, without relying on the cooperation of the tracker, at the cost of increased complexity. Extensive simulations demonstrated the effectiveness of the mechanism in reducing redundant cross-ISP traffic while maintaining high download rates.

Aggrawal *et al.* proposed a similar approach in [7]. The proposed solution is based on the provision of a peer ranking service by ISPs, called the *oracle*, that enables informed peer selection, based on a rich selection of criteria such as hop count, delay, congestion, bandwidth etc. This effort attempts to build a synergy between ISPs and P2P applications: P2P applications retrieve rich information about the underlying network conditions directly from the network operators which have the ability to provide it, while ISPs influence the peer selection process in favor of traffic localization. Simulation and experimental measurements based on the Gnutella application (see Section 1.2.1) demonstrate the ability of the proposed scheme to localize communication within AS boundaries, while preserving the graph properties of the Gnutella network.

The P4P framework also builds on the direct interaction between the network and the P2P applications, however providing a richer interaction environment [67]. In the P4P architecture, ISP operated portals, named *iTrackers*, are used to collect information about the network. Such information is represented as a map containing the *p-distances* between end-hosts or groups of end-hosts (possibly aggregated at a PoP level for scalability and privacy reasons). The p-distances may incorporate a rich set of attributes such as OSPF weights, BGP preferences, congestion, financial cost and/or the dual variables of an optimization decomposition problem.

The *Ono BitTorrent* variant also attempts to bias the peer selection process [68], by utilizing locality-related information already gathered by widely deployed CDNs (e.g. *Akamai* [33]). In this approach, peers take advantage of the mechanisms employed by CDN servers to (re-)direct end-hosts to nearby replicas of their customers' content (see also Section 1.2.2). The proposed *BitTorrent* extension is based on the observation that end-hosts that are (re-)directed to the same CDN servers are probably closely located in the network. Hence, *Ono* peers engage in the periodic DNS-based resolution of CDN names, in order to establish redirection maps denoting the set of CDN replica servers indicated by the CDN. The selection of nearby peers is then based on the exchange and comparison of the created redirection maps. The proposed solution was shown to localize *BitTorrent* traffic within ISP boundaries over 33% of the time, while considerably improving the perceived download times. The advantage of the *Ono* approach stems from the fact that the biased selection is solely based on information gathered by the peers, avoiding the reliance to centralized information collecting points in the network, operated either by the ISPs or other third parties (e.g., the trackers) as in the previously describe approaches.

In the same vein, the *TopBT BitTorrent* variant was proposed in [69]. In this approach, *TopBT* peers engage in the collection of network proximity information to guide the peer selection process. However, in contrast with *Ono*, *TopBT* clients do not rely on the operation of any type of third-party infrastructure, including non-dedicated for this purpose CDN Servers, as is the case with Ono. Each peer probes all peers returned by the tracker in order to obtain information about their relative position in the network. This probing is performed periodically and consists of `ping` and/or `traceroute` probes. The collected information is then incorporated in the *BitTorrent*'s choking algorithm so as to favor the exchange of data with peers both located in nearby network locations and yielding high download rates. Experimental measurements showed that the selected approach results in an average reduction of download traffic by 25%, while improving download times by 15%.

A different approach is proposed in [70], where ISPs deploy highly provisioned *BitTorrent* peers. The purpose of this deployment is to allow these peers to fetch the content inside their domains and at the same time to constitute a highly desirable choice during piece selection for the collocated regular peers. Papafili *et al.* show that the proposed approach can be combined with a biased peer selection scheme, further improving the localization of traffic.

2.4 Caching

Caching has been also considered as a candidate mechanism for the reduction of excessive traffic due to the observed model mismatch in the current Internet. Initially, caching targeted at removing the performance bottlenecks at hotspot Web servers, while reducing the latencies perceived at the client side [71]. The increase of traffic volumes due to the proliferation of P2P file-sharing applications (see Section 1.1.2) regenerated the research interest in the use of this mechanism.

The distinctive characteristics of the traffic generated by P2P applications (in comparison to the widely studied Web traffic) have stimulated further research in the area [72, 9, 73]. First, files exchanged in P2P applications are typically of larger size in comparison to Web objects, as they typically refer to multimedia content [72]. Based on this observation, the segmentation of content was proposed in order to achieve finer grained management of caching space [73, 74]. In [75], two cache replacement policies are proposed in the context of partial caching, namely the *Minimum Relative Size* (MINRS), in which the object with the least cached fraction is evicted and *Least Sent Bytes* (LSB), in which the object that served the least amount of bytes, while cached, is selected for eviction. In the same context, a cache replacement algorithm was proposed in [73], aiming at the allocation of caching space to large objects proportionally to their popularity. The proposed algorithm was shown to outperform both the MINRS and LSB algorithms by jointly taking into account the amount of bytes served from the cache and the relative cache size of objects. The fragmentation of content was also proposed to enable the use of the already deployed Web caches [74].

Second, the content exchanged in P2P applications was found to present different popularity patterns than Web content. In [9], extensive measurements of the workload generated by the *Kazaa* application, revealed the relation between the type of content usually distributed in a P2P application and the resulting access patterns: it was shown that the immutable character of media files (e.g., MP3 songs and video files) results in a *fetch-at-most-once* behavior in which end users do not re-issue requests for an already retrieved object as in the case of Web pages. As a result, a significant deviation was observed from the Zipf distribution, which is usually employed to describe the popularity of Web objects [76]. The same deviation was also observed in the *Gnutella* application [73]. In this work, a Mandelbrot-Zipf distribution was further shown to better express the observed popularity patterns, in which the objects at the highest popularity ranks do not present the same popularity as indicated by the Zipf distribution. This was further attributed to the large size of the shared content and the corresponding longer download times that restrict user requests to items of particular interest [73].

The potential performance gains of caching in the context of P2P traffic were investigated in a number of works [9, 72, 73]. In these approaches, a single cache location per administrative domain was considered, with the caching node typically entering the overlay network created by the participating peers. It was shown that with relatively small cache sizes, cache hit rates ranging from 35% to 65% can be achieved, reducing the generated traffic in the network.

In order to avoid the deployment of additional infrastructure for the caching of P2P traffic, the *HTTP-based Peer-to-Peer* (HPTP) framework was proposed in [74], in which content providers use the HTTP protocol to distribute their content in order to enable caching at the already available Web caches deployed by ISPs. Interested end-hosts engage in a Web cache discovery process and collaborate with the origin source in the formation of a tree structures that enable the distribution of workload among the participants. Preliminary simulation results demonstrated the gains of the proposed scheme with respect to network traffic and the end user experience.

A different approach was proposed in [77] in which caching aims at the reduction of the traffic load incurred by the limited flooding of query messages in the *Gnutella* application. Based on the observed locality of query messages, the authors propose the deployment of a transparent cache at the gateway(s) of an administrative domain, able to cache the query responses passing through. By avoiding the uneccessary forwarding of query messages, the proposed service achieves cache hit rates up to 38%.

2.5 Mobility management

As already mentioned, the need for mobility support in the Internet has drawn the attention of the research community yielding a variety of approaches. One significantly differentiating aspect in these approaches is related to the *range* of the mobility the respective protocols deal with. In the case of *macro-mobility*, protocols target at mobility ranges spanning several network administrative domains, often refering to the entire Internet. In the case of *micro-mobility*, protocols focus on the support of mobility within administrative domain boundaries.

2.5.1 Macro-mobility protocols

Mobile IP

The *Mobile IP* (MIP) protocol, whose most recent version was designed for IPv6 [12] (MIPv6), was a first attempt to support IP mobility, at the network layer. In MIP, whenever a *Mobile Node* (MN) connects to a foreign network, it is assigned a *Care of Address* (CoA) and it informs its *Home Agent* (HA)—a router in its home network which represents the mobile node when it is away—about that address. In the simplest form of MIP, every message towards the mobile node has as a destination address the mobile node's initial home address, and it is tunneled by the HA to the CoA of the mobile node. The MN either sends its messages directly to the *Correspondent Node* (CN), that is, the node currently communicating with the mobile node, (*triangular routing*), or uses the reverse direction, tunneling its packets through its HA (*bi-directional routing*). In both cases, the indirection of messages that pass through the HA results in significant overhead related both to the use of IP tunnels, and the distances traveled by the messages in the network, as they have to pass through the HA even if the MN is far from its home network and close to the CN. These problems triggered the emergence of various optimizations such as the *Route Optimization* (RO). In this scheme, the CN is informed of the mobile node's CoA so that traffic can be directly destined there. Even with RO, MIP suffers from various problems. Encapsulation adds significant overhead to each packet and handoff is not always fast enough. Furthermore, MIP was designed to handle local host mobility the same way it handles global host mobility, so the same signaling load is imposed to the Internet, regardless of a MN's mobility pattern, making MIP not scalable. However, the latter issue was later addressed by the development of the *Hierarchical MIPv6* (HMIPv6) [13] and *Fast Handover for MIPv6* schemes [14] that focused on micro-mobility (see Section 2.5.2).

Dynamic DNS

An alternative approach focused on a solution at the transport layer [78]. In *TCP Migrate*, TCP is modified in order to allow for the *migration* of established connections i.e., established TCP connections are able to resume after a change in the IP address of one of the nodes. These changes are communicated by MNs to their home network's DNS server(s) so that CNs are able to locate them while moving. However, the proposed solution does not apply for simultaneous mobility scenarios, in which both communicating nodes are mobile.

HIP

The *Host Identity Protocol* (HIP) [15] was proposed as an intermediate solution, by inserting an additional layer between the network layer and the transport layer. The purpose of this layer is to allow the decoupling of location with identity.

To this end, HIP introduces the notion of Host Identities which are used to identify end-hosts, regardless of their point of attachment to the network. These identities are employed by the transport layer to enable the communication between end-hosts transparently to the underlying IP address changes. In effect, established TCP connections are not broken due to mobility. A change of the IP address of end-host is propagated to its CNs with special UPDATE messages.

SIP

The *Session Initiation Protocol* (SIP) [17] was also proposed as an application-layer solution for the support of mobility [18]. In SIP, a MN is responsible for informing its CN(s) about any change of its IP address using INVITE messages. However, this mechanism does not guarantee the maintainance of TCP sessions across subnet changes. In the case of streaming applications, RTSP's [79] SETUP messages have been also considered as an alternative option for updating the corresponding streaming node with the new IP address upon handoff. Though generally considered as a macro-mobility solution, SIP may also allow finer grained installations based on the use of proxy servers [18].

2.5.2 Micro-mobility protocols

Micro-mobility in Mobile IP

The *Hierarchical MIPv6* [13] and *Fast Handover for MIPv6* schemes [14] were proposed as enhancements to the standard Mobile IPv6 protocol for the better support of micro-mobility. In HMIP a new network entity is introduced, the Mobile Anchor Point (MAP), with the purpose of decomposing the handling of mobility. A MAP can be deployed in a network domain in order to handle all handoff events taking place inside the domain. In each visited domain, a MN is assigned a Regional CoA (RCoA) which is registered with its CNs and the HA through standard MIPv6. The MN is also assigned an On-link Care-of Address (LCoA) by its current router inside the domain. Any handoff resulting in the change of the LCoA triggers a Binding Update (BU) message which is issued towards the MAP. Then the RCoA is updated to point to the new LCoA without any message being sent to the HA and/or the CNs.

The Fast Handover for MIPv6 enhancement aims at reducing the handoff latency by proactively preparing the Binding Update procedure. In doing so, the protocol allows a MN to obtain information about a neighboring subnet as soon as it has detected its presence (e.g. via WLAN scanning). The protocol allows for the current access router serving the MN to proactively retrieve and deliver to the MN the new subnet's prefix, as well as its IP and link layer (MAC) address. With this information, the MN is also able to obtain a new CoA while it has not yet completed the handoff. In this way, upon handoff, the Binding Update procedure may begin immediately, resulting in a faster completion of the entire handoff process. Fast Handover for MIPv6 also considers the employment of user movement prediction algorithms in order to expedite the handoff process. However the accuracy of these prediction schemes has proved to not reach a satisfactory level in some environments [80].

Cellular IP

Cellular IP (CIP) [81] is a micro-mobility protocol proposed to operate in conjunction with Mobile IP. CIP is deployed inside an administrative domain's boundaries, with Mobile IP supporting mobility across domains. In CIP, routing is based on reverse path forwarding. Beacon packets are periodically broadcasted by the domain gateway inside the access network, establishing reverse path forwarding state at each encountered router. Based on this information, MNs' packets are routed towards the gateway leaving also reverse path forwarding state that is used to forward any packet destined to them. Inactive MNs periodically transmit *paging-update* messages in order to provide the network with information on their location. CIP supports two types of handoff. In *hard handoff* a MN simply associates with the new base station and issues a *route-update* packet towards the gateway. As a result, the reverse path forwarding state is updated, but any packets in-transit to the old base station are lost. In the *semisoft handoff*, the MN triggers the routing update procedure from the new base station and immediately returns to the old station to continue receiving the in-transit packets. This results in both base stations receiving data for the MN. After a *semisoft delay*, the MN performs the handoff, associating with the new base station.

Hawaii

The *Handoff-Aware Wireless Access Internet Infrastructure* (HAWAII) [82] was also proposed for the support of micro-mobility, presenting two sets of alternative route update mechanisms. The first set is comprised of two alternative path set-up mechanisms that enable the old base station to forward the arriving packets to the new base station, before the appropriate routing update takes place at the cross-over router i.e., the router where the two paths connecting the old and the new base station to the gateway of the domain converge. The second set does not consider the forwarding of data from the old base station, focusing on the diversion of data packets at the cross-over router.

2.5.3 Multicast assisted mobility

Multicast assisted mobility was considered as an alternative solution [19, 20] soon after IP Multicast was proposed [37]. A multicast address is assigned to each mobile host resulting in the creation of a multicast tree that, in its simplest form, supports a single leaf node i.e., the base station currently serving the MN. In the opposite direction, MNs unicast their packets to their stationary[2] CNs. The attractive characteristics recognized in multicast relate to its inherent ability to decouple the sender from the receivers, while allowing the concurrent delivery of content to a dynamically changing set of locations, an appealing feature for the support of soft handoffs. In essence, multicast addresses allow for the decoupling of identity from location [20]. At the same time the hierarchical structure of the multicast forwarding trees favors the localized updating of routing information at their lower levels, without necessitating end-to-end signaling between the communicating entities, in a similar way to some micro-mobility protocols (i.e. [13, 82]). In [19] a set of *multicast-based re-establishment* schemes is proposed that take advantage

[2]Simultaneous mobility has not been investigated in any of the presented approaches.

of the underlying multicast functionality to divert data packets at the cross-over router, similarly to [82]. The MSM-IP scheme [20] followed the same approach further delving into implementation details and clarifying technical aspects regarding the use of the available multicast related protocols (e.g. IGMP [83]). In [84] a clear distinction was made between micro-mobility and macro-mobility, allowing the joint operation of IP multicast and Mobile IP. The Core Based Trees (CBT) protocol [85] was proposed for the support of mobility at an intra-domain level, while Mobile IP was suggested to handle mobility across domains. In the same vein, Helmy *et al.* investigated the use of multicast as a micro-mobility protocol, further suggesting mechanisms for scalable address management and the support of proactive handoffs [21]. The proposed M&M protocol was compared via simulations with the Cellular IP and HAWAII micro-mobility protocols, demonstrating the superior performance of the multicast based approach. An in depth study of multicast-based mobility support schemes is also presented in [22]. Based on an experimental setup, the authors show the effectiveness of the multicast-assisted mobility, also demonstrating the inferior performance of *Hierarchical Mobile IPv6*.

Chapter 3

System architecture

The purpose of this chapter is to present an overview of the proposed architecture before proceeding into the detailed description of the protocols involved. It discusses how the high level design goals presented in Section 1.3 are translated into specific design decisions. Furthermore, it provides an overview of the proposed functionality and describes the new network entities introduced for its support, discussing also deployment issues. The description of the proposed architecture is completed with the detailed presentation of the developed protocols and functions provided in Chapters 4, 5 and 7.

3.1 Design principles

As discussed in Section 1.3, one of our primary design goals is to facilitate resource sharing. To this end, we focus on two key mechanisms: multicast and caching. In *multicast*, similar requests can be served via routes that share links to carry the information only once. Since single transmissions serve multiple recipients, eventual delivery of the data takes place (almost) simultaneously at the leaves of the tree. In effect, similar requests have to be tightly correlated in time in order to take advantage of link sharing. In this sense, multicast has been traditionally considered as especially suitable for (live) streaming applications such as IPTV[1]. However, this requirement for strong temporal proximity of multicast served requests is also satisfied in the context of content distribution applications, and especially in the cases of *flash crowds* in which the popularity of a certain item increases rapidly, resulting in a large number of tightly correlated in time requests, but also diminishes rapidly [86, 46]. Therefore, multicast appears as a promising solution for the support of content distribution services as well.

However, in the context of content distribution and in contrast with streaming applications, data must be delivered correctly in its entirety and this requirement complicates the use of multicast. First, the need for reliable data transmission requires the use of error and flow control mechanisms that have proved difficult to support with IP multicast [23]. As discussed below, we address this issue by following an application-layer multicast scheme. Second, as requests for content cannot be perfectly synchronized, multicast may result in variable data availability at the end-hosts, past the end of a multicast session. Therefore, additional mechanisms are required to ensure the complete transfer of content to the end recipients. Moreover, the temporal proximity of arriving requests cannot always be considered to be high enough to allow for the use of multicast. A low temporal proximity of requests may result in the degeneration of multicast trees to a series of unicast transmissions of the content to end-hosts. Based on these observations, we consider *caching* as an appropriate mechanism to also serve asynchronous request patterns in a resource efficient manner. With caching, content is temporarily stored allowing the asynchronous access to selected pieces of data.

At the same time, in view of the successful example of Web caching [71], caching appears as a promising approach towards the enhancement of the end user experience. By placing the content close to the recipients, both the response times at the end users and the content provision overhead at the content provider side are expected to be reduced. Moreover, caching is considered as a mechanism suitable for the reduction of network traffic by cutting down the repeated transfers of the content from the content provider. The low and constantly decreasing cost of storage space further motivates the deployment of caching services. Figure 3.1

[1]This is also strongly correlated with other factors such as the difficulties faced in supporting error recovery, flow control and congestion control features in IP multicast [23].

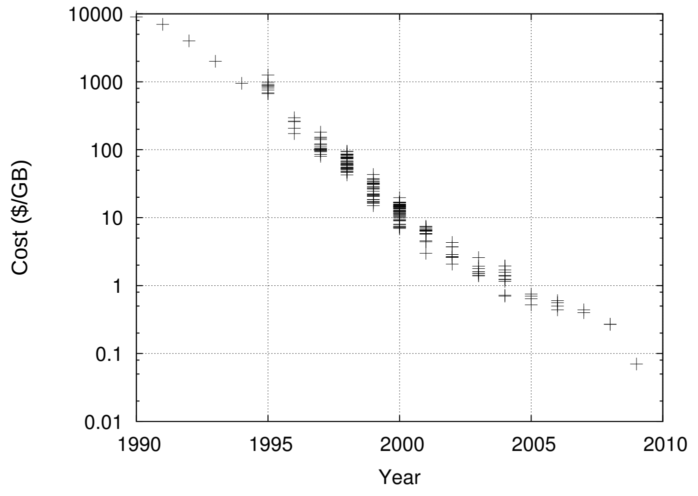


Figure 3.1: Storage cost evolution

shows the cost per GB of storage during the last 20 years [87]. Note that in 2009, this cost reached \$0.07/GB.

Considering the deployment problems faced by IP multicast as well as the success of CDN overlay deployment, we have targeted the establishment of the desired functionality as an overlay (see Section 1.3). To this end, we base the functionality of the proposed architecture on the operation of an application layer multicast scheme on top of a DHT routing substrate. Apart from facilitating deployment, the selection of a DHT-based set of routing and forwarding protocols serves several of our design goals. First, key-based routing in DHT schemes perfectly fits our target of focusing on information, providing a mechanism for identifying and referring to information inside the network, through the use of flat identifiers (see also Section 2.1). In these schemes, routing and forwarding are inherently correlated with information identifiers, facilitating the departure from the end-host centric model. Moreover, the logarithmic scalability of DHT-based schemes serves our design objective for the design of a large scale architecture.[2] We thoroughly describe and discuss the advantages of the specific overlay protocols selected in Section 3.3.

[2]The scaling properties of the protocols employed in our architecture are investigated in Section 5.4.

The information-centric character of the proposed architecture is further augmented by the adoption of the publish/subscribe (pub/sub) paradigm [54]. The pub/sub paradigm has been considered suitable to express the shift towards information centric networking, due to the way it decouples end-hosts from each other, focusing on information rather than on endpoints [39]. A pub/sub architecture divides the roles of the endpoints into two categories; publishers and subscribers. Publishers are data owners and subscribers are data consumers. Subscribers express interest in particular pieces of information, or on information patterns, in the form of subscription messages which are stored in an event notification service. Publishers make information available in the form of publications, which are also handled by the event notification service. The network is responsible for matching subscriptions with publications and notifying subscribers when a publication matches their interest or simply forwarding the available data to them. Hence, subscribers are not necessarily aware of publishers, and vice versa, as the event notification service can transparently handle all messages exchanged. Moreover there is no need for synchronization or coordination between publishers and subscribers, as they do not have to interact with the event notification service at the same time. Therefore, the pub/sub paradigm departs from the current end-to-end argument. Moreover, the anonymity and the asynchrony of the pub/sub model allows for the quick adaptation to frequent connections and disconnections, making it advantageous in a mobile network [88].

3.2 Architecture overview

The proposed architecture attempts to take advantage of information- awareness to improve the utilization of network resources via resource sharing and to further provide mobility support. To achieve this, network operators deploy and control proxy overlay routers that enable the joint provision of multicast and caching. Based on these primitives we name our architecture *MultiCache*. *MultiCache* follows the pub/sub model of communication, inherently departing from the end-to-end conversational model. Inside the network, the *Scribe* overlay multicast scheme [42] is employed to transport the content from its origin in a pub/sub fashion, thus serving synchronous requests (e.g., flash crowds) and feeding in-network caches. By taking advantage of the locality awareness of the established *Pastry* routing substrate [44], anycast queries, based on the already established overlay multicast forwarding state, are later used to locate nearby caches that can serve asynchronous requests by unicasting the cached content.

3.2.1 Deployment

We propose this architecture to be realized by the deployment of additional infrastructure, in the form of *Overlay Access Routers* (OARs), inside access networks, possibly collocated with regular access routers i.e., ISPs' PoP. A simple deployment example is given in Figure 3.2, with OARs being collocated with the corresponding access routers.

All OARs provide the following functionality:

1. They participate in the overlay routing and forwarding substrate, enabling the use of overlay multicast. This entails the maintenance of the Pastry routing information [44], as well as the functionalities of *Scribe* and *MultiCache* (see Chapter 5).

2. They act as the interface of end-hosts with the overlay, possibly aggregating data requests from multiple attached end-hosts, and also allowing the delivery of data to the overlay from content providers. For this purpose, all end-hosts establish a control connection to an available *proxy* OAR, designated during network attachment.

3. They cache the content destined to their attached end-hosts and provide it upon request. As a result, the same content is available at multiple locations in the network.

As pointed out in Section 1.3, the limited deployment of IP multicast is due to several reasons, including scalability concerns, group management and billing difficulties, and the need for ubiquitous deployment i.e., all routers in the network

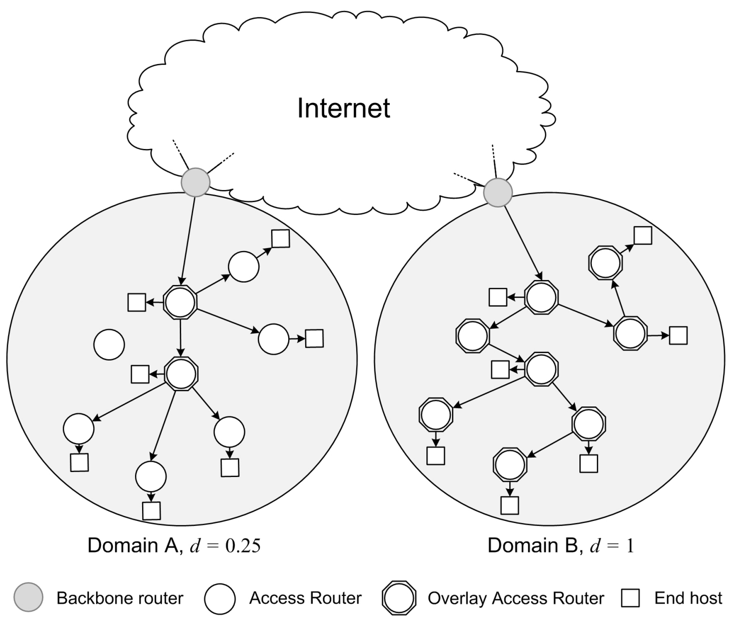


Figure 3.2: *MultiCache* deployment

must support IP multicast. Our architecture takes these factors into account, first, by employing a highly scalable application layer multicast scheme, and second, by pushing the required functionality inside access networks, but in an overlay fashion. As a result, scalable multicast is achieved without relying on the ubiquitous deployment of overlay functionality. In addition, by operating at the application layer the proposed architecture enables the exploitation of already available solutions regarding error, flow and congestion control. Finally, the proxy character of the established overlay nodes, described in Section 3.4, facilitates group management, as these nodes may act as intelligent gateways of end-hosts to the multicast substrate.

The placement of the caches in the access networks is driven by several factors. Caching the content there facilitates its discovery by the clients, by being in their vicinity [89]. As an effect, cached content is expected to travel short (networking) distances, yielding two important benefits. First, the incurred network traffic is expected to be localized, reducing the load both within domain boundaries, and across inter-domain links. In an alternative scenario, more centralized caching locations could be deployed close to the backbone (e.g., [34]). However, apart from increasing the risk of failure, this centralized solution would also require the cached content to travel across multiple intra-domain links at every cache hit, incurring higher intra-domain traffic load. At the same time, this would also require the content to be placed at a greater networking distance from the end-hosts, which brings us to the second benefit stemming from this design decision: the close proximity of cached data is expected to lower the download times perceived by the end-hosts. Finally, as discussed in [90], placing caches close to the end points of the network avoids incentive incompatibilities regarding inter-domain relationships. Namely, ASs located high in the interdomain AS hierarchy act as transport providers for their lower level customers; in effect, the provision of caching services would result in a reduction of the transit traffic and the corresponding revenues for the transit ASs.

Access Networks

The exact location of OARs in access networks can vary based on the type of the network (i.e., wireline or wireless access networks), the exact underlying network topology, and the deployment density of OARs (see Section 4.2). In cases of wireline access networks (e.g., ISPs providing DSL connections), the OARs are deployed at the ISPs PoPs (e.g. Broadband Remote Access Servers (BRASs) and/or Digital Subscriber Line Access Multiplexers (DSLAMs) in the case of ADSL.[3] The selection of the PoPs in case of sparse deployments may be based on load balancing criteria, such as the measured average traffic load on PoP or their fan-out degree (i.e., the number of current and/or expected number of subscribers served by the PoP) and can be integrated with the overall planning of the network. Nevertheless, the efficient distribution of load by the underlying *Pastry* scheme [44], coupled with its self-organization and reliability features [92, 93], especially in view of the lower churn rates expected in such a controlled environment, allow for a less demanding PoP selection process. In the remainder of this work, we have assumed a uniform distribution of OARs across access networks.

[3]The deployment in DSL networks would depend on the exact topology of the network, as multiple deployment alternatives exist [91].

In the case of wireless access domains, the deployment procedure may vary due to mobility, which may introduce high variations in the aforementioned load balancing criteria. In this case, there is a variety of alternative approaches depending on the wireless access technology in use and the intended wireless coverage. In the simplest case, the deployed wireless *access points* (APs) may act as simple bridges to the wired part of the network, thus allowing the fallback to the wireline deployment case (e.g., residential WLANs [94]). Alternatively, APs may form groups connected to a single OAR so that link layer mobility is provided in certain parts of the network, as in the case of cellular networks, or they can even act as OARs themselves.

3.3 Overlay routing and forwarding

3.3.1 *Pastry*

In the *Pastry* routing substrate [44], a uniform identifier space is distributed among participating nodes. Identifiers are managed as a sequence of digits. Each node maintains routing state that enables the forwarding of messages using prefix based routing: at each routing step, a node forwards the message to another node, whose identifier shares at least one more digit with the target identifier. During routing state creation and maintenance, *Pastry* takes network locality into consideration i.e., among equally qualified routing table entries, the one corresponding to the closest node in the network (with respect to the employed proximity metric, e.g., hop count, Round Trip Time (RTT), etc.) is selected. It is noted that as the length of the prefix match increases, the size of the *Pastry* node population providing a match decreases exponentially, providing thus an exponentially decreasing set of alternative overlay neighboring nodes. In effect, as the length of the prefix match increases, the effect of the proximity metric rapidly diminishes. However, at the earlier stages of routing, in which shorter prefix matches are considered, the set of candidate neighboring *Pastry* nodes is large enough to provide significant benefits by reducing the overlay path stretch. This feature yields *Pastry*'s *short routes* property and constitutes one of the reasons for selecting *Pastry* as the key-based routing substrate for our architecture. Additionally, according to *Pastry*'s *route convergence* property, the distance traveled by two messages originating from two distinct nodes before their routes converge towards the same destination, tends to be approximately equal to the distance between the two source nodes in the proximity space [44, 42, 95]. As described in Chapter 5, we take advantage of this property in the proposed caching scheme, in order to locate caches residing in the networking vicinity of end-hosts.

Pastry employs 128-bit identifiers (though other lengths may be used), handled as a sequence of b-bit digits (b is a configuration parameter with typical value 4). Routing state is maintained in two separate structures, the *Routing Table* and the *Leaf Set*. Each entry contains the identifier and the IP address of a node in the system. The Routing Table is a $(2^b - 1) \times \frac{128}{2^b}$ array. Each row i contains entries for nodes whose identifiers share only the first i digits with the present node. A row is populated with entries refering to identifiers whose $i + 1st$ digit has one of the possible $(2^b - 1)$ values, other than the present node's $i + 1st$ digit. The uniform distribution of identifiers across the node population results in only $\lceil log_{2^b} N \rceil$ being populated, resulting in an average Routing Table size of $(2^b - 1) \times \lceil log_{2^b} N \rceil$ entries [44]. The Leaf Set contains $|L|$ entries for nodes whose identifiers are the closest to the present node's identifier. The set is split in two parts with $|L/2|$ entries for numerically smaller identifiers and $|L/2|$ for larger identifiers. *Pastry* also maintains entries for the $|M|$ closest nodes in the network with respect to the proximity metric in the *Neighborhood Set*. These entries are not used in routing, however they are maintained to assist in filling the Routing Table. Typical values for $|L|$ and $|M|$ are 2^b and $2^{(b+1)}$.

3.3.2 *Scribe*

The *Scribe* [42] application-layer multicast scheme operates on top of *Pastry*, though it can operate with any DHT-like routing substrate (e.g., *Chord* [52] or *Tapestry* [64]). *Scribe* builds any-source trees by mapping the name of each multicast group to an identifier and making the node responsible for that identifier in the underlying *Pastry* substrate the Rendez-Vous (RV) point of the group. The

RV point of the group acts as the root of the multicast tree. The data source in a multicast session locates the RV point of the corresponding tree via the *Pastry* routing substrate. Receivers join the group by sending a join message towards the group identifier; as the message propagates towards the RV point, reverse path routing state is established until a node already in the tree is found, thus forming a multicast tree rooted at the RV point. A sender simply routes data towards the group identifier, so that the RV point may then propagate it over the established tree.

An important characteristic of *Scribe* that led to its adoption in this work is its scalability. Multicast routing state is maintained in a decentralized fashion: each node in a tree is only aware of its immediate ancestors and descendants. This is a significant scalability advantage over other overlay multicast schemes (for example, *Bayeux* [63]) as it means that *Scribe* does not require excessive signaling traffic in order to gather global state information.[4] At the same time by adhering to the pub/sub paradigm, *Scribe* departs from the end-host centric model providing an inherently information-centric rendez-vouz mechanism that allows information providers (publishers) and consumers (subscribers) to *meet* inside the network.

3.4 Interacting with the overlay architecture

The proxy role of OARs simplifies the interaction of end-hosts with the proposed architecture, as the entire overlay functionality is placed in the network's infrastructure. The information-centric character of the architecture is reflected by the central role of the flat, location independent identifiers (IDs) used to represent information. A globally unique ID residing in *Pastry*'s namespace is assigned to each information item. Collisions can be avoided by selecting a proper ID length[5] in combination with an one-way hash function such as SHA-1.

Upon attachment to the network, end-hosts establish a control connection with an OAR. The criterion for the selection of the OAR among the many available may be based on metrics such as the networking distance and the current workload.

[4]We analytically investigate this issue in Section 5.4.

[5]The default length in *Pastry* is 128-bits, though longer IDs can also be used.

In our work, we have selected the former metric, expressed in number of hops, in order to avoid lengthy paths, which are expected to incur higher latencies and increased intra-domain traffic.

As mentioned above, the usage model of the proposed architecture closely follows the publish/subscribe paradigm. In the case of end-host receivers (i.e., subscribers), a subscription request message is sent towards the proxy OAR declaring the ID of the desired piece of information. This ID is retrieved via out-of-band mechanisms e.g., a torrent-like file in content distribution applications, as described in Section 5.3. The proxy OAR is then responsible to fetch the requested item using the protocols described in Chapter 5. On the other hand, data providers (i.e., publishers) advertise their content to the network. To this end, publishers submit an advertisement message to their proxy OAR which is then responsible to locate the corresponding *Scribe* RV point, using *Pastry* routing.

All item IDs are assumed to have been appropriately selected so that the RV functionality is provided by OARs residing at the content provider's domain. This is in order to reduce forwarding dependencies [23], an issue raised by the any-source character of the created multicast trees, which results in the selection of a single RV point as the root of the tree.[6] The appropriate selection of the identifiers allows for content providers to not rely on the forwarding service provided by the RV point(s) of a randomly selected network operator. Moreover, it allows for network operators to not be forced to support the distribution of content that originated at a content provider residing at a different access network, possibly serving end-hosts also residing at different networks. This would obviously result in the consumption of network resources without any associated revenue for the network operator. A candidate mechanism for the proper selection of IDs is proposed in [53], in which nodes sample the ID space to locate IDs that lead to closely located infrastructure nodes. Alternatively, network operators may provide pools of properly selected IDs to their customers. In general, it is assumed that publishers advertise their content prior to making the respective ID available to the aforementioned out-of-band mechanisms. Hence, upon reception of the first subscription, the RV solicits

[6]In contrast with source-specific trees in which the root is selected based on the location of the source of the content.

the respective publisher which responds by delivering the advertised data to the RV point.

Chapter 4

Overlay multicast

This chapter focuses on the deployment of the multicast functionality in the proposed architecture. Initially, the gains stemming from the overlay deployment[1] of the selected *Scribe* multicast scheme are investigated, motivating network support. Subsequently, the gradual deployment process is studied with respect to both the adoption of the multicast functionality across the inter-network and the required deployment density inside domain boundaries.

4.1 Network support for application-layer multicast

As discussed earlier, the development of multicast protocols operating at the application-layer emerged as an alternative solution to the deployment problems faced by IP multicast. In essence, the lack of network support for multicast pushed this functionality to the edge of the network, at the end-hosts. Though overcoming deployment difficulties, the resulting multicast schemes inherently present some inefficiencies, exactly due to their deployment at the edge of the network. This is evidently demonstrated in the case of regular *Scribe* that relies on end-hosts. An end-host that is an interior node in some trees will limit the bandwidth available to all those trees to that of its access link. This can be avoided by exploiting the

[1]The distinction between overlay multicast and application-layer multicast terms has been introduced in Section 2.2.

properties of the underlying DHT to create a set of trees such that each node will be an interior node for only one of them [96]. However, this solution is tied to a specific overlay routing scheme (in this case, *Pastry*), while the employed tree-reconfiguration mechanisms result in the creation of parent-child relationships that violate the locality properties of *Pastry* (see Section 3.3.1) [97]. In addition, an end-host that is an interior node even for a *single* tree, may still be a bottleneck. Figure 4.1 shows an example regular *Scribe* tree. As shown in Figure 4.1(b), data in transit has to enter and exit the RV point (peer b) and the other two internal end nodes c and d, only one of which (peer c) is also a receiver, via their access links. If these access links are asymmetric, the tree bandwidth will be limited by the, typically lower, uplink bandwidth.

Another issue is that neighboring end-hosts may download the same content via separate tree branches, thus incurring unnecessary network load. For example, in Figure 4.1(b) the two peers e and f receive separately the content from their parent in the tree (peer d). Note that in this example peers b and d act as intermediate tree nodes without being receivers. This arises when nodes participating in various multicast groups share the same DHT substrate, so as to amortize DHT maintenance costs among different groups and improve DHT routing performance by increasing the available overlay paths.

The overlay deployment of the multicast functionality in the proposed architecture aims to overcome the observed inefficiencies. The proposed proxy role of OARs presents some significant advantages regarding the characteristics of the created distribution trees. This is based on the assumption that network operators are able to deploy the OARs close to regular access routers so that their inter-connection via high bandwidth links is facilitated. In the following, we will use the term *router assisted overlay multicast* (RAOM) to refer to the proposed deployment scheme. Figure 4.2 shows the overlay multicast tree created in the case of RAOM, for the example of Figure 4.1. First, as shown in Figure 4.2(b), data do not need to cross the access links of interior tree nodes at all, only crossing the, typically faster, downlink direction of those access links leading to nodes that are members of the group. For example, data will not cross the access link

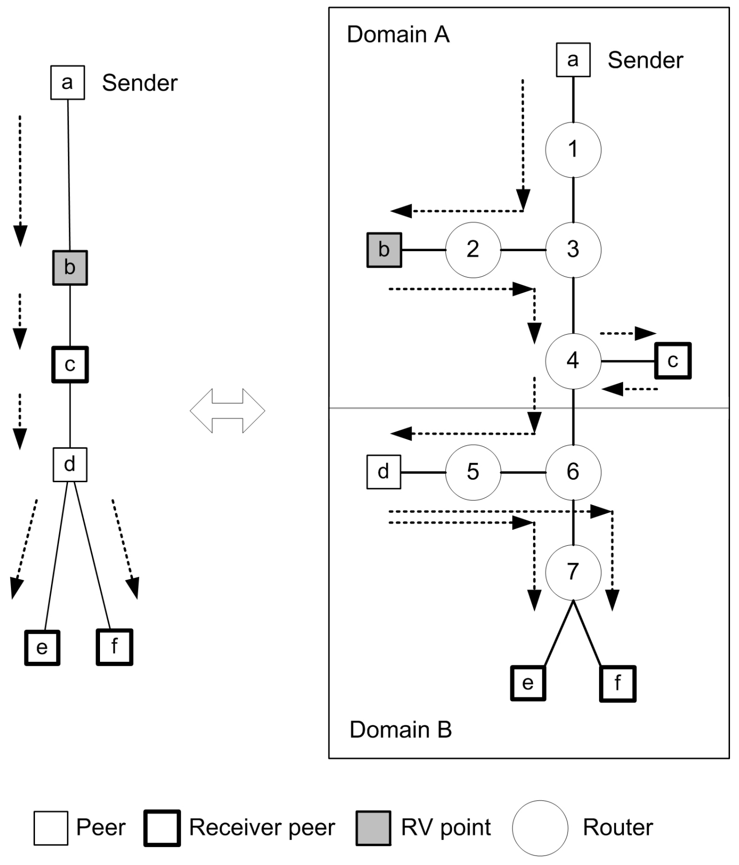


Figure 4.1: Regular *Scribe*: (a) overlay (b) underlay

between peer *d* and router *5* at all, and it will only cross the access link between peer *c* and router *4* in the downlink direction so as to deliver the content to peer *c*. Second, multiple tree branches towards end-hosts attached to the same access router can be aggregated in a single branch leading to that access router (in our example router *7*). For the entire distribution tree of Figure 4.2, router assistance means that a packet transmitted to the group will only cross 12 instead of 20 links with regular *Scribe* (see Figure 4.1) (or 8 with a source-specific, shortest-path IP multicast tree), avoiding the uplink direction of access links. Therefore, in router assisted overlay multicast the paths through the distribution trees become shorter and faster, while redundant transmissions over the access links of intermediate nodes are prevented, something especially important in an environment where asymmetric access links are prevalent, as it prevents the (typically slower)

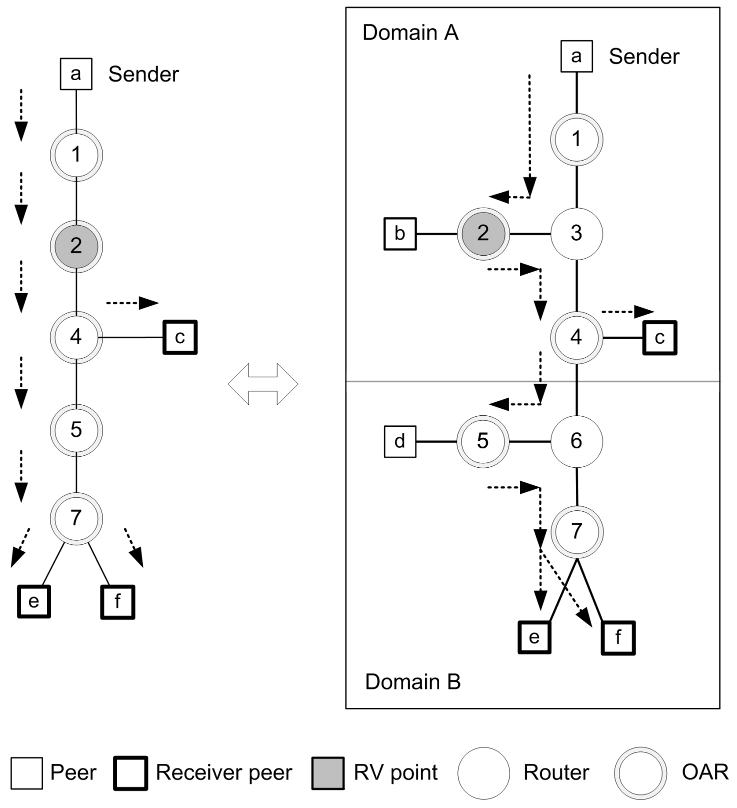


Figure 4.2: Router assisted overlay multicast: (a) overlay (b) underlay

uplinks from becoming bottlenecks.

4.1.1 Evaluation

In order to investigate the potential gains of the proposed RAOM scheme, we have performed an extensive set of simulations, based on the OverSim Simulation Framework [98], an overlay network simulation framework for the OMNeT++ simulation environment[2] [45].

Simulation scenarios

In our simulations we considered a wide range of network topologies following the Georgia Tech Internet Topology Model (GT-ITM) [47]. The considered topologies range in size, from 650 routers to 5050 routers in total. Five different

[2]Chapter 8 provides further details for the employed simulation platform.

Table 4.1: Network topology parameters

	Topo0	Topo1	Topo2	Topo3
Transit domains	5	7	9	10
Avg. routers per transit domain	5	4	4	5
Stub domains per transit router	5	7	9	10
Avg. routers per stub domain	5	7	9	10
Stub routers	625	1372	2916	5000
Transit routers	25	28	36	50
Total routers	650	1400	2952	5050

topologies were generated for each network size and all the results we present express the mean values over those instances. Table 4.1 shows the parameters used to generate the topologies. In all topologies, the default link establishment probabilities were used, that is, a link between two transit routers was established with a probability equal to 0.6 and a link between two stub routers was established with a probability equal to 0.42. The target number of end-hosts were then randomly attached to the stub routers.

Most of the results in the following refer to Topo1 with a variable number of end-hosts in the network. Specifically, we simulated three different scenarios with 500 (Sparse), 1000 (Medium) and 4000 (Dense) end-hosts, respectively, so as to explore the impact of host density on the performance of the suggested overlay multicast scheme.

Regarding multicast groups and their sizes, a *Zipf*-like distribution was used for the size of each group, that is, the r-th group had a size equal to $\lfloor Nr^{-1.25} + 0.5 \rfloor$, where N was the total number of overlay nodes, as in [42]. The first group included all overlay nodes, while the last group had a size equal to 11 nodes, a size typical of instant messaging applications [42]. Based on this lower bound and the number of participating hosts, the number of groups in the Sparse scenario was 21, in the Medium scenario was 36 and in the Dense scenario was 110. The members for each group were randomly selected from the entire end-host population, meaning that each end-host may have participated in many groups. For each group, a random identifier was chosen and a *non-member* end-host was randomly selected as the sender.

In all scenarios, end-hosts first join the overlay (*Pastry*) network; in our scheme, this means that OARs join the overlay network on behalf of their attached end-hosts. After the initialization of the overlay network has completed, participating nodes start joining, and therefore forming, the multicast trees. When all nodes have joined the respective trees, the senders send a data packet towards the RV point of each tree to inspect the path between the sender and the RV point. Note that in this evaluation, unlike in earlier ones [42, 99], we explicitly take into account not only the tree formed between the RV point and the receivers, but also the path from the sender to the RV point, since application performance is determined by the entire path between sender and receivers.

Path stretch

The most obvious metric for evaluating overlay routing schemes is the overhead incurred by not strictly following the shortest path IP routes. This is usually expressed by the term *stretch* which is defined as the ratio between the overlay path length (or delay) and the length (or delay) achieved by IP routing. In essence, *stretch* expresses the degree to which the routing performance achieved is worse than the performance of the underlying IP substrate, a penalty paid in return for the flexibility and scalability of the overlay solution. Multicast complicates this metric, not only because we need to take into account the entire distribution tree, but also because the Internet does not provide multicast routing in the first place. The usual convention, also followed here, is to assume that IP multicast would use the tree formed by merging the optimal unicast paths between the sender and each receiver.[3] In the absence of multicast, a sender desiring to reach all receivers would have to send copies of each packet over all those paths.

We define *path stretch* as the ratio between the number of IP hops comprising the path from the sender to a receiver in the multicast overlay tree to

[3]The Protocol Independent Multicast - Sparse Mode (PIM-SM) protocol [100] allows for the creation of source-specific, shortest-path trees.

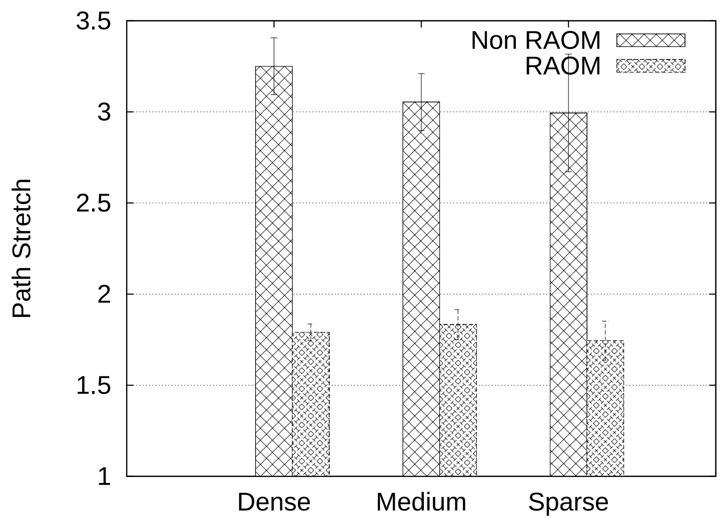


Figure 4.3: Path stretch

the number of hops that comprise the shortest unicast path between these two nodes, averaged over all trees and paths. The overlay paths are comprised of two segments, from sender to RV point and from RV point to receiver. For the first segment we use the shortest path between these two nodes: after the initial message sent to locate the RV point for a group, the sender can cache its IP address so as to avoid the overlay path for subsequent data transmissions. For the second segment we use the path constructed by the overlay scheme. In the example of Figures 4.1 and 4.2, IP multicast would result in a packet traversing 4 hops to reach node *c*, and 6 hops to reach nodes *e* and *f*. In the case of non router assisted overlay multicast, a packet requires to traverse 8 hops to reach node *c* and 16 hops to reach node *e* and *f*, resulting in an average path stretch value of 2.44. In the case of RAOM, nodes *c*, *e* and *f* would receive a packet via paths of length 6, 10 and 10 hops respectively, resulting in an average path stretch value equal to 1.61. Figure 4.3 shows the path stretch achieved for the Sparse, Medium and Dense scenarios in Topo1; by definition the stretch of IP multicast is 1. It is clear that our approach yields significantly lower stretch values in all scenarios (about 40-45% decrease), regardless of end-host density.

It should be pointed out that our results diverge from those shown in the papers evaluating *Scribe* [42, 99] in two ways. First, we measure the *entire* path between the sender and each receiver, as opposed to the path between the RV point and each receiver, since it is the entire path length that applications have to cope with. Second, we use network *hops* rather than delays to calculate stretch, since in a static environment we can only calculate the propagation and transmission delays. Since these delays may only represent a small part of the actual delays faced by applications in a dynamic environment where queuing delays due to congestion may be dominant, we believe that defining stretch using these delays would be misleading.

Transmission Stretch

Another important performance metric is the efficiency of the multicast trees produced in terms of the total load imposed on the network for data distribution. We define as the *total transmission load* the number of hop-by-hop transmissions required in order for a single packet originating from the sender of each group to reach all group members of the corresponding group. In order to provide a normalized metric, we define *transmission stretch* as the ratio between the total transmission load imposed by each overlay multicast scheme to the total transmission load imposed by IP multicast. As described earlier, in the example of Figures 4.1 and 4.2, the transmission load in the case of IP multicast would be equal to 8. In the case of RAOM, the transmission load would be equal to 12, yielding a transmission stretch equal to 1.5. In the case of regular *Scribe* the transmission load would be equal to 20, yielding a transmission stretch equal to 2.5.

Figure 4.4 shows the transmission stretch achieved for the Sparse, Medium and Dense scenarios in Topo1; by definition the stretch of IP multicast is again 1. Exploiting router assistance results in a decrease of the transmitted packets in all cases (a 7-19% decrease). This is due to the fact that the proposed router assisted multicast overlay scheme leads to the formation of multicast trees with fewer IP hops compared to the regular *Scribe* trees, leading to a more efficient utilization of the network. Recall also that, as mentioned above, our router assisted scheme

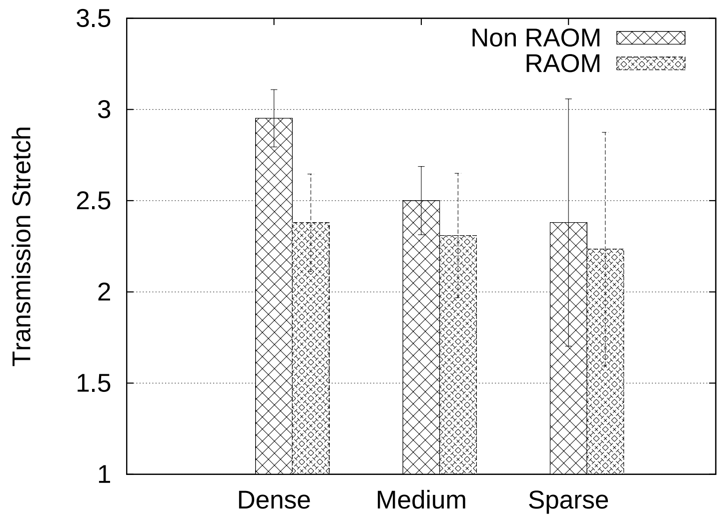
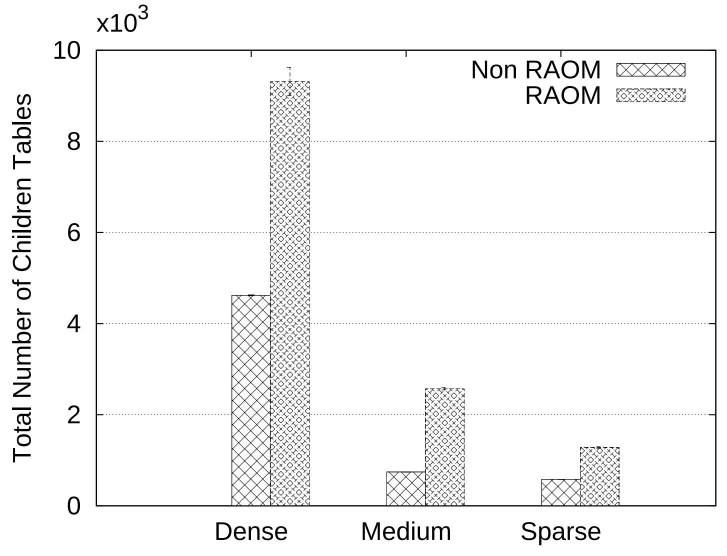


Figure 4.4: Transmission stretch

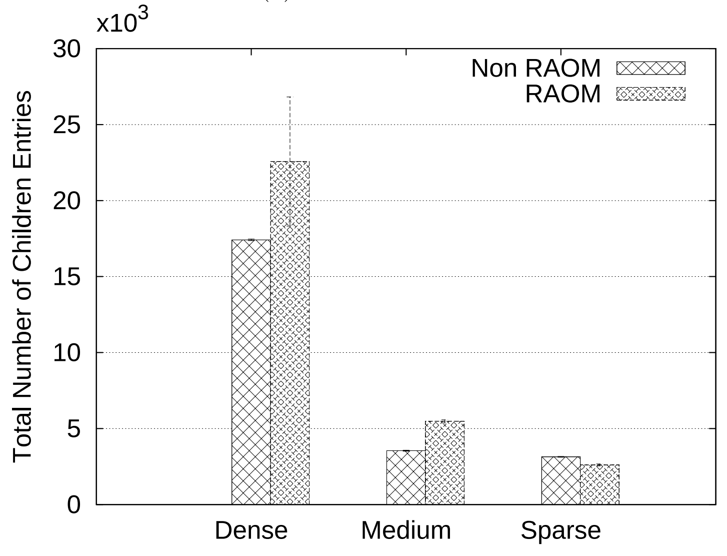
eliminates packet transmissions in the uplink direction of access links leading to intermediate nodes, thus removing a bottleneck imposed by asymmetric access links.

Node Stress

A critical metric of a multicast scheme's performance is the forwarding load it imposes on participating nodes. In a multicast scheme forwarding load can be expressed by the branching factor of each node, that is, the number of descendants it forwards traffic to; in a multi-tree scenario, attention must also be paid to the number of groups served by a node. In *Scribe* each node maintains forwarding information in a separate *children table* per group, with each table's entries referring to the node's children for that group. Hence, we calculate two metrics of node stress, as in [42, 99]. The first metric is the *number of children tables* maintained by each node, which corresponds to the number of multicast groups the node is forwarding traffic for. In our router assisted scheme, this also includes the groups for which a router is not forwarding data to other overlay nodes, but is only acting as a proxy on behalf of one or more attached receivers. The second metric is the *number of children entries* maintained by each node, which corresponds to the number of nodes it is forwarding traffic to, across all multicast groups. Again, in our scheme this also includes the group recipients attached to an access router, for which the access router acts as a proxy.



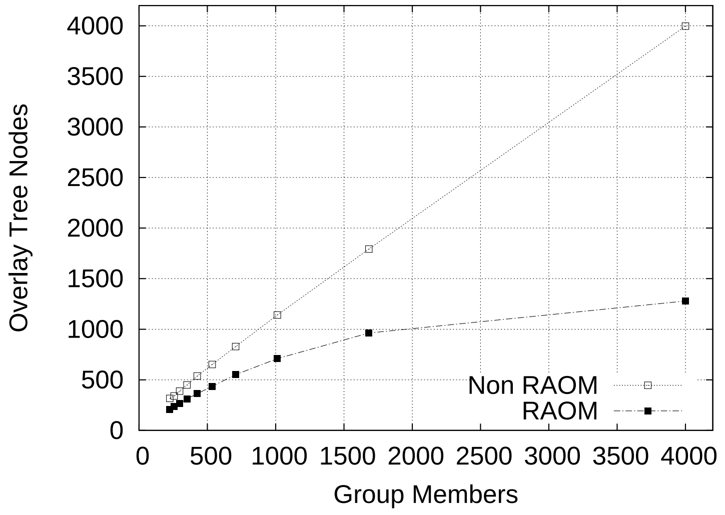
(a) Children tables



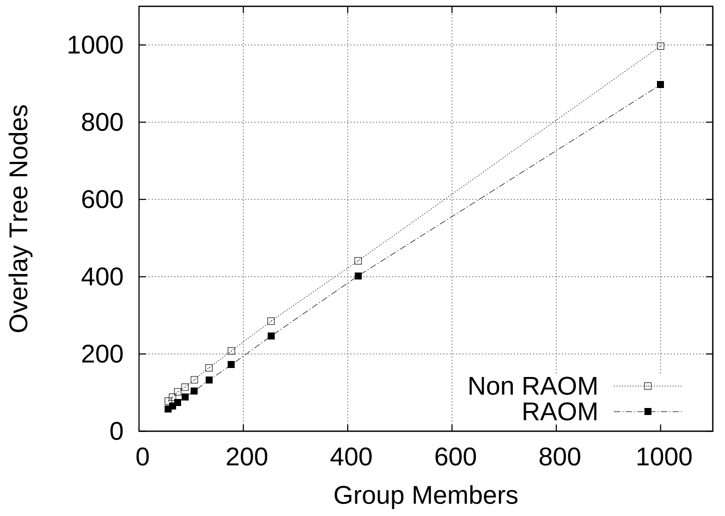
(b) Children entries

Figure 4.5: Node stress

Figures 4.5(a) and 4.5(b) depict the two node stress metrics for the Sparse, Medium and Dense scenarios in Topo1, averaged over all overlay nodes in the network; results from other topologies indicate that both metrics are mostly dependent on end-host density. The proposed router assisted scheme incurs a significant forwarding state overhead increase, especially considering that this state is concentrated on the OARs serving end-hosts participating in the overlay, rather than being distributed among the participating end-hosts as in regular *Scribe*.



(a) Topo1, Dense



(b) Topo3, Sparse

Figure 4.6: Number of overlay multicast tree nodes as function of group size

In order to assess how the amount of forwarding state affects the nodes participating in the overlay multicast routing process, Figure 4.1.1 shows the number

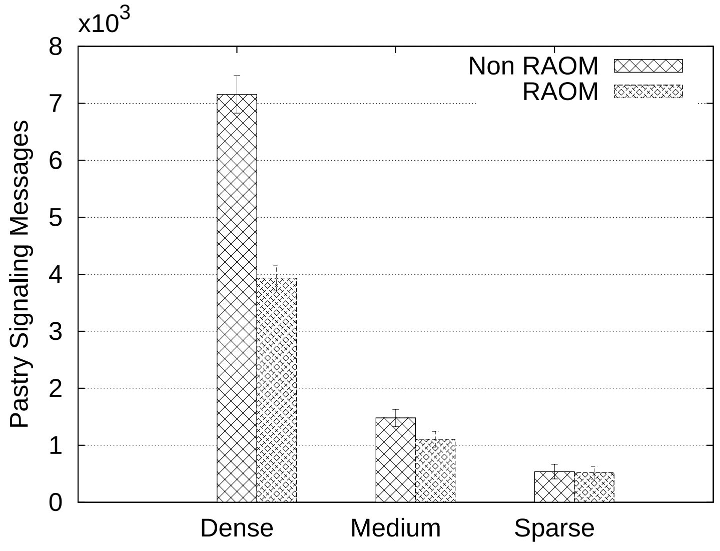


Figure 4.7: *Pastry* signaling overhead

of overlay nodes per tree, as a function of the number of end-hosts participating in the DHT substrate, either directly or via their access routers; for clarity, the figures show only the ten most popular groups in each scenario. As expected, with more end-hosts (Dense) and fewer OARs (Topo1), Figure 4.6(a) shows that the forwarding load in RAOM is concentrated on fewer nodes than in regular *Scribe*, while with fewer end-hosts (Sparse) and more access routers (Topo3), Figure 4.6(b) shows that the forwarding load in RAOM is distributed in roughly the same manner as in regular *Scribe*.

On the other hand, this reduction in the number of overlay participants also reduces the number of overlay signaling messages exchanged to establish the DHT substrate (*Pastry*). As demonstrated in Figure 4.7 for the Sparse, Medium and Dense scenarios in Topo1, the router assisted approach greatly reduces (by 45%) the signaling overhead for DHT maintenance in the Dense scenario, as in this case the access routers participate in the overlay on behalf of many end-hosts. On the other hand, in the Sparse scenario each router hardly ever serves more than one end-host, thus we see only a slight reduction (4%); in the Medium scenario the reduction lies between these extremes (25%). Note that in RAOM these messages travel shorter distances since they do not need to cross the access links (to and from the end-hosts). Therefore, there exists a tradeoff between placing the burden of forwarding and tree maintenance on the OARs and reducing path stretch, link stress, and DHT related signaling.

4.2 Incremental deployment

Having investigated the benefits stemming from the network support of overlay multicast, the question we pose next is how the proposed architecture would perform in scenarios of limited deployment i.e., we are interested in studying the potential evolutionary process towards the ubiquitous support of application-layer multicast across the Internet. In this process, we consider two fundamental dimensions, namely the Interdomain and Intradomain dimensions.

In the *Interdomain* dimension, we study the adoption of the proposed architecture by individual network operators across the Internet. Since a synchronized Internet-wide deployment seems infeasible, we believe it is important to investigate the performance of the architecture as it gradually gets deployed across the Internet i.e., as it progressively gets adopted by network operators. Specifically, we examine whether the support of overlay multicast is beneficial even if it is not universally adopted by all network operators, as well as whether there exist (performance related) incentives for a single network operator to first adopt RAOM. In the following, Interdomain deployment density is expressed as a proportion of the total number of access networks in the entire inter-network.

The *Intradomain* scenario studies the deployment density of the proposed architecture inside the boundaries of each administrative domain. Since our architecture requires the deployment of additional network infrastructure to provide the overlay functionality, it is important to examine the relationship between the size of that investment and the perceived gains for the network operator. In the following, Intradomain deployment density is expressed as a percentage of the total access domain size, denoting the proportion of access router locations gaining overlay functionality with the deployment of an OAR.

4.2.1 Evaluation

The evaluation of the incremental deployment process focuses on the gains and losses of individual *Autonomous Systems* (AS). Our purpose it to correlate the performance perceived at each AS with the degree of deployment both within the AS's boundaries and the inter-network as a whole. In this respect, we aim to reveal the potential performance related incentives for network operators to provide support for overlay multicast.

Our evaluation employs the following metrics, both measured on a per AS basis:

- *Path Length.* It refers to the number of hops traversed by data packets from the root of a tree until the end-hosts. It is expressed as the average length of all paths terminating at end-hosts within each AS, across all trees. It aims to capture the impact of network support on end user experience. Shorter path lengths are expected to increase end-user satisfaction, thus providing an advantage to a specific network operator, against operators with longer delivery paths. In the example of Figure 4.1, the path towards end-host c of Domain A has a length equal to 4, while the average path length for end-hosts in Domain B is equal to 12. In the case of RAOM, Domain A has an average path length equal to 3, while Domain B has an average path length equal to 7 (see Figure 4.2). It is stressed, that in the case of RAOM, the resulting shorter delivery paths are considered to also provide greater bandwidth by avoiding the forwarding of data over the typically slow last-mile uplink.

- *Transmission Stress.* It refers to the total number of hop by hop data packet transmissions inside an AS's boundaries for the delivery of a single data packet via all trees. It aims to capture the total traffic load imposed on an AS. It includes both transmissions required for packet delivery to end-hosts within the AS, as well as transmissions required to serve hosts in other AS's. In the example of Figure 4.1, Domain A's transmission stress is equal to 9, while the transmission stress is equal to 10 for Domain B. In the case of RAOM, Domain A has a transmission stress equal to 6 and Domain B a transmission stress equal to 5 (see Figure 4.2).[4]

In our simulations, we again employed *transit-stub* topologies following the GT-ITM model. The created topologies consisted of 24 backbone routers in 8 transit domains. Each backbone router supports a single stub domain with 50 access routers on average. The default link establishment probabilities were used again. Also, the same *Zipf*-like distribution was used for the size of each multicast group. We focus on dense topologies with 2400 users participating in 75 groups. In all scenarios, the target number of end-hosts were attached to randomly chosen stub routers. In networks without RAOM support, end-hosts enter the overlay themselves, while in networks with RAOM support, each end-host is proxied by the closest OAR.

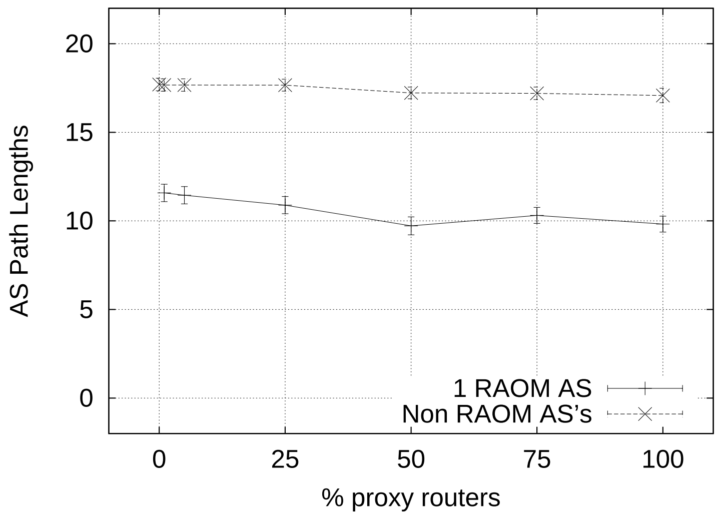
In order to evaluate the incremental deployment process described above, we created three basic scenarios that aim to capture the behavior of the proposed architecture across the interdomain and intradomain deployment dimensions.

Initial deployment

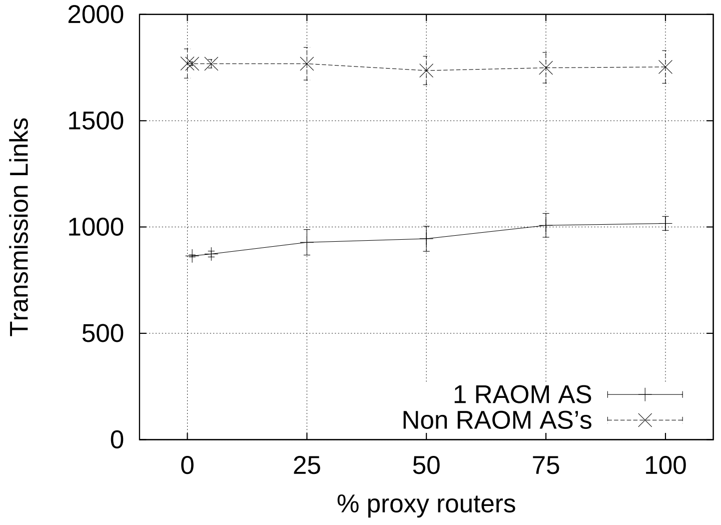
In the first scenario we focus on the initial deployment phase, where a single network operator adopts RAOM. The aim of this scenario is to investigate the potential benefits for this AS compared to other ones and whether these would provide incentives for the adoption of RAOM. As we can see in Figure 4.8 even with a relatively sparse deployment, the performance perceived at a single RAOM AS is substantially improved. More specifically, we notice an average reduction of 41% and 44% in path length and transmission stress respectively. In effect, we can see that individual operators have strong incentives to adopt RAOM, as both the traffic load on their network is reduced and the delivery of content to their customers is substantially improved.

Moreover, as intradomain deployment density increases, we notice a slight reduction of the path lengths and a slight increase of the transmission stress. In the former case, the increased number of overlay nodes results in the availability of

[4]In this example we have considered routers *4* and *6* being connected via a backbone link.



(a) Path Length



(b) Transmission stress

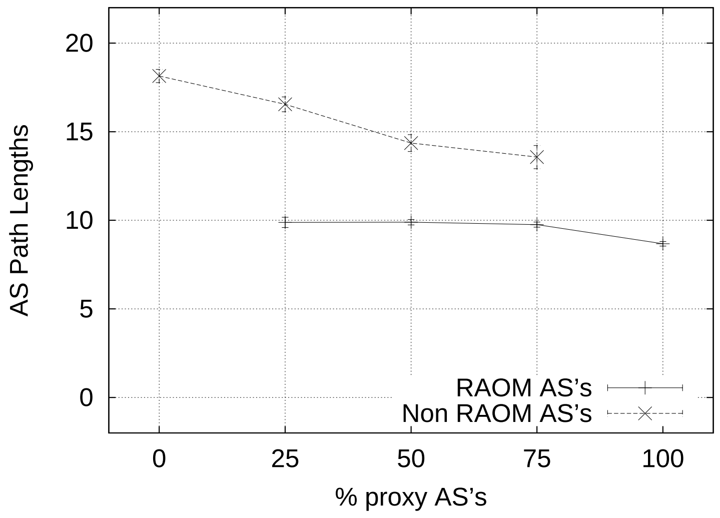
Figure 4.8: Initial deployment: 1 RAOM enabled AS vs. Non RAOM enabled AS's.

more routing alternatives in *Pastry*, allowing the proximity based selection mechanism to choose shorter paths. However, the availability of more overlay routing alternatives further allows the finer grained selection of paths with respect to the proximity in the identifier space, which combined with the slight reduction of the delivery paths denotes the increased degree of branching in the forwarding nodes of the created multicast trees. In effect, denser intradomain deployments lead to shorter trees but with increased branching. Finally, we see that the adoption of RAOM by a single network operator is not enough to affect the performance perceived in the remainder of the AS's.

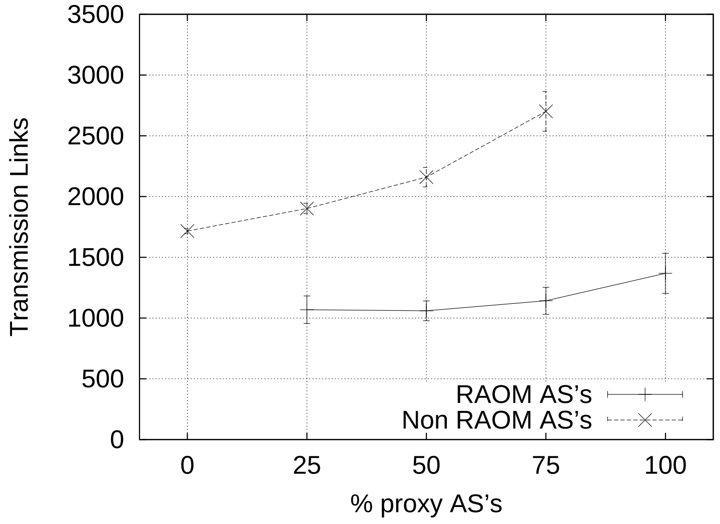
Interdomain incremental deployment

In this scenario we take a step further and investigate the performance perceived by AS's as they progressively engage in RAOM. Each participating AS enhances 50% of its routers with the proposed functionality. Figure 4.9 shows that as RAOM is adopted by increasingly more AS's, the performance perceived in these AS's is consistently better than the one perceived at AS's not supporting RAOM. In RAOM supporting AS's, data delivery paths are from 28% up to 40% shorter, while the networks of their non RAOM supporting counterparts' experience 43% to 57% higher traffic. Hence, after the initial adoption by a single AS, the incentives for deploying RAOM are still preserved throughout the progressive deployment process.

Furthermore, we notice that the inter-domain deployment density of RAOM affects both the path lengths and the transmission stress at the participating AS's. Here, in contrast with the previous scenario, the increased adoption of RAOM affects the performance perceived at non adopting AS's, since the considered multicast trees are established across the entire internetwork. Therefore, since forwarding paths may cross several AS's, RAOM enabled AS's also contribute to the reduction of forwarding path lengths in non RAOM enabled AS's. Regarding the transmission stress, we interestingly notice an increase for both types of AS's. Further investigation on this issue reveals that this increase is due to the variable deployment density across the network: in RAOM enabled networks the number of overlay nodes (i.e., OARs) is lower than the corresponding number in non RAOM networks. This is because OARs' role is to participate in the overlay on behalf of multiple end-hosts, which in the case of non RAOM networks participate in the overlay themselves. This variable density results in the increase of the forwarding tree branches established between RAOM enabled and non RAOM enabled networks. The increased overlay node population in non RAOM networks offers a correspondingly increased number of routing alternatives in the identifier space, attracting the *Scribe* Join messages from RAOM enabled AS's. This is illustrated in Figure 4.10 which shows the number of interdomain tree branches in this scenario, i.e. the total number of tree branches crossing administrative boundaries.



(a) Path Length



(b) Transmission stress

Figure 4.9: Interdomain incremental deployment (50% Intradomain deployment density).

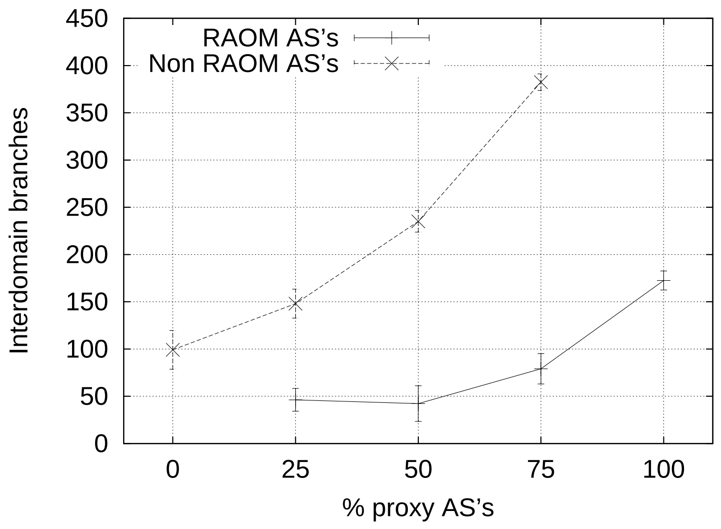
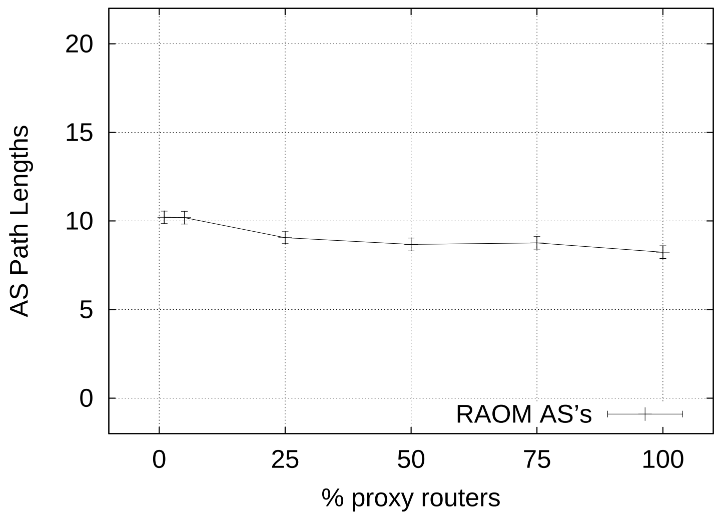


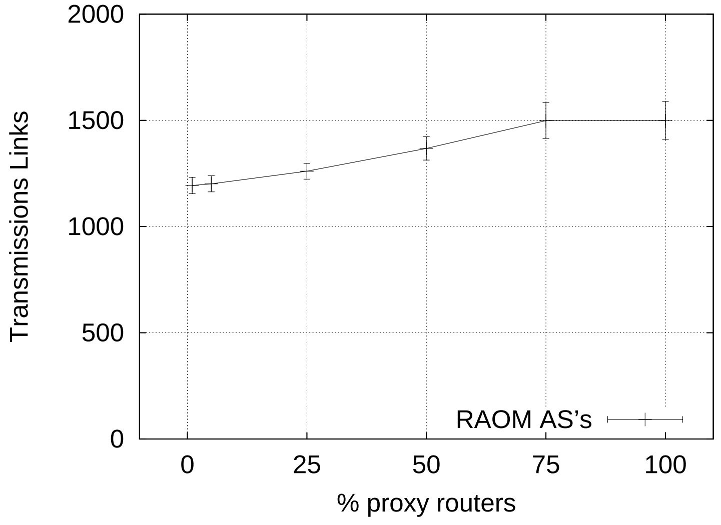
Figure 4.10: Interdomain incremental deployment: variable overlay density results in the increase of interdomain tree branches.

Intradomain incremental deployment

In this scenario we investigate the impact of deployment density inside each network in the case of ubiquitous RAOM deployment, that is, when all ASs have deployed RAOM. Figure 4.11 presents the performance perceived for different deployment densities. As the deployment level increases, so does the transmission stress on the AS's, without significant improvements on path length though. As already explained, by increasing the number of deployed overlay entities, we make the overlay routing substrate denser, therefore *Scribe* trees become denser, with more branches at each level, resulting in higher transmission stress. In this respect, an intradomain deployment level of 25-50% is sufficient for RAOM.



(a) Path Length



(b) Transmission stress

Figure 4.11: Intradomain incremental deployment (100% Interdomain deployment density).

Chapter 5

Overlay multicast and caching

This chapter presents the joint operation of multicast and caching services in the proposed architecture. First, the operation of multicast is discussed with respect to content distribution and the required signaling for the coupled operation with the caching service. Then the proposed caching scheme is presented including the required protocol signaling for the discovery and eviction of cached items, as well as the cache replacement policies employed. The chapter provides a thorough performance evaluation of the resulting content distribution architecture, also comparing it against the dominant *BitTorrent* file-sharing application. Finally, this chapter presents an analytical study of the proposed architecture's scaling properties with respect to the size of the state information maintained at each deployed OAR.

5.1 Multicast

Multicast forwarding takes place among OARs, driven by end-host requests i.e., end-hosts issue requests for desired data objects to their proxy OARs via the established control connections. These requests may be translated to corresponding *Scribe* JOIN messages, depending on the current state of the proxy OAR with respect to the indicated data item. The joining process deviates slightly from regular *Scribe* in that the JOIN messages are extended to further carry the IP address, the listening port number, the credentials of the initial issuer of the JOIN message (i.e., the proxy OAR[1]) and the 32-bit *Autonomous System* (AS) number of the proxy OAR's AS [101]. This extra information is used during cache searching and provisioning, as explained in the next section.

During the joining process, OARs establish TCP connections with their children for the reliable delivery of the requested data. When a JOIN message eventually reaches the *Rendez-Vous* (RV) point, the content provider will be solicited to deliver the data which will then start traversing the tree created via the already established TCP connections. It is assumed that the content provider has already created the respective group, and therefore has contacted the RV point. Due to the asynchronous character of request arrivals, this process may result in partial data availability at the leaves of the multicast tree, at the end of the multicast session. However, the caching mechanism ensures that these partial feeds will complete later.

A simple example of these operations is given in Figure 5.1. The subfigures depict progressive snapshots of a simple *Scribe* tree. The arrays below each OAR denote the availability of the content and will be further explained in Section 5.2.5. OAR *1* first joins a *Scribe* multicast tree via OAR *2*, followed by OARs *3* and then *5* that join during the multicast session via OAR *4*. At the end of the multicast session (end of Step 3) , OARs *3* and *5* have received part only of the multicasted content, i.e., they are missing blocks 0 to 3 and 0 to 6 respectively.

5.2 Caching

Caching plays a vital role in *MultiCache*. *MultiCache* aims at to reduce network traffic by enabling localized cache hits. To this end, caches are located at proxy OARs, i.e., at the leaves of multicast trees, and cache the content they receive via multicast or feeds from other caches. In effect, multiple caching locations are available in the entire network. In the following, we describe our caching scheme in terms of cache discovery as well as management of indexing state and caching space.

[1]Note that in cases of multi-overlay hop paths, the proxy OAR is not the node that eventually delivers the JOIN message to an already joined node.

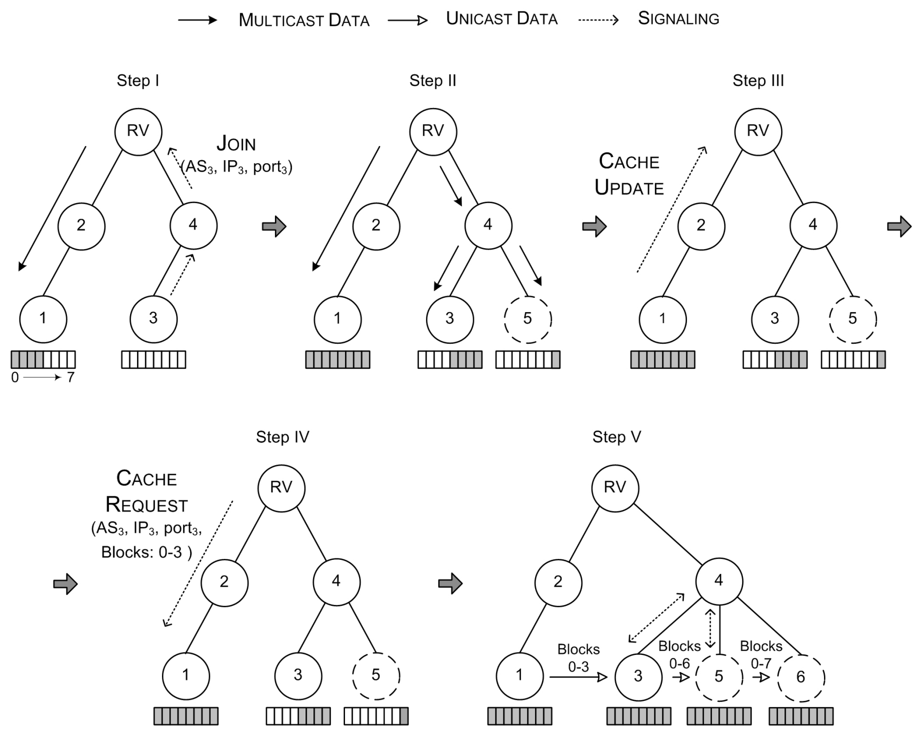


Figure 5.1: *MultiCache* example

5.2.1 Cache discovery

In order to locate an available cache, *MultiCache* uses the already established overlay multicast forwarding state. The OARs cache the forwarding state established during tree creation even after the end of a multicast transmission. As caches are created, Cache Update messages are issued by leaf OARs towards the RV of the multicast tree. The purpose of these messages is to notify ancestors about the availability of cached items downstream and allow their discovery upon cache requests. Note that caches may be fed by other caches, therefore OARs cannot rely on forwarded traffic in order to deduce cache availability below them. OARs further propagate received Cache Update messages towards the root *iff* they have not already done so for another downstream cache, thus avoiding feedback implosion.

When an end-host requests data, its proxy OAR creates a *Scribe* Join message, unless it has already a cached copy (direct cache hit). As in regular *Scribe*, a Join message is suppressed at the first OAR that has already joined the respective tree, henceforth termed as a *meta-cache* OAR. What happens next, depends on the state of the meta-cache OAR with respect to the indicated object. If the requested object has been cached by the meta-cache OAR itself, the cached data will be directly delivered to the requesting node. If the data are not cached but the meta-cache OAR has previously completed forwarding the data object to its descendants[2], it will anycast a Cache Request message to the sub-tree below it in a *depth first search* (DFS) fashion, carrying all extra information inserted in the Join message. In particular, at each level of the traversed sub-tree this message is forwarded to one of the children that have previously issued a Cache Update message. At each step, preference is given to children belonging to the same AS with the requesting proxy OAR. Among equivalent candidates, a randomized selection ensures the uniform distribution of load to the available caches. Eventually, a Cache Request reaches an OAR that has cached the data, and a TCP connection is established between the caching OAR and the proxy OAR for the delivery of the cached data.

Finally, if the meta-cache OAR is currently forwarding the requested data, it forwards the arriving multicast data to the joining node, keeping also track of the part of the data object that was not delivered due to the late arrival of the Join message. Upon the arrival of the first Cache Update notification a data receiver, a Cache Request is issued for the missing data for each partially served child of a meta-cache OAR, thereby reverting to the previous case.

In the example of Figure 5.1, due to the arrival of OAR *3*, the RV node receives a Join message from OAR *4* including OAR *3*'s details (Step I). At the end of the multicast session OAR *1* completes downloading the data and notifies its parent with a Cache Update message which is eventually suppressed by the RV point (Step III). At this point, having partially served its descendants, the RV node issues a Cache Request towards the only child known to lead to a cache, i.e. OAR *2* (Step IV). Since this node has not cached the requested item, it forwards the received message to OAR *1* which eventually serves the request (Step

[2]The anticipation of this fact is based on content fragmentation (see Section 5.2.5) and a simple block counting mechanism.

V). The same procedure takes place in the case of OARs *5* and *6*, which belong to a different AS than OARs *3* and *4*. OAR *5* joins first and receives the cached data from OAR *3*. Later, OAR *6*'s Join message is suppressed by OAR *4* that prefers to send a Cache Request message to OAR *5* than to OAR *3*, resulting in localized traffic between OARs *5* and *6*.

5.2.2 Cache eviction

In *MultiCache*, cache availability is correlated with the overlay multicast forwarding state. This allows requests for content to lead to either the multicast-based delivery of data or to a cache hit, while preserving the locality properties of the established tree structure (see Section 5.2.4) and avoiding extra control overhead for the discovery of cached objects. In practice, this means that cached items are not evicted from a cache, unless the corresponding multicast forwarding state is torn down.

To synchronize caching and forwarding state, we have slightly altered *Scribe*'s Leave procedure. A caching OAR issues a *Scribe* Leave Request message for the tree serving the cached object marked for eviction and this message propagates up the tree until either the first node with additional children or the RV point is encountered. A Leave Response message is then issued towards the leaving child, thus tearing down the forwarding state and removing the cached data. In contrast, in regular *Scribe* the forwarding state is removed immediately upon the reception of a Leave message. The new procedure ensures that eventually, all requests for data reach either a caching OAR (in the form of Cache Request messages) or the root, if no other cache location is available (in the form of *Scribe* Join messages). In the latter case, the content provider is solicited to provide the desired object again via the established multicast delivery path. The above procedure is illustrated in the simple example of Figure 5.2. Node *2* will not remove node *1* from its forwarding table until a Leave Response message is received from node *3*. In the meanwhile, node *4*'s issued Cache Request will be normally served.

This cache eviction mechanism is employed either in the context of cache replacement or in the case of invalidated cached content. In the latter case, we

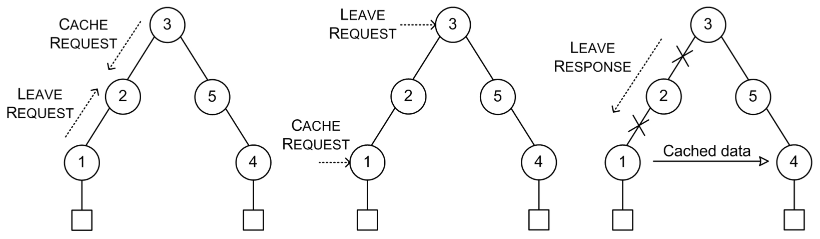


Figure 5.2: *MultiCache* Leave procedure

consider a *time-to-live* (TTL) mechanism that triggers cache eviction upon expiration. The estimation of the appropriate TTL value is application specific and out of the scope of this work. In our content distribution application described in Section 5.3, we consider non versioned content (e.g. media files) with infinite TTL values.

5.2.3 Cache replacement

As mentioned above, OARs always cache the content delivered due to an end-host request. When a request arrives at an OAR with an exhausted cache space a cache replacement policy is employed to select an item for eviction. Common replacement policies (e.g., *Least Recently Used* (LRU)) aim at adjusting cache contents to request patterns so that less popular items leave space for more popular ones, thus increasing the cache hit ratio. In *MultiCache*, cache replacement aims at taking advantage of the multiplicity of cache locations inside an administrative domain. To this end, caching OARs keep track of the popularity of each cached item with respect to the frequency and/or recency of hits from other OARs inside the domain. Since all content delivered to an OAR is locally cached, such hits imply that additional copies of the same object are probably cached nearby. In *MultiCache* we examined the suitability of the *Most Recently Used* (MRU) and *Most Frequently Used* (MFU) policies; these policies favor the selection of items that are most likely to be available at other cache locations. It must be stressed that common replacement policies such as LRU and LFU refer to the recency/frequency of requests referring to the entire data item (file). In *MultiCache*, caching, and therefore cache replacement, takes place at a fragment level (see Section 5.2.5), allowing the partial caching of files. The examined MFU and MRU policies aim at reflecting the existence of specific cache locations and therefore are enforced on fragments, regardless of the data item each fragment belongs to. In this manner there is no need for control signaling and state overhead for the association of single pieces with the corresponding data items.

5.2.4 Locality properties

MultiCache favors localized cache hits by building upon *Pastry*'s locality properties and the multiplicity of cache locations. According to *Pastry*'s route convergence property, since caching OARs are essentially leaves of a data item's *Scribe* multicast tree, a *Scribe* JOIN message from a proxy OAR is expected to reach a meta-cache OAR at a distance approximately equal to the distance between the proxy OAR and a caching OAR in the proximity space. At the same time, following *Pastry*'s prefix based routing (see Section 3.3.1), *Scribe* JOIN messages are initially expected to travel short distances at each overlay routing step. Hence, as demonstrated in [102], in cases of multiple cache locations, *Scribe* JOIN messages from proxy OARs are expected to first reach nearby meta-cache OARs, thus leading to closely located caches. In effect, cache search messages and cached data are expected to traverse short network distances, with respect to *Pastry*'s proximity metric, leading to the localization of traffic. This is further enhanced by the simple AS number-based cache selection mechanism.

5.2.5 Content fragmentation

MultiCache allows the fragmentation of large files into *pieces*, in a *Bit-Torrent* fashion, leading to the creation of a forest of *Scribe* trees, resembling SplitStream [96], but without the explicit goal of creating disjoint trees. This fragmentation serves several important goals. First, it facilitates the establishment of parallel data flows towards a recipient node, possibly exploiting the available downlink bandwidth and avoiding the sequential delivery of large files. Furthermore, it allows the partial caching of large data volumes, i.e., certain pieces can be cached independently of others, enabling the fine grained management of caching space [73]. The establishment of partial caches in different network locations favors the establishment of disjoint delivery paths, facilitating the distribution of forwarding load and the localization of traffic. However, these benefits come at the cost of forwarding state which increases with the size of the resulting forest. Pieces are further partitioned into blocks, again as in *BitTorrent*. This second level of fragmentation facilitates the provision of data from multiple sources. For example, as explained in the previous section, an OAR may join a multicast tree while data are in transit, in which case the first part of the piece shall be later provided by a cache.

5.3 Evaluation

In order to provide a realistic application model for the evaluation of the proposed architecture, we designed a *MultiCache*-based content distribution application that can be directly compared to regular *BitTorrent*. In our application, a content provider employs content fragmentation to create multiple trees for the delivery of a single file. All identifiers are retrieved by end-hosts via out-of-band means, e.g., a *MultiCache*-torrent file. Upon arrival to the network, end-hosts connect to their proxy OAR and submit requests for pieces of the file. The number of pending requests is capped, in an analogy to regular *BitTorrent*. Each node submits its requests independently of other end-hosts, since we cannot assume any form of collaboration between end-hosts.

We selected content distribution and *BitTorrent* as a representative case of information-centric communication that is forced by the current information agnostic model to be fully controlled by nodes at the edges of the network. Our target is first to examine the degree by which information awareness, as introduced in the network by *MultiCache*, enables the efficient utilization of network resources; and at the same time investigate how end user experience is affected. Taking then a step further, we delve into the details of the proposed architecture and investigate its performance with respect to key parameters.

5.3.1 Simulation environment

The evaluation of *MultiCache* is based on a detailed full stack simulation environment based on the OMNeT++ Simulator [45] and the OverSim Framework [98]. The *MultiCache* content distribution application is compared against our own *BitTorrent* implementation for OMNeT++[103]. In our simulations we again used Internet-like topologies based on the GT-ITM model (see also Section 4.1.1). For our measurements, we created topologies comprised of 1225 routers hierarchically organized in 25 stub and 5 transit domains.

We generated synthetic traces of request arrivals for several files across the network. To generate this workload we used features of the ProWGen trace generation tool [104]. To better reflect the characteristics of a P2P application we replaced the Zipf distribution of file popularities with the *Mandelbrot-Zipf* distribution proposed in [73]. A certain number of requests is generated for each file in the workload, according to the file's popularity. All file requests follow the exponentially decreasing arrival rate process described in [46], parameterized according to the popularity of the corresponding file. We interleave these single file traces by placing the first request of each file at a constant time interval after the first request for the previous file. This reflects the constant torrent arrival rate observed with *BitTorrent* in [46]. File sizes were sampled from the traces in [105]. The content providers, one per file, are uniformly distributed across the entire network. Each of the generated requests is then assigned to one of the 100 end-hosts we attach at a randomly chosen access router of the topology.

We have considered different bandwidth allocations for the uplink and the downlink directions of the access links, as current access technologies, such as ADSL, present this asymmetry. For the measurements presented below, we employed various downlink bandwidth values ranging from 4 to 24Mbps, and uplink bandwidth values of 1 or 2 Mbps. These values were distributed across end-hosts as shown in Table 5.1. Backbone routers are connected with 10GBps links, while 1GBps links connect access routers.

Finally, due to concerns related to the scalability of the simulation environment we do not consider any type of background traffic in the network. Therefore,

Uplink (Mbps)	**Downlink** (Mbps)	**Fraction**
1	4	0.20
1	8	0.40
2	16	0.25
2	24	0.15

Table 5.1: Bandwidth distribution of access links.

the data transmissions in the network are related solely to the investigated distribution of content and do not get affected of any other data flows. Hence, the possible effects of the interaction with background traffic, such as congestion triggering TCP's congestion control mechanism, are not taken into account by the created simulation environment. Such interaction would be expected to worsen the performance of both the *BitTorrent* application and our architecture due to the retransmissions of lost TCP segments. Additionally, the disc access delay in *MultiCache*'s caches as well as the delay for state management at the OARs (e.g., forwarding table lookups) are not taken into account.

5.3.2 Evaluation framework

Our comparison against *BitTorrent* first concentrates on the utilization of network resources in terms of egress inter-domain traffic (EIT) and intra-domain traffic load (ITL) incurred for the delivery of the content. In the former case, we measure the total number of bytes of egress traffic at each stub domain, thus expressing the amount of traffic that network operators must pay for, in order to reach the transit domain. In the latter case, we measure the aggregate number of block transmissions over all links of a domain. This metric allows as to assess the load imposed on each administrative domain. We then proceed in comparing the download times experienced by the end-hosts in each case.

In order to explore the properties of *MultiCache*'s caching scheme, our first metric is the achieved *cache hit ratio* (CHR). To further study the localization of traffic within AS boundaries, we also measure the *intra-domain cache hit ratio*

(CHR-Intra) which reflects only cache hits on OARs residing in the same AS as the end-host receiving the cached data. Finally, we measure the *distance to block source* (DBS), i.e., the average number of physical hops traversed by blocks arriving at end-hosts, in order to assess the overall locality properties of data transfers.

Following the methodology in [106], we consider relative cache sizes (S_r), i.e., the cache size is expressed as a fraction of the "infinite cache size", which is the minimum cache size required to avoid replacements. We also examine the effect on these metrics of *MultiCache* deployment density $d \in [0, 1]$, defined as the fraction of the access routers that are enhanced with *MultiCache* functionality. In the example deployment of Figure 3.2, the density for domain A is $d_A = 2/8 = 0.25$ while for domain B it is $d_B = 1$. In this evaluation, we assume uniform density values across all AS's. Unless otherwise stated, the presented results refer to scenarios with $d = 0.5$, i.e., we examine average case scenarios with not particularly dense or sparse deployments.

Finally, we investigate the ability of *MultiCache* to localize traffic inside domain boundaries depending on the popularity of data objects inside the domain. For this reason we define the *localizability* parameter $l \in [0, 1]$ as an indicator of the concentration of end-hosts (and the corresponding requests) inside domain boundaries. If we denote as D the total number of administrative domains in the topology, then end-hosts are uniformly distributed across $\max[(1 - l)D, 1]$ administrative domains. At $l = 1$, all end-hosts reside in the same AS, while at $l = 0$, they are uniformly distributed across all AS's.

5.3.3 Results

Traffic

Figures 5.3 and 5.4 present the CDF of the EIT and ITL metrics respectively, for the average scenarios of $d = 0.5$ and $l = 0.5$. In both cases *MultiCache* achieves a substantial decrease in the traffic incurred. On average, *MultiCache* reduces EIT by 53,19% and ITL by 61,39%. Obviously, this reduction is a consequence of caching, that enables the localization of traffic, as further demonstrated by the perceived CHR values (see Section 5.3.3). At the same time, the funda-

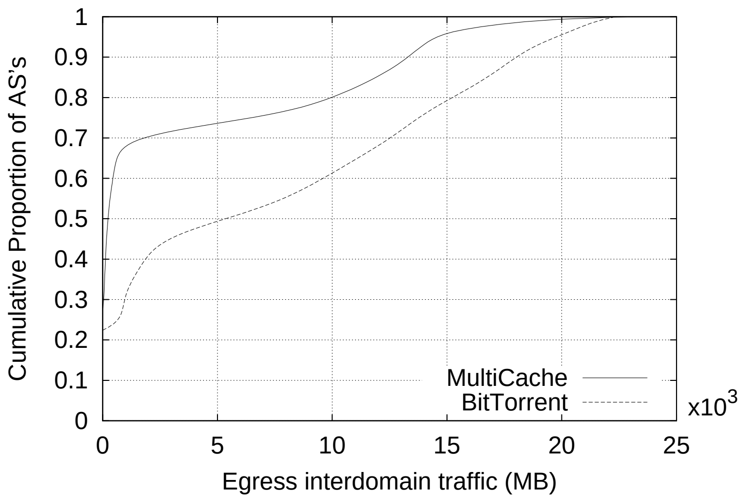


Figure 5.3: Comparison between *MultiCache* content distribution application ($l = 0.5, d = 0.5$) and *BitTorrent*: EIT

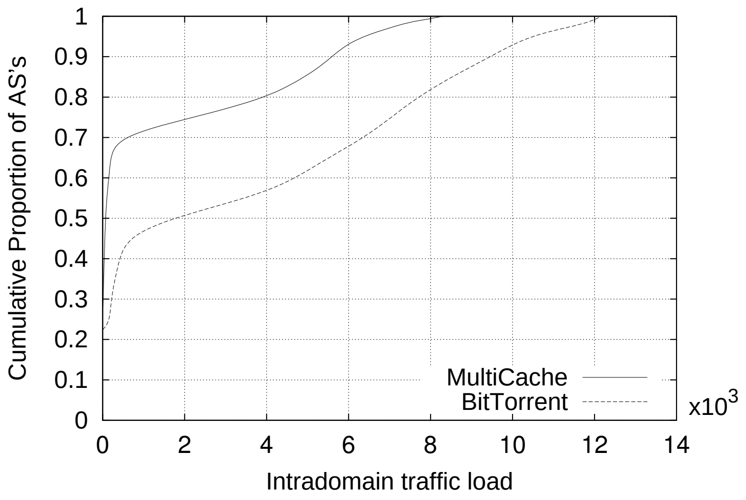


Figure 5.4: Comparison between *MultiCache* content distribution application ($l = 0.5, d = 0.5$) and *BitTorrent*: ITL

mentally different application model of *MultiCache* avoids the exchange of data between end-hosts in a P2P fashion. This is evidently demonstrated in Figure 5.3, in which we notice considerably low EIT values for almost 70% of the *MultiCache* enabled AS's i.e., end-hosts do not generate traffic in order to retrieve the desired content but benefit from previous downloads. We consider this as a direct benefit of the information-centric network model and the employed caching mechanism, that enables network operators to gain back control of the traffic carried by their networks. An emerging question then, is whether this incures a penalty in the quality of service experienced by the end users.

User Experience

Figure 5.5 presents the CDF of the download times (DT) achieved in both cases of *MultiCache* and *BitTorrent*, over all downloaded items. The download time perceived with *MultiCache* is on average 56,35% lower than in the case of *BitTorrent*. This huge reduction is accredited to three important factors. First, caching allows the content to be locally stored and provided. This is also depicted by the achieved cache hit ratios shown in the following subsection, as well as the average number of hops traversed by data blocks, i.e., in the case of *BitTorrent* it is 8.6 hops, while in *MultiCache* it drops down to approximately 6 hops. Second, in the case of *MultiCache* end-hosts do not engage in a search for the required data among participating peers. This is a direct consequence of the information-centric model, in which users simply request data from the network. Third, the download rate of end-hosts is not capped by the uplink of their peers but by the forwarding capacity of the OARs, which is typically higher.

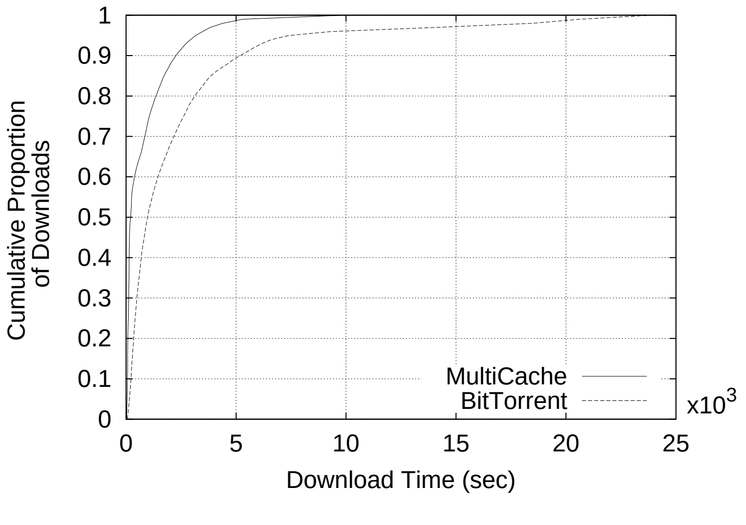


Figure 5.5: Comparison between *MultiCache* content distribution application ($l = 0.5, d = 0.5$) and *BitTorrent*: DT

Cache size and replacement policies

Figures 5.6 and 5.7 show the CHR and CHR-Intra achieved with the LRU, MFU and MRU cache replacement policies, for relative cache sizes (S_r from 0.5% to 20%). Interestingly, all these policies exhibit approximately the same behavior for all cache sizes considered. Note that the CHR reaches values up to 98.5% for higher S_r values, thus reducing the amount of data delivered via overlay multicast and taking advantage of available caches throughout the entire network. As a result, *MultiCache* reduces the impact of overlay multicast stretch and takes advantage of *Pastry*'s proximity properties in order to locate nearby copies of the desired data. This is clearly demonstrated in Figure 5.8: as the cache size increases, the average distance to block source decreases, denoting the delivery of content from nearby caches. Again, no cache replacement policy exhibits superior performance, leading us to the adoption of the MFU policy due to the simplicity of its implementation.

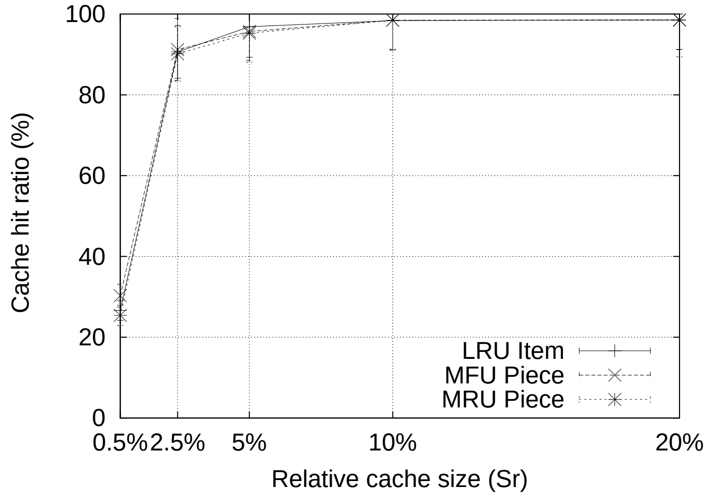


Figure 5.6: Effect of cache replacement policies on CHR ($l = 0, d = 0.25$)

Deployment Density

Since *MultiCache* necessitates the deployment of additional infrastructure (OARs) by network operators, a crucial issue for the viability of the proposed architecture is the magnitude of the investment required. Figure 5.9 shows the cache

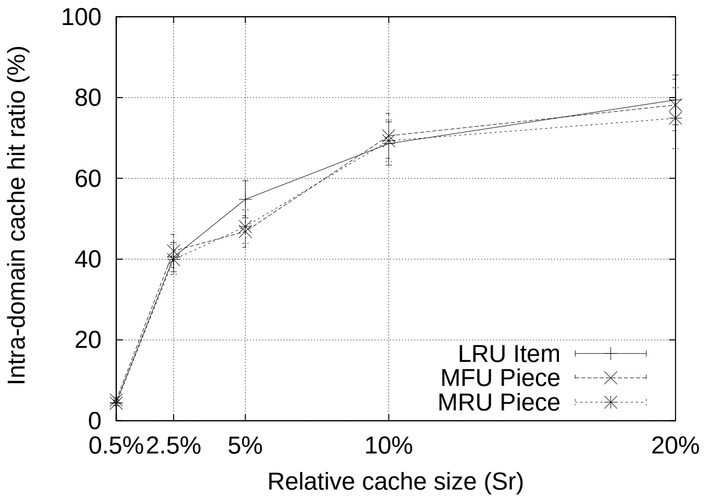


Figure 5.7: Effect of cache replacement policies on CHR-Intra ($l = 0, d = 0.25$)

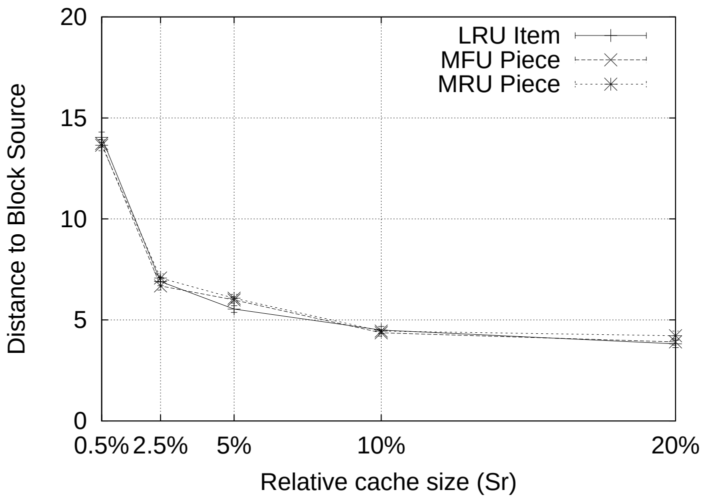


Figure 5.8: Effect of cache replacement policies on DBS ($l = 0, d = 0.25$)

hit ratio for various deployment densities and relative cache sizes. We see that CHR increases with deployment density, following the increase of the aggregate caching space in the inter-network. For relative cache sizes ranging from 2.5% to 20% of the "infinite cache size" the perceived CHR is always greater than 80% at a deployment density of 25%. Figure 5.10 shows that CHR-Intra ranges from 46% to 88% for higher relative cache sizes, meaning that this portion of traffic is held inside domain boundaries and explaining the observed reduction of EIT and ITL in comparison with the *BitTorrent* application. As the density increases however, local cache hits decrease, due to the reduced degree of request aggregation at caching OARs and the corresponding reduction of direct cache hits at proxy OARs. This is also depicted by the modest reduction of the average network distance travelled by data blocks, shown in Figure 5.11. While denser deployments result in more cache locations, and therefore shorter distances between proxy and caching OARs, they also mean that similar requests and the resulting cached content are distributed across a correspondingly larger number of locations.

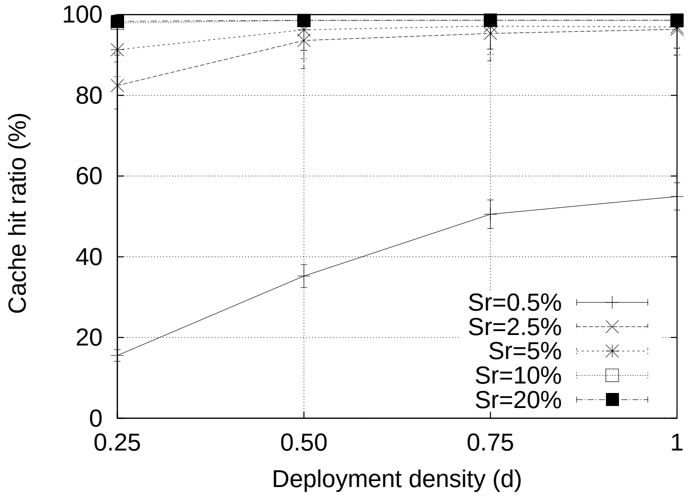


Figure 5.9: Effect of deployment density on CHR $(l = 0.5, MFU)$

Figures 5.12, 5.13 and 5.15 further examine how the perceived cache hit ratios translate to actual network traffic in the network and how they affect the DT. The traffic incured in the case of the *BitTorrent* application is also presented. We see that, in most cases, both the EIT and the ITL increase with deployment density. This is a direct effect of the decrease of CHR-Intra and the concurrent increase of CHR. As the deployment density increases, similar requests are dispersed across OARs. On the one hand, this reduces the request aggregation at caching OARs, as mentioned above. On the other hand, more cached copies are created throughout the network resulting in the increasing CHR. In essence, in denser deployment scenarios, cache hits are distributed across the network, thus increasing the traffic load. This is also in accordance with the increase of the transmission load perceived

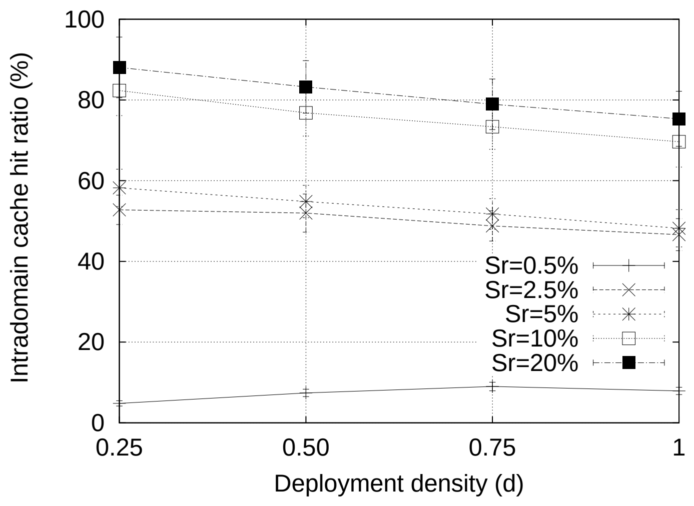


Figure 5.10: Effect of deployment density on CHR-Intra ($l = 0.5, MFU$)

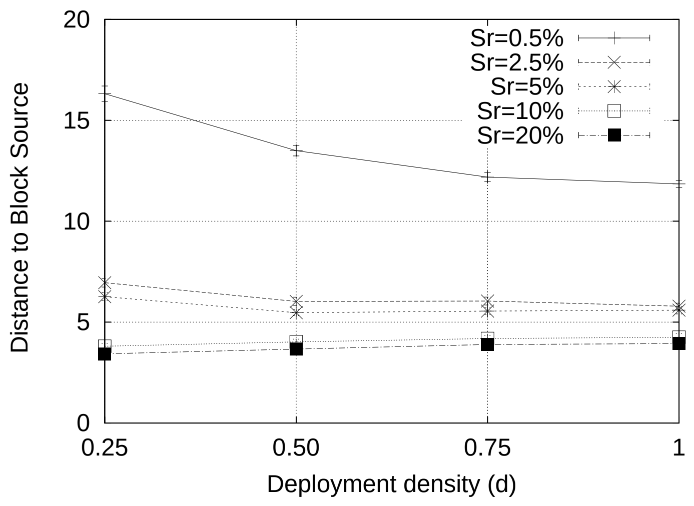


Figure 5.11: Effect of deployment density on DBS ($l = 0.5, MFU$)

for dense deployments in the case of simple overlay multicast (see Section 4.2). However, we notice that the aforementioned trend is not followed in the case of small cache sizes and sparse deployments (i.e., $S_r = 2.5\%$ and $S_r = 5\%$, for $d = 0.25$), where more traffic is generated in the network than in the case of small cache sizes and denser deployments (i.e., $S_r = 2.5\%$ and $S_r = \%$, for $d = 0.50$). Even though the sparse deployment results in higher CHR-Intra values, the small size of the caches results in the increased use of multicast. This is illustrated in

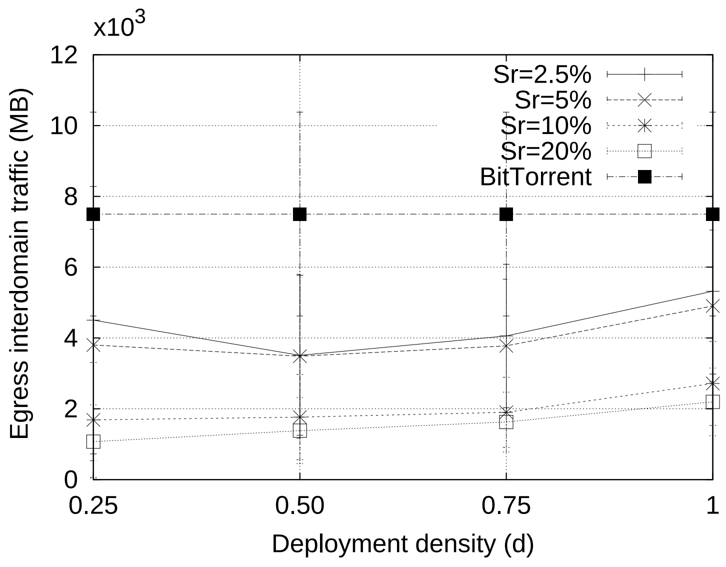


Figure 5.12: Effect of deployment density on EIT $(l = 0.5, MFU)$

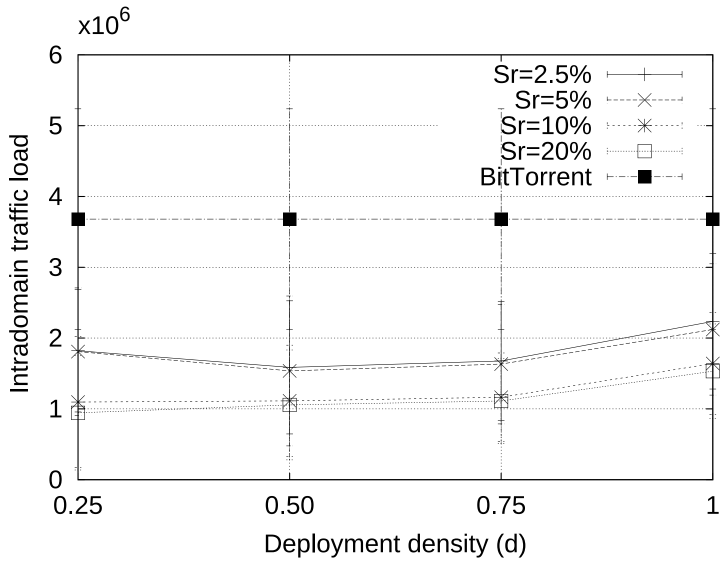


Figure 5.13: Effect of deployment density on ITL $(l = 0.5, MFU)$

Figure 5.14, which shows the average ratio of the data received at the end-hosts through multicast (denoted as Multicast ratio). In these cases, the caching space is not enough to accommodate the increased workload thus triggering the transfer of the content from the content provider and not a nearer caching location.

Figure 5.15 shows the impact of deployment density on the download time perceived by the users. Interestingly, we see the DT initially increasing with de-

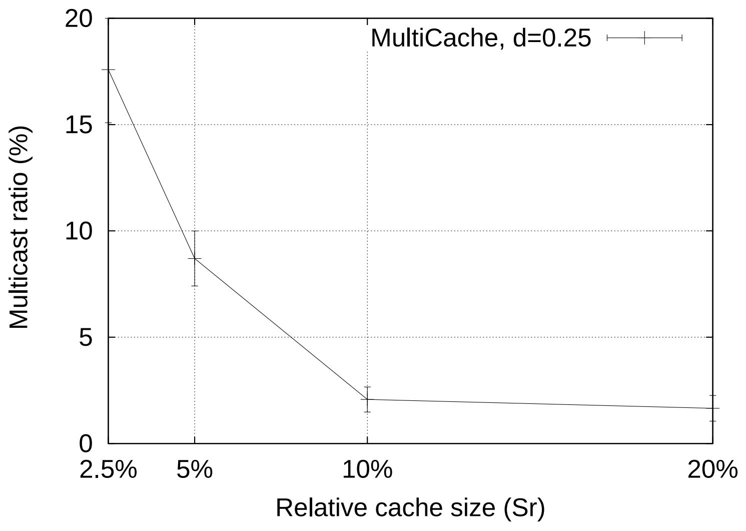


Figure 5.14: Ratio of data received via multicast in sparse deployments ($d = 0.25, S_r = 2.5\%, MFU$)

ployment density and finally decreasing for very dense deployments. This behavior reflects the tradeoff between the higher degree of request aggregation in the case of sparse deployments, against the increased caching space in the case of dense deployments. Initially, the increased CHR-Intra results in low download times as the end-hosts receive the data from nearby cache locations. However, as deployment density increases the dispersion of cache hits across the network, the perceived download times increase, up to the point where the availability of multiple cached copies results in high CHR values that allow for the reduction of the download time. It is noted that in highly dense deployments the dispersion of cache hits across the network is not reduced, however, with more caching locations in the network, the increased amount of caching space finally allows for higher cached data availability, with less cache replacement events taking place.

In all, we see that sparse deployments of high storage capacity OARs result in the best performance. While denser deployments provide a higher aggregate cache capacity in the entire inter-network, they also result in the distribution of the requests across the participating administrative domains, increasing the traffic load. At the same time, sparse deployments of small cache size OARs cannot cope with the increased workload without triggering the multicast based distribution of

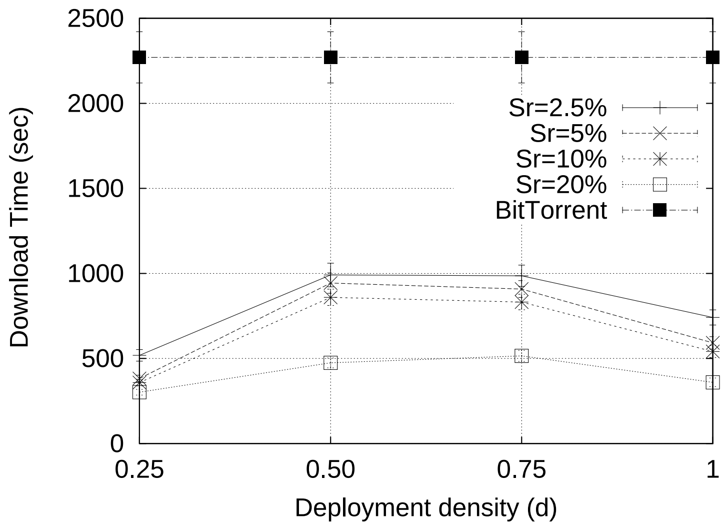


Figure 5.15: Effect of deployment density on DT ($l = 0.5, MFU$)

content, which also increases the traffic in the network, compared with the unicast transmission of content from nearby caches.

Localizability

Figure 5.16 shows the effect of the localizability factor on the cache hit ratios for S_r =2.5%. Since the delivery of content to an OAR results in the availability of that content at the corresponding cache, localizability essentially expresses the availability of content inside domain boundaries. However, as localizability increases, so does the competition for caching space: the more data being delivered to a domain, the more content has to be cached locally within a fixed cache size. Hence, the content arrives at a domain but gets evicted due to the increased load on the caches. At the same time, content concentrates at intra-domain cache locations, due to co-located requests, therefore the portion of intra-domain cache hits rises, up to the point where all cached content is provisioned by a local cache. This means that *MultiCache* does takes advantage of localized request patterns, but only up to the point where cache size limitations do not allow further improvements.

The impact of localizability on network traffic and the perceived download

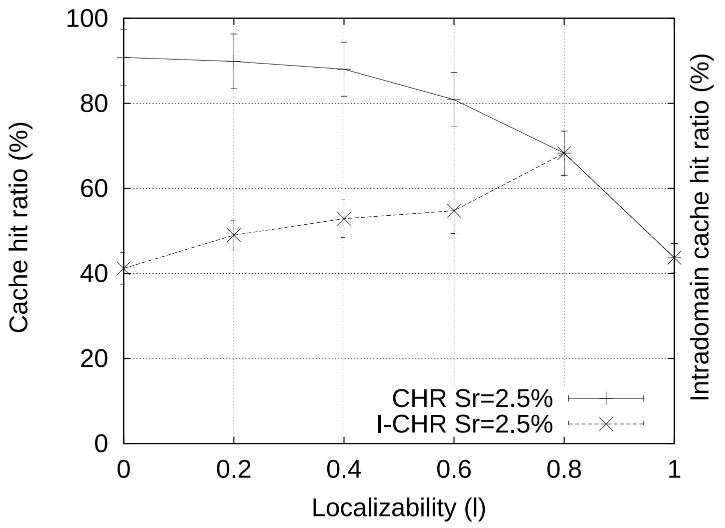


Figure 5.16: Effect of localizability on cache hit ratio ($d = 0.25, S_r = 2.5\%, MFU$)

times is illustrated in Figures 5.17, 5.18 and 5.19, which also present the performance of the *BitTorrent* application. As expected, the EIT decreases for both *MultiCache* and *BitTorrent*, as requests tend to be localized in fewer administrative domains: data need to be transfered to a smaller set of domains thus reducing interdomain traffic. *MultiCache* outperforms *BitTorrent* in almost all cases, taking advantage of the localized character of requests. However, for $l = 1$, in which case all requests are concentrated in a single domain, *MultiCache* incurs more EIT. The EIT in this scenario is incurred by the transfer of the content from the origin domains i.e., the domains where the content providers reside, to the single domain of the entire end-host population. In this process, *BitTorrent* employs unicast shortest paths connecting the content providers with (some of) the peers. On the other hand, *MultiCache* employs overlay multicast trees to bring the content to the domain of the end-hosts, and in this process the delivery paths are stretched (see Chapter 4). Hence, it is possible for the delivery paths to pass through third party domains i.e., domains that neither host content providers nor end-hosts requesting the content, thus increasing the average EIT. This issue is further studied in Chapter 6. Regarding the download times perceived by the end-hosts, we notice a slight increase in the case of *MultiCache*, which is due to the aforementioned stress imposed on the caches due the increased workload in the domain. It is noted that as localizability increases, the effectiveness of several caches across the network diminishes as a result of the absence of end-host requests which would result in the corresponding data arriving and getting cached.

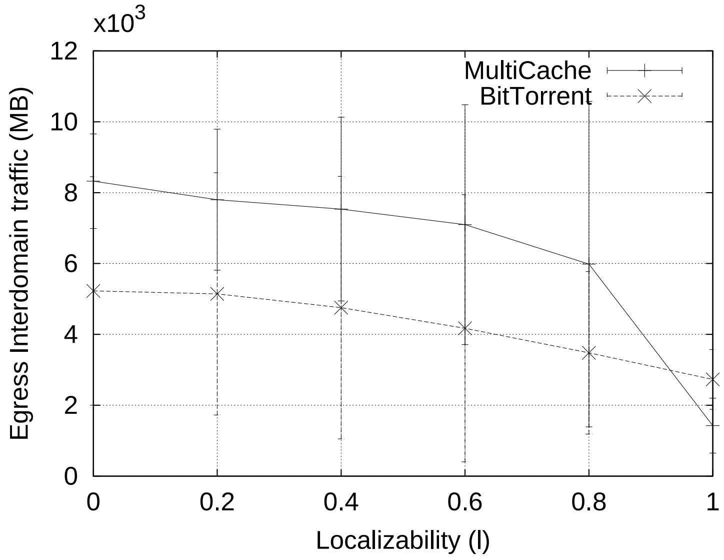


Figure 5.17: Effect of localizability on EIT $(d = 0.25, MFU)$

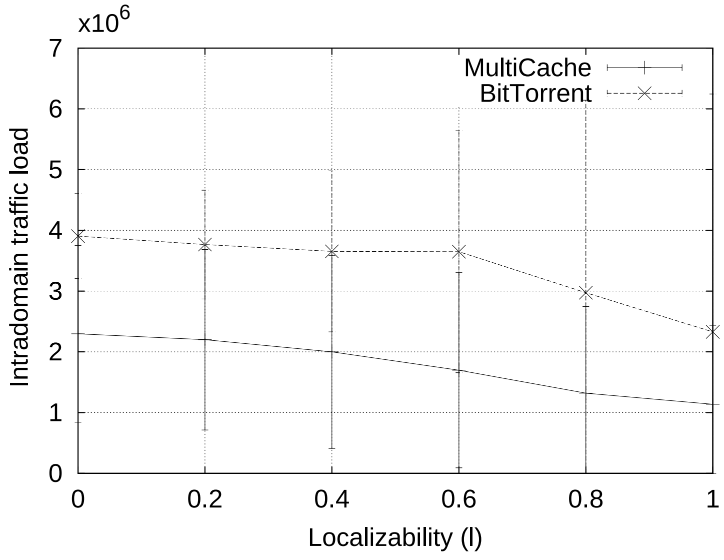


Figure 5.18: Effect of localizability on ITL $(d = 0.25, MFU)$

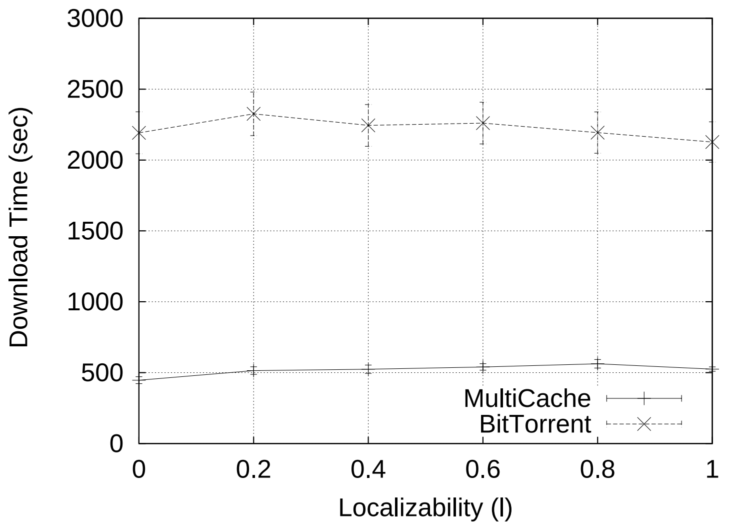


Figure 5.19: Effect of localizability on DT ($d = 0.25, MFU$)

5.4 Scalability analysis

Introducing information-awareness into the network necessitates the scalable representation and handling of information. In *MultiCache*, information is represented by flat identifiers which are handled in two layers. First, the underlying *Pastry* DHT is responsible for the key-based routing that enables the overlay multicast functionality. As demonstrated in [44], Pastry presents good scaling properties, as the required routing state increases logarithmically with the size of the overlay. In addition, OARs are expected to present lower churn rates in comparison with regular deployment at the edges, further reducing the amount of routing information exchange. Second, the *Scribe* multicast scheme necessitates the establishment of additional forwarding state, which is further augmented by *MultiCache* specific state i.e., the AS Number of a joining OAR and a bitfield for monitoring the delivery of content to lower layers (see Sections 5.1 and 5.2.1). In the following, we investigate the scalability properties of *MultiCache* in terms of the size of this state. This will allow us to assess the amount of resources required to support the proposed architecture. To this end, we study two important aspects that heavily affect the load imposed on OARs, namely, the structural characteristics of the Scribe overlay multicast trees, which dictate the distribution of the forwarding load, and the size of the imposed workload. Table 5.2 provides a summary of the notation employed hereafter.

U	Number of end hosts
N	Number of access routers
N_o	Number of overlay access routers
D	Number of domains
R_i	Number of access routers in domain i
d	Deployment density
b	Pastry configuration parameter
C_i^j	Number of end host requests for item j at tree level i
J_i	Average number of forwarding links at tree level i
K_i	Average number of forwarding state entries per node at tree level i
M	Number of distributed files
P	MultiCache piece size (MB)
Q	Average file size (MB)

Table 5.2: Analysis notation

5.4.1 Overlay multicast tree properties

The impact of deployment density on the properties of the resulting multicast trees is expressed through the size of the overlay, i.e., the total number of deployed OARs, N_o. To simplify our analysis we assume that all AS's adopt *MultiCache*, and focus on the intradomain deployment density (d) which is assumed to be the same across all domains. Hence, we have $N = \sum_{i=0}^{D-1} R_i$ and $N_o = dN$. We assume that all end-hosts are uniformly dispersed in the network and all requests submitted to the same OAR for the same data are aggregated and ultimately served via a single subscription.

Following [107] and [108], we express the shortest overlay distance between two nodes in Pastry with a binomial distribution with $p = \frac{1}{2^b}$:

$$L_i = \binom{\log_{2^b} N_o}{i} p^i (1-p)^{\log_{2^b} N_o - i} \tag{5.1}$$

If C is the total number of end hosts that request a certain item, we calculate the average number of leaves C_i that are i hops away from the root of the respective tree as follows. Equation (5.2) actually expresses the aforementioned aggregation of similar requests at proxy OARs.

$$C_i = min[CL_i, N_o L_i] \tag{5.2}$$

Then, we calculate the average number of forwarding links J_i, at each level i of a multicast tree, by considering the number of leaf nodes at each level of the tree, as well as the number of forwarding links that transfer data for leaf nodes residing at lower levels. Two forwarding links from level i to $i+1$ of a multicast tree, stem from the same node at level i with probability:

$$S_i = 1 - (1 - \frac{1}{N_o \cdot L_i})^{J_i} \tag{5.3}$$

Equation 5.3 expresses the portion of forwarding links merging at nodes of level i. At each level i of a tree, $S_i J_i$ links merge on some node(s) of that level on average. On the one extreme, these links may all merge at a single node at level i (hereafter called the *MaxMerge* case), thus incurring a single forwarding link at level $i-1$. On the other extreme, they may merge in pairs incurring $\frac{S_i J_i}{2}$ forwarding links at level $i-1$ (hereafter called the *MinMerge* case). Figure 5.20, illustrates both cases. We calculate J_i beginning from the lowest level of the tree where all forwarding links lead to leaf nodes. Then, traversing the levels of the tree towards the root, we take the two boundary merging cases and calculate the number of forwarding links that are required so that the data reaches the lower, already visited levels of the tree. At each level i, part of these forwarding links passes through non-leaf, forwarding nodes at level $i+1$, and the remainder passes through leaf, forwarding nodes of that level i.e., nodes that both consume and forward the received data. Additional forwarding links are also considered for all leaf nodes of the next level that do not act as forwarders. Hence, J_i is calculated as follows:

$$J_i = F_i(1 - \min[\frac{C}{N_o}, 1]) + C_{i+1} \tag{5.4}$$

$$F_i = \begin{cases} (1 - S_{i+1})J_{i+1} + 1 & , \textit{MaxMerge} \text{ case} \\ (1 - S_{i+1})J_{i+1} + \frac{S_{i+1}J_{i+1}}{2} & , \textit{MinMerge} \text{ case} \end{cases} \tag{5.5}$$

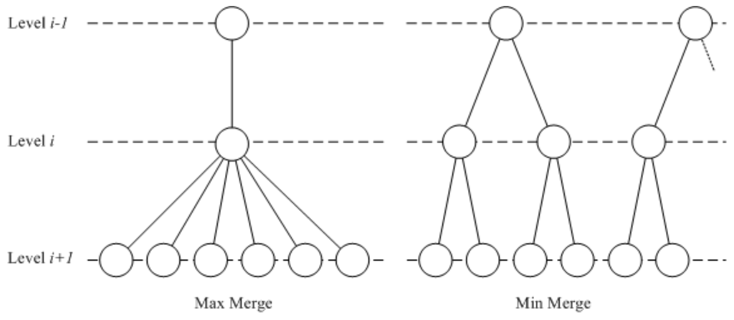


Figure 5.20: Boundary merging cases

Next, we can calculate K_i, the average number of children entries maintained at an OAR residing at level i of a multicast tree. The nodes residing at each level i of the tree equally share the forwarding links to the next level of the tree.

$$K_i = \begin{cases} \frac{J_i}{J_{i-1}} & , i > 0 \\ J_i & , i = 0 \end{cases} \tag{5.6}$$

5.4.2 Workload impact

Based on the properties of the created overlay multicast trees, we determine the workload imposed by the end users and then calculate the amount of forwarding state per OAR required to support it. To this end, we focus our analysis on a simple content distribution scenario in which a *MultiCache* enabled network serves the delivery of M files to the end hosts of the network. We simplify our analysis with the following assumptions:

1. All delivered files have equal size Q.
2. Each file is fragmented into pieces of size P and $Q\%P = 0$ i.e., for each distributed file exactly $\frac{Q}{P}$ multicast trees are formed.
3. We study the properties of the resulting trees once all recipient OARs for each file have joined the respective multicast trees.

The size of the resulting workload depends on the popularity of the considered files that ultimately defines the size of the resulting delivery trees. Again, we model the popularity of files $j \in [1, M]$, with a *Mandelbrot-Zipf* distribution $P(j)$ [73]. We normalize this distribution so that the most popular item is requested by all end hosts (U). Hence, we calculate the total number of requests per file j:

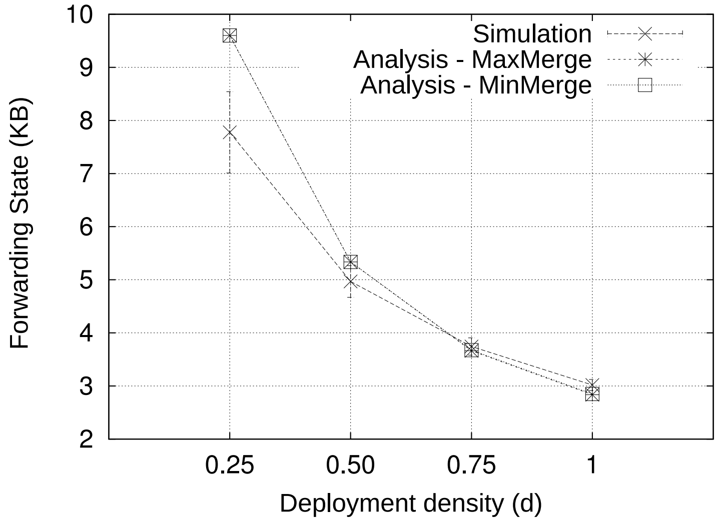
$$C^j = \frac{P(j)}{P(0)} \cdot U \tag{5.7}$$

We must stress at this point that the respective number of actual leaf nodes is determined by the deployment density as shown in Equation 5.2. Moreover, due to content fragmentation, C^j translates to an equal number of requests for each of the $\frac{Q}{P}$ resulting pieces.

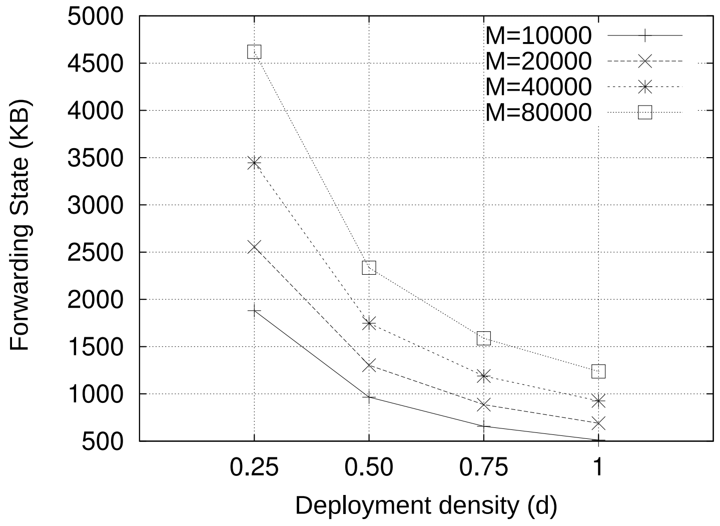
Based on the above, we can calculate the forwarding state load for each created tree by first deriving the total number of requests and then employing Equations 5.1 to 5.6. Note that, on average all OARs may act as forwarders for each tree with equal probability. Additionally, a forwarding OAR is equally probable to reside at any level of the respective tree. Hence, we evenly distribute the resulting load to all N_o OARs in the network.

Figure 5.21(a) presents the forwarding state required per OAR for a sample scenario with $N = 2400$, $U = 1500$, $M = 500$, $Q = 16$MB, $P = 16$MB, derived via both analysis and simulation. The agreement between the two methods is notable, especially in denser deployments. For sparser deployments, the analysis overestimates the forwarding state required.

Due to scalability restrictions of the simulator, we turn to the analytical model in order to investigate the scaling properties in the case of much larger topologies and workloads. Figure 5.21(b) presents the aggregate forwarding state at each OAR, for various workload sizes and deployment densities. In the scenario depicted, we have considered a network of 15000 access routers ($N = 15000$) with 10000 end hosts uniformly dispersed. The various workload sizes refer to files of size $Q = 656$MB, i.e. close to the median file size (651MB) observed in [105], and $P = 16$MB. We consider a 256-bit memory footprint for each forwarding entry, consisting of the 128-bit Pastry ID of the target OAR along with its 32-bit IP address, the 32-bit AS number and a 64-bit *bitfield* for monitoring the forwarded content. The results demonstrate the average of the *MaxMerge* and *MinMerge* cases. We can see that even in the highest workload case, the memory footprint of the forwarding state does not exceed 5MB, demonstrating the scalability of MultiCache.



(a) Analysis Vs. Simulation



(b) Analysis: high workload

Figure 5.21: Forwarding state per OAR

Chapter 6

Adapting to the inter-network structure

A crucial aspect of overlay routing relates to the degree to which overlay paths adapt to the structure of the underlying physical network topology. This chapter illustrates the shortcomings of the current *flat* overlay design and then proceeds with the high level description of the hierarchical DHT *Canon* paradigm [43]. It then presents the proposed application of this paradigm to the selected *Pastry* DHT, which aims at better adapting to the inter-domain network topology and ultimately providing network operators with finer control on interdomain service provision in the context of MultiCache. Finally, an extensive set of simulation-based measurements are presented, comparing the hierarchical design of *Pastry* with *Chord* and *Pastry* DHTs, as well as *Crescendo*, the hierarchical version of *Chord*.

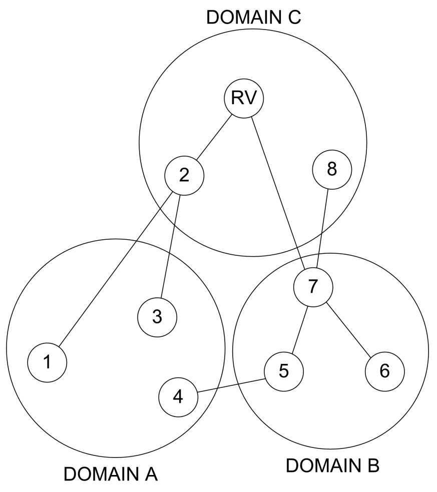
6.1 The need for further adaptation

As discussed in Section 3.3.1, one of the important features of the *Pastry* routing scheme is its ability to adapt to the underling network topology, by employing a proximity metric during the creation and maintenance of the overlay routing tables. However, this mechanism leads to routing decisions that only express preference with respect to a certain measurable criterion, such as the hop count or the Round Trip Time (RTT) between nodes. Though yielding low stretch routes [44], this feature of *Pastry* is possible to allow the formation of overlay paths that unnecessarily cross the administrative boundaries of a network. This is illustrated in the example of Figure 6.1, in which overlay nodes residing in three administrative domains form a *Scribe* multicast tree. The current design of *Pastry* is possible to result in the establishment of redundant forwarding links. Namely, Domain A receives the distributed content via three distinct inter-domain links and Domain C receives the content via an interdomain forwarding link with Domain B, while the data originates from Domain C. At the same time, Domain C forwards the same content to Domain A via two distinct overlay links. This is because the employed proximity metric does not aim at reflecting the underlying inter-domain level structure of the network and does not capture the routing preferences of network operators. For instance, the RTT between nodes 4 and 5 in our example may be lower than that between nodes 4 and 3, leading to the establishment of a forwarding path between domains B and A. This is an undesirable situation as the establishment of inter-domain delivery paths is usually translated into costs for network operators that pay their transit providers to forward their traffic to the destination network [109].[1]

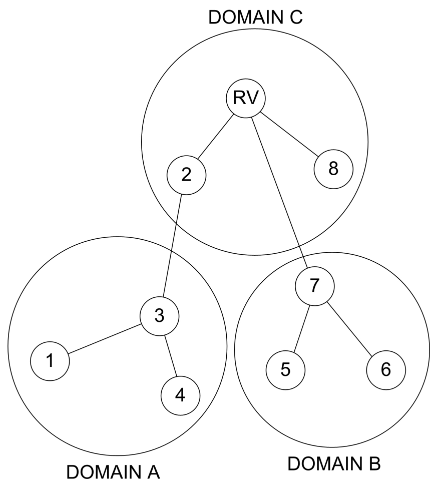
6.2 The *Canon* paradigm

The importance of the DHT adaptation to the underlying network structure was identified by Ganesan *et al.* in [43], where they proposed the *Canon* DHT paradigm. *Canon* intervenes in the construction process of a DHT in order to enable the progressive merging of individual DHT constructions, assuming a hierarchical structure of the inter-domain topology. In the lower level of this hierarchical process, each domain preserves its own DHT structure. At the higher level, the DHT structures of sibling domains are merged so that a single DHT structure is created. This process continues up to the highest level, in which all individual DHT's have been merged into a single construction. The purpose of

[1]A notable exception is the case of peering agreements in which Autonomous Systems (AS's) establish bilateral agreements in order to exchange traffic, usually at no cost.



(a) *Scribe* tree over *flat Pastry*



(b) *Scribe* tree over *Canonical Pastry*

Figure 6.1: The effect of *Canonical* DHT design on *Scribe* multicast trees

this process is to satisfy the following requirements:

1. *Inter-Domain path convergence*: all messages originating from the same domain A and targeting the same node located at a different domain, should exit domain A through the same node (see Figure 6.2).

2. *Locality of intra-domain paths*: all overlay routes connecting nodes of the same domain should never exit the domain boundaries (see Figure 6.3).

3. *Preserve logarithmic state size*: the adaptation to the underlying hierarchy should not inflate routing state.

4. *Preserve logarithmic routing hops*: the number of routing hops connecting any two overlay nodes should remain logarithmic to the size of the overlay network.

In order to satisfy these requirements, the merging process is based on the incremental refinement of the overlay routing state while traversing the domain hierarchy towards the entire network structure i.e., nodes of a certain administrative domain create and maintain routing entries to nodes in another sibling domain *iff* there are no better entries in their local DHT routing tables.

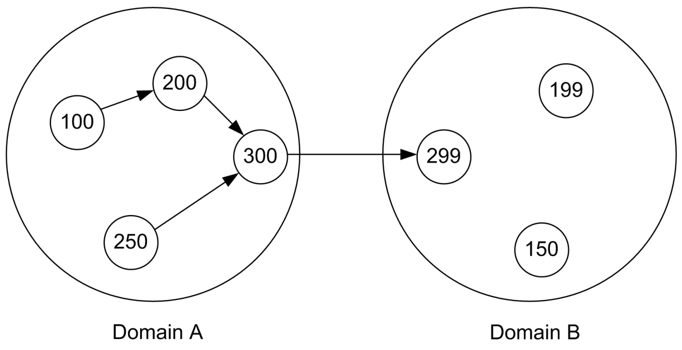


Figure 6.2: Path convergence: Nodes 100, 200 and 250 route towards 299, all via node 300.

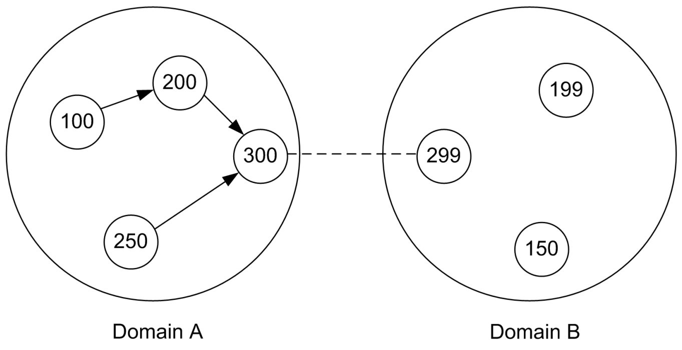


Figure 6.3: Locality of intra-domain paths: Nodes 100, 200 and 250 route towards 300 without passing through 299.

6.3 Hierarchical *Pastry*

The proposed DHT merge process was illustrated in [43] based on the *Chord* DHT [52], yielding *Crescendo*, the hierarchical version of *Chord*. The *Canonical* version of several other DHT's were also discussed, except for the case of *Pastry*, due to the complexity introduced by the two-level structure of the routing table i.e., the routing table and the leaf table. In this work, considering the benefits stemming from the short routes and route convergence properties of *Pastry* (see Section 3.3.1), and the lack of equivalent characteristics in the *Crescendo* scheme, we engaged in the design of the *Canonical* version of *Pastry* i.e., *Hierarchical Pastry* (*H-Pastry*).

In the following, we consider two levels of hierarchy: local and global. The former concerns the DHT operation in each individual domain and the latter the DHT operation across all participating domains. This two-level scheme can be recursively employed to allow for the adaptation on hierarchies with more than two levels. Distinct routing tables are maintained for each level, allowing for compliance with Requirement 2. Each node maintains in its global routing tables information about nodes that are numerically closer to certain points in the identifier space (i.e., its own identifier and the identifiers's required to fill the Routing table (see Section 3.3.1)) than any other node in its domain. Consequently, global traffic is expected to always leave the domain from the node whose identifier is numerically closer to the destination (Requirement 1). Figure 6.4 shows the identifier space partitioning in *H-Pastry*; the bullet in the center represents the node's identifier, l^- and l^+ are the numerically smallest and biggest local node identifiers respectively, g^- and g^+ are the numerically smallest and biggest global node identifiers respectively and $l^-/2$ and $l^+/2$ denote the middle of the distance between l^-, g^- and l^+,g^+ in the identifier space respectively. Every node maintains routing information only for the identifier space between $l^-/2$ and $l^+/2$. The shaded area denotes the actual identifier space covered by each node, as every key that is on the left of $l^-/2$ is numerically closer to l^- and every key that is on the right of $l^+/2$ is numerically closer to $l^+/2$.

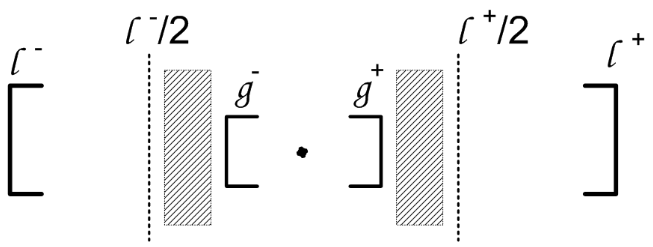


Figure 6.4: ID space partitioning

6.3.1 Joining procedure

Joining *H-Pastry* is a two step procedure; local DHT join and global DHT join. The entire procedure is illustrated in Algorithm 1. Table 6.1 provides a summary of the notation employed in the algorithm. Every joining node (JN) initially issues a Join message to its local *Pastry* DHT via a local bootstrap node (LBN) as in regular *Pastry* (line 2). The Join message arrives at the numerically closest local node completing the joining process for the local domain (lines 5-15). This local node becomes the bootstrapping node for the second step, the global DHT joining procedure (lines 20-32). The global join procedure will be based on the global routing state already established. If global state is not available yet, a join message will be sent to a randomly selected, nearby neighboring domain (line 17). During the global join procedure the JN receives global routing state (line 21), which it further filters so that only entries for the $[l^{-}/2, l^{+}/2]$ are kept (line 22). This global routing state is being communicated to the local leaf set nodes as it may contain entries that should be present in their global routing tables. These nodes may also exchange their global routing information with it (line 31). The global join procedure completes with the exchange of global state between the JN and the discovered global nodes, as dictated by the regular *Pastry* join procedure (line 32).

CuN	Current Node
JN	Joining Node
LBN	Local Bootstrap Node
GBN	Global Bootstrap Node
LRT	Local Routing Table
GRT	Global Routing Table

Table 6.1: *H-Pastry* Join algorithm notation

6.3.2 Routing

Based on the collected routing information during the Join procedure, routing in *H-Pastry* comprises again of two distinct steps. Initially, an *H-Pastry* node

Algorithm 1 *H-Pastry* Join algorithm

```
1: CuN ← JN
2: Request LBN
3: if LBN ≠ JN then                                        ▷ Local Pastry exists
4:     LocalJoinComplete ← $false$
5:     while LocalJoinComplete ≠ $true$ do
6:         CuN.ExchangeLocalStateWith(JN)
7:         NextHop ← FindLocalNextHop()
8:         if NextHop=CuN then                             ▷ Local Join Completed
9:             LocalJoinComplete ← $true$
10:        else
11:            ForwardJOIN(NextHop)
12:            CuN ← NextHop
13:        end if
14:    end while
15:    JN.ExchangeLocalStateWith(nodes in LRT)                  ▷ End of local join
16: else                                                 ▷ First Node in Local Pastry
17:    Request GBN
18: end if                                                    ▷ Begin Global Join
19: GlobalJoinComplete ← $false$
20: while GlobalJoinComplete ≠ $true$ do
21:    CuN.ExchangeGlobalStateWith(JN)
22:    JN.FilterGlobalState($\frac{l^-}{2}$,$\frac{l^+}{2}$)
23:    NextHop ← FindGlobalNextHop()
24:    if NextHop=CuN then                                 ▷ Global Join Completed
25:        GlobalJoinComplete ← $true$
26:    else
27:        ForwardJOIN(NextHop)
28:        CuN ← NextHop
29:    end if
30: end while
31: JN.ExchangeGlobalStateWith(nodes in LRT leaf set)
32: JN ExchangeGlobalStateWith(nodes in GRT)                   ▷ End of global join
```

routes a message towards the destination key based on the local routing information. Local routing terminates at the local node responsible for the selected key i.e., the node that the message would terminate at if there was no other domain in the internetwork. Next, if this node holds global routing information that further refines the search i.e., knows about the existence of an *H-Pastry* node with an identifier numerically closer to the target key, it will forward the message outside the domain. This two step routing procedure guarantees both design Requirements 1 and 2. In the former case, all messages sent from within the domain towards the same key will eventually reach the same node, before exiting the domain. In the latter case, by initially using only the local routing information, an intra-domain destination is guaranteed to be reached via a routing path comprising solely of intra-domain nodes.

6.3.3 *H-Pastry* properties

The described Join procedure satisfies the first two requirements set by the *Canon* paradigm (see Section 6.2). Based on the observation that each *H-Pastry* node only maintains the routing state that further refines its local *Pastry* entries, and considering the fact that local routing is unaffected by the *Canon* principles, while global routing follows the same *Pastry* principle, we expect that the remaining two requirements are also satisfied by the presented design. We analytical study this issue in the following. Table 6.2 provides a summary of the notation used hereafter. We assume that all N overlay nodes are uniformly dispersed across the D equally sized domains participating *H-Pastry*.

N	Number of overlay nodes
2^b	Pastry numbering base
k	Key length (digits)
D	Number of participating domains, ($D > 1$)
S_L	Size of local routing state
S_G	Size of global routing state
S	Total size of routing state

Table 6.2: *H-Pastry* analysis notation

Routing State

As described above, the two level hierarchy of *H-Pastry* results in the maintenance of overlay routing state in two sets of routing tables, namely the local and the global.Since the local overlay routing state remains unaffected by the appliance of the *Canon* paradigm, following [44], we have:

$$S_L = \left\lceil log_{2^b} \frac{N}{D} \right\rceil \times (2^b - 1) \tag{6.1}$$

Next, we turn our attention to the additional state required to support the overlay routing at the global level. As described above, a single node maintains a global routing table only for the identifier space between $l^-/2$ and $l^+/2$. Assuming a uniform distribution of identifiers in the identifier space, and a uniform distribution of the N overlay nodes across the D domains, the length of this space (A) can be calculated as follows:

$$A = \frac{D2^{bk}}{N} \tag{6.2}$$

Our target then is to calculate the average number of node identifiers actually residing inside this space i.e., calculate the number of nodes for which an *H-Pastry* node maintains routing state in its global routing tables. To this end, we first calculate the average identifier space between all overlay nodes (B):

$$B = \frac{2^{bk}}{N} \tag{6.3}$$

It follows that a continuous part of the identifier space of length A, will contain on average $\frac{A}{B} = D$ nodes. Therefore the $[l^-/2, l^+/2]$ interval of each node will contain $(D - 1)$ nodes residing at different domains. In effect, the routing state for $D - 1$ nodes will have the following size:

$$S_G = \lceil log_{2^b}(D - 1) \rceil \times (2^b - 1) \tag{6.4}$$

Therefore, the total size of the routing state maintained by an *H-Pastry* node is expressed by the following equation:

$$S = S_L + S_G = \left\lceil log_{2^b}(\frac{D-1}{D}N) \right\rceil \times (2^b - 1) \tag{6.5}$$

Hence, Requirement 3 is satisfied. We notice also that *H-Pastry* requires the maintenance of less routing state than in the case of plain *Pastry*, which in the considered scenarios is equal to $\lceil log_{2^b} N \rceil \times (2^b - 1)$.

Path Length

It follows from the calculated size of the routing state that the maximum number of overlay hops between any pair of *H-Pastry* nodes is equal to $(\left\lceil log_{2^b}(\frac{D-1}{D}N) \right\rceil)$ [44].

6.3.4 Evaluation

In order to further evaluate the properties of the *Canonical* version of *Pastry*, we implemented *H-Pastry* for the OverSim Simulation framework [98]. In the following, we present the results of this performance evaluation, comparing *H-Pastry* against regular *Pastry*, as well as *Chord* [52] and its *Canonical* version *Crescendo* [110]. Our evaluation is based on a simple scenario in which the overlay host population is uniformly dispersed across 112 stub domains of a GT-ITM *transit-stub* topology (See Section 8.1.1). Each overlay node generates 1000 messages, each destined to a randomly chosen ID.

In order to first gain an overview of *H-Pastry*'s routing properties with respect to the adaptation to the underlying network structure, we categorize each of the overlay hops comprising an overlay path into the following three categories.

- *Local intra-domain hops*: overlay hops taken by a message within the sender's domain boundaries i.e., the number of overlay hops taken before the message (possibly) leaves the domain.
- *Inter-domain hops.* Overlay hops taken across domain boundaries i.e., connecting nodes residing in different domains.

	Chord	**Pastry**	**Crescendo**	**H-Pastry**
Local Intra-domain hops	0.96	21.29	36.52	38.66
Inter-domain hops	98.21	77.05	57.85	56.01
Remote Intra-domain hops	0.82	1.64	5.62	5.32

Table 6.3: Distribution of overlay hops (%)

- *Remote intra-domain hops*: overlay hops between nodes residing in domains other than the domain of the sender.

Then, delving into the details of *H-Pastry*, we examine the effect of our design on the length of intra-domain overlay paths i.e., the overlay paths connecting nodes residing in the same domain. Our target is to illustrate the benefits of the *Intra-domain path locality* property. Then, we investigate the *Convergence of Inter-domain paths* property of *H-Pastry.* To this end, we measure the *hop overlap fraction* of two overlay paths originating from the same domain towards the same destination ID [43]. Finally, we examine the stretch of the created overlay paths.

Routing properties

Figure 6.5 presents the distribution of overlay routing hops across the aforementioned categories for all cases of *Chord*, *Pastry*, *Crescendo* and *H-Pastry*, in a scenario with 4096 overlay nodes[2]. Table 6.3 also presents the numerical values depicted in Figure 6.5.

Comparing with the case of regular *Pastry*, we notice a substantial reduction of the Inter-domain part of the resulting overlay paths in the case of *H-Pastry* (approximately 27%), while the Intra-domain part of the paths increases both in the case of the Local and Remote domains. This is considered as a direct concequence of both the *Locality of intra-domain paths* and *Convergence of inter-domain paths* properties of the applied *Canon* paradigm, which constrain overlay routing within domain boundaries, avoiding unecessary inter-domain hops when possible. *H-Pastry* performes only slightly better than *Crescendo* due to the exact

[2]Similar results were derived for other population sizes.

same method used in both cases to isolate traffic as possible. Moreover, we also notice that the proximity awareness of *Pastry* manages to substantially reduce the percentage of Inter-domain hops in comparison with *Chord* (approximately by 21%).

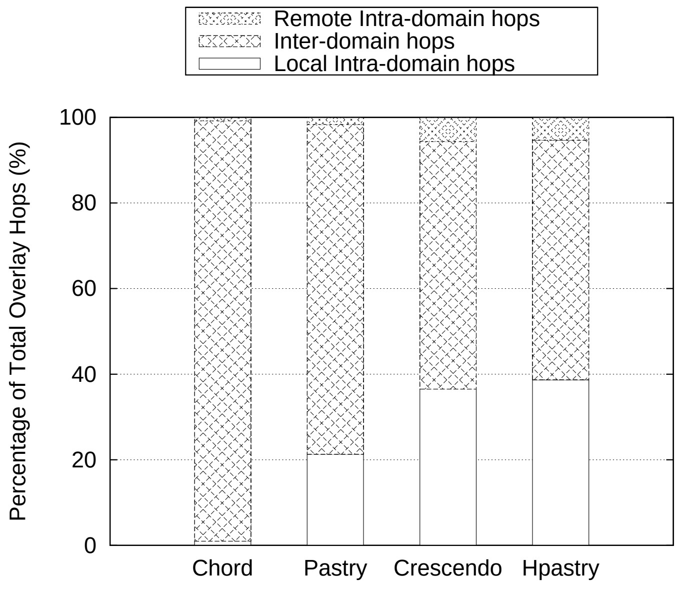


Figure 6.5: Distribution of overlay hops

Intra-domain routing

Figure 6.6 illustrates the average length (measured in number of IP hops) of paths connecting nodes residing in the same domain, in all considered cases, for various overlay sizes. *H-Pastry* outperforms regular *Pastry* in all cases, yielding on average 33% shorter intra-domain paths. This is atributed to the Locality of intra-domain paths property which results in paths between nodes of the same domain to be confined inside domain boundaries. In the case of regular *Pastry*, routing is based on global routing information i.e., routing tables may contain entries for nodes residing on the same or different domains, making no distinction between the two cases. Due to this reason, paths connecting nodes in the same domain, may leave the domain and pass through nodes at different domains. Moreover, the proximity awareness of *H-Pastry* manages to yield shorter paths in comparison with *Crescendo* that does not incorporate any proximity metric. The average path length reduction in this case reaches a percentage of 44%. Interestingly, we also see that regular *Pastry* also outperforms *Crescendo*, demonstrating the importance of the proximity awareness in building the routing tables. *Chord* performs dramatically worse that any other considered alternative due to the lack of any mechanism for the adaptation to the underlying network structure.

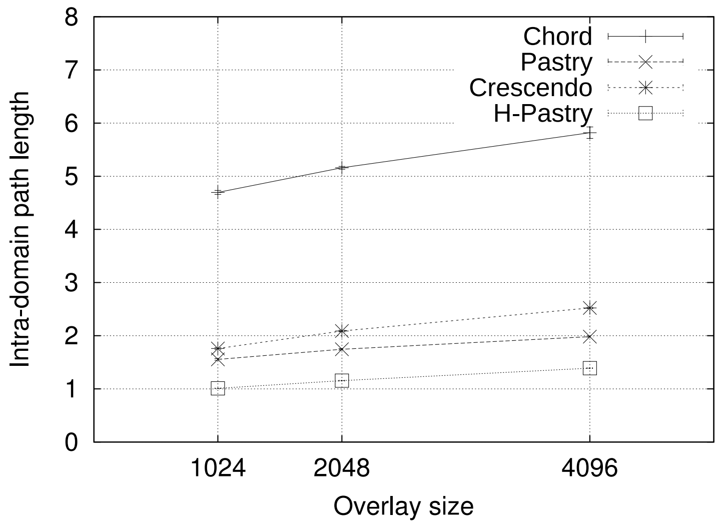


Figure 6.6: Intra-domain path locality

Convergence of inter-domain paths

Figure 6.3.4 presents the average hop overlap fraction in all considered cases, for various overlay sizes. The average hop overlap fraction is defined as follows. An overlay node v is first chosen randomly from the entire overlay node population, and it uses the deployed DHT scheme to route a message towards a destination node residing at a different administrative domain. The destination node is randomly selected among all candidate nodes. Let P denote the overlay path taken by v's message. Then, another node u is randomly selected among the overlay node population in node v's domain and it also sends a message towards the same destination node. Let *P'* denote the overlay path taken by *u*'s message. The hop overlap fraction is the fraction of *P'* that overlaps with *P*. We denote the first common node of both paths as *c*. The purpose of this metric is to capture the benefit stemming from the Convergence of inter-domain paths property. We notice that *H-Pastry* consistently outperforms *Pastry*. In the largest population size (4096 overlay nodes), the average hop overlap fraction is 72% for the case of *H-Pastry*, while it only reaches 50% in the case of *Pastry*. This increase in the hop overlap fraction verifies and quantifies the improvement of *Scribe* trees in the case of *H-Pastry*, as discussed in Section 6.1. As the overlaping fraction of paths is served by a single tree branch in *Scribe*, larger values imply better resource utilization, as branching, and packet duplication, take place closer to the receiving nodes (smaller non-overlaping path fractions). Furthermore, we see that both *Crescendo* and *H-Pastry* perform similarly, as they are based on the same mechanism to select a common gateway node that will act as the convergence point for both paths. It is noted, that this metric does not aim to capture the effect of the locality awareness of any of the considered schemes, as the convergence of the paths is primarily controled by the structure of the routing tables and not their exact entries.

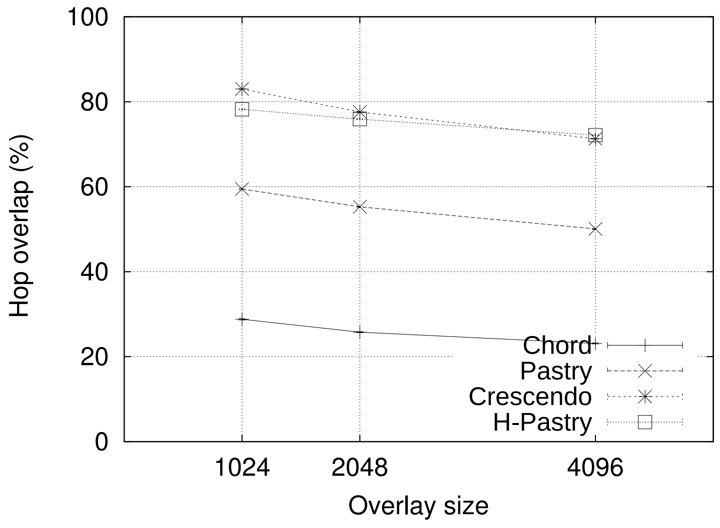


Figure 6.7: Convergence of inter-domain paths

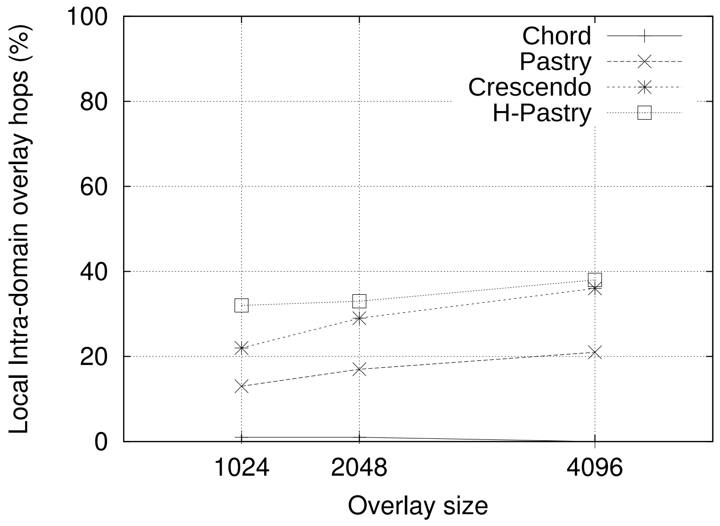


Figure 6.8: Local Intra-domain overlay hops

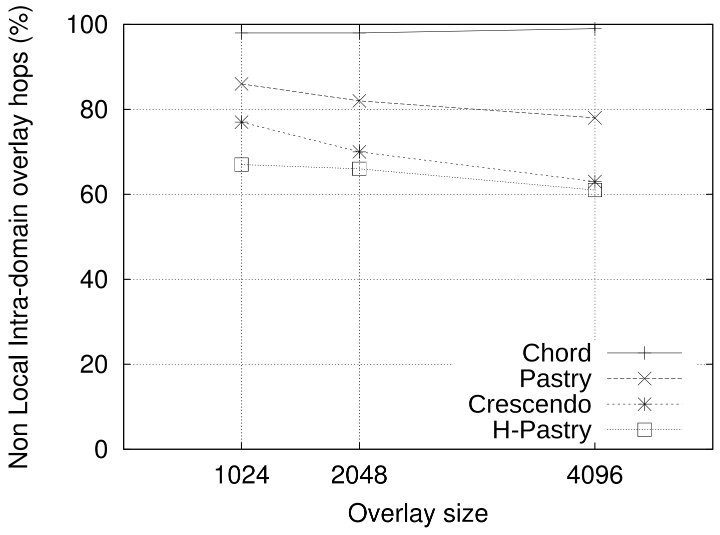


Figure 6.9: Non Local Intra-domain overlay hops

Interestingly, we notice that the hop overlap fraction decreases with the size of the overlay network, denoting that the convergence of interdomain paths takes place closer to the destination node. In the case of the examined *Canonical* DHTs, paths converge at a node c, which, by design, resides at the origin domain.

Therefore, the decreasing hop overlap fraction denotes the fact that the Local Intra-domain proportion of the overall paths increases with the size of the overlay population. Figure 6.8 presents this proportion, verifying our assumption. This is attributed to the fact that larger overlay sizes result in longer overlay paths, due to the increased number of overlay nodes, that allow finer grained routing decisions i.e., the routing tables of the overlay nodes are populated with more entries.[3] In Section 4.2 it was shown that denser deployments result in slightly shorter delivery paths. However, these results refer to the total number of IP hops comprising an overlay path. In essence, larger overlay networks result in overlay paths comprised of more overlay hops, but yielding a reduced total number of IP hops. Figure 6.9 depicts the aggregated proportion of the Inter-domain and Remote Intra-domain hops, denoted as Non Local Intra-domain hops (Figure 6.9). We notice that the proportion of Non Local Intra-domain hops decreases with the number of overlay nodes. Based on the above, this leads us to the assessment that larger overlay network sizes result in finer grained routing that yields longer routes in which node c resides closer to the destination node.

Stretch

Figure 6.10 shows the *stretch* of the delivery paths in our setup. We define stretch as the ratio between the time required for a message to reach its destination following an overlay path, to the corresponding time required by an identical message to reach the same target following the shortest path. We see that both *Pastry* and *H-Pastry* present similar behavior, outperforming both *Chord* and *Crescendo*. *H-Pastry*'s stretch is on average 72% and 60% lower than *Chord* and *Crescendo* respectively, reaching an average value of 1.71. This substantial decrease is due to the employment of the proximity metrics during the creation and maintenance of routing information in *Pastry* and *H-Pastry*.

[3]Equation 5.1 in Section 5.4.1 verifies this intuition since the expected path length equals to $E(i) = np$, which increases monotonically with the size of the overlay network.

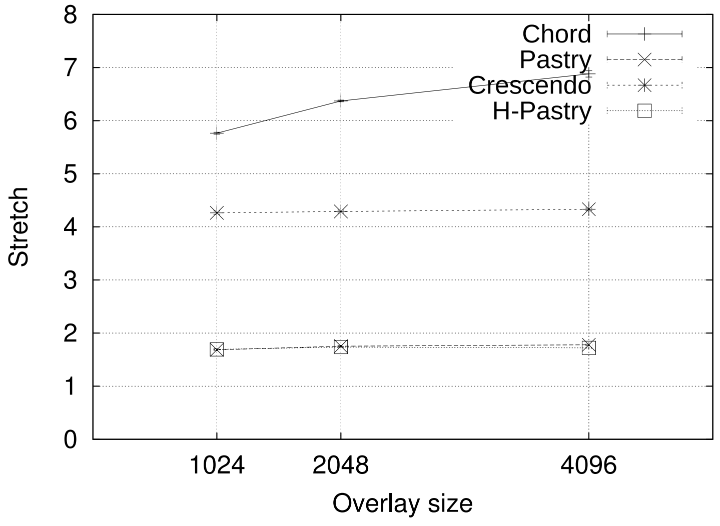


Figure 6.10: Stretch

Chapter 7

Mobility support

This chapter presents the mobility support features of the proposed architecture. These features rely on the information-centric model of communication and the publish/subscribe nature of the deployed routing and forwarding protocols. The proposed Overlay Multicast Assisted Mobility (OMAM) scheme, aims to take advantage of the established multicast functionality and the decoupling of end-hosts, as achieved by *Scribe*, as well as the locality properties of the *Pastry* routing substrate. This chapter describes how these high-level characteristics of the architecture are translated to specific signaling in the network, highlighting the advantages of the selected approach. In doing so, this chapter also provides an analytic comparison of the proposed signaling with Mobile IPv6. Finally, it presents simulation results demonstrating the effectiveness of the proposed approach.

7.1 Overlay multicast assisted mobility

As described in Section 3.2, end nodes do not participate into the overlay themselves, but rather access the respective services via a proxy OAR designated during network attachment. In the case of mobile devices, the access link acting as the first/last *mile* towards the proxy OAR is wireless, typically being served by the currently associated *Access Point* (AP). Figure 7.1 shows a generic network topology in which a mobile node (MN), initially residing at its home network, moves around the network from OAR to OAR while communicating with a correspondent node (CN) via an RV.

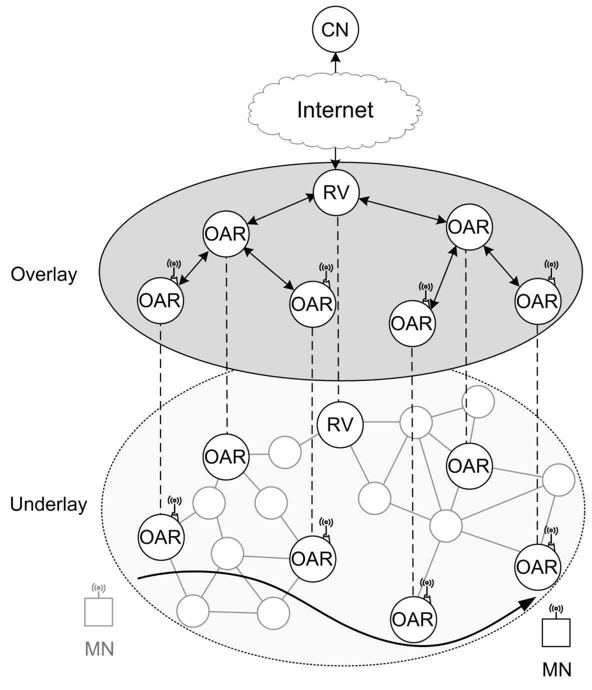


Figure 7.1: Example topology.

As described in Chapter 3, whenever a (mobile) node wishes to act as a sender, it advertises its data to the network and upon solicitation, it simply delivers it to its proxy OAR which is then responsible to forward it to the proper RV point. Whenever a (mobile) node wishes to act as a receiver, it sends a subscription request message to its OAR. When an OAR receives a subscription request for the first time, it issues a *Scribe* JOIN message in order to join the multicast group of the publication. What actually changes with mobility is that as MNs move from one AP to another, it is possible that they will change their OAR as well[1], in which case they must transfer their subscriptions from their old OAR (OAR_{old}) to the new one (OAR_{new}). This is accomplished by each MN re-issuing its subscription request(s) to OAR_{new}. The latter then simply joins the respective multicast tree(s), unless it is already a member[2].

[1]Depending on the intra-domain deployment density (see Section 4.2).

[2]This may happen because another end node that resides on OAR_{new} has already subscribed to the same publication or because, due to the overlay nature of *Scribe*, OAR_{new} has joined the

At OAR$_{old}$, a MN's subscription(s) removal process depends on the type of application, the underlying wireless access technology and the OAR deployment density. In general, when caching is used by the application, OAR$_{old}$ triggers the Leave procedure following the cache eviction mechanism presented in Section 5.2.2. Otherwise, the Leave procedure is driven by the mobility of the users. When the concurrent association of mobile terminals with two neighboring APs is feasible (e.g., CDMA), a *make-before-break* handoff procedure is employed. A MN first establishes connectivity with OAR$_{new}$ and re-issues its subscriptions, actively initiating the Leave procedure at OAR$_{old}$ when data starts arriving from its new AP. This feature is obviously helpful in reducing packet loss during handoffs. On the other hand, duplicate packets will inevitably reach the MN in this case. However, these can be dealt with via the use of sequence numbers.

When a *make-before-break* handoff is not feasible, the Leave procedure is initiated *after* a period of time from the moment OAR$_{old}$ anticipates the need to leave the respective multicast tree(s). This anticipation is subject to the deployment density of the overlay network of OARs. In cases of dense deployments, in which the OARs are collocated with the APs, it can be based on the state of the association of each currently attached MN to the wireless AP. In these cases, an OAR schedules the LEAVE REQUEST message for a specific group to be sent some time after the last MN that is a member of that group has disassociated from the AP. The reason why an OAR does not immediately leave the group is that whenever an OAR leaves a group, the respective multicast tree is (at least partially) destroyed. Depending on the network traffic conditions, the topology of the network and the speed of the moving MN, it is possible for a *Scribe* LEAVE message issued by OAR$_{old}$ to reach the lowest *common ancestor* of both OAR$_{old}$ and OAR$_{new}$ in the multicast tree (OAR$_{CA}$) before the corresponding *Scribe* JOIN message issued by OAR$_{new}$. In this case, it is not possible for a moving MN to take advantage of the multicast nature of *Scribe* and the Route Convergence property of *Pastry*, unless if another MN in the same area has subscribed to the same tree. If the deployment density does not allow for OARs to access the association state

tree in order to serve another OAR as a forwarder. In that case OAR$_{new}$ only needs to forward the data towards the MN.

of MNs, then a soft state mechanism can ensure the initiation of the leave procedure. In these types of handoff, packet loss is inevitable as the packets delivered to the OAR$_{old}$ during the handoff procedure are lost and the new subscription(s) at the OAR$_{new}$ result only in subsequent packets being delivered to the MN(s). The caching capabilities of OARs and the fragmentation of the content into blocks (see Section 5.2.5) can be utilized to overcome this limitation. OARs can temporarily cache the last n blocks they have forwarded to their children in the tree. The exact value of n can be calculated based on an estimation of the handoff latency in the supported network. This is obviously subject to the topology of the wireless access network, the speed of MN's and the selected block size (in our work we have considered 16KB blocks). Based on this mechanism, the OAR$_{CA}$ can provide the packets lost during handoff to the OAR$_{new}$.

When a MN moves from one OAR to another, traffic is diverged to the new point of attachment at the lowest common ancestor of the two visited OARs in the created multicast tree (see Figure 7.2). Hence, re-routing of data does not need to take place at the CN as in the case of Mobile IPv6 [12], SIP[17] and HIP[15] protocols. At the same time, due to the Route Convergence property of *Pastry*, the OAR$_{CA}$ is expected to be close enough to the new point of attachment so that the routing update can be performed without large delays. The intuition here is that when a MN moves from one AP to another, these APs are expected to be close enough in the proximity space, so the routing update message of the MN will meet the OAR$_{CA}$ after traveling a short distance in the proximity space resulting in a low delay for the propagation of the routing update.

This indirect mode of communication in OMAM presents some significant characteristics. First, it allows for the *location privacy* of MNs as their IP addresses, which may lead to a rough inference of their physical location, are not exposed to their CNs. Moreover, no modifications are required to the protocol stack of either the CNs or the MNs. A stationary CN simply sends/receives the data to/from its proxy OAR, while the only action taken by a MN due to mobility is the re-declaration of its interest to its active sessions[3], to each new proxy OAR.

[3]We use the term *session* to also refer to streaming data.

At the same time, OMAM supports the simultaneous mobility of communicating end-hosts. In this case a multicast tree can be built in each direction.

7.2 Evaluation

7.2.1 Signaling

We investigate the performance of the proposed architecture and compare it analytically against Mobile IPv6. We chose Mobile IPv6 as it is in itself a standardized solution, and it can also serve as a means for an indirect comparison of our approach with other mobility assisting schemes. We first compare our scheme against plain Mobile IPv6 and then proceed with a comparison against its micro-mobility enhancement, Hierarchical Mobile IPv6. Our purpose is to demonstrate the ability of the proposed scheme to better localize routing updates, while acting as a unified solution in both cases of macro- and micro-mobility .

We chose to use the handoff delay as the primary performance metric since it reflects the service disruption experienced by the user. Specifically, we consider the amount of time required for a MN to be able to resume communication after a change of network location, that is, once the MN has associated with the new wireless AP. We will refer to this metric as the *Handoff Latency* (HL). Obviously, this time is heavily affected by the signaling required for the involved network entities to be informed about the MN's change of position i.e., the time required for the routing substrate to adapt to the movement of the MN.

OMAM Vs. Mobile IPv6

Figure 7.2 presents an example scenario in which a MN performs several consecutive handoffs while communicating with a corresponding node, in the case of OMAM. Our target here is to investigate the impact of the signaling procedure required in the cases of Mobile IPv6 and the proposed architecture so that the MN can again become reachable after changing its point of attachment to the network.

We denote by $d_{x \to y}$ the delay of a message sent from network entity x to y. For simplicity we assume that $d_{x \to y} = d_{y \to x}$. Using the same topology for both

OMAM and Mobile IPv6 (see Figures 7.2 and 7.3), we have:

$$d_{OAR_{k-1} \to OAR_k} = d_{AR_{k-1} \to AR_k} \tag{7.1}$$

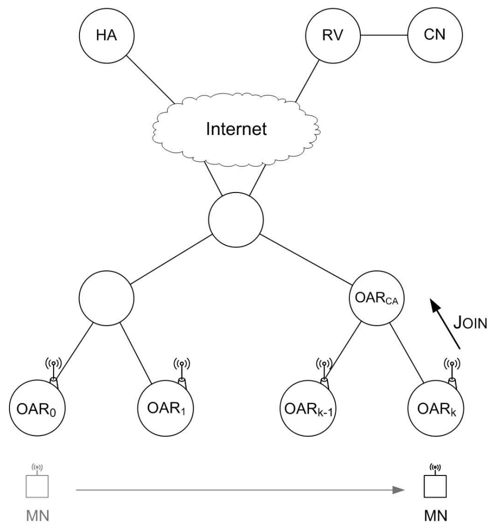


Figure 7.2: Overlay Handoff

Mobile IPv6 without RO. First, we consider the case of the simple Binding Update procedure, without employing the Route Optimization (RO) procedure (see Section 2.5.1). Figure 7.3 shows the corresponding scenario under Mobile IPv6. A Binding Update (BU) message is sent towards the HA of the MN and a Binding Acknowledgment (BA) message is returned to the MN.[4] Hence:

$$HL_{MIPv6} = 2 \cdot (d_{MN \to AR_k} + d_{AR_k \to HA}) \tag{7.2}$$

In the case of OMAM, a subscribe message is sent over the wireless medium by the MN towards the newly visited OAR, which in turn generates a *Scribe* JOIN message towards the CA. Considering also the acknowledgment of this packet[5], we have:

[4]We omit acknowledgment messages from the figures for clarity reasons.

[5]It is noted that the routing update takes place before the arrival of the acknowledgement packet to the MN. However, we take these messages into account for reasons of completeness and in order to provide a fair comparison of the considered protocols.

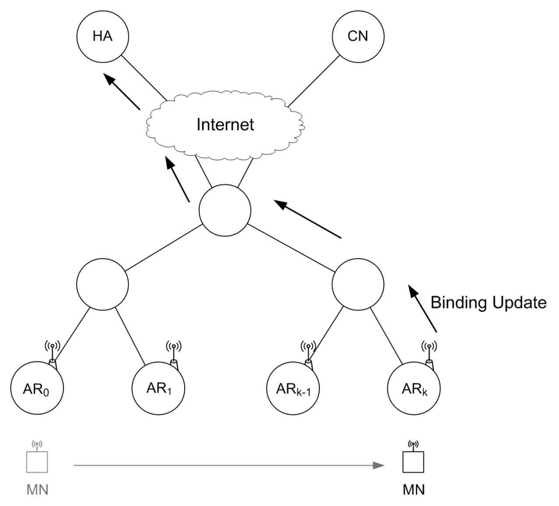


Figure 7.3: Mobile IPv6 Handoff (without RO)

$$HL_{OMAM} = 2 \cdot (d_{MN \to OAR_k} + d_{OAR_k \to CA}) \tag{7.3}$$

The following equation expresses *Pastry*'s route convergence property.

$$d_{OAR_k \to CA} = a \times d_{OAR_{k-1} \to OAR_k}, a \to 1 \tag{7.4}$$

Considering Equation 7.1 and 7.4, and assuming the delay of wireless transmissions to be the same in both scenarios, our scheme results in a smaller HL value when:

$$HL_{OMAM} < HL_{MIPv6} \Leftrightarrow a < \frac{d_{AR_k \to HA}}{d_{AR_{k-1} \to AR_k}}$$

Since according to the route convergence property $\alpha \to 1$, it is clear that our architecture results in a reduced HL compared to MIPv6, as in most cases the distance between neighboring OARs is expected to be smaller than the distance between the current OAR and the home network of the MN. Moreover, when OAR$_k$ is already a member of the publication's multicast tree, the HL is expected to be further reduced i.e., $HL_{OMAM} = 2 \cdot d_{MN \to OAR_k}$, in which case OAR$_k$ simply acknowledges the request of the MN. This may happen because another MN attached to OAR$_k$ has already expressed interest for the same publication, or because

OAR$_k$ is acting as a forwarding node for another OAR.

Mobile IPv6 with RO. Next, we consider the case of the Route Optimization (RO) procedure. In RO the Binding Update also targets the CNs the MN is actively communicating with, apart from the HA, in order to avoid triangular routing. RO is based on the Return Routability procedure, which is used to enable a CN to assure that the MN is triggering the Binding Update. Table 7.1 summarizes the signaling messages of the entire RO procedure. Initially, the MN issues a BU message towards its HA in order to enable the Return Routability procedure, as explained below. This message is acknowledged by the HA. Then, the MN issues two messages towards the CN, at the same time. The first message (Home Test Init) passes through the HA of the MN, while the second (Care-of Test Init) is send directly to the CN. As a response, the CN issues the Home Test and Care-of Test messages that follow the reverse paths of the HoTi and CoTi messages respectively. The purpose of this exchange is to allow the CN to verify that no other node but the MN has required the data to be sent towards another network address. The Return Routability procedure is completed with the reception of the HoT and CoT messages by the MN, which can then issue a BU message.

#	**Message Type**	**Source**	**Destination**
1	Binding Update (BU)	MN	HA
2	Binding Ack. (BA)	HA	MN
3	Home Test Init (HoTi)	MN	HA $\rightarrow$ CN
4	Care-of Test Init (CoTi)	MN	CN
5	Home Test (HoT)	CN	HA $\rightarrow$ MN
6	Care-of Test (CoT)	CN	MN
7	Binding Update (BU)	CN	MN
8	Binding Ack. (BA)	MN	CN

Table 7.1: Return Routability procedure.

Steps 3 and 4 and steps 5 and 6 of the Return Routability procedure are considered to require very little processing and thus to take place in parallel [12]. Assuming that the HoTi and HotT messages take longer to reach there destinations compared to the CoTi and CoT messages respectively, as they pass through the HA, we can express the HL of Mobile IPv6 with RO as follows:

$$HL_{MIPv6} = 4d_{MN \to HA} + 2d_{MN \to CN} + 2d_{HA \to CN} \tag{7.5}$$

Based on the above, our scheme results in a smaller HL value when:

$$a < \frac{2d_{MN \to OAR_k} + d_{OAR_k \to CN} + d_{HA \to CN}}{d_{OAR_{k-1} \to OAR_k}} \tag{7.6}$$

Again, since $a \to 1$ and based on the expected proximity between neighboring OARs, we expect the HL in OMAM to be lower than in the case of Mobile IPv6 with RO. It must be noted though, that the OMAM signalling does not aim at authenticating the MN triggering the handoff, as in the case of the RO procedure. However, when consecutive handoffs are performed by a MN, the HoTi, CoTi, HoT and CoT messages are cached in the CN and the MN in order to reduce the HL. In these cases, Steps 1 through 6 are omitted, and by following the same reasoning, OMAM yields a faster handoff when:

$$a < \frac{d_{AR_k \to CN}}{d_{AR_{k-1} \to AR_k}} \tag{7.7}$$

OMAM Vs. Hierarchical Mobile IPv6

Hierarchical Mobile IPv6 reduces the handoff delay by considering the deployment of a Mobility Anchor Point (MAP) at the gateway router of each administrative domain. The role of the MAP is then to handle all Binding Updates locally, hiding any intra-domain handoff events from the HA and the CNs. Figure 7.4 shows the corresponding Binding Update procedure for the considered example. In this case we have:

$$HL_{HMIPv6} = 2 \cdot (d_{MN \to AR_k} + d_{AR_k \to MAP}) \tag{7.8}$$

Similarly with the case of regular Mobile IPv6, OMAM results in a smaller HL value when:

$$HL_{OMAM} < HL_{HMIPv6} \Leftrightarrow a < \tfrac{d_{AR_k \to MAP}}{d_{AR_{k-1} \to AR_k}}$$

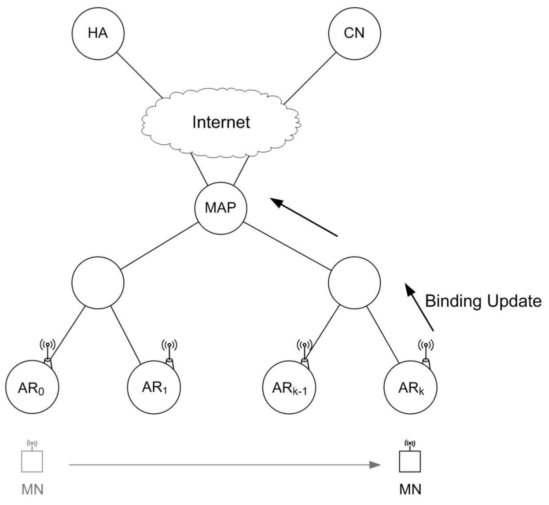


Figure 7.4: Hierarchical Mobile IPv6 Handoff

Again, the distance between the MAP and the current point of attachment is expected to be greater than the distance between two consecutive OARs, yielding a higher delay in the case of Hierarchical Mobile IPv6. This is because, as mentioned earlier, the MAP is located at the gateway router of the current administrative domain.

7.2.2 Simulation results

To evaluate the performance of our scheme under dynamic conditions, we used the OMNeT++ simulation framework [45] enhanced with xMIPv6 [111] and OverSim [98]. In our simulations we considered a simple network topology comprising multiple OARs deployed in a grid-like topology. All OARs run the full TCP/IP protocol stack, as well as *Pastry* and *Scribe* in the case of the proposed architecture. All wired connections are implemented with Ethernet links. One IEEE802.11b AP is directly connected to each OAR, with neighboring APs operating in disjoint channels. We were restricted by the xMIPv6 model which does not provide an implementation of the Hierarchical Mobile IPv6 enhancement, it mandates the use of the Route Optimization procedure and also necessitates the deployment of only a single home network with a single MN in any topology.

Both the HA and the CN are attached to randomly chosen, disjoint OARs of the topology. The AP in the Home Network of a MN is directly connected to the HA rather than to the respective OAR. In the case of the proposed architecture and in order to produce comparable scenarios, we placed an extra OAR between the AP and the randomly chosen OAR acting as the initial point of attachment of the MN to the network. Figure 7.5 shows an example topology. The circles around each AP denote its transmission range. The square bounded area denotes the part of the topology actually accessible by the MN. We have chosen a topology providing full wireless coverage in order to restrict the anticipated service disruption to each protocols' operation.

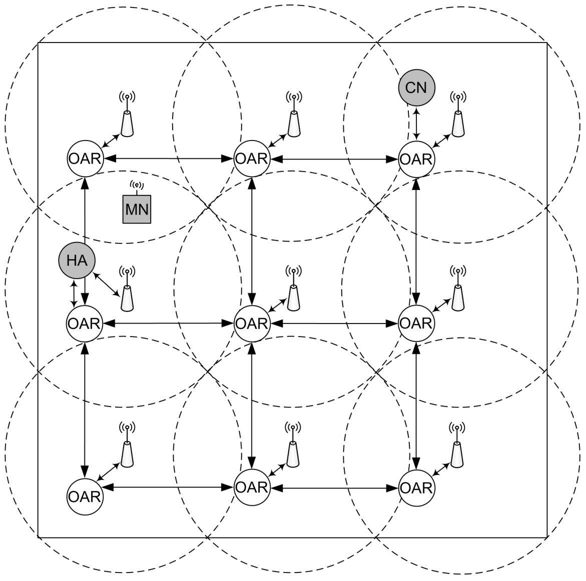


Figure 7.5: Grid-like example topology.

Our scenarios consider a single MN initially connected to its home network. Upon initialization, the MN starts to move following the mobility model described in [112]. In this model a MN moves in a straight line, makes a turn and starts over. Its speed and direction are updated every x seconds, with x following a normal distribution with a mean of 10 seconds and a standard deviation of 0.1 seconds. The speed of the MN is normally distributed with a mean of 1.39 meters/sec (approximate walking speed of 5 Km/h) and a standard deviation of 0.01 meters/second. The change in direction also follows a normal distribution with an average of 0 degrees (no turn) and a standard deviation of 5 degrees. Whenever the MN reaches the limits of the simulated area it bounces with the same angle and speed.

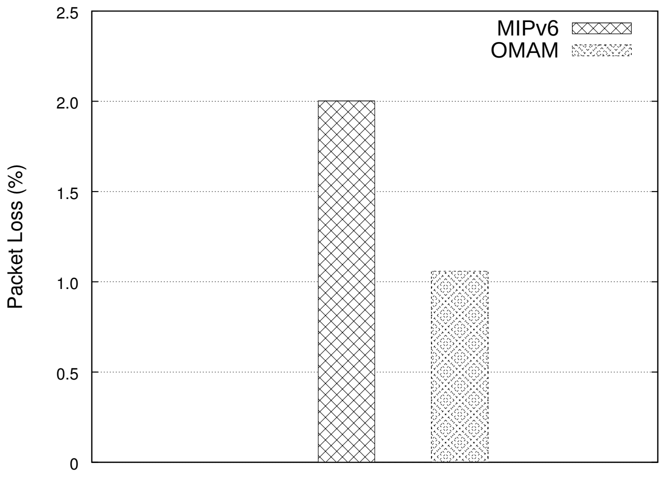
Our measurements were based on a simple application scenario in which a stationary node sends a sequence of UDP datagrams (CBR traffic) towards the MN. The UDP stream resembles a H.264, Level 1 SQCIF video stream with 30.9 frames per second [113]. In the case of MIPv6, the CN is initially only aware of the MN's home address and uses it as the destination of its data. While moving, the MN is responsible for updating its bindings with its HA and the CN in the case of Route Optimization. In the case of the proposed architecture, the CN simply sends its UDP packets towards a rendez-vous point whose position in the networks is determined by a randomly generated key and *Pastry*'s functionality. The complete set of parameter values used in our simulation environment are provided in Table 7.2.

Parameter	**Value**
Grid size	30 x 30
Number of MNs	1
Number of CNs	1
Wired connections type	100 Mps Ethernet
Propagation delay (ms)	0.5
Data rate (Kbps)	64
Packet size (bytes)	26
Total number of packets sent	556200

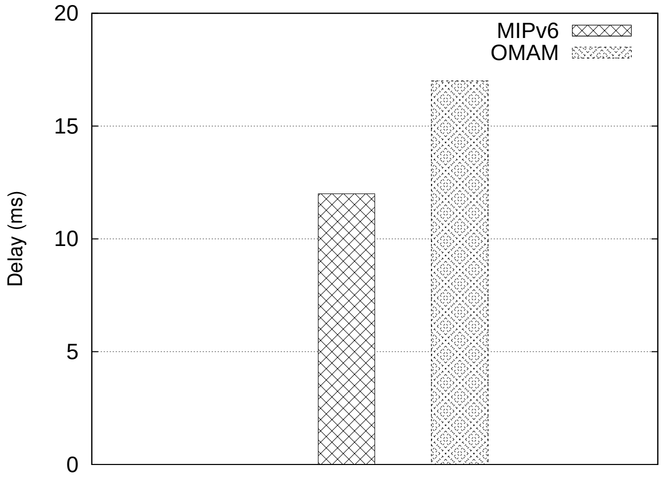
Table 7.2: Simulation scenario parameters.

Figure 7.2.2 presents the results derived from the simulation scenarios described above. These results constitute the first step towards a thorough evaluation of the proposed architecture with respect to mobility.

As expected, our scheme significantly lowers packet loss in comparison with Mobile IPv6. This is justified by the localized handling of mobility. Upon each change of network position Mobile IPv6 launches the Return Routability procedure in order to apply Route Optimization. As shown in the previous section, this causes an exchange of signaling packets both with the HA and the CN. Considering the fact that these nodes may be located in distant parts of the network, this signaling may considerably delay the establishment of new routes for the traffic destined to the MN. On the other hand, in our architecture this signaling overhead is reduced since only the CA node in the multicast tree is notified about the MN's change of position. The significance of the reduced Resume time is depicted in the Packet loss metric where we see a noticeable difference between the two considered approaches. Furthermore, it must be noted that by localizing routing updates our architecture enables the actual delivery of packets in transit in the scenarios where the CN resides in a distant area of the network. However, these improvements come at the cost of end-to-end delay. Indeed, as we see in Figure 7.6(b), our approach results in a higher end-to-end delay. As explained earlier, this is due to stretch imposed on the routing due to the reliance on a DHT substrate. We must note however, that this increase could be acceptable for non-interactive streaming applications as the ones considered in this work.



(a) Packet Loss



(b) Delay

Figure 7.6: Overlay Multicast Assisted Mobility Vs. Mobile IPv6

Chapter 8

Simulation environment

In our work we used the OMNeT++ Simulation Engine [45]. We chose this platform due to its simplicity, its high degree of modularity, and the availability of several protocol implementations ranging from a complete TCP/IP protocol stack (provided by the INET framework), to a large set of overlay protocols, encapsulated in OverSim [98] framework. However, the need for proper evaluation of the considered systems and protocols together with the Internet-wide scale of their deployment, neccecitate both the thorough investigation of the simulation engine's capabilities with respect to its scalability, and the enrichment of the provided framework[1] with features and applications that would fit the needs of our research.

8.1 Contributions

8.1.1 *GT-ITM* topologies

Our initial concern in building a simulation environment relates to the underlying network topology. OverSim's *SimpleUnderlay* model provides a scalable routing substrate since no network protocols are actually in operation. In this model, packets are directly sent to end-hosts by simply using a global routing ta-

[1]Most of our simulations were carried out on OMNeT++ v.3.3, INET v.20061020 and OverSim v.20080416

ble, with packet delivery delay being determined by the two communicating ends' distance in the Euclidean space. Furthermore, each end-host can be assigned to a logical access network for which the access delay, bandwidth and packet loss characteristics may be set. Even though it has been shown that this model provides a scalable solution for the simulation of large numbers of overlay nodes [98], it suffers from serious limitations: the lack of protocol functionality and step-by-step routing are major drawbacks of the model, since important aspects of a real system are neglected, such as the queuing of packets in intermediate nodes (routers) and therefore the packet delays, and even losses, that arise due to network congestion. As a result, this model cannot be used to evaluate the dynamic performance properties of realistic applications.

On the other hand, OverSim's *IPv4Underlay* model provides a good approximation of real networking conditions by incorporating the operation of almost all widely deployed networking protocols. We have therefore focused on this model for our work, exploring its memory and processing time requirements for the simulation of very large network topologies. Apart from scalability, the other major drawback of this model is that it lacks support for routing policy weights, such as those produced by the Georgia Tech Internet Topology Model (GT-ITM) [47] that has been used for previous studies of Scribe performance [42, 99]. Instead, the model employs an unweighted shortest path algorithm (Dijkstra's) which calculates the shortest paths between any pair of network nodes, regardless of their placement on the network. Hence, in contrast with reality, it is possible for a path between two routers in a single stub network to pass through several transit routers, something very likely to influence the results produced, especially when it comes to routing issues. By not taking routing policy weights into account, routing paths may become shorter than in reality, and since Pastry employs proximity metrics in the selection of overlay neighbors, this could result in the selection of the wrong node as an overlay neighbor.

In order to avoid routing inaccuracies, we constructed a conversion tool that allows the use of GT-ITM topologies within the OverSim platform. Our tool was implemented as an extension to the BRITE topology generator export tool [45] which already allows the parsing of GT-ITM topologies and the conversion of BRITE topologies into OMNeT++ format. However, the existing tool did not provide any support for weighted topologies, nor did it make a distinction between transit and stub routers, producing flat topologies. We solved these problems by piggybacking the routing weights inside the channel definition of the produced OMNeT++ topology, using fields whose values are not provided by the GT-ITM model, but are instead later read from configuration files. Furthermore, we incorporated the distinction between *transit* and *stub* routers in the tool and translated it into *IPv4Underlay*'s distinction between *backbone* and *access* routers. The support for routing policy weights was completed by employing a weighted shortest path algorithm, leading to a platform that captures all the intricacies of GT-ITM and is therefore comparable to earlier simulation studies based on it.

Despite these important enhancements, the resulting GT-ITM based IPv4-Underlay model retains two more significant limitations. The first is revealed when considering the locality properties (with respect to the consumption of ISP-specific resources) of data exchanges in P2P content distribution applications, such as *BitTorrent*. As discussed in Chapter 1, *BitTorrent*'s (as well as other P2P applications') network-agnostic peer selection protocol has an adverse impact on capacity related ISP costs by allowing downloads from peers residing in external domains, even when the desired data are already present locally. As presented in Chapter 2, this has triggered several research efforts that would benefit from a simulator providing the flexibility to study ISP level aspects of the protocol's performance. Hence, while OverSim's IPv4Underlay model provides no access to such information, we further enhanced our topology conversion tool to also preserve the unique Autonomous System numbers [101] produced by BRITE and to export them to a separate configuration file so that each router, as well as each attached end-host, can be assigned the corresponding AS number. Direct access to this information is provided to the simulation programmer, facilitating the investigation of the aforementioned locality properties and protocol inefficiencies, as well as the implementation of location-aware schemes [66].

Second, OverSim does not make any distinction between the uplink and downlink characteristics of access links (i.e., bandwidth). However, this distinction is important for providing realistic networking environments, since typical current access technologies, such as ADSL, do present this asymmetry. This issue becomes more important due to the fact that bandwidth heterogeneity results in a systematic unfairness of the peer-wire protocol, as the download rates achieved are based on the tit-for-tat mechanism. Hence, we further enhanced OverSim's IPv4Underlay model to support a range of *channel*[2] characteristics for the two directions of each access link. Specifically, the simulation programmer is able to specify different channel options for the uplink and downlink of each access link type, along with the fraction of the total access links across the entire network that each channel type is assigned to.

8.1.2 Simulation scalability

As shown in [98], OverSim enables the simulation of scenarios with even 100.000 overlay nodes. However, in the simulations presented in that paper, the underlying network was either too simplistic (*SimpleUnderlay*) or too small (*IPv4Underlay* with 20 backbone and 20 access routers). If the same number of overlay end-hosts is used with both models, then the memory requirements of the *IPv4Underlay* model will be larger due to additional network elements (routers and links) and their operation. It is also expected that the memory footprint of routers will differ from that of overlay end-hosts.

We investigated this issue in series of simulations focused on the required memory and computational resources.The performance results reported below were obtained on a system running Ubuntu Linux 7.10 on an Intel Core 2 Duo P9500 2.4 GHz processor with 4GB RAM. The simulator memory requirements for the considerably larger network sizes described in Table 4.1 are presented in Figure 8.1(a): the x-axis indicates the size of the network in terms of the total number of participating routers, while the curves indicate the memory footprint of scenarios either without any end-hosts or with 1000 end-hosts in the proposed router as-

[2]We adopt the OMNeT++ definition of a channel, which includes the data rate, error rate, and propagation delay characteristics of a link.

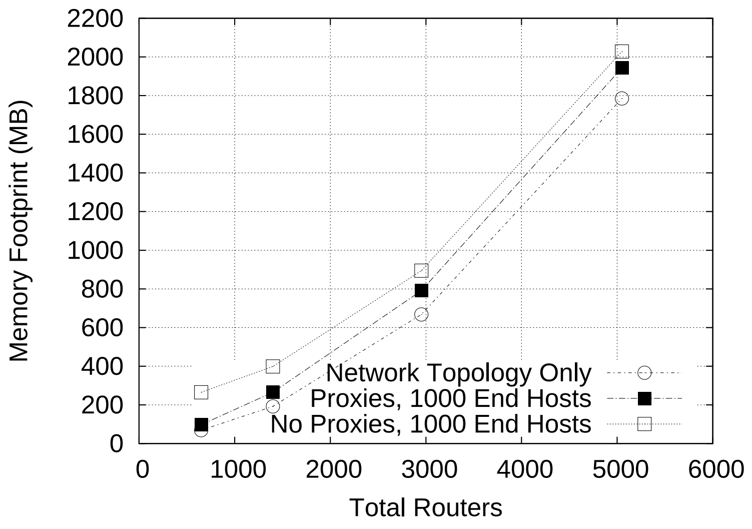
sisted (*Proxies*) or in the regular Scribe scheme (*No Proxies*). It is clear that the memory footprint of the networking topology increases dramatically with the number of participating routers, in a non-linear fashion. This increase is due to the increasing number of links between the participating routers. The memory requirements of the network topologies have a severe impact in the feasibility of large scale scenarios, since, for example, a topology with 5000 access routers and 50 backbone routers, already requires approximately 1800 MB of memory; for a realistic simulation, we also need to add end-hosts, along with their access links.

Our proposed RAOM scheme requires less memory than the regular Scribe scheme: as all overlay functionality is provided by the access routers, we did not create the actual end-hosts at all, so as to reduce the memory footprint of the simulation. Furthermore, our scheme requires fewer messages for the establishment of the overlay, since each router joins the overlay only once, regardless of the number of end-hosts that it is a proxy for. Note also that these messages travel smaller distances since they do not need to cross the access links; only a single message is required for an end-host to initially ask its access router to be its proxy.

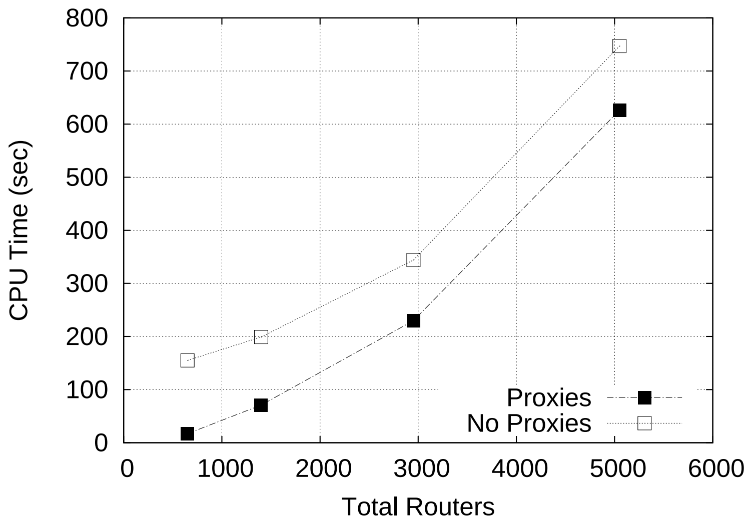
Figure 8.1(b) shows the processing (CPU) time for the simulation of each scenario, which includes constructing a network topology with 1000 end-hosts, running Pastry to establish a DHT substrate, running Scribe to establish multicast trees and sending a message to the RV point of each multicast tree. Again, the resource requirements of the *Proxies* scenario are lower than those of the *No Proxies* scenario. This can be attributed to the lower number of running simulation modules as well as to the avoidance of multiple overlay maintenance messages when many hosts are attached to the same router.

8.1.3 *BitTorrent* simulation modules

While *BitTorrent* has drawn strong interest from researchers, most studies have concentrated on the performance evaluation of the protocol and its potential variations via the study of real world trace data sets [46, 8]. This approach has significant advantages with respect to the reliability of the extracted results, but it is characterized by inflexibility: there is no control over the participating



(a) Memory requirements



(b) Processing time

Figure 8.1: Overlay multicast simulation scalability

peer characteristics and major protocol variations cannot be studied without first implementing and then deploying them. Moreover, the trace collection process is cumbersome and the data gathered may be incomplete. For instance, collecting information for peers behind firewalls is difficult, while gathering information about the swarm's size and structure might be hindered by the tracker protocol itself [114]. Analytical studies are even more problematic due to the highly dynamic character of *BitTorrent*: peers dynamically enter and leave the swarm, establish and tear down connections, decide on the preferred pieces of a file and chose to exchange data with peers or not. Simulation appears to be a more promising alternative, as it allows fast prototyping, provides the ability to perform large scale experiments, and offers a common reference platform for experimentation. Nonetheless, current *BitTorrent* simulators either consider coarse-grained representations of the underlying network, thus reducing the realism of the simulation, or omit many important features of the *BitTorrent* protocols.

Therefore, aiming for the advantages of a realistic simulation environment, we engaged in the development of a full featured and extensible implementation of the *BitTorrent* protocol for the OMNeT++ simulation environment [103, 115]. In order to increase the degree of realism in our simulation environment, we also developed a *churn generator* model that activates *BitTorrent* nodes in a network topology by following an arrival process derived from the analysis of actual *BitTorrent* traces [46]. In this study, based on the analysis of *BitTorrent* user traces, it was observed that the peer arrival rate for a torrent follows an exponential decreasing rule with time t:

$$\lambda(t) = \lambda_0 e^{-\frac{t}{\tau}},$$

where λ_0 is the initial arrival rate when the torrent starts and τ denotes the file popularity. Based on this distribution it can be shown that $N_{all} = \lambda_0 \tau$, where N_{all} is the total population size. Hence, by retrieving values for N_{all} and λ_0 from the configuration files, our model can generate random arrival times for each *BitTorrent* peer.

BitTorrent **simulation scalability**

Having established all the desired protocol features under a realistic simulation environment, including the underlying network models, we investigated the resource demands of the created simulation setup. The performance results reported below were obtained on a system running Ubuntu Linux 8.04 on an Intel Dual Core E5200 2.5 GHz processor with 4GB RAM.

Parameter	**Value**
Transit domains	7
Avg. routers per transit domain	4
Stub domains per transit router	7
Avg. routers per stub domain	7
Stub routers	1372
Transit routers	28
Total routers	1400

Table 8.1: Network topology parameters.

Based on our findings on the memory footprint of GT-ITM based topologies of several sizes (see Section 8.1.2), we created an IPv4Underlay topology that strikes a balance between the memory consumption and complexity of the routing substrate, consisting of 1372 stub and 28 transit routers. The detailed characteristics of the employed topology are presented in Table 8.1.3. We used the default link establishment probabilities, as described in Section 4.1.1. Each peer is attached via an access link to a randomly selected stub router upon arrival. In addition, we used a SimpleUnderlay topology consisting only of peers attached to access links, with the propagation delays between the peers being drawn from CAIDA's Skitter project [116].

In both cases, peers enter the swarm according to the distribution described in the previous section with an initial arrival rate of 0.0166 peers/sec. We varied the access link bandwidths across the network in order to achieve the heterogeneity present in real environments. The uplink / downlink bandwidth values of the various access link types as well as their distribution are presented in Table 5.1. The tracker and the initial seeder (content provider) have 10 Gbps symmetric access links however. We chose to set such high values for the access link of the tracker in order to avoid creating a bottleneck, so as to focus on the performance of the peer-wire base protocol. The same access link attributes were used for the initial content provider in order to simulate a dedicated high bandwidth seeding node.

In the selected simulation scenario the size of the file to be distributed via *BitTorrent* is 200 MB. We vary the swarm size between 30, 60, 90 or 120 peers.

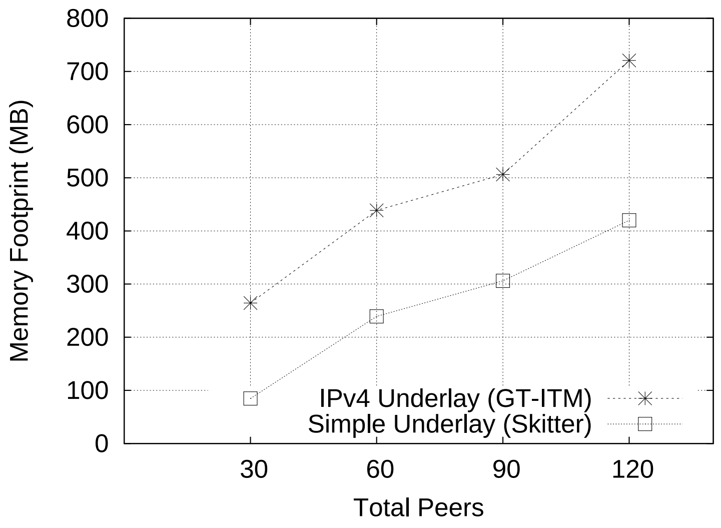


Figure 8.2: *BitTorrent* simulation memory requirements.

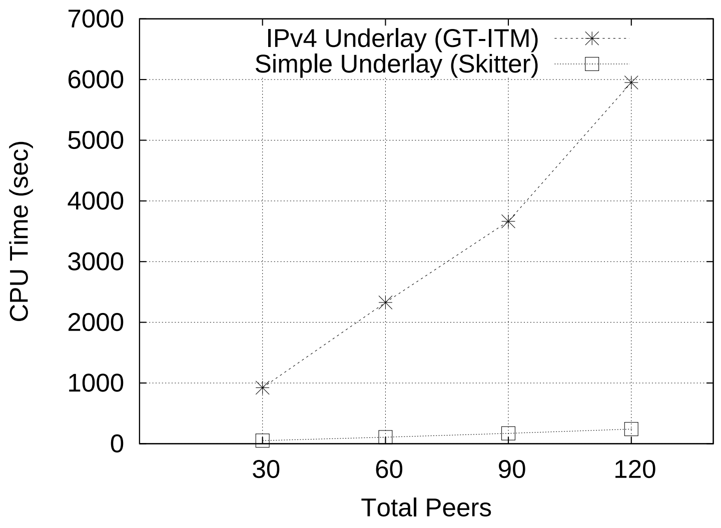


Figure 8.3: *BitTorrent* simulation processing time.

The simulator memory requirements are shown in Figure 8.2. The x-axis presents the total number of peers comprising the swarm, while the curve indicates the maximum size of the memory footprint recorded during the execution of the corresponding scenario. The difference between the IPv4Underlay and SimpleUnderlay models clearly shows that a significant part of the former's memory footprint is due to the underlying routing substrate. This is an indication of the well known tradeoff between simulation scalability and realism. However, considering the fact that the IPv4Underlay results refer to a full fledged simulation environment, the total memory footprint of 720 MB for the simulation of a swarm of 120 peers can be considered as an acceptable cost when realism is important.

The tradeoffs between the two models are more evident in Figure 8.3 which shows the processing (CPU) time for the same scenarios. In the IPv4Underlay case the processing time reaches 98 minutes, due to the operation of the full protocol stack for 120 end-hosts and 1400 routers. On the other hand, in the SimpleUnderlay case, the processing time for the same number of end-hosts is only 4 minutes, indicating that our simulator can easily be used for studies that focus on the *application logic* of *BitTorrent*. However, in this case the impact of the underlying network operation on protocol performance is completely neglected (see Section 8.1.1).

Chapter 9

Discussion and future work

This chapter first aims at positioning our work in the related research fields. To this end, it provides a thorough comparison of the proposed architecture with related approaches, highlighting the similarities with other approaches as well as the distinctive characteristics of our approach. It then takes a step further in discussing the identified open issues and research challenges.

9.1 Comparison with related work

9.1.1 Information-centric architectures

In Section 2.1 we presented a set of information-centric networking architectures proposed in literature. Though some of them are still being shaped (i.e., PSIRP [39] and CCN [51]), we try to underline both the design aspects that differentiate and relate our work with them.

Internet Indirection Infrastructure

The *Internet Indirection Infrastructure* (*i*3) architecture shares several important features with *MultiCache*. Both approaches rely on the establishment of a key-based routing infrastructure in order to enable a rendez-vous primitive that achieves the indirect character of communication. Also mobility support is based on this form of indirection in both architectures. However, both the multicast and anycast services present considerable differences. In the former case, *i*3 relies on the *a priori* selection of sets of identifiers that enable the distribution of the forwarding load to multiple *i*3 nodes residing at intermediate levels of the multicast tree. In effect, an out of band coordination mechanism is required for the generation of these identifiers, limiting the scalability of the proposed scheme. In contrast, in the employed *Scribe* multicast scheme, these intermediate forwarding nodes are automatically selected during the overlay routing of a join request, taking advantage of the underlying *Pastry* routing scheme. *MultiCache* also takes advantage of the locality properties of *Pastry*. In the case of anycast, the desired proximity-based server selection criterion is inherently built in the routing process of *MultiCache*, while *i*3 requires the incorporation of this metric in the selected anycast group identifier. Moreover, in our work we have further investigated the potential benefits of the proposed architecture in the context of content distribution, paying particular attention to the achieved network traffic savings and the end-user experience.

The proposed OMAM scheme (see Chapter 7) shares some significant characteristics with the mobility support scheme in *i*3 (see Section 2.1). Both schemes enable the mobility-driven routing state updates to take place inside the network, transparently to the Correspondent Nodes (CNs), based on a DHT-enabled rendezvouz mechanism. Moreover, in both schemes, senders may engage in a probing process for the selection of a suitable RV point. However, on the receiver side, OMAM does not necessitate this probing mechanism as it relies on the already available proximity related measurements of the underlying *Pastry* scheme, revealed through the selection of the appropriate nodes in the created multicast trees. Apart from reducing the amount of probing and the resulting overhead both on mobile nodes and the network, the multicast assistance of OMAM allows for the further localization of routing updates at the OAR_{CA} (see Section 7.1) rather than the RV point as in the case of *i*3. However, as shown in Section 7.2.2, OMAM suffers from the stretched overlay paths followed by data. In contrast, *i*3 allows the use of IP paths, shortening the distance traveled by data packets. Though this is also possible in OMAM, we take advantage of (overlay) multicast routing to reduce the inter-OAR handoff latency. As discussed in the following, the emerging tradeoff is considered as an important subject for future work.

Routing on Flat Labels

The focus of the proposed architecture on the use of flat identifiers as well as the employment of the Canon paradigm (see Chapter 6) indicate a resemblance with the ROFL scheme (see Section 2.1). However, there are fundamental differences between these two approaches. In contrast with ROFL, our architecture builds as an overlay on top of the existing infrastructure, allowing the operation of the already established TCP/IP protocol stack. As such it has a significant deployment advantage over ROFL. But more importantly, by removing any notion of location from the routing protocols, ROFL results in highly stretched paths while also increasing the amount of the required routing state. The employment of the locality-aware *Pastry* scheme along with the use of caching, allows for the proposed architecture to achieve short delivery paths (see Section 5.3.3).

DONA

The *MultiCache* architecture shares some important design features with the DONA architecture [50]. Based on the use of flat identifiers, both approaches construct an overlay network that allows the discovery of information providers, also supporting an anycast primitive. However, unlike DONA, *MultiCache* relies on the operation of the *Pastry* routing scheme and the *Scribe* multicast scheme that efficiently distribute the required forwarding state among the participating infrastructure nodes. At the same time, *MultiCache* also takes advantage of the locality properties of the *Pastry* scheme, thus allowing more efficient forwarding via the anycast service, with the *H-Pastry* design providing tighter adaptation to the underlying network structure. Moreover, the proposed architecture focuses on resource sharing and further delves into the details of the data delivery plane, enabling the joint provision of multicast and caching services. In *MultiCache*, deploying multiple OARs inside an administrative domain becomes the norm, allowing the existence of multiple caching locations that enable the localization of traffic and the enhanced end user experience. Moreover, unlike *MultiCache*, DONA allows clients to only benefit from caching locations on the path towards the root level of the RH hierarchy.

Publish/Subscribe Internet Routing Paradigm

The PSIRP project aims at redesigning the Internet architecture, unconstrained by the already established functionality, at the same time it raises deployment concerns. As a *clean-late* approach, PSIRP aims at the replacement of the existing functionality heavily disrupting the current, well-established model and incurring skepticism about its deployment potential [41]. As already discussed in Section 1.3, our work aims at overcoming these concerns by rather following an evolutionary approach and building on top of the existing architecture as an overlay. Nevertheless, the proposed architecture shares some important features with the PSIRP approach. The use of flat identifiers for the representation of information is a fundamental design choice present in both architectures. The absence of any semantic information in these identifiers simplifies protocol signaling and also allows the employment of scalable flat identifier handling schemes such as the Distributed Hash Tables (DHTs) (e.g. Chord [52], *Pastry* [44], Tapestry [64]).

Furthermore, the employed *Scribe* scheme also adopts the Publish/Subscribe paradigm inherently decoupling information providers from the respective consumers (see Section 3.3.2). However, in contrast with PSIRP, the imbalance of powers in favor of the sender is still present in the underlying architecture, due to the overlay character of our approach and our reliance on the basic Internet protocols, allowing unsolicited traffic and distributed denial-of-service (DDoS) attacks.

At the same time, the inherent coupling of the multicast communication model with information-centrism has also received considerable attention in both architectures. Nevertheless, the overlay character of our approach has inevitably resulted in the creation of stretched multicast trees, though significantly improving with respect to pure end-host deployments (see Chapter 4). In contrast, following a source-routing scheme and based on the complete knowledge of intra-domain topology concentrated at the topology managers, the PSIRP architecture manages to establish non-stretched trees. However, inter-domain routing and forwarding in PSIRP is still being shaped making a complete comparison infeasible. In this issue, our work on DHT adaptation to the inter-network structure has yielded promising results with respect to the efficiency of the inter-domain forwarding links established (see Chapter 6).

In our work we have considered the extensive use of caching as a means for the reduction of unnecessary traffic in the network. Though at an initial stage, PSIRP also considers caching as a promising candidate, though also considering alternative approaches such as short term packet caching at routers (e.g. [117]).

Finally, in our work we have considered the potential benefits stemming from the availability of multicast forwarding, for the support of end-host mobility. The fundamentally different multicast model however does not allow for a direct application and comparison of the proposed approach in the context of PSIRP. In the presented OMAM scheme (see Chapter 7), mobile nodes actively trigger the reconfiguration of the multicast tree supporting them, by directly issuing the appropriate control plane message, taking into account the locality properties of the overlay infrastructure. The source-based routing in PSIRP however would necessitate the communication of a handoff event to the Topology Manager in order to update the created zFilter and would eventually result in the routing changes reaching the corresponding node, presumably incurring a high delay cost.

Content Centric Networking

The CCN research program also departs from the end-host centric model, taking an information-centric approach, however, in contrast to PSIRP, it also aims at maintaining compatibility with the current architecture by taking advantage of unused features of currently deployed routing protocols. Our architecture is also compatible with the current Internet architecture, however, we follow a fundamentally different approach by not incorporating the proposed functionality in the currently deployed routing protocols. This design choice is tightly related with the fundamentally different model adopted for the representation and handling of information. In contrast with the flat identifiers employed through *Pastry* and

Scribe, CCN adopts a hierarchical structure for the information labels aiming at the aggregation of labels when possible, much like the aggregation of IP ranges of addresses in current IP routing protocols. However, the flooding of `Interest` packets in the network and the enormous size of the content namespace (disproportional to the IP address space) raise significant scalability concerns when considering inter-domain level routing and forwarding. In our proposed architecture, these concerns are directly faced, with the logarithmic properties of the underlying *Pastry* DHT and the demonstrated scalability of *Scribe* that efficiently distributes multicast forwarding information [42]. We further investigated this issue in Section 5.4. Also, in contrast to *MultiCache*, CCN does not currently address mobility support.

9.1.2 Multicast and caching

Most approaches considering caching of P2P traffic require the caching servers participating in the considered P2P overlay networks i.e., caching takes place at the application layer requiring the integration of the caching functionality to each specific P2P application. In view of the domination of a few P2P applications [5], this is currently considered feasible. However, the need for adaptation to new emerging applications still remains.

A closely related approach to *MultiCache* is the LSAM Proxy Cache scheme, where multicast is used to distribute popular web pages to proxy caches [118]. LSAM also builds on the proximity of the deployed caches in order to alleviate content providers from the increased workload in cases of popular items and to achieve improved response times for the end users. A major differentiation is that *MultiCache* takes advantage of the established multicast forwarding state to locate caching locations in the network. *MultiCache*'s anycast functionality enables the automated discovery of closely located cached objects taking advantage of the multiplicity of cache locations. In contrast, LSAM client requests only follow the hierarchy of *filters* advancing one level each time the desired content is not located in the current filter.[1] Moreover, in LSAM caches are only fed via multicast, while in

[1]In [118], a search for *sibling* cache locations is implied but no further information is provided regarding the collection and maintenance of the required state for this operation.

MultiCache caches may be fed by other caches, resulting in traffic localization and further reduction of content provider load. Finally, we consider the deployment of a highly scalable overlay multicast scheme paying particular attention to its locality properties, with the aim of localizing cache hits (see Section 5.2.4). In contrast, LSAM is based on a hybrid unicast-broadcast-multicast scheme [119], better suited for low or medium scale deployments i.e., the centralized character of the *secretary* node functionality poses significant scalability concerns in cases of wide area deployment.

Povey *et al.* [120] introduced the idea of employing only leaves of server hierarchies as caching locations, discovered through their ancestors. *MultiCache* shares this design aspect by not placing cached copies of the data at intermediate, forwarding nodes. However, an important distinction is that in [120] the servers of the hierarchy are explicitly and statically distinguished to leaves, i.e., caching nodes, and intermediate, non caching nodes. This obviously introduces an administration and maintenance overhead, in carefully designing the distributed cache topology, also concentrating the entire caching workload on specific servers. In contrast, in *MultiCache*, all OARs present the same functionality and their role in the hierarchy is dynamically determined based on end-host requests. For instance, a forwarding OAR may also cache the content if an attached end-host issues a requests for the specific item, becoming a leaf node for the created tree and providing an additional cached copy for the network area it serves. Moreover, *MultiCache* further departs from this approach by employing multicast forwarding on top of the established hierarchy.

ChunkCast is a peer-to-peer content distribution scheme based on a distributed indexing service that allows the discovery of peers with the desirable content [121]. In ChunkCast, peers participate in the Bamboo DHT [92] and the Vivaldi network coordinates system [122]. Each file in ChunkCast is fragmented into individual *chunks* as in BitTorrent. For each distributed file, a multicast tree is constructed based on the Bamboo key-based routing of peer messages i.e., each peer issues a message towards the root of the respective tree indicating the chunks it has already downloaded. At each level of the created hierarchy, nodes aggregate this information, keeping track of their descendant peers as well as their chunks. The higher the tree level, the larger the amount of information. The maximum size of indexing state is limited for scalability reasons. A peer issues a request towards the root of the tree, indicating the requested chunks as well as the desired number of peers in the final response. Each encountered tree node returns all matching information and forwards the request to higher layers until the desired number of peers has been returned or the root is reached. A preliminary evaluation of ChunkCast demonstrated the effectiveness of the locality properties of Bamboo and Vivaldi, as employed in the proposed scheme, achieving a reduction of the median download time by 32% compared with an basic BitTorrent-like scheme. However, the amount of network traffic generated by the proposed scheme was not investigated and the potential savings stemming from the employement of the Bamboo and Vivaldi schemes remain to be seen.

SplitStream [96] stripes content to produce a forest of disjoint *Scribe* trees in an effort to efficiently distribute the forwarding load. In general, we consider *MultiCache* as an orthogonal approach, as it is heavily based on the caching functionality, which is not considered in the streaming context of SplitStream. By employing, multiple trees per information item, *MultiCache* also targets at the distribution of the forwarding load and the parallelization of content transfer (see Section 5.2.5). However, in our work we do not aim at the creation of disjoint *Scribe* trees. As shown in [97], the employed tree-reconfiguration mechanisms may result in parent-child relationships violating the locality-related properties *MultiCache* is based on.

9.1.3 Mobility support

As discussed in Chapter 2, a multitude of different approaches have been proposed for the support of end-host mobility in the Internet. Yet, the proposed OMAM scheme presents some distinctive characteristics not available in these approaches. Comparing to the Mobile IPv6 protocol and its enhancements, OMAM achieves better localization of the routing updates incurred by mobility. In the case of plain Mobile IPv6, this is translated to a 47% reduction of packet loss, however at the cost of stretch overlay routing (see Section 7.2.2). At the same time, OMAM achieves the location privacy of MNs as their network location is not revealed to their CNs, contrary to Mobile IPv6, HIP and SIP protocols. Moreover, in constrast with Mobile IPv6, OMAM does not rely on the operation of a single network entity - the Home Agent (HA) in standard Mobile IPv6 and the Mobile Ancor Point (MAP) in Hierarchical Mobile IPv6 - for the diversion of data towards the appropriate location in the network. Based on the inherently highly distributed character of the overlay functionality, the necessary routing updates are distributed evenly to the participating OARs, avoiding single points of failure, and further fascilitating the scalability of the proposed architecture. Regarding previous multicast based approaches [19, 20], the overlay character of OMAM circumvents the deployment limitations of IP multicast, yet at the cost of stretched forwarding paths.

9.2 Discussion

Content delivery networks

The proposed architecture bears significant similarities with Content Delivery Networks (CDNs), exactly due to the overlay character of the deployment process. Indeed, the overlay nature of the proposed architecture does allow its deployment by third-parties in the form of CDNs providing their resource saving content distribution services to both network operators and content providers. However, in this work we consider the deployment being controlled by network operators as a step towards the reform of the prevailing networking paradigm that would better fit the current end-user needs. As P2P applications have in many cases alleviated content providers from the high workload of content provisioning, the burden has been shifted to network operators (see Section 1.1.2). In this respect, we consider network operators as inherently motivated to act towards the proposed direction.

Illegal content

One important aspect of P2P file sharing applications, is that they have enabled the illegal distribution of copyrighted content. Operated by ISPs, *MultiCache* cannot be used for the unlicensed distribution of copyrighted content. In this respect, the illegal content would still be distributed via P2P methods that enable the anonymization of the participating users. It is noted that though in both cases the network infrastructure is ultimately used by the end users to reach the content, in the case of P2P applications, the illegal transfer of copyrighted content is covered by the network neutrality principle [11]. On the contrary, in *MultiCache* illegal content is distributed through the ISPs' OAR caches, raising serious legal concerns. However, the ability of *MultiCache* to alleviate content providers from performance bottlenecks and provide a better service to the end-users while reducing the network load, could possibly open a new perspective for ISPs as they could engage in content distribution service provision, themselves, replacing CDNs, while gaining better control of their network traffic. By establishing contractual agreements with content providers (e.g. software vendors, movie distributors, the music industry, etc.), ISPs in this case could provide legal content to their customers.

9.3 Future work

9.3.1 *MultiCache*

Forwarding enhancements

In the *MultiCache* architecture, data is provided either via the multicast tree or via unicast from a discovered caching OAR. In the former case, the RV point (re-)solicits the origin server (which is not a member of the respective multicast tree) to provide the content and all (pseudo-) concurrent requests are served via multicast. Our evaluation demonstrated however that in the case of multicast distribution, the followed paths suffer from the stretch imposed by the overlay character of routing. In order to mitigate the effects of stretched routing, an alternative approach would entail the origin server joining the multicast tree(s) serving the offered content. In terms of the current protocol, the content provider would actually appear to the rest of the system as an ordinary caching location. However, in this case multicast forwarding would be replaced by multiple unicast transmissions of the content towards the leaf nodes. The effect of this alternative approach on both the generated traffic and the perceived download times is subject to further investigation.

Security issues

The departure from the conversational model in our information-centric architecture raises some important security related concerns. First, the indirect nature of the communication with content providers requires the authentication of the content itself. To this end, PKI based mechanisms such as those proposed in [50] may be employed to verify the source of the content and/or apply integrity checks. Moreover, though the actual publish/subscribe character of communication restores the current imbalance of powers (now favoring the sender), *Denial-of-Service* (DoS) attacks in the form of bogus subscriptions that consume the resources of the rendez-vous system, are identified as an open issue. Finally, overlay network access control and data confidentiality is of major importance as the architecture must ensure that in certain application environments only authorized end-hosts receive the distributed content. To this end, a first high-level protection mechanism is based on the careful, off-line, distribution of the flat identifiers that enable the rendez-vous process for the selected piece of information, to authorized end users only.

Alternative deployment scenarios

The deployment of the proposed architecture was shown to provide substantial benefits both for network operators, in terms of the generated network traffic, and for end-hosts, in terms of download times (see Section 5.3.3). Moreover, it was shown that there are consistent performance related incentives for network operators to invest on the proposed architecture and that low deployment densities are enough to achieve these benefits. Nevertheless, the incurred investment cost is still considered as an obstacle to the adoption of *MultiCache*. An alternative deployment scenario would involve the implementation of the proposed functionality on ISP-controlled set-top-boxes, possibly in the context of streaming services (e.g. IPTV); in which the deployment cost could be distributed to the respective clients. In the envisioned scenario, the end-user devices collaborate to improve both the perceived download times and reduce the network traffic. Free-riding concerns would neccesitate the limited control over the set-top-box functionality, restricting user control to the selection of the desired content. In the case of streaming services, issues regarding the timeliness of the received content necessitate further investigation, especially in view of the last-mile bottlenecks in multicast forwarding.

9.3.2 *H-Pastry*

The enforcement of the Canon paradigm on the *Pastry* routing scheme was shown to provide the desired routing characteristics, i.e., the convergence of inter-domain paths and the locality of intra-domain paths. In effect, the underlying inter-network structure was reflected in the resulting design yielding routing decisions isolating traffic within domain boundaries as much as possible. Due to its hierarchical nature, *H-Pastry* is able to capture the customer-provider relationships formed between ISPs. However, the current design does not reflect peering relationships between ISPs. In effect, it is possible for these policies to be neglected by the overlay routing of *H-Pastry*, resulting in traffic unnecessarily passing through provider domains. Therefore, a proper adaptation of *H-Pastry* is required to address this issue. It must be noted though, that this inefficiency has its roots in the Canon paradigm itself. Focusing on the adaptation to a hierarchical structure, Canon assumes a tree domain level graph. However, it is possible for peering relationships to introduce cycles in the domain level graph. Our plans for this issue, include the detailed evaluation of *H-Pastry* with respect to this inefficiency in order to discover the degree by which the employed proximity metrics fail to compensate for this problem. In a next step we aim at the evaluation of an enhancement inspired by the ROFL design [49] in which peering agreements are reflected by the explicit creation of DHT rings by the peering domains.

9.3.3 Overlay Multicast Assisted Mobility (OMAM)

Performance evaluation

In Chapter 7 the proposed mobility support scheme was evaluated against Mobile IPv6 both analytically and via simulation. The evaluation demonstrated the effectiveness of the proposed approach in better localizing the routing updates incurred by mobility. However, the lack of a Hierarchical MIPv6 (HMIPv6) implementation in the considered simulation environment limited the completeness of the evaluation. Though the analytic comparison of the proposed scheme against HIPv6 supports the expectation of lower Handover Latencies for OMAM, a simulation based comparison is necessary for the completeness of the comparison. Taking a step further, we also consider the comparison of OMAM against the *i*3 mobility support scheme. In this context, we aim at the establishment of a richer evaluation environment based on the use of more realistic topologies (e.g., GT-ITM topologies [47]) that will allow the detailed evaluation of the considered protocol both at a micro-mobility and a macro-mobility level. Moreover, our plans include the evaluation of the proposed mobility support scheme in the context of additional, non-streaming applications, in which the content must be delivered correctly and in its entirety.

Paging

Mobility support in the proposed OMAM scheme (see Chapter 7), has been focused on the seamless delivery of content to moving end-hosts, trying to minimize the service disruption incurred by handoffs. In the proposed solution, simultaneous mobility is supported with the establishment of one multicast tree in each direction. In publish/subscribe terms, end-hosts subscribe to the information published by their counterparts. The subscribers are tracked by the architecture via the forwarding state established during their subscription and the publishers are assumed to have advertised their data to the appropriate RV point, which is then responsible for soliciting their data upon subscription. This solicitation is based on information provided with the advertisement i.e., the IP address of the sender. However, in case of mobile publishers it is possible for this information to have become stale as they may have moved to another area served by a different OAR. Similarly, mobile subscribers may have subscribed for data which will be delivered at some point in the future (e.g., a mobile node subscribes for calls of another node). In these cases, a location management scheme is required for locating mobile nodes. Maintaining accurate information on end-hosts' location would require the continuous update of the respective routing information upon a handoff event. Nevertheless, this practice is considered as impractical both due to the associated signaling overhead and the high energy consumption at the mobile terminal because of the elimination of idle periods. Therefore, a *paging* mechanism is required to balance the tradeoff between this overhead and the accuracy of the location information.

Our plans on this issue include the exploitation of *Pastry*'s Neighborhood Set (see Section 3.3.1) for the dynamic definition of *paging areas*. Inactive mobile nodes i.e., mobile nodes not engaged in a communication session, fall into an idle state, only periodically informing the network about their location (e.g., every t seconds). This is accomplished by periodically issuing a subscribe request for a unique identifier towards their current OAR. Note that this identifier is used only for location management purposes and does not refer to a specific data item requested by a mobile node. Possible subscriptions for data expected in the future, are not maintained as they are expected to create additional overhead in the network. Such subscriptions are re-submitted upon data availability, once the described mechanism has allowed the discovery of the MN's exact location. Each OAR in the resulting multicast tree maintains a timer for the respective forwarding entry (as in regular *Scribe* but with a properly defined value as explained below). Subscription requests towards the same OAR (i.e., no inter-OAR handoff has taken place) simply refresh these timers. The *Scribe* JOIN messages targeting only at a location update are further augmented with an *overlay hop counter* (OHC) which is augmented on each overlay hop traversed by the message. We denote these messages as JOIN-L messages. The respective forwarding entry at each encountered OAR is also augmented to keep track of the OHC value. Upon reception of a JOIN-L message, or when the corresponding subscription is refreshed, an OAR initializes its forwarding entry timer with a value proportional to the current OHC value (e.g., $OHC \times t$). In this way, OARs located several overlay hops away from the last known location of a mobile node will preserve the respective forwarding entry for longer period of time than their descendants in the multicast tree. Note also that the networking distances between successive nodes in the tree, exponentially increase as we approach the RV point (see Section 3.3.1). Therefore, the OAR network is expected to maintain co. Upon expiration of this timer, the corresponding forwarding entry is deleted. JOIN-L messages at OAR$_{CA}$ result in the deletion of the older forwarding entry pointing to the old location of the MN. When data is available for an inactive node, a *paging* message is sent down the respective multicast tree. The current leaf node initiates a limited flooding scheme by broadcasting this paging message through all APs currently served by it. Additionally, depending on its OHC value it also forwards this paging message to the nodes of its Neighborhood Set. The OHC value is inserted in this message and is decremented by one at the receiving OAR. The process recursively continues until the OHC value in the propagated message reaches zero[2]. Once the MN is located it issues its data-related subscriptions. Publishers of information simply deliver their data to the respective RV node.

[2]Several optimizations can also be applied reducing the total number of messages incurred by the paging mechanism. However, they are considered out of the scope of this description.

Chapter 10

Conclusions

The high level objective of the proposed architecture was to address the model mismatch between the information-centric nature of the prevailing communication patterns in the Internet and the end-host centric design of the currently available communication platform, a mismatch that has resulted in the excessive use of network resources. Our work aimed at expressing the introduction of information-awareness inside the network with the deployment of resource sharing mechanisms, without however disrupting the well-established Internet architecture. In effect, our efforts focused on the creation of a resource-efficient information-centric networking environment on top of the current Internet architecture, that would better fit the current communication patterns, while enhancing the end user experience.

We revisited resource sharing mechanisms such as multicast and caching, but in a somewhat different context. The desired functionality was approached as a set of network services supported by network operators with a two-fold target: the improvement of network resource utilization and the enhancement of the end-user experience. Therefore, the deployment of the selected mechanisms inside the network instead of at the edges became important.

We initially investigated the potential benefits of the overlay deployment of multicast, with respect to both inter-domain and intra-domain dimensions. Our simulations first demonstrated the benefits stemming from pushing the functionality from the edges of the network to the network infrastructure. By avoiding forwarding on last-mile links, both (i) the content delivery paths are improved and (ii) the resulting network traffic is substantially reduced. We further showed that an evolutionary path towards ubiquitous deployment across the Internet is feasible, providing consistent advantages to network operators adopting the proposed architecture.

Further enhancing the overlay architecture and taking into consideration the low cost (and decreasing trend) of storage, we proposed a hybrid protocol enabling the joint operation of multicast and caching. The direct comparison with the prevailing communication model revealed the significant benefits of the proposed architecture: compared to the BitTorrent application, which characteristically expresses the aforementioned model mismatch, the proposed architecture significantly reduces network traffic as well as the download times, yielding significant deployment incentives for network operators. In an average deployment scenario, MultiCache achieves the reduction of egress interdomain traffic by 53,19%, also reducing the the intradomain traffic load by 61,39%. At the same time, the download time is reduced by 56,3%. Our investigation showed the importance of request aggregation for resource sharing. Sparser deployments of high storage capacity caches are preferable over denser deployments of lower capacity caches, as they allow multiple end host requests to be served by fewer overlay nodes, increasing the benefits from caching. On the other hand, in denser deployments, requests are distributed across the network, increasing the traffic load.

The creation of an overlay information-centric architecture for the distribution of content on top of the end-host centric Internet, raised two important issues. First, focusing on information requires the establishment of a scalable control plane, able to efficiently handle the representation of information, given the unlimited size of the information domain. Our analytical investigation of this issue demonstrated the high scalability of the employed protocols, showing the applicability of the proposed overlay approach. Second, adopting an information-centric model on top of the end-host centric Internet would be expected to yield a highly stretched routing/forwarding plane due to the exact model mismatch. At a first level, the choice of a locality aware overlay routing scheme such as *Pas-*

try proved to provide good adaptation properties. However, when considering the costs incurred by inter-domain traffic, stricter guarantees should be provided. We addressed this problem by applying the Canon paradigm on *Pastry*, yielding the design of *H-Pastry*. Our evaluation showed that guarantees on inter-domain forwarding decisions are established, reducing as a result the creation of cross-ISP forwarding links, while also reducing at the same time the amount of routing state required. *H-Pastry* constitutes an improvement over the previously proposed *Crescendo* scheme, reducing the stretch of overlay routing by up to 60%.

Moreover, we revisited mobility support in the context of information-centric networking. Taking advantage of the established multicast functionality, we demonstrated the effectiveness of multicast assisted mobility, providing a unified solution inherently supporting both macro-mobility and micro-mobility. Our simulations showed that the proposed mobility support scheme can reduce the packet loss during handoffs by up to 47% compaired with Mobile IPv6.

During research, a set of simulation tools were developed. The BitTorrent simulation modules, as well as the tools that support the use of GT-ITM topologies in the widely used OMNeT++ Simulator have been contributed to the community, with the hope of facilitating research in this area. The set of BitTorrent modules has already been used in the context of the ICT 4WARD project [123], as documented in deliverable D6.2 [124].

There are opportunities for further research in the area. Alternative deployment models may be investigated, while protocol enhancements and variations are of particular interest for improving the performance of the content distribution architecture. Based on the lessons learned from the adaptation of *Pastry* to the underlying internetwork structure, further work targeting the reflection of the inter-AS relationships in the design of *H-Pastry*, can be a primary research target. Finally, additional aspects of mobility management, such as paging in the context of location management, call for further investigation and systematic evaluation.

Appendix A

Acronyms

AP Access Point

AS Autonomous Systems

BA Binding Acknowledgment

BRAS Broadband Remote Access Server

BU Binding Update

CBT Core Based Trees

CCN Content Centric Networking

CDN Content Delivery Network

CHR Cache Hit Ratio

CHR-Intra Intra-domain Cache Hit Ratio

CIP Cellular IP

CN Correspondent Node

CoA Care of Address

CoT Care-of Test

CoTi Care-of Test Init

CuN Current Node

DBS Distance to Block Source

DDoS Distributed Denial-of-Service

DFS Depth First Search

DHT Distributed Hash Table

DONA Data-Oriented and beyond Network Architecture

DoS Denial-of-Service

DSLAM Digital Subscriber Line Access Multiplexer

DT Download Time

EIT Egress Inter-domain Traffic

GBN Global Bootstrap Node

GRT Global Routing Table

GT-ITM Georgia Tech Internet Topology Model

HA Home Agent

HAWAII Handoff-Aware Wireless Access Internet Infrastructure

HIP Host Identity Protocol

HL Handoff Latency

HMIPv6 Hierarchical MIPv6

HoT Home Test

HoTi Home Test Init

HPTP HTTP-based Peer-to-Peer

i3 Internet Indirection Infrastructure

ID Identifier

IPR Intellectual Property Rights

ITF Inter-domain Topology Formation

ITL Intra-domain Traffic Load

JN Joining Node

LBN Local Bootstrap Node

LCoA On-link Care-of Address

LRT Local Routing Table

LRU Least Recently Used

LSB Least Sent Bytes

MAP Mobile Anchor Point

MFU Most Frequently Used

MINRS Minimum Relative Size

MIP Mobile IP

MN Mobile Node

MRU Most Recently Used

OAR Overlay Access Router

OHC Overlay Hop Counter

OMAM Overlay Multicast Assisted Mobility

P2P	Peer-to-Peer
PIM-SM	Protocol Independent Multicast - Sparse Mode
PoP	Point-Of-Presence
RAOM	Router Assisted Overlay Multicast
RCoA	Regional CoA
RHs	Resolution Handlers
RO	Route Optimization
ROFL	Routing on Flat Labels
RTT	Round Trip Time
RV	Rendez-Vous
SCX	ScatterCast proXie
SIP	Session Initiation Protocol
TM	Topology Manager
TTL	Time-To-Live
URI	Uniform Resource Identifier
URN	Uniform Resource Name

Bibliography

[1] D. Clark, "The design philosophy of the DARPA Internet protocols," in *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*. New York, NY, USA: ACM, 1988, pp. 106–114.

[2] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, no. 4, pp. 277–288, 1984.

[3] YouTube. (2010). [Online]. Available: http://www.youtube.com

[4] Google. (2010). [Online]. Available: http://www.google.com

[5] IPOQUE. (2009) Internet Study 2008/2009. [Online]. Available: http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009

[6] D. Clark, B. Lehr, S. Bauer, P. Faratin, R. Sami, and J. Wroclawsk, "Overlay Networks and the Future of the Internet," *Communication & Strategies*, vol. 63, 2006.

[7] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPS and P2P users cooperate for improved performance?" *SIGCOMM Computer Communications Review*, vol. 37, no. 3, pp. 29–40, 2007.

[8] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should internet service providers fear peer-assisted content distribution?" in *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. Berkeley, CA, USA: USENIX Association, 2005, pp. 6–6.

[9] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2003, pp. 314–329.

[10] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez, "Deep diving into bittorrent locality," *CoRR*, vol. abs/0907.3874, 2009.

[11] E. Bangeman. (2008, January) FCC officially opens proceeding on Comcast's P2P throttling. [Online]. Available: http://arstechnica.com/tech-policy/news/2008/01/fcc-officially-opens-proceeding-on-comcasts-p2p-throttling.ars

[12] D. Johnson, C. Perkins, and J. Arkko, "Mobility support in IPv6," RFC 3775, Jun. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3775.txt

[13] H. Soliman, C. Castelluccia, K. E. Malki, and L. Bellier, "Hierarchical Mobile IPv6 Mobility Management (HMIPv6)," RFC 4140, Aug. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4140.txt

[14] R. Koodli, "Fast Handovers for Mobile IPv6," RFC 4068, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4068.txt

[15] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host Identity Protocol (HIP)," RFC 5201, Jun. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc5201.txt

[16] P. Nikander, T. Henderson, C. Vogt, and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol," RFC 5206, Apr. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5206.txt

[17] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, Apr. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt

[18] H. Schulzrinne and E. Wedlund, "Application-layer mobility using SIP," *SIGMOBILE Mobile Computer Communications Review*, vol. 4, no. 3, pp. 47–57, 2000.

[19] K. Keeton, B. A. Mah, S. Seshan, R. H. Katz, and D. Ferrari, "Providing connection-oriented network services to mobile hosts," in *MLCS '93: Mobile & Location-Independent Computing Symposium on Mobile & Location-Independent Computing Symposium.* Berkeley, CA, USA: USENIX Association, 1993, pp. 8–8.

[20] J. Mysore and V. Bharghavan, "A new multicasting-based architecture for internet host mobility," in *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking.* New York, NY, USA: ACM, 1997, pp. 161–172.

[21] A. Helmy, M. Jaseemuddin, and G. Bhaskara, “Efficient micro-mobility using intra-domain multicast-based mechanisms (M&M),” *SIGCOMM Computer Communications Review*, vol. 32, no. 5, pp. 61–72, 2002.

[22] A. Festag, H. Karl, and A. Wolisz, “Investigation of multicast-based mobility support in all-IP cellular networks,” *Wiley Wireless Communications and Mobile Computing*, vol. 7, no. 3, pp. 319–339, 2007.

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

[24] Napster. (2010). [Online]. Available: http://www.napster.com

[25] eDonkey. (2010). [Online]. Available: http://www.edonkey2000.com/

[26] eMule. (2010). [Online]. Available: http://www.emule-project.net

[27] Gnutella. (2010). [Online]. Available: http://www.gnutella.com

[28] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, “Making gnutella-like P2P systems scalable,” in *SIGCOMM ’03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2003, pp. 407–418.

[29] D. Hughes, G. Coulson, and J. Walkerdine, “Free Riding on Gnutella Revisited: The Bell Tolls?” *IEEE Distributed Systems Online*, vol. 6, 2005.

[30] B. Krishnamurthy, C. Wills, and Y. Zhang, “On the use and performance of content distribution networks,” in *IMW ’01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement.* New York, NY, USA: ACM, 2001, pp. 169–182.

[31] C. Huang, A. Wang, J. Li, and K. W. Ross, “Understanding hybrid CDN-P2P: why limelight needs its own Red Swoosh,” in *NOSSDAV ’08: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video.* New York, NY, USA: ACM, 2008, pp. 75–80.

[32] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, “Drafting behind Akamai: inferring network conditions based on CDN redirections,” *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1752–1765, 2009.

[33] Akamai, June 2010. [Online]. Available: http://www.akamai.com

[34] Limelight, June 2010. [Online]. Available: http://www.limelightnetworks.com

[35] Velocix, June 2010. [Online]. Available: http://www.velocix.com

[36] I. D. C. (IDC). (2010, May) The Digital Universe Decade - Are You Ready? [Online]. Available: http://www.emc.com/collateral/demos/microsites/idc-digital-universe/iview.htm

[37] S. E. Deering, "Multicast routing in a datagram internetwork," Ph.D. dissertation, Stanford, CA, USA, 1992.

[38] H. Eriksson, "MBONE: the multicast backbone," *Communications ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[39] PSIRP Project, *PSIRP Project Home Page*, http://www.psirp.org, 2010.

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

[41] R. Guerin, "Does the Networked Generation Need a Next Generation Network?" Keynote presentation at the 5th Euro-NGI conference on Next Generation Internet networks (NGI), Aveiro, Portugal, July 2009. [Online]. Available: http://www.seas.upenn.edu/ guerin/presentations/NGI-2009.pdf

[42] M. Castro, P. Druschel, A.-M. 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. 100–110, 2002.

[43] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in G Major: Designing DHTs with Hierarchical Structure," in *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 263–272.

[44] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*. London, UK: Springer-Verlag, 2001, pp. 329–350.

[45] A. Varga, *OMNeT++ Simulator Home Page*, http://www.omnetpp.org.

[46] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of BitTorrent-like peer-to-peer systems," *IEEE Journal on Selected Areas in Communication*, vol. 25, no. 1, pp. 155–169, 2007.

[47] E. Zegura, K. Calvert, , and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM '96: Proceedings of the 15th IEEE International Conference on Computer Communications*, vol. 2, San Francisco, CA, USA, March 1996, pp. 594–602.

[48] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," *IEEE/ACM Transactions on Networking*, vol. 12, no. 2, pp. 205–218, 2004.

[49] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: routing on flat labels," *SIGCOMM Computer Communications Review*, vol. 36, no. 4, pp. 363–374, 2006.

[50] 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 *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2007, pp. 181–192.

[51] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies.* New York, NY, USA: ACM, 2009, pp. 1–12.

[52] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[53] 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.

[54] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, 2003.

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

[56] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: line speed publish/subscribe inter-networking," in *SIGCOMM '09: Proceedings of the 2009 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2009, pp. 195–206.

[57] J. Vasseur, N. Shen, and R. Aggarwal, "Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information," RFC 4971, Jul. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4971.txt

[58] L. Berger, I. Bryskin, A. Zinin, and R. Coltun, "The OSPF Opaque LSA Option," RFC 5250, Jul. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5250.txt

[59] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., "Overcast: reliable multicasting with an overlay network," in *OSDI'00: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation.* Berkeley, CA, USA: USENIX Association, 2000, pp. 14–14.

[60] Y.-h. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast (keynote address)," in *SIGMETRICS '00: Proceedings of the 2000 international conference on Measurement and modeling of computer systems.* New York, NY, USA: ACM, 2000, pp. 1–12.

[61] S. E. Deering, "Multicast routing in internetworks and extended LANs," in *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols.* New York, NY, USA: ACM, 1988, pp. 55–64.

[62] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2002, pp. 205–217.

[63] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz, "Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination," in *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video.* New York, NY, USA: ACM, 2001, pp. 11–20.

[64] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and routing," Berkeley, CA, USA, Tech. Rep., 2001.

[65] Y. D. Chawathe, "Scattercast: an architecture for internet broadcast distribution as an infrastructure service," Ph.D. dissertation, 2000, chair-Brewer, Eric A.

[66] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems.* Washington, DC, USA: IEEE Computer Society, 2006, p. 66.

[67] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in *SIGCOMM '08: Proceedings of the 2008 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2008.

[68] D. R. Choffnes and F. E. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," *SIGCOMM Computer Communications Review*, vol. 38, no. 4, pp. 363–374, 2008.

[69] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang, "TopBT: A Topology-Aware and Infrastructure-Independent BitTorrent Client," in *INFOCOM 2010: Proceedings of the 29th IEEE International Conference on Computer Communications*, San Diego, CA, USA, March 2010.

[70] I. Papafili, S. Soursos, and G. D. Stamoulis, "Improvement of BitTorrent Performance and Inter-domain Traffic by Inserting ISP-Owned Peers," in *ICQT '09: Proceedings of the 6th International Workshop on Internet Charging and Qos Technologies.* Berlin, Heidelberg: Springer-Verlag, 2009, pp. 97–108.

[71] J. Wang, "A survey of web caching schemes for the Internet," *SIGCOMM Computer Communications Review*, vol. 29, no. 5, pp. 36–46, 1999.

[72] N. Leibowitz, A. Bergman, R. Ben-shaul, and A. Shavit, "Are file swapping networks cacheable? Characterizing P2P traffic," in *WCW'02: Proceedings of the 7th International Workshop on Web Content Caching and Distribution*, 2002.

[73] M. Hefeeda and O. Saleh, "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1447–1460, 2008.

[74] G. Shen, Y. Wang, Y. Xiong, B. Y. Zhao, and Z. li Zhang, "HPTP: Relieving the tension between ISPs and P2P," in *IPTPS '07: Proceedings of the 6th International Workshop on P2P Systems*, 2007.

[75] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak, "Cache replacement policies revisited: the case of P2P traffic," in *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 182–189.

[76] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," in *INFOCOM '99: Proceedings of the 18th IEEE International Conference on Computer Communications*, vol. 1, New York, NY, USA, March 1999, pp. 126–134.

[77] S. Patro and Y. C. Hu, “Transparent Query Caching in Peer-to-Peer Overlay Networks,” in *IPDPS ’03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing.* Washington, DC, USA: IEEE Computer Society, 2003, p. 32.1.

[78] A. C. Snoeren and H. Balakrishnan, “An end-to-end approach to host mobility,” in *MobiCom ’00: Proceedings of the 6th annual international conference on Mobile computing and networking.* New York, NY, USA: ACM, 2000, pp. 155–166.

[79] H. Schulzrinne, A. Rao, and R. Lanphier, “Real Time Streaming Protocol (RTSP),” RFC 2326, Apr. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2326.txt

[80] L. Song, D. Kotz, R. Jain, and X. He, “Evaluating location predictors with extensive Wi-Fi mobility data,” in *INFOCOM ’04: Proceedings of the 23d IEEE International Conference on Computer Communications*, vol. 2, 2004, pp. 1414–1424.

[81] A. T. Campbell, J. Gomez, S. Kim, A. G. Valko, C. yih Wan, and Z. R. Turanyi, “Design, implementation, and evaluation of Cellular IP,” *IEEE Personal Communications*, vol. 7, pp. 42–49, 2000.

[82] R. Ramjee, K. Varadhan, L. Salgarelli, S. R. Thuel, S.-Y. Wang, and T. La Porta, “HAWAII: a domain-based approach for supporting mobility in wide-area wireless networks,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 396–410, 2002.

[83] W. Fenner, “Internet Group Management Protocol, Version 2,” RFC 2236, Nov. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2236.txt

[84] A. Mihailovic and M. S. A. Aghvami, “Multicast for mobility protocol (MMP) for emerging Internet networks,” in *PIMRC ’00: Proceedings of the 11th IEEE International Symposium Personal Indoor Mobile Communixations*, London, UK, Sep. 2000.

[85] T. Ballardie, P. Francis, and J. Crowcroft, “Core based trees (CBT),” *SIGCOMM Computer Communications Review*, vol. 23, no. 4, pp. 85–95, 1993.

[86] T. Stading, P. Maniatis, and M. Baker, “Peer-to-Peer Caching Schemes to Address Flash Crowds,” in *IPTPS ’01: Revised Papers from the First International Workshop on Peer-to-Peer Systems.* London, UK: Springer-Verlag, 2002, pp. 203–213.

[87] (2010, May) Historical Notes about the Cost of Hard Drive Storage Space. [Online]. Available: http://ns1758.ca/winch/winchest.html

[88] Y. Huang and H. Garcia-Molina, "Publish/subscribe in a mobile environment," *Wireless Networks*, vol. 10, no. 6, pp. 643–652, 2004.

[89] R. Tewari, M. Dahlin, H. Vin, and J. Kay, "Design Considerations for Distributed Caching on the Internet," in *ICDCS '99: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 1999, p. 273.

[90] J. Rajahalme, M. Särelä, P. Nikander, and S. Tarkoma, "Incentive-compatible caching and peering in data-oriented networks," in *CoNEXT '08: Proceedings of the 4th international conference on Emerging networking experiments and technologies*. New York, NY, USA: ACM, 2008, pp. 1–6.

[91] AgilentTechnologies, "Understanding DSLAM and BRAS Access Devices," White Paper, Feb. 2006. [Online]. Available: http://cp.literature.agilent.com/litweb/pdf/5989-4766EN.pdf

[92] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[93] R. Mahajan, M. Castro, and A. Rowstron, "Controlling the Cost of Reliability in Peer-to-Peer Overlays," in *Peer-to-Peer Systems II*, ser. Lecture Notes in Computer Science, vol. 2735. Springer, Oct. 2003, pp. 21–32.

[94] E. C. Efstathiou, P. A. Frangoudis, and G. C. Polyzos, "Stimulating participation in wireless community networks," in *INFOCOM 2006: Proceedings of the 25th IEEE International Conference on Computer Communications*, Barcelona, Catalunya, Spain, April 2006, pp. 23–29.

[95] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," Microsoft Research, Tech. Rep. MSR-RT-2002-82, 2002.

[96] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: high-bandwidth multicast in cooperative environments," in *SOSP '03: Proceedings of the 19th ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 2003, pp. 298–313.

[97] A. R. Bharambe, S. G. Rao, V. N. Padmanabhan, S. Seshan, and H. Zhang, "The Impact of Heterogeneous Bandwidth Constraints on DHT-Based Multicast Protocols," in *IPTPS '05: Proceedings of the 4th International Workshop on P2P Systems*, 2005.

[98] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *Proc. of the IEEE Global Internet Symposium*, 2007, pp. 79–84.

[99] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast using peer-to-peer overlays," in *INFOCOM '03: Proceedings of the 22nd IEEE International Conference on Computer Communications.* IEEE, March-April 2003.

[100] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601, Aug. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4601.txt

[101] IANA. (2010, May) Autonomous System (AS) Numbers. [Online]. Available: http://www.iana.org/assignments/as-NUMBERs/as-NUMBERs.xml

[102] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scalable Application-level Anycast for Highly Dynamic Groups," in *NGC '03: Proceedings of the 5th International Networked Group Communication (COST264 Workshop)*, 2003.

[103] K. Katsaros, V. Kemerlis, C. Stais, and G. Xylomenos, "A BitTorrent Module for the OMNeT++ Simulator," in *MASCOTS '09: Proceedings of the 17th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, London, UK, September 2009, pp. 361–370.

[104] M. Busari and C. Williamson, "ProWGen: a synthetic workload generation tool for simulation evaluation of web proxy caches," *Computer Networks*, vol. 38, no. 6, pp. 779–794, 2002.

[105] A. Bellissimo, B. N. Levine, and P. Shenoy, "Exploring the use of BitTorrent as the basis for a large trace repository," University of Massachusetts Amherst, Tech. Rep., June 2004.

[106] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.

[107] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh, "Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* New York, NY, USA: ACM, 2003, pp. 395–406.

[108] V. Pappas, D. Massey, A. Terzis, and L. Zhang, "A comparative study of the DNS design with DHT-based alternatives," in *INFOCOM 2006: Proceedings of the 25th IEEE International Conference on Computer Communications*, Barcelona, Catalunya, Spain, April 2006.

[109] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transaction on Networking*, vol. 9, no. 6, pp. 733–745, 2001.

[110] G. Nomikos, "Implementation of Hierarchical Chord (Crescendo), according to Canon paradigm and evaluation, via simulation over realistic network topologies, of the Crescendo's advantages in comparison to Chord," Master's thesis, Mobile Multimedia Laboratory (MMLab), Athens University of Economics and Business, Athens, Greece, 2009.

[111] F. Z. Yousaf, C. Bauer, and C. Wietfeld, "An accurate and extensible mobile IPv6 (xMIPV6) simulation model for OMNeT++," in *SIMUTools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops.* Brussels, Belgium: ICST, 2008, pp. 1–8.

[112] C. Perkins and K.-Y. Wang, "Optimized Smooth Handoffs in Mobile IP," in *ISCC '99: Proceedings of the 4th IEEE Symposium on Computers and Communications.* Washington, DC, USA: IEEE Computer Society, 1999, p. 340.

[113] ITU, "H.264 : Advanced video coding for generic audiovisual services," http://www.itu.int/rec/T-REC-H.264, 2007.

[114] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The Bittorrent P2P file-sharing system: Measurements and analysis," in *IPTPS '05: Proceedings of the 4th International Workshop on P2P Systems*, Feb 2005, pp. 205–216.

[115] K. Katsaros, V. Kemerlis, C. Stais, and G. Xylomenos. (2009, June) BitTorrent module for OMNeT++. [Online]. Available: http://www.mm.aueb.gr/research/bittorrent/

[116] CAIDA. (2010, May) CAIDA skitter Topology Traces. [Online]. Available: http://imdc.datcat.org/collection/1-0006-1=CAIDA+skitter+Topology+Traces

[117] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," in *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication.* New York, NY, USA: ACM, 2008, pp. 219–230.

[118] J. Touch and A. S. Hughes, "LSAM proxy cache: a multicast distributed virtual cache," *Computer Networks and ISDN Systems*, vol. 30, no. 22-23, pp. 2245–2252, 1998.

[119] J. R. Cooperstock and S. Kotsopoulos, "Why use a fishing line when you have a net? an adaptive multicast data distribution protocol," in *ATEC '96: Proceedings of the 1996 annual conference on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 1996, pp. 28–28.

[120] D. P. John and J. Harrison, "A Distributed Internet Cache," in *In Proceedings of the 20th Australian Computer Science Conference*, 1997, pp. 5–7.

[121] B. Chun, P. Wu, H. Weatherspoon, and J. Kubiatowicz, "ChunkCast: An Anycast Service for Large Content Distribution," in *Proceedings of the 5th International Workshop on Peer-to-Peer Systems*, 2006.

[122] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 15–26.

[123] 4WARD Project, *4WARD Project Home Page*, http://www.4ward-project.eu, 2010.

[124] ——, "D-6.2 Second NetInf architecture description," Project Deliverable, January 2010. [Online]. Available: http://www.4ward-project.eu/index.php?s=file_download&id=70