Freie Universität Berlin

Department of Mathematics and Computer Science
Institute of Computer Science

Diploma Thesis

Scalable Adaptive Group Communication on Bi-directional Shared Prefix Trees

Matthias Wählisch
waehl@inf.fu-berlin.de

July 2008

Examiner: **Prof. Dr.-Ing. Jochen Schiller**
Tutor: **Georg Wittenburg, M.Sc.**

Abstract

Efficient group communication within the Internet has been implemented by multicast. Unfortunately, its global deployment is missing. Nevertheless, emerging and progressively establishing popular applications, like IPTV or large-scale social video chats, require an economical data distribution throughout the Internet.

To overcome the problem the limitations of multicast deployment, we introduce and analyze BIDIR-SAM, the first structured overlay multicast scheme based on bi-directional shared prefix trees. BIDIR-SAM admits predictable costs growing logarithmically with increasing group size. We also present a broadcast approach for DHT-enabled P2P networks. Both schemes are integrated in a standard compliant hybrid group communication architecture, bridging the gap between overlay and underlay as well as between inter- and intra-domain multicast.

Acknowledgments

Many thanks to Georg Wittenburg for his very valuable feedback throughout this diploma thesis. Despite of his deep involvement in his own projects, he always had time for fruitful discussions. It has been a real pleasure working with him.

I would like to thank Jochen Schiller and Günter Rote for their understanding of unexpected problems.

Stephan Krause and Ingmar Baumgart are gratefully acknowledged for providing me with a preliminary version of the Scribe enabled P2P network simulator OverSim at the beginning of this project.

Last but not least, I would like to express my gratitude to Thomas Schmidt for continuous discussions on this work, and his help to clarify my mind not only on probability affairs. Beyond this thesis, I would like to thank him for ten years of joyful collaboration. Thanks, for being inspiring.

This work has been supported by the German Bundesministerium für Bildung und Forschung within the project Moviecast. This funding offered the opportunity to work on this subject without a thought on my subsistence – acknowledgements to the ministry for doing the right thing.

Contents

List of Figures

1 Introduction

When the Internet was still in its early, premature state of development, the idea arose to extend unicast capabilities by a multicast group service [5]. The corresponding distribution function allows to inject data *once* into computer networks, but feed *multiple* receivers of the same stream. Packets will be automatically duplicated at suitable branching points along the paths from the source to the listeners, thereby economizing transmission capacities.

Multicast communication techniques have been under debate since Stephen Deering introduced the host group model to the Internet layer [29, 30]. Until today, the initial approach of Any Source Multicast (ASM) routing remained hesitant to spread beyond limited, controlled environments. Meanwhile, new demands for group communication are arriving with increasing intensity, e.g., multimedia streaming and conferencing in mobile environments, service discovery in service-oriented architectures or self-configuring components in autonomous networks.

However, the deployment of IP multicast in general has been slow over the past 15 years, even though all major router vendors and operating systems offer a wide variety of implementations to support multicast [31]. A fundamental dispute arose on multicast concepts in the end-to-end design principle by Saltzer *et al.* [84], questioning the appropriate layer, where group communication service should reside on. For several years, the focus of the research community turned towards application layer multicast, and only recently reconsidered the relevance of IP layer multicast.

In the past, vendors and technicians, trying to promote multicast functionality, focused their marketing arguments on network efficiency and unintentionally degraded its paradigm to unidirectional, broadcast-type services. Since then multicast suffers from a reputation of being merely useful for non-interactive, 'archaic' mass distribution tasks. Large-scale interactive group applications like massive multiplayer games, conferencing in restricted regimes or complex collaborative environments have only recently drawn attention. In parallel, mobile multimedia group communication appeared as an emerging applications field. Multicast services in mobile environments may soon become indispensable, when multimedia distribution services such as DVB-H and IPTV will develop as strong business cases for portables. As IP mobility will unfold dominance and efficiency in costly radio environments, the evolution of multicast protocols will naturally follow mobility constraints [12].

In the following, we discuss the current state of the art of IP multicast, its potentials, problems and solutions. A brief introduction in overlay multicast technologies is given.

1.1 IP Layer Multicast

1.1.1 Intra-Domain Multicast

A large number of today's enterprise networks provide multicast services within their local domains to facilitate administrative tasks as well as shared group applications. This is indicated by the wide availability of intra-domain multicast protocols such as IGMP [19], MLD [100],

DVMRP [104, 74], PIM-DM/SM/SSM [3, 39], Bidir-PIM [42] in routers and end systems and the fairly uniform presence of multicast capabilities in lower layer protocols, i.e., in IEEE 802.3 Ethernet, 802.11 WLAN, 802.16 WIMAX [49] or in 3GPP MBMS [1] and DVB-H [34].

This deployment success on the one hand can be attributed to the large number of nodes installed in common enterprise domains, which immediately profit from multicast distribution services, on the other hand complex routing services are much easier established, controlled and also restricted within a single administrative domain. Multicast admission and scoping in general, and prevention of misuse in DDoS attacks in particular, can be managed with reasonable effort at intra-domain level, while these tasks turn into critical challenges in an inter-provider context. Furthermore higher spare capacities of routers and systems at Internet edges allow for concurrent operation of multicast management burdens, while at the same time scaling limitations inherent to most of the present protocols remain invisible within most enterprise networks.

Nevertheless, intra-domain multicast routing is not considered complete, but remains an active research field. The major reason for discontent results from the handling of data-driven multicast distribution states. They are required at the routing layer, which breaks the paradigm of stateless forwarders and opens the door for flow-state attacks directed against the routing infrastructure. Recent work on bi-directional PIM by Handley *et al.* [43] has advanced this debate by utilizing a group-specific shared tree within limited domains. States for this bi-directionally operational, but not uniformly optimal distribution tree are established at multicast activation and group creation and thus fully decouple from the data plane.

1.1.2 Inter-Domain Multicast

In contrast to the success at an internal level, inter-domain multicast deployment largely failed. Inexplicit benefits, complexity and scalability issues with multicast BGP-4 extensions, robustness and security concerns, as well as the threat of intransparently interwoven service models kept Internet Service Providers (ISPs) from adding multicast burdens onto their already notoriously overloaded core routers. At present, the key issues for inter-domain multicast deployment may be seen as:

Control on groups will allow ISPs to explicitly restrict (or charge for) distribution services, and thus must be considered an important part of a consistent business model.

Controlled load on backbone routers in terms of table spaces, computational and signalling demands will be required for a predictable service quality.

Scalable protocols build the essential foundation for a large-scale deployment.

State aggregation within shared trees will be a technical demand to control the router load.

Forward routing will be of vital importance due to asymmetric backbone routes. Many multicast routing protocols depend on Reverse Path Forwarding and thereby erroneously assume symmetric routes.

Explicit benefits will provide the reasons for ISPs to deploy multicast. Aside from a simple gain analysis, arising applications or new, e.g., mobile services may stimulate appropriate business cases for multicast.

Recent advancements led IP multicast routing in divergent directions. Source Specific Multicast (SSM) [11, 46] broke with Deering's open host group model to achieve greatly simplified, domain-transparent routing. In contrast to Any Source Multicast (ASM), optimal multicast source trees are constructed immediately from subscriptions at the client side towards the multicast sender, without using network flooding or rendezvous points. Source addresses are to be acquired by out of band channels, limiting its applicability to service-aware parties. By this lack of generality, SSM remains unsuitable for self-configuration tasks of distributed systems. Moreover the single source model does not allow for state aggregation in shared trees, while the common PIM-SSM routing [39] uses Reverse Path Forwarding for Internet backbone traversal.

BGMP [95] at the Internet backbone attains a somewhat complementary role of Bidir-PIM by supporting bi-directional shared trees between domain-level rendezvous points, thereby overcoming limitations of scalability. However, BGMP continues to rely on route symmetry throughout the Internet backbone.

Only two years ago at SIGCOMM, Ratnasamy *et al.* [76] again urged for the adaptation of an any source multicast service on the IP layer. The authors propose BGP extensions to exchange group membership announcements decoupled from multicast route discovery. Routing follows a forward path approach achieved by a tree-based source routing on top of BGP. As BGP routing tables are unaware of global contexts, the authors need to encode the entire distribution tree within forwarded packets. While incorporating original, valuable ideas, this monolithic Free Riding Multicast (FRM) protocol suffers from the drawbacks of not only requiring a complete change of the BGP layer, but also placing the heavy burden of evaluating the distribution tree in the Internet core and performing correspondent source routing.

1.1.3 Mobile Multicast

Multicast mobility management has to accomplish two distinct tasks: handover operations for mobile listeners and senders. While many solutions exist for roaming receivers [81], [88], very few schemes have been specified for mobile multicast sources. Following a handover, multicast data reception can be fairly easily regained by a remote subscription approach [52], possibly expedited by agent-based proxy schemes [86]. In contrast, a multicast sender either defines the root of a source-specific shortest path tree (SPT), distributing data towards a rendezvous point or receivers, or it forwards data directly down a shared tree. Aside from tunneling or shared trees, forwarding along source-specific delivery trees will be bound to a topological network address due to reverse path forwarding (RPF) checks. At the same time a mobile sender must not change source address while re-associating in a different network, since addresses are associated on the application layer, e.g., with RTP media streams.

Within intra-domain multicast routing, the employment of shared trees may considerably relax mobility-related complexity. Relying upon a static rendezvous point, a mobile source may continuously submit data by encapsulating packets with its previous topologically correct or home source address. Constraints even diminish, when bi-directional PIM is used. Intra-domain mobility is transparently covered by bi-directional shared trees, which are built from a 'virtualized rendezvous point', eliminating the need for tunneling data to reach the rendezvous point.

However, issues arise in inter-domain multicast scenarios, whenever notification of source addresses is required between distributed instances of shared trees. Problems increase with Source Specific Multicast (SSM) operated on the IP-layer, as it requires active subscription to

contributing sources, thereby relying on topologically correct addresses. On the occurrence of handovers and in the presence of source filters, any mobile SSM routing protocol is required to transform a given source-specific (S_{old}, G) state into (S_{new}, G), while listening applications continue to receive multicast data streams admitting a persistent source address S.

Facing multicast deployment problems, it is desirable that any solution to mobile multicast should leave routing protocols unchanged. Mobility management in such deployment-friendly schemes should preferably be handled at the Internet edges, preserving the core routing infrastructure in mobility-agnostic condition. Facing the current state of proposals, the urgent search for such a simple, infrastructure-transparent solution remains, even though there are reasonable doubts about whether this can be achieved for SSM.

The search of a solution for a mobility agnostic multicast routing protocol was the original motivation for this thesis and directed us to overlay multicast.

1.2 Overlay Multicast

In recent years the Internet community experienced two significant disruptions. The advent and overwhelming success of Napster and successors from 1999 on demonstrated an imperative desire of Internet users to take advantage of transparent end-to-end application services. The Internet, originally designed as a logical end-to-end overlay on top of heterogeneous physical networks, apparently had failed to serve these needs in its current server-centric and NAT-burdened state of deployment.[1]

In the year 2001, when Napster failed legally and early versions of Gnutella broke down technically, proposals for using an abstract name space to combine nodes and content emerged, which organized within Distributed Hash Tables (DHTs). The introduced solutions admit the routing geometry of rings as in Chord [93], trees as in Tapestry [107] and Pastry [82] or a d-dimensional toroidal geometry as in CAN [77, 79]. Their common concept of distributed indexing by Plaxton *et al.* [72], which had been initially developed for distributed memory computer architectures, stimulated many of ideas and continues to inspire routing in Mobile Ad-hoc (MANET) networks [73], [106], as well as to heat up the debate on a *clean slate* reinvention of the Internet [37].

Structured peer-to-peer systems offer multicast services in an infrastructure-agnostic fashion. They are reasonably efficient and scale over a wide range of group sizes. However, they do not allow for layer 2 interactions and thus do not facilitate unrestricted scaling in shared end system domains. Stability issues for tree-based overlay multicast under churn arise as well, as the departure of branching nodes close to the root may have disastrous effects on data distribution. These drawbacks may be mitigated by hybrid approaches, where overlay multicast routing only takes place among selected nodes, which are particularly stable and form a virtual infrastructure. Similar initial propositions have recently been introduced to IRTF [16]. Such adaptive schemes of cooperative routing in underlay and overlay bear the potential to optimize stability and performance, while sustaining ample flexibility for deployment.

The performance gap between IP and application layer multicast widens, when mobility is introduced. Frequent handoffs and topological re-arrangements degrade the stability of distribution trees and the efficiency of proximity selection. Garyfalos & Almeroth derived from fairly

[1] Characteristically, an ongoing combat arose of P2P suppression on the infrastructure management side and barrier evasion on the application layer.

generic principles efficiency measures for source specific multicast in different metrics [40]. Overlay trees uniformly admitted degradations up to a factor of four over native IP layer multicast in the presence of MIPv6 mobility management. To overcome mobility obstacles, the authors introduce the Intelligent Gateway Multicast, which assists in reactive handovers at the network access.

1.3 Hybrid Multicast

Hybrid multicast schemes inherit major efficiency from the IP layer, while sustaining ease in deployment and infrastructure-transparency from selected group distribution in overlay networks. Such approaches differentiate the end-to-end design argument with respect to the inhomogeneous nature of the global Internet: While customer-oriented end system networks, which are mainly built on top of multicast enabled subnetwork technologies, do significantly profit of utilizing network layer multicast services, the flow-oriented transition networks of the Internet core do not.

The design of interconnecting end system domains on the basis of a structured overlay gives full multicast admission control to local operators and may be interpreted as globally distributed service peering. It will enable inter-domain distribution trees to multicast group services, which remain invisible to the Internet core, while inheriting the full potential of scalability, self-organization, redundancy and error resilience from the overlay network protocols in use.

The recently sketched Hybrid Shared Tree architecture [102] follows the lines of the evolutionary construction scheme of the Internet. Its focus originates from a customer network or an ISP domain, where multicast services are locally deployed. Multicast service exchange is then expected to be implemented like unicast peering, in a dedicated but isolated step. It will operate following the activation of a gateway service, which interconnects the local multicast routing with the distributed peering on the structured overlay. Note that a separation of inter-domain multicast from unicast routing will lead not only to a simplified, more stringently structured approach, but will also segregate malfunctions due to misconfiguration or component overload.[2]

1.4 Problem Statement and Approach to Solution

Currently, multicast is not globally deployed. The aim of this thesis is to provide new ideas and solutions in the context of hybrid group communication networking, which may enforce global multicast and thus help towards efficient data distribution within the Internet. In this thesis we are focusing on two main ingredients for hybrid multicast: An efficient large-scale overlay multicast protocol suitable for inter-provider as well as end system multicast, and an underlay-overlay aware group communication network stack.

Such a network stack will enable router vendors as well as application developers to implement multicast transparent group services. Software equipped with this stack allows end users to build up anywhere widely spread multicast groups without dedicated infrastructure support. Additionally, a relay service in concordance with the hybrid shared tree architecture should be designed. This component forwards data automatically between underlay and overlay multicast

[2] Caused by experiences with early PIM-SM implementations, there is a common fear of multicast to degrade the unicast forwarding.

networks. With respect to an easy deployment, it accounts for different multicast scenarios on the one hand, and current Internet standards on the other hand.

Bridging the gap of the inter-domain multicast deployment problem using overlay multicast imposes special demands for the protocol in use. First and foremost the protocol needs to balance replication load between data forwarders with respect to peering fairness. Assuming IPTV scenarios with provider centered sources, large receiver groups for one TV channel occur easily when public attractive events, e.g., soccer world cup, will be broadcasted. A single overloaded peer results in a bottleneck. Interferences will generate higher jitters or packet loss. Both affect video compression schemes and thus annoy users.

Providers possess a strong interest in calculating load on network components before rollout. Further on, the performance of the overlay protocol should be predictable, such that hybrid infrastructure components can grow with upcoming requirements. For an inter-provider deployment it needs to scale with very large groups.

In this thesis we present two group communication protocols for overlay networks. The first one is prefix flooding, a simple broadcasting mechanism that operates in the prefix space of structured overlay networks without signaling. Starting from simple models of recursive k-ary trees, we analytically derive distributions of hop counts and the replication load. Extensive simulation results confirm our observations.

The second protocol we introduce is BIDIR-SAM: The first multicast routing protocol for structured overlay networks operating on a bi-directional shared prefix tree, which permits source-specific paths. Optimized tree construction and data transmission throughout the underlay are key controls for efficient group communication in structured overlay networks. Our prefix-guided group management strictly adheres to forward-directed establishment of distribution trees. BIDIR-SAM omits any dedicated infrastructure component like a rendezvous point. It exhibits strictly predictable logarithmic costs and is best-suited for large-scale group communication.

Comparisons of the prefix flooding and BIDIR-SAM are drawn to Scribe, taken as a general reference model for group communication according to the shared, rendezvous-point-centered distribution paradigm.

The remainder of this thesis is organized as follows: In chapter 2 we discuss basic related technologies. Chapter 3 introduces and analyzes prefix flooding, a broadcast scheme for structured overlay networks. Our work extends the results of Castro *et al.* [23] by a generalization of the protocol and its theoretical prove as well as by a thoroughly analytical study and simulations. We introduce and evaluate our overlay multicast routing protocol, BIDIR-SAM, in chapter 4. The evaluation is based on a very detailed theoretical analysis and simulations. In chapter 5 we explain our overlay group communication architecture accommodating the prefix flooding and BIDIR-SAM. This architecture includes the design of the hybrid shared tree components and the middleware for structured P2P group communication. We present our protocol implementations within a P2P simulator in chapter 6. Chapter 7 summarizes our results and gives an outlook for future directions.

2 Related Technologies

2.1 Structured Overlay Networks

Distributed Hash Tables

Locating resources is a common task in computer networks and distributed systems, and the motivation why name services like the DNS have been foreseen. Resources, i.e., data or information, may be arranged directly in a decentralized store, but also indirectly via pointers indicating their location. Based on a unique resource identifier, the device holding the data then needs to be resolved. For this reason, network members have to provision a mapping between device and resource descriptor. The simplest implementations of such a search algorithm either flood the information within the entire network or save the tuple on a centralized server. Both approaches do not scale with the number of nodes. Flooding allows to minimize memory requirements, but the resulting communication costs increase at least linearly. Centralized approaches operate oppositely. Therefore, a common solution in distributed systems is the pre-structuring and dissemination of the required information, such that they can be located in logarithmic time.

Distributed Hash Tables (DHTs) implement efficient data storage over multiple nodes on the one hand, and its fast localization on the other hand. The most prominent examples are Chord [93], Pastry [82] and CAN [77, 79]. The common concept of DHTs comes from distributed indexing inherited from distributed memory architectures in the 80ies. Commonly, memory consumption and lookup complexity range in the order of $O(\log N)$ with N number of nodes. Each node within a DHT maintains information corresponding to a resource descriptor. Determining the storage location is based on the following idea: A DHT node and a resource will be assigned to an address from the same (ordered) identifier space. The DHT node with an adjacent address to the resource ID is responsible for the stored data. Adjacency is defined on a DHT specific metric. The node is called the *key's root*. Consequently, the destination for a resource request can be calculated based on the descriptor ID.

Internet devices have an address, and application data descriptors are decoupled from host identifiers. To fulfill the DHT requirement of an identical address space, both, the original resource name and a unique host token will be mapped to a set of keys via the same hash function. Thus, network nodes participating in a DHT obtain an additional (hash) address, which represents an ID on the application layer.

To locate distributed keys, the DHT creates an abstract structure independent of the underlying network. This network will be used to forward a key request. The DHT connects at least two nodes to application layer paths based on their DHT identifier. The actual topology depends on the individual DHT. A network arises on top of the underlying layers, an *overlay* network.

Routing tables will be calculated for the logical network. Overlay DHT nodes, thus, own either only information to forward keys towards other nodes or additionally a mapping between resource key and data item.

Hence, a DHT represents a data structure storing distributed keys and includes simultaneously an algorithm, which implements forwarding between the nodes of the DHT to lookup the key. A DHT is supplemented by a communication protocol, which maintains the data structure and exchanges the required routing information.

Key-based Routing

Key-based Routing (KBR) delivers a message destined for a given key to a the key's root. In contrast to DHTs, the routing request does not need to result in the retrieval of a stored key-value-pair, but only to reach the key's root. The KBR, thus, implements a more generic message routing and passes on the conceptual distinction between (resource) key and overlay (node) ID. However, as discussed above, each DHT implements inherently a key-based routing scheme. Colloquially, the terms DHT and KBR are often used interchangeably.

A KBR layer provides applications with a structured overlay on top of a network layer protocol. As no overlay node plays a distinct role, the equal members will be called *peers*. Each peer owns a key as overlay address. The forwarded messages are application specific. In contrast to network layer routing, the KBR scheme guarantees, that each address from the identifier space is associated with a peer according to the scheme of locating the key's root in DHTs.

In the current Internet, KBR approaches are deployed on top of an existing global underlay routing. In general, there is no direct correspondence between the node's location in the underlying topology and its placement within the overlay, the latter following the properties of hash functions. Hence, peers, which are far away in the overlay, may be close to each other in the real network, and the other way around. A KBR scheme not considering this case may induce two penalties: On the one hand, it may imposes delays as unnecessary hops are taken with respect to the native routing. On the other hand, it may stress native network connections with needless transmissions of the same data over the same link based on cross-routing between peers. This disadvantages grow worse in group communication scenarios, as packets have to be distributed to multiple destinations.

However, there are approaches, which account for properties of the underlying network using the flexibility of key-based identifiers. In the following we will discuss a KBR/DHT substrate implementing *proximity neighbor selection* in its forwarding rules.

2.2 Pastry

In contrast to underlay unaware DHTs, Pastry [82] accounts for the distance of two nodes in the physical topology while constructing its overlay routing table. This distance can be evaluated according to different metrics, such as RTT, hops or packet loss. Thus, a peer does not only forward a message to an adjacent node in the logical network, but also selects the peer with the minimal delay with respect to the physical topology. For this purpose, a Pastry node maintains a prefix-based routing table. Each prefix covers a set of keys. Thus, multiple overlay nodes potentially exist as destinations for message forwarding.

The main concept of Pastry provides a peer with an aggregated view of the overlay based on prefixes, which are used for message routing. As visualized in figure 2.1, peers can be connected by a prefix tree. This structure is constructed by labelling recursively inner vertices with the longest common prefix of their children. Leaves represent overlay nodes. Inner vertices define

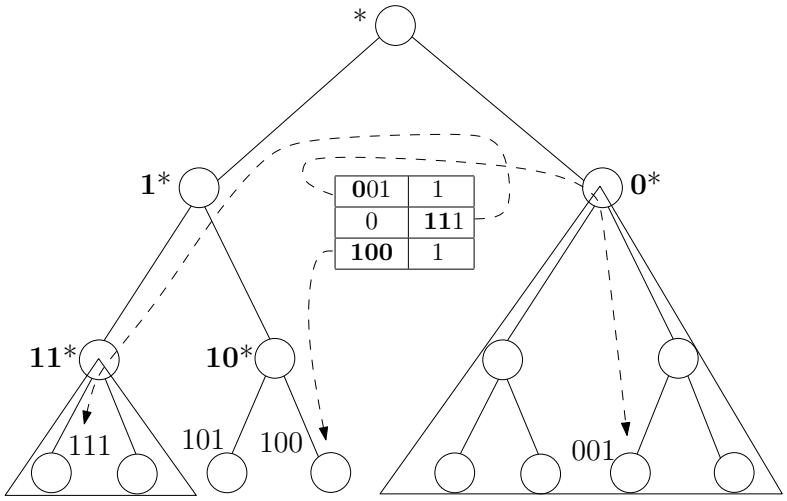


Figure 2.1: Routing Table for Peer 101 and the Corresponding Spanning Prefix Tree Using a Binary Key Space. Next Hop Pointers are Highlighted by Dashed Lines.

the root of prefix subtrees. Forwarding can be performed along the following idea: A node sends a message to the subtree root, which shares digits with the destination. The message will be intercepted by a member of this subtree and internally be passed towards the key's root. Thus, a peer only needs to store one vertex of each subtree within its routing table (cf. figure 2.1). The selection process follows Pastry's proximity metric.

In the following, we will describe the routing table and forwarding algorithm of Pastry in detail and more formal way.

To gain insight into the Pastry routing, we first clarify the routing table structure. Let k denote the base of the overlay identifier space $\{\mathcal{K}_n\}$, and h denotes the maximal number of digits for an identifier. Additionally, we denote $prefix(m, \mathcal{K})$ as a function mapping the first m digits of a value $\mathcal{K}$ with $prefix(m, *) := NULL$. Then a Pastry routing table consists of at most h rows and k columns. Each cell in the i-th row and j-th column ($1 \leq i \leq h,\ 1 \leq j \leq k$) represents the following semantic: Based on a transformation $T(i, j)$, the table index approximates a Pastry overlay key. The table entry contains the corresponding underlay address. The index transformation creates prefixes with respect to the overlay node identifier. For a peer with ID $\mathcal{K} \in \{\mathcal{K}_n\}$, the transformation is given by:

$$T(i, j) = prefix(i - 1, \mathcal{K}) \cdot j_k, \text{ with } z \cdot j_k := \ z \text{ concatenated with } j \text{ in base } k. \tag{2.1}$$

Thus, each column of a row i corresponds to the first $i-1$ digits of the peer's overlay identifier. Further on, for each row there exists a column l, which equals the first i digits of the peers ID ($T(i, l) = prefix(i, \mathcal{K})$). This cell can be ignored, as it references the node itself. Based on the calculation rule (2.1), each entry can be accessed in constant time. The routing table includes $O(k \cdot \log_k N)$ entries asymptotically.

The routing table entries are selected according to the following scheme: For a prefix $\mathcal{P}$ with length $|\mathcal{P}|$, all overlay members carrying an ID $\mathcal{L}$ are available if:

$$prefix(|\mathcal{P}|, \mathcal{L}) = \mathcal{P}, \text{ with } \mathcal{L} \in \{\mathcal{K}_n\}.$$

Pastry chooses the overlay member with the minimal underlay latency to the peer. This process is called proximity neighbor selection and seamlessly adapts to underlay performance without interfering with the key based routing logic.

Figure 2.1 visualizes a sample routing table of node 101 for $k = 2$ and $h = 3$.[1] The first column of the first row represents prefix 0*, column two refers to the node itself. As next hop for all keys starting with 0, Pastry node 101 has chosen peer 001. Entries in the second row conforms to the second digit of an overlay key. Iterating over the prefix alphabet, but fixing the first digit, the first column is pointing to 101 itself and the second column includes a peer with ID 11*. Hence, the last row contains all direct neighbors.

Whenever a Pastry node receives a routing request, it selects the table cell with a prefix that is at least one digit longer than the prefix that the key shares with the peer's ID (cf. rule (2.1)). If no such node is found, the message is forwarded to a node whose overlay ID shares a prefix with the key as long as the current node, but is numerically closer to the key than the peer's ID. Thus, in each routing step the key will be forwarded to a Pastry node, whose identifier is closer to the key. The routing algorithm terminates when the key's root has been found.

To enhance the efficiency of the routing scheme, each Pastry node additionally maintains a *Leaf Set*. This includes a constant number of direct neighbors, which are located to the right and left of the peers node ID. Hence, a larger leaf set results in a higher rate of direct peer access.

The Pastry routing table maintenance is improved by a neighborhood set. In contrast to the leaf set, this includes overlay members close to the peer with respect to the underlay proximity metric. The set is not used for the message forwarding itself, but to feed the routing table.

2.3 Scribe

Scribe implements an overlay multicast scheme on top of Pastry [83, 22]. Pastry is used as underlying (unicast) key-based routing, guiding a proximity-aware forwarding. Scribe operates on a multicast distribution tree, which is constructed in correspondence to the shared tree approach of the native multicast routing protocol PIM-SM [39]. Multicast sources send their data to a dedicated rendezvous point, the root of the distribution tree (cf. figure 2.2. This root node is directly derived from the multicast group ID. Forwarding states are stored in a multicast children table in parallel to Pastry's routing table.

Scribe operates in detail as follows: A multicast source creating a group delegates a *create* message including security credentials to the Pastry instance. The message is destined to the overlay multicast group address, i.e., an identifier created by the KBR hash function. Pastry initiates regular overlay routing towards the numerical closest peer. The Pastry instance of the key root forwards the message towards Scribe, which checks the credentials. On success, the corresponding group will be activated and the peer acts as rendezvous point (RP): Each overlay node is now enabled to send its multicast data to the RP, which distributes the data, if the credentials hold.

The multicast distribution tree will be erected by receiver subscriptions. An overlay multicast listener sends a *join* message to the known multicast group address based on Pastry routing. Overlay nodes on the route from the receiver to the rendezvous point intercept and forward this message to the local Scribe instance. The Scribe instance adds the previous hop, i.e., its

[1] For simplification we store the overlay ID instead of the IP address.

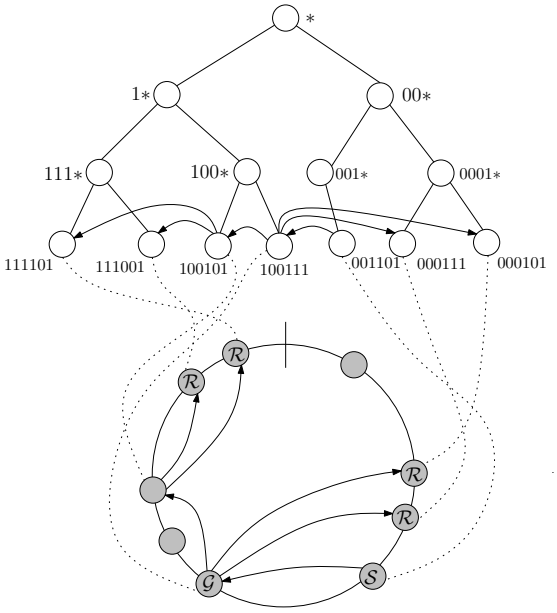


Figure 2.2: A Source $\mathcal{S}$ Distributes Data to $\mathcal{G}+1$. The Key's Root $\mathcal{G}$ Forwards the Data Towards all Receivers $\mathcal{R}$.

overlay key and IP address, to the local multicast children table, if there is not already an entry for it. The message will be passed back to Pastry only if no previous group member has been identified in the table. Hence, a path from a receiver to the RP will subsequently be built by children pointers. Branching occurs whenever multiple joins for one group arrive at the same peer. Henceforth, *publish* messages can be distributed via the rendezvous point towards the receivers by forwarding data based on the children table entries.

The multicast tree will be deconstructed using the Scribe *leave* message. To leave a group a peer sends this state update towards the rendezvous point. Intermediate forwarders distribute the leave, if there are no other entries in the corresponding children table. Thus, multicast branches will be pruned.

2.4 The Hybrid Shared Tree Architecture

In this section we will introduce our previous work, the Hybrid Shared Tree (HST) [102], a hybrid architecture designed to enable global multicast peering at the ISP or enterprise level, while sustaining end system transparency in utilizing well-established group distribution services.

The basic concept of HST preserves multicast routing and lower layer packet transmission within local multicast domains, while bridging the inter-domain gap with the help of a structured overlay network to overcome the multicast deployment problems [31].

In combining a well established DHT with a new overlay multicast routing scheme, this approach addresses in particular the following design objectives:

- Provide scalability, robustness and inter-domain transparency for shared distribution trees;
- Detach multicast routing from the Internet core and restrict the backbone infrastructure to plain unicast forward routing;

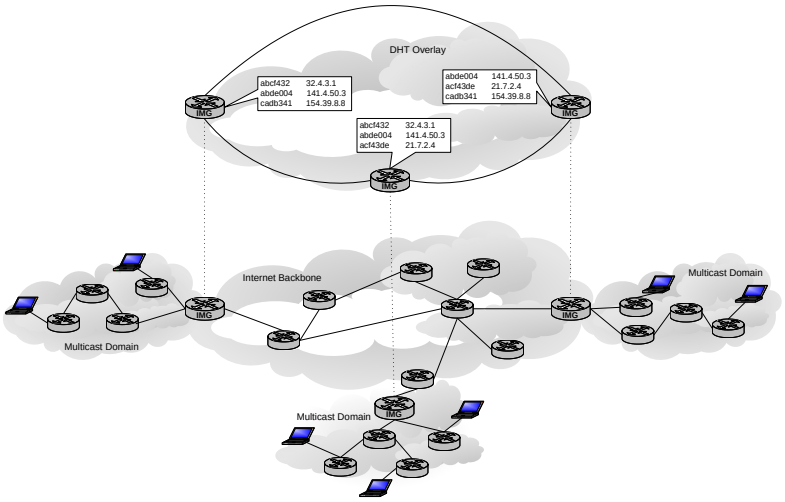


Figure 2.3: The Hybrid Shared Tree Architecture

- Decouple group membership registration from route discovery;
- Decouple multicast state management from the data plane;
- Grant control on group admission to local operators;
- Open a lightweight deployment perspective for mobile multicast services.

Architectural Overview

The Hybrid Shared Tree architecture introduces the Inter-domain Multicast Gateway (IMG) as a new architectural entity, which provides a gateway functionality between the overlay it is a member of, and the multicast routing at the intra-domain underlay that it resides in (cf. figure 2.3). Those gateways will participate in multicast traffic originating from its residential network, which it will forward into the overlay according to the distributed multicast receiver domains of this group, and will also advertise group membership and receive data according to any subscription from its domain. On the overlay, the IMGs will jointly operate a bi-directional overlay multicast routing scheme, which we will present in detail in section 4.2.

The IMG may be positioned anywhere within the multicast domain, but need to provide a protocol interface to the locally deployed multicast routing. To avoid zigzag transmission, the IMG may be situated at the domain border router, though, in the example of a PIM-SM or Bidir-PIM domain, the IMG could also be co-located with the rendezvous point or the rendezvous address. Note that the IMG function may either be built as a dedicated system entity or may consist of an additional intelligence on existing routers.

Activation of inter-domain multicast gateway services requires only a small amount of selected information for bootstrapping, i.e., an arbitrary contact member of the structured overlay, and authentication and authorization credentials, if applicable. The IMG remains under the administrative control of the local network operator, who may restrict admission, scoping and QoS characteristics of the group traffic flowing in and out of the intra-domain. Aside from general multicast peering policies, a service provider is thus enabled to implement firewall-type of packet filters at, or co-located with, these multicast gateways.

This architecture supports flexible operations in several ways. A domain operator is enabled to connect to several multicast overlays in parallel, may choose to replicate IMGs for load balancing or redundancy purposes or may transparently take advantage of the fail-safe unicast peering realized by multi-homed network connectivity. Replication operations will be seamlessly empowered by the self-organization capabilities of the DHT overlay.

A Note on Tunneling

Multicast islands may also be interconnected by underlay tunnels instead of using a structured overlay. Tunnels between multicast listener and sender domains can be created manually or automatically [94]. The first approach does not grant a flexible solution as self-organization remains unsupported. Like in the early Mbone, tunnels are set up manually by operators and do not adapt to user requirements nor network changes [67]. The second case, however, induces further deployment issues as it is based on an anycast discovery mechanism. Anycast paths are not widely available in the current Internet. In contrast to these properties, overlay multicast operates in a self-organizing way, without the need of special requirements provided by the underlying network.

A tunnel-based infrastructure encapsulates data of a multicast source domain and delivers it via unicast to receiver domains. In this sense, the hybrid shared tree architecture operates similar to a dynamic tunneling using application layer encapsulation. However, a tunnel approach does not include branching and imposes a significant replication and state load on tunnel entries [50]. Each tunnel entry feeds all receiver domains of a multicast group, building a corresponding *point-to-point* mesh. Overlay multicast pursues an efficient distribution strategy starting at the source domain. The hybrid shared tree architecture, thus, includes a scalable solution for the inter-domain gap.

3 Broadcast in DHTs

3.1 Introduction

Distributed Hash Tables (DHT) define an overlay and enable a routing layer thereon, which scales logarithmically in both, memory requirements and forwarding path lengths.[1] DHTs like Chord and Pastry have not foreseen any broadcast mechanism, i.e., a mechanism to reach all participants of one DHT instance without listener participation.

The broadcast mode admits two unique features. The a priori awareness of the data flooding task may significantly enhance efficiency, e.g., by taking advantage of network or (shared) media specifics. Further on it enables a message exchange among mutually unknown parties without a requirement of specific service awareness or any form of signaling. Broadcast is thus the fundamental mechanism for unselective data synchronization and for the autonomous coordination of distributed systems.

On the application layer, there are likewise versatile use cases for broadcast communication. Applications range from broadband data dissemination in video conferencing or data replication, over the area of service and peer discovery up to the implementation of a virtual link layer in VPN-type solutions.

Broadcast is a special case of multicast. This distribution mechanism guarantees to reach not only a subset of nodes, but all nodes of a dedicated domain without explicit registration. The set of *all nodes* is also called the broadcast domain. It is worth noting that a broadcast domain can be arranged on different layers with varying inherent capabilities. Connecting nodes, e.g., with an Ethernet hub to a shared segment facilitates packet distribution based on the physical network structure. It is limited by the supporting medium, i.e., the range of signal propagation which may be repeated. The equivalent holds for the wireless domain, where the medium is always shared, but of restrictive propagation ranges. Participating nodes do not need a specific network logic in sending and receiving broadcast data on the physical layer. Broadcast support, however, on a dedicated layer should be independent of the underlying tier, which may accelerate it. In the example of IP, broadcast addresses will be directly mapped to the Ethernet broadcast address, such that all Ethernet hosts of one segment receive the data independent of their subnet assignment, but in contrast to network access, packets can be forwarded on the network layer beyond physical bounds.

Broadcast support is commonly provided on the network and data link layer. Analog to the realization of IP layer broadcast, application layer broadcast can be mapped to the IP broadcast. Listing on a predefined port then defines membership of the broadcast domain, i.e., unsolicited reception of data on the application layer. Obviously, on the one hand the dimension of the broadcast domain is limited by the subnet, because IP layer broadcast will not be routed in general. On the other hand, all nodes of the IP subnet receive the packets, regardless if they are running the specific application, which may not only cause unnecessary replication

[1] For the sake of accuracy it should be remarked that there are DHTs like CAN with different scaling behavior.

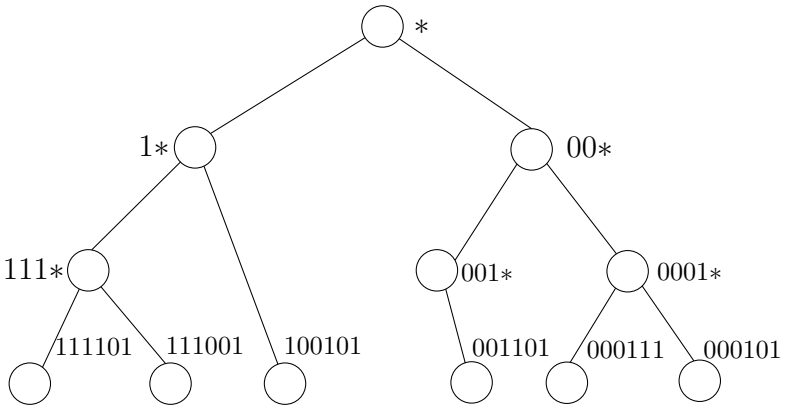


Figure 3.1: A Binary Prefix Tree

load on bridges and switches, but also a compatibility and security problem, whenever another application uses the same port. As a consequence, the flooding on the application layer should remain restricted to de facto domain members if possible.

In general, broadcast in logical networks can be enabled by passing data incrementally to direct overlay neighbors. If the graph of nodes is connected and contains the source, all nodes will be reached. DHT structures allow to derive such a connected neighborhood graph. Any node can send packets to an address adjacent to its own key space. In contrast to IP, every possible address is associated with one overlay peer. Such a simple ring broadcast scheme sends the packet to exactly one neighbor. Thus, the distribution degenerates to a linear chain. The broadcast arrives at all n DHT peers after n hops. The neglected parallelism can be improved by simply sending the data not only along one direction of the ring address space, but also counter clockwise, which will halve the time complexity. Asymptotically, the complexity remains linear.

As an alternative approach to the case of unknown neighborhoods, a dedicated, well-known replicator can be placed in the network like the Broadcast and Unknown Server in ATM. Such a rendezvous point-based approach requires extra signaling to register receivers. The parallelism of distribution is bounded by the replicator, which sustains the overall duplication load and may be a single point of failure.

In the following, we will present a general broadcast algorithm along with optimizations for Pastry, that uses the DHT structure more efficiently and replicates data stepwise to all neighbors in prefix space. This scheme works without peer involvement, especially without signaling. In contrast to the ring broadcast, the neighbor set is derived from the prefix relationship of the source to the other peers, increasing parallelism in data distribution efficiently, but still staying within a fairly bounded replication load per peer. We model and analyze the approach theoretically and in simulations, drawing comparison to a generic rendezvous point approach derived from Scribe [22].

3.2 A Prefix Flooding Approach

Efficient broadcasting on the application layer needs a strategy for data replication on the overlay. In a DHT the peer identifiers are composed of an alphabet with k digits and have a predefined length. All nodes of a structured overlay can be naturally arranged in a prefix tree, branching recursively at longest common prefix of k neighboring vertices. The leaves are labeled

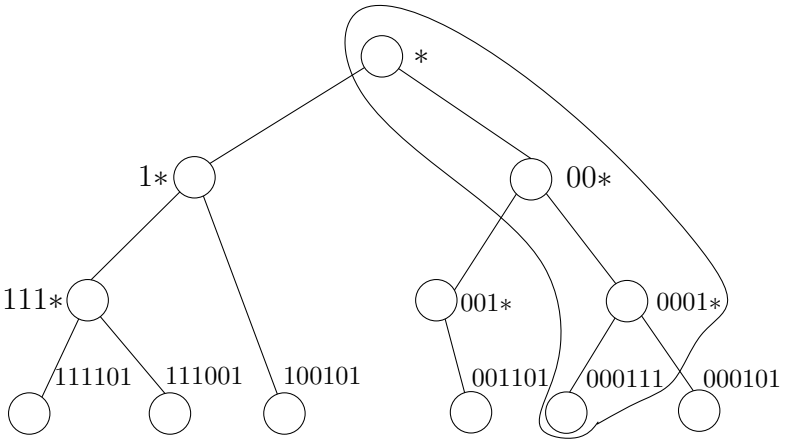


Figure 3.2: DHT Node Placement in a Prefix Tree – All Vertices Associated with Node 000111 are Highlighted. Adjacent Vertices Represent Prefix Neighbors.

with the overlay identifiers of the DHT members and the inner vertices represent the shared prefix (cf. figure 3.1).

This tree can be interpreted as a distribution tree, defining the broadcast domain of a specific DHT instance. If a broadcast packet is sent starting from the root of the tree towards the leaves, the packet will be replicated where prefixes branch. Actually, the broadcast domain (prefix tree) decomposes in many smaller broadcast sub-domains (subtrees), in which the propagations continue in parallel. Following the nature of broadcast, a packet will be forwarded locally, after it has arrived at a root of a subtree.

This approach allows to reach all peers of a DHT, because the data is flooded to the leaves, which represent the overlay nodes. To decide on packet replication, a peer receiving a broadcast is required to determine the current branching position on the distribution tree. This context awareness can be achieved by sending broadcast packets carrying the prefix currently addressed, which we call *destination prefix*. This destination prefix will grow in length with every forwarding hop while descending the tree.

We denote the length of a prefix $\mathcal{A}$ by $|\mathcal{A}|$. Given two prefixes $\mathcal{A}$ and $\mathcal{B}$, the longest common prefix will be written $\mathcal{L} = LCP(\mathcal{A}, \mathcal{B})$. The relation of $\mathcal{L}$ being a prefix of $\mathcal{A}$ is written as $\mathcal{L} \subseteq \mathcal{A}$. Consequently $\mathcal{L} \subseteq \mathcal{A}$ and $\mathcal{A} \subseteq \mathcal{L}$ if and only if $\mathcal{L} = \mathcal{A}$.

A proper specification for data distribution, i.e., routing procedure on prefix trees, requires further definitions. The two sub-problems that need to be solved are a *routing to a prefix* and the *association of nodes with prefixes*:

Definition 3.1 *A prefix $\mathcal{L}$ is associated with an overlay node of ID $\mathcal{N}$, if and only if $\mathcal{L} \subseteq \mathcal{N}$.*

As shown in figure 3.2, all inner vertices on the shortest path from the root to a node are associated with that node.

Concordantly, a *prefix routing* can be defined as forwarding a packet to the node the destination prefix is associated with. In general, there may be several nodes owning an associated prefix, since prefix-to-node mapping is only assured to be unique for prefixes of full key length.

Taking the shortest path from an arbitrary leave $\mathcal{K}$ to the root of the tree, all intermediate vertices $\mathcal{V}_1, \ldots, \mathcal{V}_n$ are labeled with a prefix of $\mathcal{K}$. They are associated with $\mathcal{K}$. If the routing entries $\mathcal{V}_{\{1,\ldots,n\}}$ of $\mathcal{K}$ do not point to $\mathcal{K}$ itself, the routing along the prefix tree can be obviously

inefficient. Suppose the case in which $\mathcal{V}_{\{1,\ldots,n\}}$ refer to $\mathcal{K}$'s overlay neighbor $\mathcal{K}'$ and the corresponding entries of $\mathcal{K}'$ point back, then the replication on the path from the root to $\mathcal{K}$ alternates between $\mathcal{K}$ and $\mathcal{K}'$ until the tree is descended. For this reason and for the sake of simplicity, the concept of prefix flooding requires at least that associated prefixes of a peer do not refer to another overlay node in the local routing table, but to itself.

For flooding a prefix tree, a forwarding peer needs to route packets to all neighboring prefixes (cf. figure 3.2). Consequently a peer must store corresponding nodes for each prefix adjacent to its associated vertices in a prefix neighbor set. Even though details of neighbor set maintenance belong to the underlying DHT, it is important that these tables are complete.[2] A *complete neighbor set* meets the following condition: Whenever an overlay node exists for a given prefix, then the neighbor set will provide an entry for this prefix. In particular it follows that each overlay node is a destination in at least one set, since node keys are uniquely assigned.

A source initiates a broadcast by starting with the empty destination prefix. This corresponds to delivering the data to all prefix neighbors $\mathcal{N}_i$. At each neighbor a packet will be further replicated. The destination prefix is replaced with the new target address. In detail, the algorithm works as follows:

PREFIX FLOODING

```
   ▷ On arrival of a packet with destination prefix C at a DHT node
1  for all N_i IDs in prefix neighbor set
2        do if LCP(C,N_i) = C          ▷ N_i is downtree neighbor
3              then C_new ← N_i
4                   FORWARD PACKET TO C_new
```

If all peers have a complete set of prefix neighbors, the scheme guarantees that all overlay nodes will be accessed, no peer receives a broadcast packet more than once and the algorithm terminates.

Theorem 3.1 (Coverage) *If the prefix neighbor sets are complete at all nodes, then the* PREFIX FLOODING *assures packet distribution to all overlay nodes.*

Proof by induction. We assume that the prefix neighbor sets $\{\mathcal{N}_i\}$ are complete and correct on all peers. Induction is done with respect to the number of overlay nodes n.

Base case: We consider an overlay with $n = 2$ nodes, a source and one destination. Both vertices are then adjacent to each other. The source sends the broadcast directly to its neighbor in the first step where $|\mathcal{C}| \geq 0$.

Induction step: Assume that the PREFIX FLOODING covers all overlays with n nodes. We have to show covering holds for $n+1$. We number the peers from $1, \ldots, n+1$ arbitrarily. According to the induction hypothesis, the nodes $1, \ldots, n$ receive the broadcast. Based on the assumption of complete neighbor sets, there exist one or several nodes holding node $n+1$ in their neighbor sets. Among those nodes select j, $1 \leq j \leq n$, such that node j shares the longest prefix $\mathcal{C}'$ among all neighbors with node $n+1$. As j received the broadcast, the node has selected $n+1$ as one $\mathcal{N}_i$ corresponding to line 1 of the PREFIX FLOODING.

[2] This is a realistic assumption as some DHTs include routing maintenance schemes. If such mechanisms are missing, they can easily be implemented.

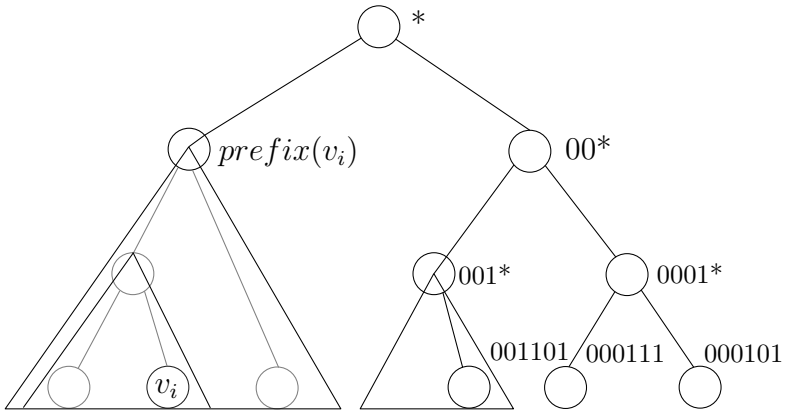


Figure 3.3: A Prefix Tree Decomposes into Self-similar Subtrees Under Routing Initiated from Source 000101

The destination prefix $\mathcal{C}_j$ carried by the packet at j now may be equal, shorter or longer than $\mathcal{C}'$. If $\mathcal{C}_j \subseteq \mathcal{C}'$, the condition in line 2 of the algorithm holds and the node $n+1$ receives the broadcast (line 4).

Otherwise, if $\mathcal{C}_j \supseteq \mathcal{C}'$, $\mathcal{C}_j \neq \mathcal{C}'$ the prefix $\mathcal{C}'$ must have been a destination prefix in a previous hop on the routing path to node j, say at node k. Then node k shares the same longest common prefix $\mathcal{C}'$ with the $(n+1)$-th node. Thus the destination prefix of this hop $C_k = C' \supseteq C'$ and node $n+1$ is flooded via the neighbor set of node k. □

Theorem 3.2 (Uniqueness) *Each overlay node will receive a broadcast packet at most once using the* Prefix Flooding.

Proof. Each overlay node forwards a broadcast only in the range of the destination prefix, which follows from line 1 and 2 of the Prefix Flooding. Let $\mathcal{N}_i$ be a valid destination prefix. Then there exists only one overlay node v receiving the broadcast for $\mathcal{N}_i$ due to the uniqueness of the set of prefix neighbors. Corresponding to line 3 and 4, $\mathcal{N}_i$ defines a new destination prefix $\mathcal{C}_{new}$, which extends the digits of $\mathcal{C}$, i.e., $\mathcal{C} \subseteq \mathcal{C}_{new}$ and $|\mathcal{C}| < |\mathcal{C}_{new}|$. $\mathcal{C}_{new}$ can be interpreted as the root of a new self-similar subtree (cf. figure 3.3) including a subset of the leaves $\mathcal{L}_j$. Thus, it is proved that a node will receive a packet at a given prefix length only once. It remains to show that broadcast packets do not cross between disjoint subtrees. However, this follows directly from the observation, that the current prefix $\mathcal{C}_{new}$ cannot be matched outside the subtree it defines. Consequently the condition in line 2 will always fail. □

From theorem 3.2 it follows that the Prefix Flooding does not induce loops, proving the assumption that the algorithm terminates.

Implementation for Pastry

The idea of prefix routing is implemented in Pastry. The Pastry routing table of a peer reflects directly the elements of a prefix tree (cf. section 2.2). Every row in a Pastry routing table is related to a level of the prefix tree and every column represents a child of an inner vertex. Each peer carries a subset of the prefix tree in its routing table. Merging the routing tables of all

peers, would form the global distribution tree. Pastry peers flood their routing tables. Thereby they flood the prefix tree, which corresponds to the overlay broadcast described by the PREFIX FLOODING. In detail, the idea is as follows: A source sends its data to all routing table entries. Each destination prefix corresponds to the root of a broadcast sub-domain. The receiving peers determine their position in the tree, i.e., the height D in the prefix tree, at which they receive the data, and forward the packets downwards. This is equal to sending data to all routing table entries starting at row $D+1$. Note that the tree position can easily be derived by denoting the row number, which reduces the packet size in contrast to encoding the entire key. For Pastry the PREFIX FLOODING reads in pseudo code:

PASTRY PREFIX FLOODING

▷ On arrival of a packet with destination prefix length D
▷ at Pastry node of ID $\mathcal{K}$ with routing table A containing l rows and k columns

```
1  for all i ← D + 1 to l
2      do for all j ← 1 to k
3          do if a_{i,j} ≠ Unspecified ∧ a_{i,j} ≠ K
4              then D_new ← i
5                   FORWARD PACKET TO a_{i,j}
```

If the routing table is filled correctly, all theorems for the PREFIX FLOODING are also valid for Pastry, since the Pastry routing table corresponds to the set of prefix neighbors $\{\mathcal{N}_i\}$. Even though Pastry guarantees that each overlay node will be covered, the routing table may not be complete [82], which conflicts with the PREFIX FLOODING. To fix this issue, Pastry can be augmented with a proactive routing maintenance mechanism as described in section 6.

It is worth noting that the PREFIX FLOODING approach is applicable to arbitrary DHTs. If the DHT does not support forwarding on prefixes inherently, a supplementary prefix routing table can always be constructed on top of the DHT. Obviously this may result in additional communication overhead, because required information cannot be inherited from the DHT directly, but must be acquired by usual DHT lookups.

3.3 Performance Analysis

The prefix flooding approach to broadcasting introduces prefix trees as a control plane to packet forwarding. This simple mechanism operates without additional signaling, which is an apparent advantage. The quality of the routing as inherited from a hash-generated prefix tree needs closer inspection. Ideally, packet distribution should be fast and minimize traffic and replication load in the network. To obtain an overall insight into the routing quality, we evaluate the prefix flooding scheme according to the following metrics and compare our results to Scribe [22], which serves as a generic reference model based on the same DHT, Pastry.

Traffic load measures the mean UDP traffic per peer generated during the simulation. All application layer packets will be encapsulated in UDP. To eliminate the base load, i.e., data appearing in both schemes like the usual Pastry maintenance, the relative traffic load per peer is calculated as well.

Packet replication load quantifies the number of packets a single peer has to forward. This metric reflects the number of direct neighbors per node in the distribution tree. The overall

characteristic for the prefix routing is then given by the distribution of the replication load obtained from all forwarding nodes.

Travel time describes the time a data packet travels from the source until it reaches a receiver measured in seconds. This absolute value depends on the one hand on the number on hops between the nodes and on the other hand of the transmission time inherited from the hop by hop link delays and the packet size of the transmitted data.

Relative delay penalty measures the ratio of the travel time for data packets delivered via Scribe and the travel time resulting from the prefix flooding scheme. This relative factor gives an indication of the parallelism of packet forwarding.

Hop count counts the number of overlay routing traversals that a packet needs on its way from the source to the destination. Note, that the hop count affects the travel time, because every additional hop results directly in an additional transmission time. In this sense the travel time is correlated with the hop count.

3.3.1 Analytical Results

To understand the performance of the prefix flooding scheme, we first present analytical considerations. Based on the shape of the prefix tree, we gain insight in the structural behavior of protocols for traversing prefix distribution trees. As this analysis is only based on the tree itself, fringe effects known from simulations are isolated. Actually, simulation results depend always on the structural phenomena of the studied mechanisms. The theoretical analysis can therefore be used to verify the outcome of our simulations (cf. 3.3.2) and to explain the measurements on a firmer basis.

Replication Load

In the following we want to derive the distribution of the replication load in a prefix tree. For the general case of prefix flooding in a structured overlay of N nodes using a prefix alphabet of k digits, the following upper bound of the replication load can be derived immediately.

Theorem 3.3 *Any overlay node in a prefix flooding domain of N receivers and an alphabet with $k \geq 2$ digits will replicate a data packet at most $\log_2(N)(k-1)$ times.*

Proof. Packet replication is performed to distribute data to all neighbors. It will be shown that the number of possible neighbors of a node falls below the claimed bound.

Any overlay node is situated as a leaf in the prefix tree and has all vertices on the shortest path to the root associated with it. Thus the number of neighbors equals the sum of the neighbors at each associated vertex. For an alphabet of k digits the latter is bound by $k-1$. The number of vertices towards the tree root is limited by the height of the path compressed tree, which is maximal if all branches are binary. Consequently a prefix tree with N leaves has a maximal height of $\log_2(N)$. Combining both estimates, a node cannot have more than $\log_2(N)(k-1)$ prefix neighbors. □

For the distribution function of the replication load in a full prefix tree, we need to determine replication values along with their frequencies. Recalling the picture of a full prefix tree for

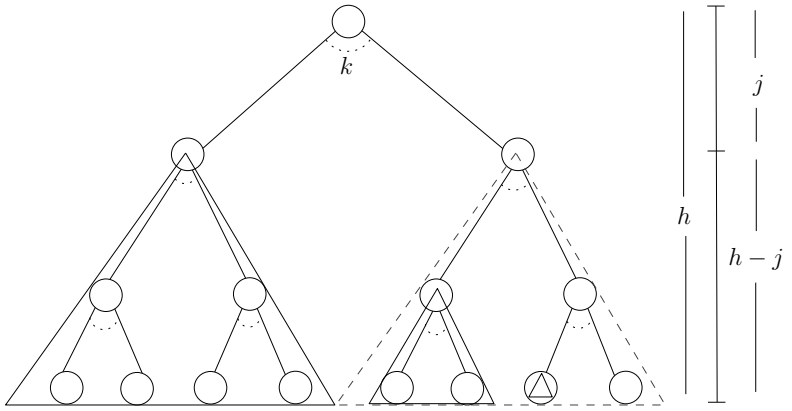


Figure 3.4: A Prefix Tree with Inner Vertices Defining the Root of Subtrees with Self-similar Properties due to the Recursive Nature of k-ary Trees

an alphabet with k digits, every node except the leaves has k children. The number of packet replications for an overlay peer is equal to the overall number of forwarding neighbors, which depends on the tree position, when the peer receives the packet. Per level the replication load is $k-1$. For example, the source starts forwarding a packet from the root of the prefix tree with height h, resulting in a constant load of $h \cdot (k-1)$. Prefix neighbors receive the data on the tree level $h-1$ and duplicate the data $(h-1) \cdot (k-1)$ times. Consequently, in a fully populated k-ary prefix tree of height h, replication occurs only at multiples of $k-1$, the number of neighbors in prefix space. For $j \geq 0$ we denote these discrete values by $v_{h,k}(j) = (h-j)(k-1)$.

To derive the replication frequency, we quantify the occurrence of the replication load $v_{h,k}(j)$. Since we know the load of a peer forwarding packets at height j, the frequency can be calculated by counting the number of peers that fulfill the replication condition. The latter corresponds to the number of (sub-)trees with height $h-j$, because every peer serves as forwarder for one tree. Starting at the source in a full prefix tree, the structure decomposes in $k-1$ subtrees with height $h-1$, $k(k-1)$ subtrees of height $h-2$, etc. (cf. figure 3.4). At every level of the full prefix tree, there is an exponential growth in the number of inner vertices representing the root of new subtrees. Thus, the frequency of $(h-j)$-size subtrees must increase exponentially with their decreasing height. In detail there are $k^{j-1} \cdot (k-1)$ subtrees of height $h-j$, which account for a replication load of $(h-j) \cdot (k-1)$.

Theorem 3.4 *Given a fully populated k-ary prefix tree of height h. Then the frequency $f_{h,k}(v_{h,k}(j))$ for a replication load $v_{h,k}(j) = (h-j)(k-1)$ is given by*

$$f_{h,k}(v_{h,k}(j)) = \begin{cases} 1 & \text{for } j = 0 \\ k^{j-1} \cdot (k-1) & \text{for } 0 < j \leq h. \end{cases} \tag{3.1}$$

Proof by induction. We assume a full k-ary prefix tree of height h. The case $j = 0$ corresponds to the (single) source that replicates data to $h(k-1)$ neighbors as derived above.

The induction is done with respect to $h-j$, the height of a subtree.

Base case: Is $h-j = 1$, we have to show that the replication load $v_{h,k}(h-1)$ appears $(k-1)$-times. In a tree of height 1, the source sends the data to all further leaves directly, which equals $k-1$.

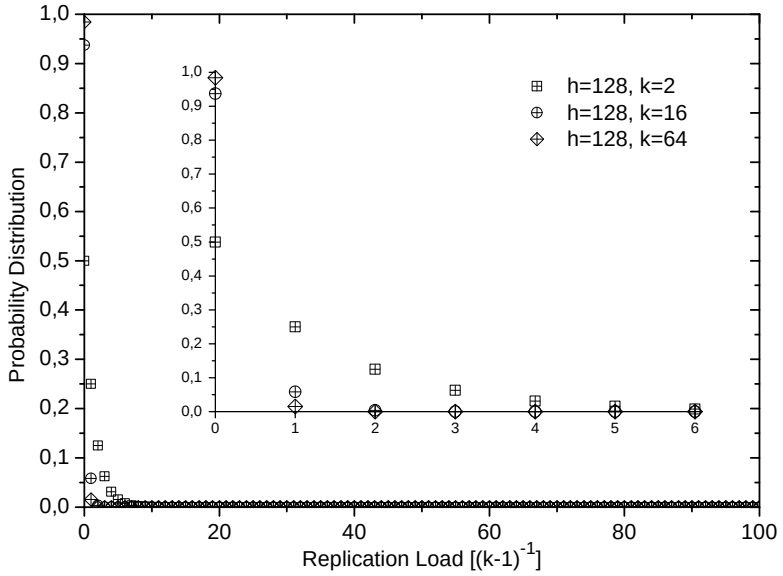


Figure 3.5: Normalized Probability Distributions $P_{h,k}(h-j)$ for the Replication Load Cut at 100 in Full Prefix Trees. Comparisons of a Small ($k=2$), Regular ($k=16$) and Large ($k=64$) Alphabet in a Realistic Key Space ($h=128$).

Induction step: Assume the statement holds for $h-j$. We have to show that the statement holds for $h-j+1$, i.e., $f_{h,k}(v_{h,k}(h-j+1)) = k^{h-j}(k-1)$.

This can be seen as follows: Each full prefix tree of height $h-j+1$ consists of k subtrees of height $h-j$. The replication load of a node in a tree of $(h-j+1)$ equals the sum of all neighbors in these k subtrees. Using the induction hypothesis the overall replication load $k \cdot f_{h,k}(v_{h,k}(h-j)) = kk^{h-j-1}(k-1) = k^{h-j}(k-1)$. □

The overall number of packet replications is easily identified as the number of leave nodes, since there are no packet duplications and each peer receives the broadcast. The number of leaves of a full k-ary tree of height h equals k^h, such that we arrive at the following

Corollary 3.1 *The probability distribution $P_{h,k}$ for packet replication multiplicities reads*

$$P_{h,k}(v_{h,k}(j)) = \begin{cases} k^{-h} & \text{for } j = 0 \\ k^{j-h-1} \cdot (k-1) & \text{for } 1 \leq j \leq h \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Proof. As we know the frequency distribution from theorem 3.4, we can calculate the normalization factor based on the geometric series by

$$\sum_{j=0}^{h} f_{h,k}(v_{h,k}(j)) = 1 + \sum_{j=1}^{h} k^{j-1}(k-1) = 1 + (k-1)\sum_{j=1}^{h} k^{j-1} = 1 + (k-1)\frac{k^h - 1}{k-1} = k^h.$$

□

The normalized probability distribution $P_{h,k}(h-j),\ 0 \leq j \leq h$ is plotted in figure 3.5. The height of the tree, which corresponds to the key length in a DHT, is fixed to $h = 128$ and the

prefix alphabet size k is varied from 2 to 16 and 64. It can nicely be seen that in a full prefix tree most of the peers are stressed solely with a small replication load. A large replication load, however, is highly unlikely for these exponentially decaying distributions. The variation of the branching parameter k influences significantly the balance of replication values smaller than 10. If the tree broadens, more leaves will receive a broadcast directly from the same neighbor, while the distribution is more balanced for narrower trees.

Corollary 3.2 *The average replication load for a node in a full prefix tree $T_{h,k}$ is given by $1 - k^{-h}$, its standard deviation by $\sqrt{k} + \mathcal{O}(k^{-h})$.*

Proof. Omitting vanishing terms, the average is expressed by

$$\begin{aligned}
\sum_{j=0}^{h} v_{h,k}(j) \cdot P_{h,k}(v_{h,k}(j)) &= h(k-1) \cdot k^{-h} + \sum_{j=1}^{h} (h-j)(k-1) \cdot k^{j-h-1}(k-1) \\
&= k^{-h} \left\{ h(k-1) + \sum_{j=1}^{h} (h-j)(k-1) \cdot k^{j-1}(k-1) \right\} \\
&= \frac{k-1}{k^h} \left\{ h + (k-1) \sum_{j=0}^{h-1} (h-(j+1))k^j \right\} \\
&= \frac{k-1}{k^h} \left\{ h + (k-1) \left(\frac{h-1}{1-k} + \frac{k(k^{h-1}-1)}{(1-k)^2} \right) \right\} \\
&= \frac{k-1}{k^h} + \frac{k^h - k}{k^h} = 1 - k^{-h}.
\end{aligned}$$

The sum in the third line is an arithmetic-geometric progression and evaluated according to [2].

To evaluate the standard deviation, we first calculate the second moment

$$\begin{aligned}
\sum_{j=0}^{h} (v_{h,k}(j))^2 \cdot P_{h,k}(v_{h,k}(j)) &= k^{-h} \left\{ h^2(k-1)^2 + \sum_{j=1}^{h} (h-j)^2(k-1)^2 \cdot k^{j-1}(k-1) \right\} \\
&= \frac{(k-1)^2}{k^h} \left\{ h^2 + (k-1) \sum_{j=0}^{h-1} (h-(j+1))^2 k^j \right\} \\
&= \frac{(k-1)^2}{k^h} \left\{ h^2 + \frac{(k-1)}{(k-1)^3} \left((k^h-1)(1+k) - h^2(k-1)^2 - 2h(k-1) \right) \right\} \\
&= k^{-h} \left\{ (k^h-1)(1+k) - 2h(k-1) \right\} \\
&= 1 + k - k^{-h} \left\{ (1+k) + 2h(k-1) \right\}
\end{aligned}$$

Combining these terms leads to the standard deviation for the replication load (RL)

$$\begin{aligned}
\sigma_{h,k}(RL) &= \sqrt{< RL^2 > - < RL >^2} \\
&= \sqrt{1 + k - k^{-h} \left\{ (1+k) + 2h(k-1) \right\} - (1-k^{-h})^2} \\
&= \sqrt{k - k^{-h} \left\{ (2h-1)(k-1) + k^{-h} \right\}} \\
&= \sqrt{k} + \mathcal{O}(k^{-h})
\end{aligned}$$

□

Surprisingly, the average replication load for a realistic key space is almost independent of h and k and very close to 1. This is mainly due to the high number of nodes, which are not required to forward data at all. Suppose an overlay with 256 keys and a binary alphabet, the average equals $0,996$. Commonly, the key space is much larger to avoid collisions in addressing. Similarly the variation of the replication load is almost only a function of the prefix alphabet k, but fairly independent of the tree height and thus the number of nodes.

Observing the weak dependence of the replication load distribution on h and k, i.e., the tree shaping parameters, it can be assumed that the model is sufficiently general to grant insights into the qualitative replication behavior of partly populated k-ary trees. We will see in section 3.3.2 that the simulations support this assumption.

Hop Count

As for the replication load, we firstly derive general measures of the number of hops a packet travels from the source to any destination in the prefix flooding scheme.

Theorem 3.5 *Any overlay node in a structured broadcast domain of N receivers and an alphabet with $k \geq 2$ digits will receive a packet from prefix flooding after at most $\log_2(N)$ hops. In the presence of Pastry overlay routing, the number of hops attained on average equals $\log_{2^b}(N)$ with $k = 2^b$.*

Proof. Prefix flooding increases the destination prefix hopwise to the next longest common prefix as obtained from the routing table. Longest common prefixes are represented by branching vertices on the prefix tree. In each forwarding step, the prefix tree is consequently descended by one vertex. Thus the maximal number of hops is limited by the height of the tree, which is bound by $\log_2(N)$ as shown in the proof of theorem 3.3. This shows the first claim.

Flooding based on the Pastry substrate is equivalent in path length to issuing packets from the source to all destinations by unicast routes. The average unicast path length in Pastry routing equals $\log_{2^b}(N)$ hops [82], which proves the second claim. □

We now want to return to considering a fully populated prefix tree and derive the hop distribution thereof. The main idea is similar to the replication load: A forwarding peer sends the broadcast to $k-1$ prefix neighbors, all of them rooting an equally structured subtree of height $h-1$. We are counting the number of paths with a length reduced by one herein. Additionally we count the frequency of paths for the calculated hop count in the virtual subtree containing the forwarder. This recursion results in

Theorem 3.6 *Given a fully populated k-ary prefix tree of height h, the frequency $f_{h,k}(j)$ of a hop count j occurring in prefix flooding is given by*

$$f_{h,k}(j) = \binom{h}{j}(k-1)^j. \tag{3.3}$$

Proof. A flooding packet arriving at node n after j hops will admit a current destination prefix of length j. Being located in a subtree of height $h-j$, n will forward the packet to its downtree neighbors, thereby partitioning its subtree into $k-1$ further subtrees of height $h-j-1$ (cf. figure 3.4). Due to the recursive nature of the k-ary prefix tree, the frequency distribution satisfies the recurrence relation

$$f_{h,k}(j) = f_{h-1,k}(j) + (k-1) \cdot f_{h-1,k}(j-1) \text{ with } f_{1,k}(0) = 1, f_{1,k}(1) = k-1. \tag{3.4}$$

Inserting $f_{h,k}$ leads to

$$\begin{aligned} f_{h,k}(j) &= \binom{h}{j}(k-1)^j = \binom{h-1}{j}(k-1)^j + (k-1)\binom{h-1}{j-1}(k-1)^{j-1} \\ &= f_{h-1,k}(j) + (k-1) \cdot f_{h-1,k}(j-1), \end{aligned}$$

which proves the theorem. □

This result can be interpreted in two different ways. At first, among all legitimate paths in downtree routing, i.e., of length h, those of length j are selected and branch $k-1$ times at each of the j intermediate prefix nodes. At second, flooding corresponds to a node discovery process, where a node discovers its $v_{h,k}(j) = (h-j)(k-1)$ neighbors which in turn discover their neighbors in the following step. Subsequent neighbor discovery requires connect to the j-th part as only $(h-j)(k-1)/j$ nodes have further neighbors.

Following a similar argument as in corollary 3.1, it is clear that normalization for hop count frequencies is given by k^h, the number of leaf nodes in the full prefix tree.

Corollary 3.3 *The probability distribution $H_{h,k}(j)$ of the hop count for flooding a full prefix tree $T_{h,k}$ evaluates to*

$$H_{h,k}(j) = k^{-h} \cdot \binom{h}{j}(k-1)^j. \tag{3.5}$$

Proof. Using the binomial expansion $(1+x)^n = \sum_{j=0}^{n} \binom{n}{j} x^j$ [2],

$$\sum_{j=0}^{h} f_{h,k}(j) = \sum_{j=0}^{h} \binom{h}{j}(k-1)^j = (1+k-1)^h = k^h.$$

□

Corollary 3.4 *The average hop count at which a packet is received from flooding in a full prefix tree $T_{h,k}$ is given by $< H_{h,k} >= (k-1)/k \cdot h$, the standard deviation of the hop count distribution (3.5) equals $\sigma_{H_{h,k}} = \sqrt{(k-1) \cdot h}/k$.*

Proof. An evaluation of the average sum with the help of the symbolic calculator Maple yields

$$< H_{h,k} >= \sum_{j=0}^{h} j \cdot H_{h,k}(j) = \sum_{j=0}^{h} j \cdot k^{-h} \cdot \binom{h}{j} (k-1)^j = k^{-h} \cdot \frac{k^h (k-1) \cdot h}{k} = \frac{k-1}{k} \cdot h.$$

The second moment evaluates to

$$< H_{h,k}^2 >= \sum_{j=0}^{h} j^2 \cdot H_{h,k}(j) = \sum_{j=0}^{h} j^2 \cdot k^{-h} \cdot \binom{h}{j} (k-1)^j = \frac{k-1}{k} \cdot h + \left(\frac{k-1}{k}\right)^2 \cdot h(h-1),$$

hence combining these terms leads to the standard deviation

$$\begin{aligned} \sigma_{H_{h,k}} &= \sqrt{< H_{h,k}^2 > - < H_{h,k} >^2} = \sqrt{\frac{k-1}{k} \cdot h + \left(\frac{k-1}{k}\right)^2 \cdot h(h-1) - \left(\frac{k-1}{k}\right)^2 \cdot h^2} \\ &= \sqrt{h \cdot \left\{ \frac{k-1}{k} - \left(\frac{k-1}{k}\right)^2 \right\}} = \frac{\sqrt{h \cdot (k-1)}}{k} \end{aligned}$$

□

This average is almost independent of the prefix alphabet k and can be in some sense interpreted as the counterpart of the average replication load as seen in corollary 3.2. As the average number of per hop replications is close to one, packets travel down the entire tree and reach most of their receivers after nearly h hops. The width of the hop count distribution, its standard deviation, admits a weak dependence on k, slowly decaying from its maximum at $k = 2$ as $k^{-1/2}$.

In contrast to the replication load distribution, which showed only a weak dependence on the tree shaping parameters, the hop count results strongly depend on h for the fully populated k-ary tree. The height h is directly related to the number of nodes k^h in this tree, which does not hold for realistic scenarios. Thus a direct transfer to sparsely populated random trees is questionable.

To derive a distribution for general distribution trees, evaluations are required on the class of all *random k-ary* trees[3]. Unfortunately this turns out to be difficult. Proceeding in a significantly simpler, but reasonable approach, we restrict the analysis to the class of *random recursive k-ary* trees with a *homogeneous* probability p for independent edges. In this model, each vertex branches to each of its $k-1$ possible out degrees independently with probability p, thereby preserving the recursive nature of the fully populated k-ary tree. Instead of equation 3.4, the hop frequency of routing on this random recursive tree will be governed by the modified rate equation

$$f_{h,k}(j) = f_{h-1,k}(j) + p \cdot (k-1) \cdot f_{h-1,k}(j-1) \quad \text{with } f_{1,k}(0) = 1, f_{1,k}(1) = p(k-1). \tag{3.6}$$

This can be solved analogously to 3.4 and immediately yields

[3] A random k-ary tree is a tree with nodes of out degrees $\leq k$ that follow some random distribution.

Corollary 3.5 *The probability distribution $H_{h,k}^{(p)}(j)$ of the hop count for flooding a random recursive k-ary prefix tree $T_{h,k}^{(p)}$ with homogeneous, independent edge probability p evaluates to*

$$H_{h,k}^{(p)}(j) = (1 + p(k-1))^{-h} \cdot \binom{h}{j} \cdot (p(k-1))^j, \tag{3.7}$$

which attains the average value $< H_{h,k}^{(p)} > \; = \frac{p(k-1)}{1+p(k-1)} \cdot h$, *and the standard deviation* $\sigma_{H_{h,k}^{(p)}} = \frac{\sqrt{p(k-1)\cdot h}}{1+p(k-1)}$.

The introduced edge probability p is not a 'free' parameter, but a function of the total number of leaf nodes $N = (1 + p(k-1))^h$ in the tree. Solving this relation for p,

$$p = \frac{\sqrt[h]{N} - 1}{k-1},$$

and inserting typical Pastry parameters for $k = 4, 16$, $h = 128$ and node numbers of our simulations, will lead to the relatively small edge probabilities, mean hop counts and standard deviations displayed in table 3.1.

	$k = 4,\ h = 128$				$k = 16,\ h = 128$			
N	10	100	1.000	10.000	10	100	1.000	10.000
p	0.0061	0.0122	0.0185	0.0249	0.00122	0.00244	0.00370	0.00497
$< H_{h,k}^{(p)} >$	2.30	4.52	6.73	8.90	2.30	4.52	6.73	8.88
$\sigma_{H_{h,k}^{(p)}}$	1.50	2.09	2.53	2.88	1.50	2.09	2.53	2.87

Table 3.1: Selected Link Probabilities, Mean Hop Counts and Standard Deviations for Characteristic Parameter Sets.

The corresponding probability distributions for a small and large overlay are plotted in figure 3.6 using a fixed key length of 128 and an alphabet size of 16. $p = 1$ represents a full k-ary prefix tree. Although the tree structure changes by decreasing the number of leaves, the shape of the distribution persists and results mainly in an adjustment of the centering to < 20 hops preserving the same qualitative behavior. It is interesting to observe that the width of the hop count distribution weakly oscillates as a function of the sparsity parameter p, approaching 0 as $p \downarrow 0$ and about $\sqrt{h/k}$ as $p \uparrow 1$, while crossing its maximum $\sqrt{h}/2$ at $p = 1/(k-1)$. For realistic settings in partly populated prefix trees, the distribution narrows around small mean values. Thus, a broadcast is delivered after an acceptable number of hops to most of the peers.

These analytical results will not only support a qualitative insight into the mechanisms of prefix-based packet distribution, but also show ample agreement with the simulation results presented in the subsequent section. The latter specifically holds for the overall shape of the distribution, even though the actual centers and mean values are overestimated with respect to simulations and theorem 3.5. This is due to the limited validity of this homogeneous approach in very sparse networks. As will be derived in the multicast section 4.2, edge probabilities are not homogeneous in prefix length, but exponentially decaying, whenever most of the prefix key space remains unpopulated. The advantage of this simpler model should be seen in being completely solvable, as well as in the smooth transition that it grants between a densely and a partly populated prefix space.

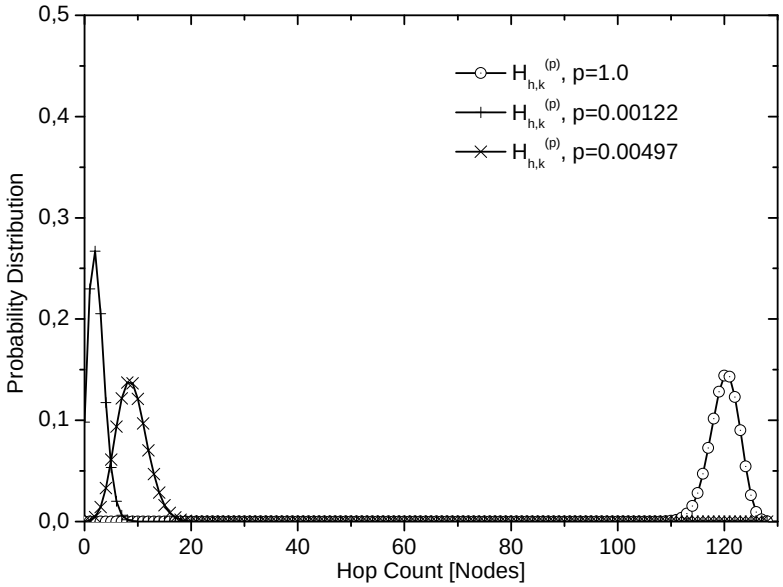


Figure 3.6: Probability Distributions $H_{h,k}^{(p)}$ for the Hop Count in Prefix Trees With Varying Homogeneous Edge Probabilities p and Fixed $k = 16, h = 128$

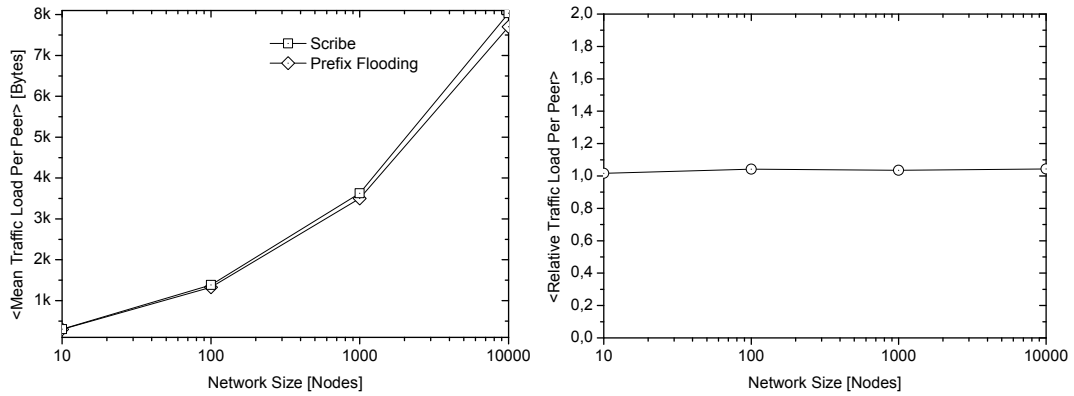
3.3.2 Simulation Results

In this section, we will analyze the performance of the prefix flooding based on a stochastic discrete event simulation. Prefix flooding distributes a broadcast via virtual peers along optimal paths according to a prefix tree. For comparative reasons, we also investigate the behavior of a rendezvous point-based approach operating on a prefix tree. For the latter, Scribe [22] is chosen as a generic approach for structured trees rooted at a rendezvous point. Both, the prefix flooding and Scribe, are implemented on top of a proactive version of the DHT substrate Pastry. The proactive routing maintenance will be initiated after the peer creation process ended to ensure a complete Pastry routing set.

In detail, our simulations are performed on the network simulator platform OMNeT++ 3.3 [99], supplemented by a preliminary version of the overlay simulation package OverSim [10] including Scribe and extended by the prefix flooding implementation. Pastry has been configured as in its original version [82]. Especially, we use a key length of 128 and an alphabet size of 16, if not mentioned otherwise. To investigate the scaling behavior of the protocols, the simulations are conducted for a number of peers varying by three orders of magnitude.

None of the relative metrics described in section 3.3 depend on the underlay. Thus the Simple model [9] has been applied as the underlying network with a homogeneous link delay of $1\,ms$ to analyze the network properties inside the overlay. Underlay properties such as link delay variations would influence the proximity selection and stretch the travel time, which should produce the same impact on both approaches and would cancel out by relative metrics. The Simple model also eliminates processing steps of non-overlay nodes between peers, which would be an unnecessary detail for our study.

The analysis is not focusing on reliability aspects, which allows us to neglect churn. In particular, any effects of volatile nodes would be completely maintained by Pastry for the prefix flooding and partially for Scribe. Rendezvous point (RP) based schemes have to reorganize



(a) Absolute UDP-Traffic per Peer L

(b) $L_{\text{Scribe}}/L_{\text{Prefix Flooding}}$

Figure 3.7: The Mean UDP Traffic Volume per Peer in Prefix Flooding and Scribe for Overlays of Different Sizes

the distribution tree due to failing RPs, resulting in DHTs by new key associations, which nevertheless is not addressed here.

Each simulation is sampled with the same parameter settings until it is converged.[4] The convergence time has been determined by performing calculations for increasing sample sizes until the variation of values remain two orders of magnitude below actual results. Each sample consists of a new key association to peers randomly selected according to uniform distribution. One sender is chosen from the peer set per sample with equal weights. For Scribe, we also uniformly choose a key as multicast group, which all nodes join except the source. After one broadcast is delivered to all receivers, the measurements stop. We average the results over all samples with the same settings.

Summarizing the simulation scenario, we calculate the flooding performance on an arbitrary ($k = 16$)-ary prefix tree with a fixed maximal height and a varying number of leaves interconnected by links of identical weight. The broadcast will be initiated by a randomly selected leaf.

The simulation has been thoroughly tested. Manual packet tracing has been performed for selected networks to compare simulations with the routing algorithms. Plausibility checks are based on the results as described below.

Traffic Load

The mean traffic load accumulates the UDP data volume per peer. The relative traffic load includes all overhead of Scribe in contrast to the prefix flooding, which is free of additional signaling. The results are plotted in figure 3.7. In general, the traffic load per peer increases polynomially. However, Scribe requires sending of slightly more data to accomplish a broadcast (cf. figure 3.7(b)). The reason for this is twofold: The prefix flooding inherits all routing information directly from Pastry without the requirement of further signaling and hence carries no overhead. In contrast, Scribe has to perform an additional active group and tree management, including receiver subscriptions and paths maintenance towards the rendezvous point. The application header of Scribe is also marginally larger, as the full overlay key is transmitted to

[4] The number of runs per scenario depends on the overlay size and ranges from $500 - 2500$.

identify the multicast group. The prefix flooding requires only an indication of the prefix length to identify the destination prefix.

Surprisingly, the traffic load will be dominated by the KBR overhead. The amount of data for Scribe and the prefix flooding differs in additional traffic for maintenance in Scribe. Eliminating regular Pastry data for both approaches by a relative measure reveals that BIDIR-SAM and Scribe creates almost the same amount of data (cf. figure 3.7(b)), while the absolute traffic volume increase is dominated by Pastry (cf. figure 3.7(a)).

Although the signaling of Scribe is negligible in our analysis, it is worth noting that a signaling free scheme like the prefix flooding is less vulnerable to distortions. A join message which has been lost, restrains receivers from packet reception.

Replication Load

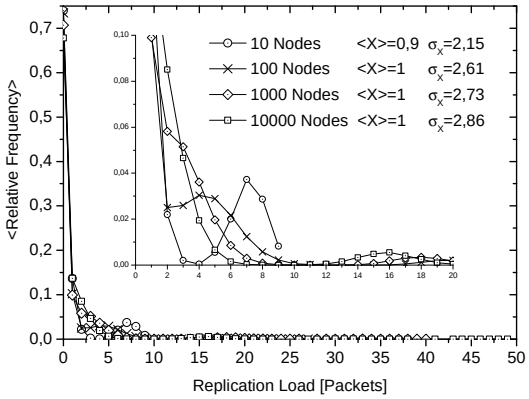
The distributions of the peer replication load for prefix flooding and Scribe are displayed in figure 3.8. Both schemes show an exponential decay around their common average value of 1. However, the shapes of the distributions for the two approaches vary significantly, which becomes apparent at first from standard deviation values. While the widths of the distributions for prefix flooding are small and almost independent of network sizes, the corresponding values for Scribe grow large, about linearly in the number of nodes.

Both broadcasting schemes produce a large number of replications of values 0 and 1, but frequencies drastically drop for higher multiplicities. Prefix flooding distribution attains a much smoother decay, leaving significant probability to replication values of $2 - 10$. Smoothness is even more pronounced for smaller alphabets as visualized in figure 3.8(e). In contrast, Scribe decreases faster from its average, decaying rapidly to probabilities below 1/100 for replications larger than 2, fairly independent of the alphabet k.

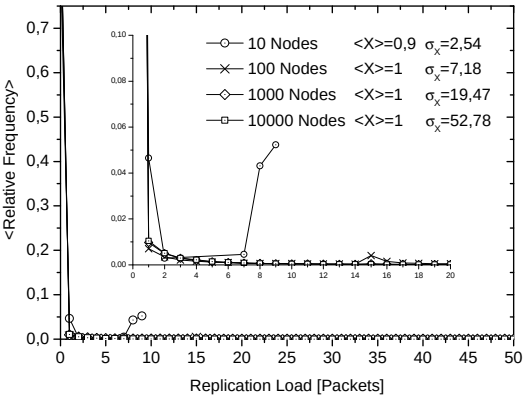
An exception from this overall shape can be observed for the distribution of 10 peers in Scribe. Here, the frequencies of replication values around 9 are strongly enhanced. This border effect for very small networks can be understood from analyzing distribution tails. As visualized in the log-log plot 3.8(d), the distribution of Scribe is heavy-tailed according to a power law decay, representing remarkably high probabilities for very large replication values up to 7800. Corresponding probabilities are accumulated for small sized overlays.

In contrast, the prefix flooding distribution admits a strict exponential decay, with tail weights vanishing at 50. Replication values in prefix flooding are superimposed by oscillating frequencies as visible in figure 3.8(c). The resulting probability "bumps" are noticeable on different scales for all overlays. The observation of oscillating tails can be explained by our theoretical analysis, which reveals an exponential decay within the range of multiples of $(k - 1)$. Compared to the prerequisites of corollary 3.1, the simulated overlays do not operate on full k-ary prefix trees. Hence replication values do not only occur as multiples of the branching factor, but level out with neighboring values. Nevertheless, regarding the peaks of the bumps, the population and replication pattern of the k-ary trees remain clearly visible.

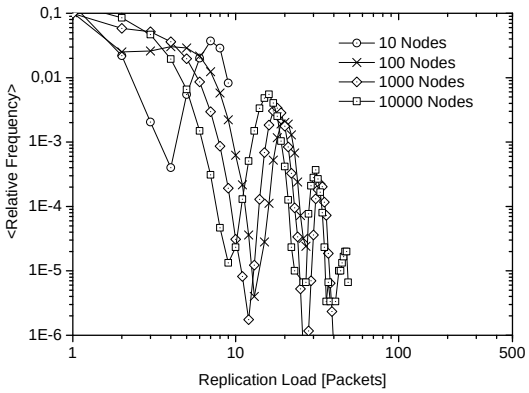
In both approaches, most of the peers receive the broadcast without a need to forward it further. Scribe thereby stresses a small number of peers to serve a much higher replication load. Instead, the prefix flooding reduces the maximal replication load by distributing the load fair and evenly over the neighbors.



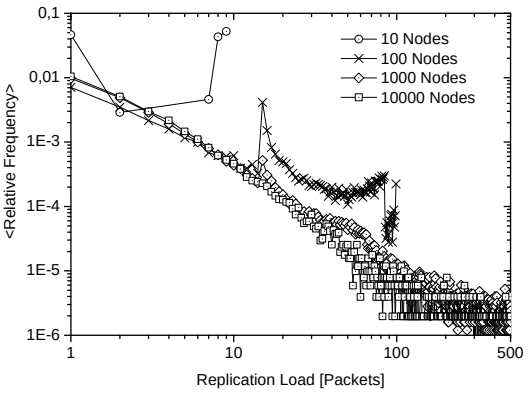
(a) Prefix Flooding, $k = 16$



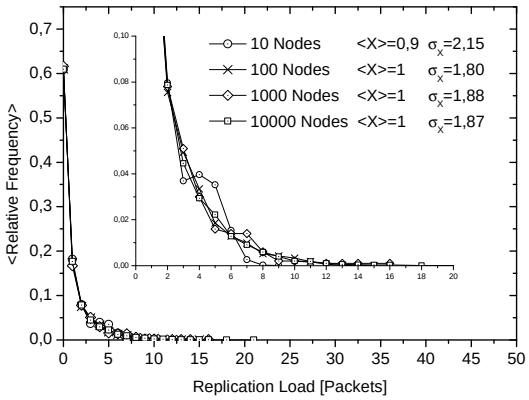
(b) Scribe, $k = 16$



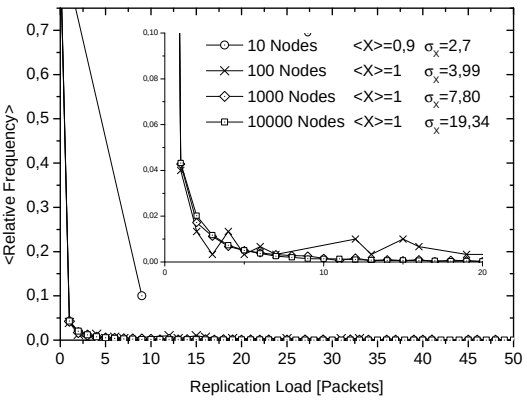
(c) Detail: Tail for Prefix Flooding, $k = 16$



(d) Detail: Tail for Scribe, $k = 16$

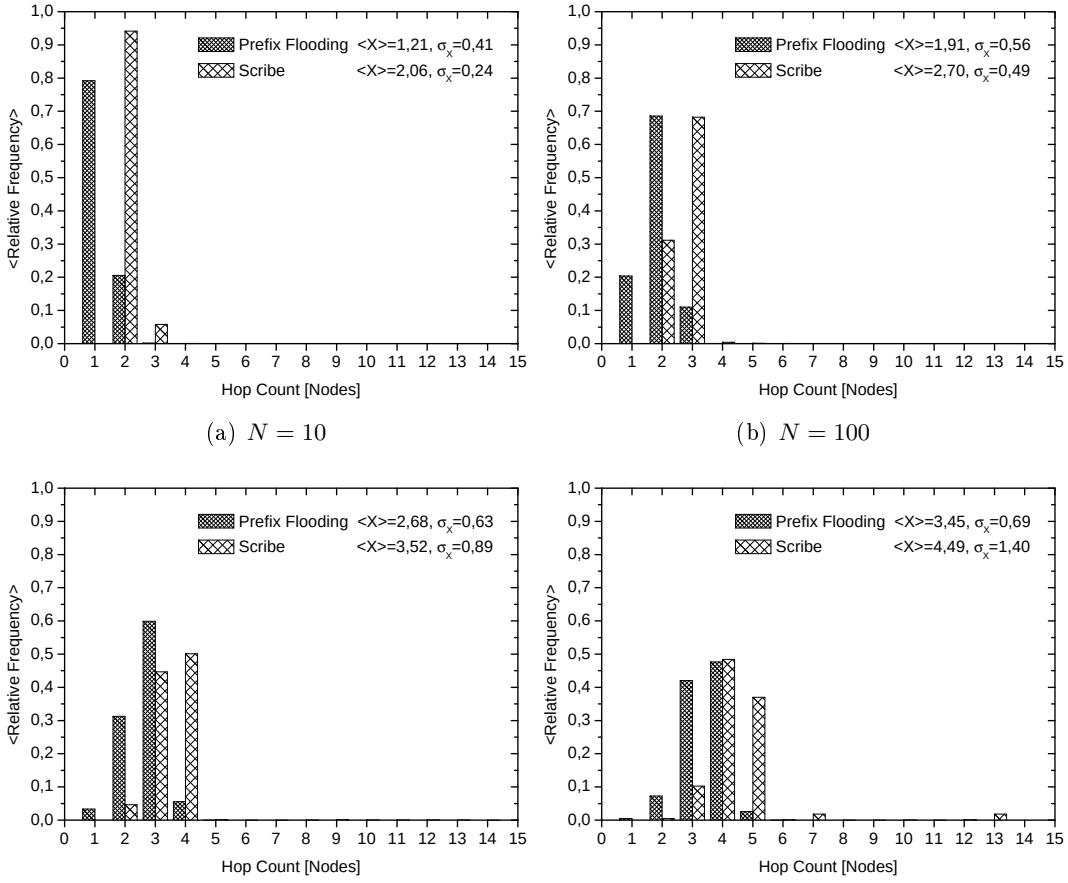


(e) Prefix Flooding, $k = 4$



(f) Scribe, $k = 4$

Figure 3.8: Distribution of Packet Replication Comparing Prefix Flooding With Scribe for a Number of Peers Using a Fixed Key Length of 128 and a Varying Prefix Alphabet Size k



(a) $N = 10$ (b) $N = 100$

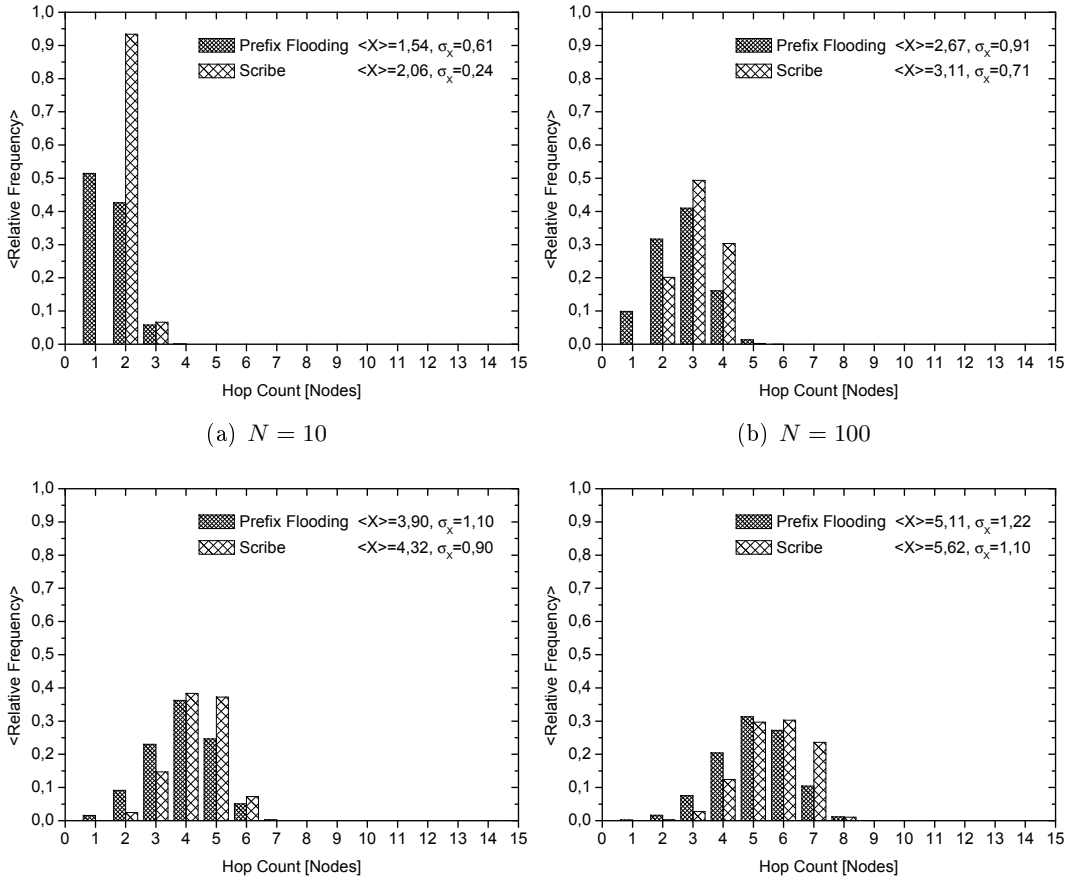
(c) $N = 1.000$ (d) $N = 10.000$

Figure 3.9: Hop Count Distribution for an Overlay of Size N, $k = 16$

Hop Count

The mean hop count distribution for different overlay sizes is shown in figure 3.9. In general, both schemes show the logarithmically growing hop path length dependent on the number of peers. With an increasing quantity of leaves, the height of prefix trees will increase logarithmically, as well, resulting in longer paths from the source and intermediate forwarders to the receivers. Figure 3.9(a) visualizes nicely that the path length in rendezvous point schemes is elongated by at least one hop, which clearly holds independent of the receiver numbers. The mean hop count $< X >$ for Scribe highlights approximately one additional node in contrast to the prefix flooding.

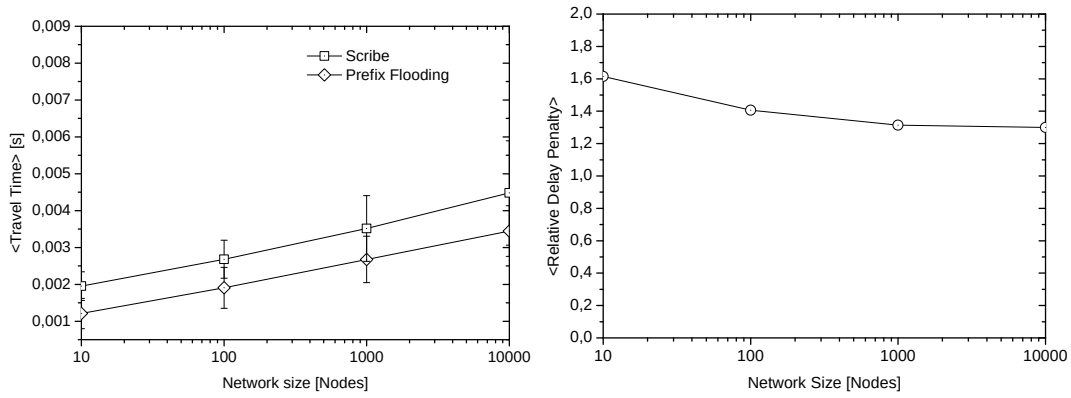
For a sufficiently large $N > 10$, the average of the distribution for the prefix flooding attains directly the calculated mean hop count in theorem 3.5, at which all other hop count values are centered. The hop count distribution in Scribe shows a heavy-tailed behavior, which increases with the overlay size as indicated by the approximate linear growth of the standard deviation. In contrast, the prefix flooding almost attains a constant variation. Consequently, in prefix flooding the path lengths are tightly concentrated around the logarithmically bounded average, while Scribe builds up longer branches with higher weights.



(a) $N = 10$ (b) $N = 100$

(c) $N = 1000$ (d) $N = 10000$

Figure 3.10: Hop Count Distribution for an Overlay of Size N, $k = 4$



(a) Absolute Travel Time

(b) Travel Time$_{\text{Scribe}}$/Travel Time$_{\text{Prefix Flooding}}$

Figure 3.11: Travel Time and Relative Delay Penalty for Prefix Flooding & Scribe

Travel Time

Figure 3.11 shows the absolute travel time and the relative delay penalty depending on the network size for Scribe and prefix flooding. The prefix flooding outperforms a rendezvous point-based approach by approximately a factor of 1.4 in larger networks (cf. figure 3.11(b)). Due to the sparsely filled prefix space in small networks, the maximal path length from the rendezvous point (RP) to the receivers is similar to the prefix flooding, which is visualized in figure 3.11(a). Figure 3.8(b) confirms this observation: For an overlay with 10 peers almost all receivers will be addressed directly by a node, which replicates in the number of receivers, i.e., the rendezvous point (RP). The more keys are allocated, the more branching points are located closely to the RP resulting in longer paths and less efficient parallelism in contrast to the prefix flooding.

The travel time has a direct correlation with the path length, which reproduces figure 3.11(a) by comparison with figure 3.9. Recalling the constant link delay of $1\,ms$, the average of the absolute travel time corresponds directly to the mean hop count for both approaches.

3.4 Related Work

The principal approach for implementing broadcast on a pure DHT derives from recursive partitioning of the key space with data distribution following partition ranges. The prefix flooding operates in this sense, defining numerical interval boundaries from prefix transitions. The first idea of a broadcast based on nested intervals was proposed in [33]. The authors observe that lookup (routing) in DHTs is similar to performing a distributed k-ary search (routing along a k-ary tree). The broadcast is sent to intervals of exponentially increasing scale as derived from the Chord routing table. Each interval decomposes recursively, creating a broadcast distribution tree for Chord, where peers recursively flood the routing table. The messages are distributed to all peers without redundancies under the assumption of complete finger tables. However, the correctness of the algorithm is not formally proven probably due to the simplicity of Chord associating uniquely a finger table entry to a peer. The paper lacks a detailed analysis of the approach, as the authors only measure the average messages per node and the ratio of message redundancy in simulations.

A generalization of [33] is proposed in [41]. In addition to a design independent of Chord, the

authors enhance their algorithm by reliability routines, which guarantee a broadcast distribution independent of the routing table states. This is performed by delegating data delivery for missing entries to subsequent forwarders. It is not obvious that the algorithm terminates, and has not been proven. Node coverage and unique delivery of messages are only verified by simulations. These simulations focus on the performance of the reliability mechanism.

The authors in [59] introduce a scheme, which splits the key space in d partitions of equal size and selects the first node in clockwise direction as the responsible forwarder. Similar to [33], this approach refrains from using uneven, logarithmic partitioning. The focus of the work, however, lies in reliable broadcast under churn. The proposed algorithm is complemented with a recursive acknowledgement mechanism, i.e., all children of a parent node have to confirm the data reception. If the ACK failed after a specific timeout, a new responsible peer for the partition is selected and the broadcast is retransmitted. The performance is analyzed with respect to different churn rates.

An approach, which cannot ensure a broadcast distribution without data redundancy, is presented in [65]. The authors combine a slightly enhanced version of [33] with an epidemic distribution. All broadcast forwarders send the data periodically to a randomly chosen neighbor, whereby the protocol cannot guarantee that a neighbor receives the broadcast only once. Although epidemic flooding enforces the reliability of data delivery, it conflicts with the design goal of a broadcast to operate without redundancy.

All of the approaches mentioned above lack formal verification, as well as general analytical considerations regarding data distribution according to k-ary prefix trees. Most of the algorithms are implemented on top of Chord, none of them on Pastry, which natively offers a proximity-aware prefix routing.

A generalized construction scheme to partitioning the key is space is presented in [57]. The authors observe that any contractive self-mapping function P of the key space with a single fixed point α, i.e., $P(\alpha) = \alpha$, gives rise to a parent relationship. Based on the parent relation $P(\alpha)$, a reverse path can be set up for any node α, leading to a broadcast distribution tree with the root α. Different parent functions thus give rise to different trees at variable roots, which may be used for load-sharing or redundancy purposes.

DHT specific flooding has been introduced in the early work [78] for CAN (Content Addressable Network). In contrast to Chord or Pastry, CAN maps node IDs to regions representing coordinates in a partitioned d-dimensional space. CAN broadcasts the data to all geographical neighbors, thereby accounting for predecessors and foreseeable redundancies. However, the partitioning of the d-dimensional space may be uneven and result in data duplication at subregions. An extensive simulation study of flooding and tree based overlay multicast over CAN and Pastry with respect to the underlay is presented in [23]. The authors show that CAN flooding is outperformed by Pastry flooding, which relies on a more efficient tree structure adaptive to the underlay.

Our implementation of the prefix flooding is almost identical to the Pastry flooding of Castro *et al.* [23]. The main difference lies in the reactive routing maintenance, which may result in data redundancy at the fallback forwarder in [23]. The focus of their analysis of broadcast distribution concerns the context of overlay multicast. The measured metrics reflect performance issues focusing on efforts imposed on the underlying network. In this sense, our results can be understood as complementary.

The contribution of this work lies in a structural analysis of flooding on k-ary prefix trees,

both, analytically and in simulations. Further on, we proved the correctness of the prefix-based broadcasting.

3.5 Discussion

Typical applications for broadcasting are met in IPTV and large-scale conferences or virtual events. While the first application is of uni-directional nature, but distributes higher data volumes, the latter operates under a tight real-time regime as required for immediate feedback. Both application areas are sensitive to disturbances resulting from delays or jitter. Additionally, streams of high data volume may overload intermediate forwarding nodes that are requested to replicate traffic at high multiplicities.

In this chapter we have presented and analyzed broadcasting within distributed hash tables. A prefix flooding, distributing data along prefix branches directly to receivers, is compared to a rendezvous point-based scheme, which utilizes a shared tree rooted at a predefined anchor peer. Several phenomena of general interest could be observed.

Divergent Path Length Distributions

Our simulation results confirm the mean hop difference of one between the prefix flooding and the rendezvous point-based approach Scribe. This additional, triangular hop in the overlay becomes noteworthy when stretched in the underlay and then may put stress on several links. The major advantage of the prefix flooding, though, is its quite stable concentration of path length distribution around the average, attaining low variations independent of the overlay size. In general, P2P networks consist of volatile nodes. If we assume an overlay with regular churn, i.e., session times in the range of minutes or larger, and a persistent number of peers on average, the DHT moderately reorganizes key associations. Such structural modifications lead to changing paths within the overlay and in the worst case, a single arrival or departure of a node may cause a data path to change drastically. In the prefix flooding, the path length only changes moderately for new and existing peers due to its narrow distribution.

This property differs for Scribe, which not only creates longer paths, but also admits a higher hop count fluctuation which grows with network size. A change of the overlay structure may thus elongate delivery branches significantly. This results in a higher jitter at end nodes. Such unstable behavior can cause disturbances, in particular for voice and video applications. Large jitter has to be compensated by buffers at the application side, which in turn add delay. The heavy-tailed overlay hop count distribution of Scribe produces a largely inhomogeneous travel time, which complicates synchronous applications.

Varying Replication Load

A high variation can also be identified for the packet replication in Scribe. Similar to the prefix flooding, it is rather likely that peers forward with low replication load. Nevertheless, in a long tail distribution nodes are required to replicate many more packets with values up to 7800 in large sized overlays of 10.000 peers. The distribution of packet replication is thus strongly unbalanced, requiring very low and very high values to be served within the same scenario.

In contrast to Scribe, the prefix flooding guarantees a replication load closely balanced around its average of about 1. It can be tuned directly by the branching factor k. As we know from

the theoretical analysis of section 3.3.1, packet replications occur as multiples of $k-1$ in full prefix space. Decreasing k adjusts the maximum number of replications to smaller values.

On the contrary, distribution trees in Scribe are data-driven and not prefix-shaped. They do not allow for a direct configuration of the peer load via the prefix parameter. As adaptation to node capacities is not inherently provided in DHTs, group communication schemes should independently balance packet replications.

In scenarios of lightweight end devices with low battery power connected by low capacity links, each additional packet transmission causes a faster reduction of life time as well as early congestion. If a peer – responsible for large number of replications – resides inside a wireless cell, a broadcast domain may collapse, while packets storm the shared medium. Further on, in the sense of traffic pattern recognition, a well located high duplication rate of packets is identified as anomaly. This may result in blocking subsequent packets by intrusion detection systems.

An Overloaded Single Peer

The peers with extraordinarily high packet replication load in Scribe have been identified as the rendezvous points (RP). An appropriate treatment of such service nodes becomes more important under the aspect of unbalanced packet replication, but poses a severe conceptual problem in DHTs: The placement of this entity should account for node and network capacities, but in a DHT is bound to the structural mapping of the multicast group identifier to an overlay key. Any alternative approach, e.g., selecting the RP address independently of the group address, will break the key space semantic with the result that an overlay node cannot derive the RP distribution address automatically.

Packet replications may also be a problem for standard PCs. Each broadcast packet will be internally passed towards the application layer. Thus copying data is done by CPU and usually within the user space. Even though current processors are powerful enough to handle a significant amount of data, overlay peers are not specifically optimized for this dedicated task. Processing application layer data competes foremost with other running programs on the peer, possibly causing a higher delivery delay, especially in the event of high replications. This gets even worse for RPs that serve as forwarder for a significant number of peers.

In our scenario we have only focused on a single source. Multiple senders in the same broadcast domain would deliver their data to the same RP, which may increase the replication load tremendously. In contrast, the prefix flooding distributes the data along multiple trees rooted at the different sources. For each of the distribution trees, all of the previously derived results hold.

The Problem of Asymmetric Routes

Observing the hop count and packet replication distribution, the question arises about the more fundamental reasons why the prefix flooding consistently outperforms Scribe. Leaving aside the RP-issues discussed above, the main conceptual difference between data-driven tree approaches and the prefix flooding follows from the method of tree establishment. In general, data-driven trees will be constructed from the receivers towards the source (or RP). This is also known as reverse path forwarding. The tree is optimal as long as the routing table entries are invertible. But if links between nodes admit asymmetrical weights, a source may deliver data along suboptimal paths. Such a problem does not arise, if the source constructs its tree

according to forward routes.

In DHT-based group communication, the direction of tree establishment is even more important. The distribution tree in Scribe is built from receiver subscriptions towards the RP, but the packets flow in inverse direction. As the association of prefixes to nodes is not unique, two peers may select a different destination for the same prefix. Thus diverse paths will be established, even though packets could uniformly traverse the reverse directions following the RP point of view. This results in many, reversely selected paths. For this reason, the RP is burdened with an unbalanced replication load. On the opposite, the prefix flooding distributes the broadcast along paths that are optimally chosen from the source location, limiting direct branches to the number of prefix neighbors. Prefix flooding solely uses forward-oriented directives, extracted directly from unicast DHT routing control.

Optimized tree construction and data transmission throughout the underlay are key controls for efficient group communication in DHTs. This work has identified that reverse path selection in overlay and underlay turns into a severe problem in the presence of asymmetric routing.[5] DHTs like Pastry implement a proximity selection criteria for filling their routing tables, which is designed for unicast routing and in forward direction. Employing the Pastry proximity-selected neighborship to build data-driven trees as it is done in Scribe, takes counter-efficient effects on the construction of distribution trees.

Our prefix-guided group management strictly adheres to forward-directed establishment of distribution trees. It could be shown to generate efficient group communication structures. The approach is thus particularly promising for overlay multicast services.

Asymmetric routing paths are also a problem for native group communication, because common multicast routing is based on data-driven trees. Establishing forward paths in the Internet is not as easy as it is in DHTs due to scaling issues. The prefix flooding changes the paradigm of data-driven trees to source-driven distribution: Each source represents the root of an implicitely defined distribution tree under appropriate performance values.

Rendezvous point-based distribution trees may degenerate to a large number of short-cuts, solely branching at the RP. This leads to some exceptionally long routes from source to receivers in the overlay as well as the underlay. In contrast, the prefix flooding guarantees a logarithmically limited replication load per peer with a balanced path length enabling a broadcast distribution in large overlays. The maximal numbers of replications can be adjusted by the alphabet size of the overlay keys. These predictable and adjustable performance characteristics are valuable properties of the overlay multicast solution, which is presented in the following section.

[5] A large-scale analysis by Paxson revealed that 50% of the virtual Internet paths are asymmetric at the end of 1995 [70].

4 Prefix-based Overlay Multicast

4.1 Introduction

Group communication based on a multicast function has been discussed since more than two decades [5], but its global large-scale deployment is missing until now. As providers remained hesitant to implement native multicast on the network layer, many ideas arose to distribute data by replicating streams at end user devices. Such overlay schemes are now part of selected applications. Nevertheless, the efforts towards widely accessible multicast services recently reactivated due to the roll out of appealing, but bandwidth intensive mass applications like IPTV, or new, cheaply available broadcast-oriented transmission channels like DVB-H/IPDC. In contrast to previous attempts, pragmatic solutions are now considered strategic, raising deployment simplicity to prime focus. Hybrid schemes, which provide native multicast in end user domains and utilize overlay multicast at inter-provider transitions [105], are particularly attractive from this perspective.

Overlay multicast can be constructed on top of structured and unstructured P2P networks. In its current state of development, the latter operate in a hybrid way by aggregating peers at fully meshed 'super peers'. Such schemes may burden a high load onto their aggregators and tend to not scale well to a large number of nodes due to high maintenance and routing costs. Unstructured overlay routing also limits reliability by introducing false negatives as caused by restricted route information exchange. In contrast, structured overlays that implement Distributed Hash Tables (DHTs), guarantee routing correctness within logarithmic bounds for key-based message forwarding and state storage. Further on, based on their coherent routing layer, they easily allow for the deployment of additional services. Several debates rank around DHT performance and the question, whether structured overlays loose their scalability under churn, when high maintenance overhead is required. Current studies reveal that general objections do not hold and structured approaches clearly outperform the unstructured [20, 75].

Supplementing Internet services by DHTs is currently enforced by activities of the IETF P2PSIP working group. Its generic peer-to-peer protocol will include a mandatory support of a DHT [8]. Thus, it may be reasonable to assume that DHT substrates will populate the future Internet. These may then also be used as underlying routing infrastructure for multicast protocols.

Typical Internet-wide group applications, which largely benefit from multicast, are of the media broadcasting type. On the one hand, services like IPTV characterize a single source scenario. On the other hand, collaborative social platforms, e.g., video chats and massive multiplayer online games require a multi-source distribution tree. Both application classes imply large multicast groups starting from some hundred receivers.

Current structured overlay multicast approaches either implement a source-specific network flooding, a source specific or a shared tree. Castro *et al.* have shown that flooding schemes are significantly outperformed by tree based routing [23]. Bayeux, the only source specific approach,

is exposed to scalability problems, as each request to join a group is routed to a single node managing that group. In contrast, shared trees introduce a dedicated overlay node, e.g., a rendezvous point for data distribution. Multicast traffic is concentrated on this single point of failure. Providers as well as end systems, though, need a balanced mechanism, which scales with large group size and provides predictable costs to control network provisioning. This is not guaranteed by current approaches and requires a new, out-of the box thinking.

In this chapter we will present Scalable Adaptive Multicast on Bi-directional Shared Trees (BIDIR-SAM), a novel overlay multicast approach for structured overlay networks. BIDIR-SAM constructs a bi-directional shared tree, which enables an arbitrary overlay node to distribute data along forward-oriented source-specific paths. It does not rely on any kind of rendezvous point or bootstrapping and operates directly on top of a deployed, proximity-aware DHT using a prefix-based routing scheme. BIDIR-SAM exhibits strictly predictable costs, which scale logarithmically with receiver sizes. BIDIR-SAM attains a similar multicast efficiency scaling factor as native group communication protocols, making it especially suitable for large and very large multicast groups.

At first we will lay out the algorithm and discuss its basic properties along with four immediate optimization options to enhance overlay redirections, underlay proximity selection, load sharing and redundancy of the distribution system. Thereafter we model and analyze BIDIR-SAM theoretically and in simulations, deriving its characteristic performance measures in comparison to the generic shared tree approach Scribe. A brief reference to corresponding analytic work and a discussion of the results conclude this chapter.

4.2 BIDIR-SAM – Scalable Adaptive Multicast on Bi-directional Shared Trees

Efficient multicast packet distribution is based on distribution trees, where branching nodes duplicate packets. A distribution tree is constructed on top of an unicast network, which is provided in structured overlay networks by a key-based routing (KBR) layer. Typically, the multicast tree spanning all receivers is rooted at the source or a rendezvous point. In contrast to traditional approaches, which generate the distribution tree from (reversed) packet transmission, BIDIR-SAM uses a prefix tree, which is built solely of overlay addresses of receivers. The prefix alphabet size is configurable. This tree will serve as a source-specific distribution tree valid for all sources anywhere throughout the network.

4.2.1 The Core Protocol

The main idea of BIDIR-SAM is to construct a prefix-based multicast distribution tree, in which a leaf is labelled with the overlay ID of a multicast listener. Multicast branching is performed at inner vertices. Each inner vertex can be mapped to a DHT member if the label represents a prefix of the overlay node address. The corresponding peer for a prefix will be resolved based on a proximity-aware neighbor set.

In the following, we will describe the group management functions of BIDIR-SAM used to maintain the distribution tree, and to forward data packets thereon.

Group Membership Management

BIDIR-SAM uses a prefix-based routing scheme as underlying KBR, which is provided by Distributed Hash Tables (DHTs) like Pastry. A broadcast prefix tree can be constructed by identifying leaves as overlay keys and labelling recursively inner vertices with the longest common prefix of their children (cf. section 3.2).

Sending a packet from the root to the leaves of the broadcast prefix tree will reach all overlay peers, as branching will be performed at the inner vertices. This can be implemented as described for the PREFIX FLOODING in section 3.2. In contrast to broadcast, multicast implements a selective distribution strategy, where final (multicast) receivers represent a subset of the peers. Thus, the multicast distribution tree in BIDIR-SAM is created from overlay keys of the receivers. Any sender, which itself forms a leaf in the prefix tree, will 'shift' the packet up to the (virtual) root of the tree and initiate a forwarding according to prefixes populated by receivers. In this way, the prefix tree is bi-directionally traversed for a sender.

All BIDIR-SAM peers will derive semantically identical trees in prefix space, but will hold only a selected, location-dependent knowledge therefrom. Routing correspondences are to be extracted from KBR's routing table and thereby differ from node to node. Multicast nodes need not memorize the entire group specific multicast tree, but will only be required to persist the prefix neighbors of all associated vertices.

Each peer is a potential multicast forwarder, serving as an intermediate destination for a prefix it shares. Consequently, a new multicast receiver has to be announced to enable all overlay nodes to store the corresponding neighboring prefix for forwarding multicast data. This prefix neighbor represents the root of a subtree, which subsumes multiple multicast listeners. Thus, only the first join and last leave has to be propagated outside this subtree.

To distribute data along a multicast distribution tree, a BIDIR-SAM peer K with overlay ID $\mathcal{K}$ maintains a multicast forwarding table for each multicast group. This list contains all prefixes, which serve as destinations adjacent to K. For a group G, we denote the multicast forwarding table by MFT_G.

To join or leave a multicast group, a BIDIR-SAM node injects a state update into the unicast prefix tree. The first and last receiver of the group flood their join and leave message in the complete (unicast) overlay network. For all further group members, the state update is propagated within the smallest subtree including receivers and covering the multicast listener. The algorithm works as follows:

BIDIR-SAM JOIN/LEAVE INJECTION

▷ Invoking this function at peer $\mathcal{K}$ for group G
1 **if** $MFT_G = \emptyset$
2 **then** PREFIX FLOODING Join/LeaveMessage To *
3 **else** Select $\mathcal{L} \in MFT_G : |\mathcal{L}| \geq |\mathcal{L}'|, \forall\, \mathcal{L}' \in MFT_G$
4 $\mathcal{C} \leftarrow LCP(\mathcal{L}, \mathcal{K})$ ▷ Creates root of subtree to flood
5 PREFIX FLOODING Join/LeaveMessage To $\mathcal{C}$

On the reception of a multicast state update the following function will be called to include or delete multicast forwarding entries and to route the message down the unicast prefix tree:

BIDIR-SAM RECEIVE

▷ We denote the prefix of length l and a key $\mathcal{A}$ with $prefix(l, \mathcal{A})$
▷ On arrival of message m for group G from peer $\mathcal{P}$ at node $\mathcal{K}$
1 $\mathcal{L} \leftarrow LCP(\mathcal{P}, \mathcal{K})$
2 $\mathcal{L}' \leftarrow prefix(|\mathcal{L}| + 1, \mathcal{P})$
3 **if** type(m) = LEAVE
4 **then** $MFT_G \leftarrow MFT_G \setminus \mathcal{L}'$
5 **elseif** type(m) = JOIN
6 **then** $MFT_G \leftarrow MFT_G \cup \mathcal{L}'$
7 PREFIX FLOODING m TO $\mathcal{L}$

The distribution of the state update is built upon the PREFIX FLOODING. According to our observations in section 3.2, the BIDIR-SAM join/leave algorithm, thus, terminates and sends the group membership messages to all peers of the 'local' subtree. Further on, it guarantees a multicast spanning tree:

Theorem 4.1 *If the overlay unicast prefix neighbor sets are complete at all nodes, then the multicast join algorithm of BIDIR-SAM constructs a spanning tree at each peer covering all receivers.*

Proof by induction. We assume that the unicast prefix neighbor sets are complete and correct on all overlay nodes. Induction is done with respect to the number of receivers g.

Base case: We consider a multicast group with one receiver, $g = 1$. In this case the multicast listener sends a join message to all overlay peers by using the PREFIX FLOODING (cf. theorem 3.1). According to line 1 and 2 of BIDIR-SAM RECEIVE each peer creates the multicast prefix neighbor towards the receiver by storing the prefix in its multicast forwarding table (cf. line 7). Consequently, the desired forwarding entries are established at all nodes.

Induction step: Assume that the BIDIR-SAM join algorithm creates a spanning tree covering g receivers. We have to show that a join injection for listener $g + 1$ will initiate the required forwarding table entries. According to the induction hypothesis, multicast listeners $1, \ldots, g$ are part of the multicast distribution tree. Among all forwarding entries at node $g + 1$ select one, which attains the longest prefix included in MFT_G (cf. line 3 BIDIR-SAM JOIN/LEAVE INJECTION). Evaluate the initial prefix $\mathcal{C}$ for distributing the join message (cf. line 4).

For the prefix $\mathcal{C}$ either $\mathcal{C} = *$ or $\mathcal{C} \neq *$ holds. The first case is equal to the subscription of the first receiver and all overlay peers add the required entry which is successful according to the base case.

Considering case $\mathcal{C} \neq *$, the join message will be sent to all peers within the subtree rooted at $\mathcal{C}$. Then all overlay nodes, which are located in this prefix subtree will add the prefix neighbor for $g + 1$ according to the prefix flooding (cf. line 5 of BIDIR SAM JOIN/LEAVE INJECTION and line 1 and 2 of BIDIR SAM RECEIVE). As $\mathcal{C} \neq *$, there is at least one receiver j, $1 \leq j \leq g$ inside this subtree, such that the longest common prefix of j and $g + 1$ is $\mathcal{C}$. With respect to the induction hypothesis, j – and hence the subtree of prefix $\mathcal{C}$ – is covered by the multicast forwarding entries of the remaining peers. But members of this subtree will implement a forwarding to node $g + 1$, which thereby is covered, as well. □

The inverse operation of joining a multicast group, to leave it, deletes prefixes from multicast forwarding tables. Leave signaling operates fully symmetric to join. With respect to line 4 of BIDIR-SAM RECEIVE and theorem 4.1, for the leave procedure holds:

Corollary 4.1 *The BIDIR-SAM leave function initiated by a peer $\mathcal{P}$ deletes the corresponding prefix neighbor $\mathcal{C}$ on an arbitrary node $\mathcal{P}'$, if and only if there is no other receiver covered by $\mathcal{C}$.*

It is worth noting that node failures are covered by the BIDIR-SAM maintenance routine (cf. section 5.4.1), which are based on this arguments. In section 4.2.2, we will explain a data redundancy scheme for BIDIR-SAM.

Data Dissemination

Based on its group membership functions, BIDIR-SAM constructs a bi-directional shared tree covering all overlay multicast listeners. The prefix neighbors towards receivers are stored in a decentralized multicast forwarding table MFT_G, which is controlled individually by each overlay node. An arbitrary peer can act as multicast source, while it sends the data to all entries in MFT_G. The packets will then be forwarded to the leaves of the multicast tree (cf. figure 4.1). Conceptually this corresponds to the PREFIX FLOODING approach, whereas branching is guided by the multicast forwarding table:

BIDIR-SAM FORWARDING

▷ On arrival of packet with destination prefix $\mathcal{C}$
▷ for group G at DHT node of ID $\mathcal{K}$
1 **for** all $\mathcal{N}_i$ IDs in MFT_G
2 **do if** $LCP(\mathcal{C}, \mathcal{N}_i) = \mathcal{C}$ ▷ $\mathcal{N}_i$ is downtree neighbor
3 **then** $\mathcal{C}_{new} \leftarrow \mathcal{N}_i$
4 FORWARD PACKET TO $\mathcal{C}_{new}$

As the forwarding algorithm equals the PREFIX FLOODING, only based on a selective routing information base, the observations in section 3.2 hold for BIDIR-SAM, as well: Data is sent to roots of sub-trees known from the multicast forwarding table, which is a subset of the unicast prefix-table. Packet distribution follows the proved routing rules. Thus, all multicast listeners receive the data exactly once and the algorithm terminates.

4.2.2 Optimization Options

The BIDIR-SAM core protocol creates and manages a generic shared family of source trees in prefix space, which allow for unique multicast data transmission from any node in a prefix-optimized fashion. This basic scheme is open to adding optimizations or additional features as desired by the application or network scenario. In the following, we sketch options to optimize underlay performance, i.e., to minimize hops and improve proximity, and to add load sharing and scalable redundancy to the protocol. These improvements come into operation without increasing the BIDIR-SAM signaling load or management overhead.

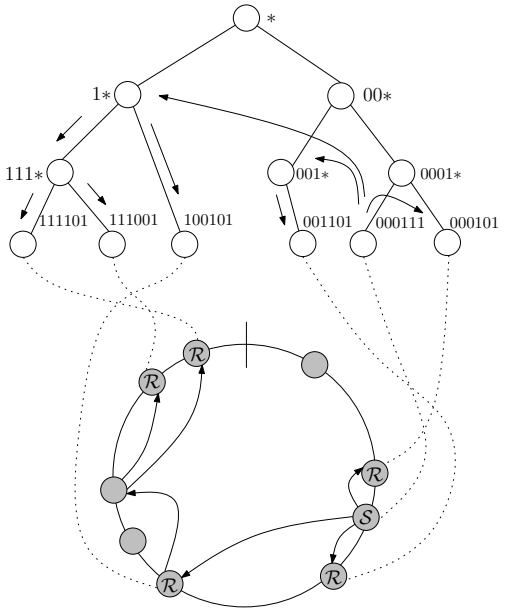


Figure 4.1: BIDIR-SAM Routing Directed by a Binary Prefix Tree

Option 1: Reducing One-way Branches

BIDIR-SAM creates the multicast distribution tree by recursively building longest common prefixes among receivers and its parent vertices. Thus, one-way branches are eliminated and the symbolic path-compressed tree is rooted at "*". However, the underlying prefix routing, which forwards data to an arbitrary overlay node matching the destination prefix, may induce single branching paths. This will be inefficient, if intermediate forwarders do not belong to the multicast group.

An example is the following scenario: Assuming a non-multicast peer sharing the prefix neighbor of the source with its identifier, the unicast prefix-based routing table may include this peer as destination entry. If then only one receiver is located within this subtree, routing proceeds via the non-multicast peer to the receiver and a one-way branch occurs.

To overcome this problem, BIDIR-SAM can be extended to store the IP address of the peer initiating the multicast join in addition to the prefix neighbor identifier in its multicast forwarding table. The data distribution will then proceed according to these underlay information. Thus, packets will be delivered directly to peers, which in turn act as multicast receivers.

With respect to the underlay proximity selection provided by the KBR layer, we suggest a configurable parameter to apply this optimization for prefix neighbors at a specific tree level. As observed in the example of Pastry, proximity selection shows a noticeable effect only for distant keys, i.e., short prefixes [82]. Consequently, the optimization should be used for levels greater than this parameter, e.g., three for Pastry, and may be adjusted for other DHTs.

Option 2: Receiver Proximity Selection

Overlay peers participating in a BIDIR-SAM multicast network will receive membership messages regularly. At an overlay node, the number of joins received from different peers may be significant in the case of large receiver numbers within a prefix neighborhood. These information can be harvested to improve proximity selection.

In detail, a multicast join will be received at the KBR layer and delivered to the upper BIDIR-SAM tier. During this operation, an overlay node may seamlessly interpret the message originator as candidate for proximity selection. In the example of Pastry, the sender of a join would simply be examined for selection into the Pastry neighborhood set. The latter could proceed conventionally by proximity probing or – in a time-synchronized scenario – by time stamps included in join messages. Such continuous proximity harvesting will cause

- a seamless improvement of the general proximity information base in Pastry and thereby increase the unicast overlay routing efficiency;
- an enhanced likelihood of multicast receivers in prefix neighborhood to be chosen by proximity awareness as part of the distribution tree. Hence intermediate non-multicast hops will be implicitly reduced.

From the perspective of a layered overlay architecture as discussed in detail in section 5.1, this approach must be considered as cross-layer design. In cases where layer violations face objections, a receiver proximity selection may be separately implemented by maintaining a receiver neighborhood table on the BIDIR-SAM tier and issuing suitable nodes as `nextHopNode` hint in the message `forward` call of the common API described in section 5.1.

Option 3: Load Sharing via Relays

All peers in an overlay network that operates BIDIR-SAM multicast services is equally suited to serve as a content root for a given group. This capability may be exploited for load sharing purposes in the following way. Any originator of voluminous packet streams e.g., in an IPTV application, may choose one or several relays to assist in data distribution by simply transmitting selected packets with a destination prefix of zero length. A node receiving a packet with empty destination prefix will forward it across the prefix tree root '*', which is equivalent of being the root of the multicast distribution tree. Hence, relays can be activated without signaling.

Utilizing relays will disburden the source of its replication load in parts, but add an extra hop to the transmission path. Further on the originator may receive relayed packets back, whenever it is a receiver or located on the distribution tree of the relay. To avoid packet replay and redistribution, sender and relay can proceed as follows. The source selects a relay that shares a prefix $\mathcal{C}$ as long as possible with its own ID. It then forwards data to the subtree defined by $\mathcal{C}$, while the relay omits packet replication to the same subtree. In proceeding this way, data continues to be disseminated in a unique fashion. As the analysis in sections 4.3.1 and 4.3.2 will reveal, the forwarding and replication load decreases exponentially in prefix length. Consequently, if the sender succeeds in selecting a relay with a long common prefix, it will rarely experience the need of additional packet forwarding.

This relay-based load sharing approach does appear very close to SplitStream [21] when regarded from a superficial perspective. Like BIDIR-SAM load sharing, SplitStream splits streams into slices and distributes slices independently via different rendezvous points. These multiple Scribes are obtained by modifying the initial prefix of the RP address and likewise leads to a unique data dissemination. Aside from additional signaling requirements in SplitStream, however, this approach accumulates the large fluctuations of Scribe with the result of huge additional delays and intolerable jitter at the receiver site. Simulating a group of 256 receiver without node failures, Birrer and Bustamante [14] report on an overall increase of about 100 % in

mean and 1.000 % in standard deviation of the latency. Conversely, BIDIR-SAM load sharing always operates on the identical prefix tree, and admits low, rigorously bound performance variations. Relaying packets will add the delay of a single, unicast hop and will retain its overall performance.

Option 4: r-Redundancy for Data and Paths by Network Coding

BIDIR-SAM multicast delivers packets uniquely via a well defined, deterministic tree. In the presence of churn, node and link failures or other disturbances it is desirable to add a certain degree of data redundancy to the distribution system. More precisely, an r-redundancy tolerating the loss of one packet out of each sequence of r may be sufficient to sustain information integrity at an appropriate degree of confidence.

An efficient method to procure redundancy at the packet level has been recently introduced by network coding [6, 58]. Its underlying idea is to create a redundant packet from a sequence of r by 'adding' them all using XOR operations. If the resulting $r+1$ packets are transmitted, all original r packets can be recovered under the loss of any one of the transmitted datagrams. A straight application of this scheme leads to an r-redundancy for data.

Many disruptive scenarios like link and node failures are likely to cause damage at more than one occasional packet. Redundant paths are required to circumvent these defects. BIDIR-SAM provides two natural options for path redundancy on top of DHT error resilience. At first, if a forwarding node experiences a neighboring link or node failure, it can pass packets with unchanged destination prefix to any other overlay node sharing the same prefix. The receiver will then act as a relay and continue to disseminate the traffic flow substitutional. At second, in scenarios where loss of the UDP data cannot be foreseen, e.g., in wireless mesh networks, r-redundant distribution paths can be organized by using an approach similar to the load sharing.

A source willing to distribute data at an r-redundant level will perform network coding for r-sequences of its packet stream. It will further select r relays, preferably of differing prefix initials, as discussed in the previous option on load sharing. Each sequence of r packets as well as the coded datagram are then distributed among the $r+1$ senders. Choosing senders of different prefix initials will minimize a coincidence of the $r+1$ prefix trees and will thus lead to the highest frequency of disjoined paths. Any node/link failure occurring on a single overlay distribution tree will then cause loss of one packet per r-sequence and can be fully compensated by the remaining r data units. Thus BIDIR-SAM can provide a full r-redundancy in data and paths without modification or additional signaling.

4.3 Performance Analysis

In this section we will analyze the multicast performance of BIDIR-SAM in comparison with the rendezvous point-based scheme Scribe. Both approaches require an explicit group management to dynamically construct the distribution tree. In addition to the metrics introduced in section 3.3, i.e., the **traffic load**, **replication load** and **hop count**[1], we therefore extend the measurement to the following quantities:

Multicast forwarding entries corresponds to the number of downstream entries required at a peer. This value represents the storage requirements at a multicast peer. It also

[1] We omit the travel time and delay penalty, as results follow directly from the broadcast scenario.

characterizes the number of children per overlay node in the distribution tree. Thus, it describes an upper bound for the packet replications in BIDIR-SAM and corresponds to the replication load in Scribe.

Signaling load measures the average number of join messages initiated by the multicast routing protocol in response to the subscription of a new multicast listener. This absolute value quantifies the cost at peers of incorporating a new receiver in the multicast delivery tree.

Join injection level describes the position at which a multicast subscription will be injected into the path compressed prefix tree. This metric is only applicable in BIDIR-SAM and reflects the height of subtrees, which need to be provisioned by a join.

Replication load per tree level counts the number of packet replications for a multicast forwarder at a specific level in the distribution tree. This differential measure characterizes the branching shape of the source specific tree.

Forwarding fairness is the ratio of multicast receivers acting as forwarders and the overall number of forwarders. This metric expresses the fairness of group members over non-multicast nodes in an overlay, which maintains multiple services.

Multicast efficiency defines similar to [26] the ratio of the average number of traversed *overlay* hops by distributing the data via multicast and the average *overlay* unicast path length. This normalized measure reflects the economic effect of multicast over repeated unicast.

4.3.1 Analytical Results

The theoretical analysis of the BIDIR-SAM multicast scheme will require more subtle considerations of the operations in prefix space than in the case of prefix flooding. The more complex modeling approach pursued in the following admits the potential of explaining processes more accurately and reproducing simulation values in significant detail, but expressions partly fail to reduce to simple closed forms.

To clarify the underlying model of our analysis, we will first give an overview of the concepts and notations used further on. Common properties of the prefix-based group model are outlined, which will be needed in the following line of arguments.

The General Model and Basic Properties

For a given key space of alphabet size k and key length h, we consider the corresponding k-ary prefix tree as basic structure, cf. section 3.2. Therein N overlay nodes $\{\mathcal{N}\}$ are placed at leaf nodes of the prefix tree, such that their keys are uniformly chosen from the prefix alphabet. In particular, for any node K with key $\mathcal{K}$, the probability of attaining a specific digit x reads

$$P(\delta_i(\mathcal{K}) = x) = \frac{1}{k}, \text{ where } \delta_i(\mathcal{K}) \text{ denotes the } i\text{−th digit of } \mathcal{K} \tag{4.1}$$

Consider an arbitrary prefix $\mathcal{C}$ of length j. The probability for a random overlay node to share this prefix equals $(\frac{1}{k})^j$. Inspecting the distribution of all overlay nodes in prefix space, it immediately follows that any (ordered) sequence of keys, l keys with prefix $\mathcal{C}$ and $N - l$ keys

not sharing $\mathcal{C}$, occurs with probability $(\frac{1}{k^j})^l(1-\frac{1}{k^j})^{N-l}$. Accounting for all possible orderings, this yields the node distribution in prefix space,

$$P\left(|\{\mathcal{N}\in\{\mathcal{N}\}|\ LCP(\mathcal{C},\mathcal{N})=\mathcal{C}\}|=l\right)=\binom{N}{l}\left(\frac{1}{k^j}\right)^l\left(1-\frac{1}{k^j}\right)^{N-l}, \tag{4.2}$$

which is Binomial with mean $N\left(\frac{1}{k^j}\right)$ and variance $N\left(\frac{1}{k^j}\right)\left(1-\frac{1}{k^j}\right)$ [2].

Recalling that the prefix $\mathcal{C}$ of length j correspondents to the root of a subtree T_{h-j} of height $h-j$ as visualized in figure 3.4, equation 4.2 also describes the distribution of nodes populating this subtree.

The keys representing the overlay nodes span a *random recursive k-ary tree* with *inhomogeneous branching rates* $p_j(k-1)$, which extends the discussion of section 3.3.1. Consider an inner vertex representing the prefix $\mathcal{C}$ at level $j-1$ on the path to a given node S with key $\mathcal{S}$. The prefix tree will branch at this vertex, if a node K with key $\mathcal{K}$ exists, such that $LCP(\mathcal{S},\mathcal{K})=\mathcal{C}$. The latter is equivalent to the existence of a node that attains a dedicated prefix of length $j-1$, and any of $k-1$ from k values at the j-th digit. The probability that none of the $N-1$ remaining nodes carry a dedicated prefix of length j equals $\left(1-\frac{1}{k^j}\right)^{N-1}$. Hence the branching probability of the overlay prefix structure at level $j-1$ reads

$$P(\text{Branch at niveau } j-1)=\left(1-\left(1-\frac{1}{k^j}\right)^{N-1}\right)\cdot(k-1)=p_j\cdot(k-1). \tag{4.3}$$

Values of p_j are not constant, but decrease exponentially in j, leading to high branching probabilities close to the tree root, but rapidly decaying as the prefix tree is descended. Note that a consistent transition to constant branching probabilities at a fully populated prefix tree derives from the limit $\left(1-\left(1-\frac{1}{k^j}\right)^{N-1}\right)\longrightarrow_{N\to\infty}1$.

Any multicast group arranges within this overlay prefix structure. Consider a group G of g receivers. We assume that receivers are independently chosen among overlay nodes with the uniform probability $r_g=\frac{g}{N}$.[2]

The BIDIR–SAM algorithm aggregates multicast receivers according to longest prefixes. For a given prefix $\mathcal{C}$ of length j, the probability that a receiver shares $\mathcal{C}$ is therefore of general relevance.

Theorem 4.2 *For a multicast group G resident in a prefix-structured overlay of k-ary alphabet and N nodes, the probability that a given prefix $\mathcal{C}$ of length j is attained by at least one out of g receivers is given by*

$$\begin{aligned} P\left(|\{\mathcal{G}\in G|\ LCP(\mathcal{C},\mathcal{G})=\mathcal{C}\}|\geq 1\right) &= 1-\left(1-\frac{g}{k^jN}\right)^N && (4.4)\\ &= 1-e^{-\frac{g}{k^j}}+\mathcal{O}\left(\frac{1}{N}\right) && (4.5)\end{aligned}$$

Proof. Assume the number $N_{\mathcal{C}}$ of nodes with prefix $\mathcal{C}$ in the overlay equal to l. The conditional probability that none of the nodes is a multicast receivers then reads

$$P\left(|\{\mathcal{G}\in G|\ LCP(\mathcal{C},\mathcal{G})=\mathcal{C}\}|=0\parallel N_{\mathcal{C}}=l\right)=(1-r_g)^l \tag{4.6}$$

[2] This assumption is supported in both, theory by [71] and Internet measurements by [24].

Using equation 4.2, the unconditional probability that no receiver shares $\mathcal{C}$ evaluates to

$$\begin{aligned}
&P\left(|\left\{\mathcal{G}\in G|\ LCP(\mathcal{C},\mathcal{G})=\mathcal{C}\right\}|=0\right)\\
&=\sum_{l=0}^{N}P\left(|\left\{\mathcal{G}\in G|\ LCP(\mathcal{C},\mathcal{G})=\mathcal{C}\right\}|=0\parallel N_{\mathcal{C}}=l\right)\cdot P(N_{\mathcal{C}}=l)\\
&=\sum_{l=0}^{N}(1-r_g)^l\binom{N}{l}\left(\frac{1}{k^j}\right)^l\left(1-\frac{1}{k^j}\right)^{N-l}\\
&=\sum_{l=0}^{N}\binom{N}{l}\left(\frac{1-r_g}{k^j}\right)^l\left(1-\frac{1}{k^j}\right)^{N-l}\\
&=\left(\frac{1-r_g+k^j-1}{k^j}\right)^N=\left(1-\frac{g}{k^jN}\right)^N,
\end{aligned}$$

where the last line was obtained by evaluating the binomial expansion series [2]. Taking the complementary weight and observing that $e^x=\left(1+\frac{x}{N}\right)^N+\mathcal{O}(\frac{1}{N})$ proves the theorem. □

It is worth noting that in large overlay networks the prefix distribution of multicast receivers is effectively independent of the overlay size.

Size of Multicast Forwarding Tables

The multicast distribution tree forms a subtree within the entire prefix tree and shares the structural properties of the overlay. Restricting considerations to this substructure, we can adapt theorem 3.3, which was not bound to modeling assumptions.

Theorem 4.3 *For any overlay node in a k-ary prefix tree with g leaf nodes (receivers), the number of adjacent vertices is limited by* $\log_2(g)(k-1)$. *This bound equally limits the number of multicast forwarding table entries.*

Proof. Any overlay node is situated as a leaf in the prefix tree and has all vertices on the shortest path to the root associated with it. Thus the number of neighbors equals the sum of the neighbors at each associated vertex. For an alphabet of base k the latter is bound by $k-1$. The number of vertices towards the tree root is limited by the height of the path compressed tree, which is maximal when all branches are binary. Consequently for a prefix tree with g leaves, the maximal height is given by $\log_2(g)$. Forwarding within the overlay is solely performed to on-tree neighbours, whose IDs are the states any overlay member needs to store in its forwarding table.

□

Returning to our prefix distribution model, we now want to determine the distribution of multicast forwarding states on the prefix tree. At every level j of the prefix tree, an overlay node may face 0 to $k-1$ neighboring vertices connecting different receivers, which follow a binomial distribution:

Theorem 4.4 *In the BIDIR-SAM multicast scheme of a group with g receivers, the probability distribution* $P(j,l)$ *that a given overlay node holds l multicast forwarding entries at prefix level j reads*

$$P(j,l)=\binom{k-1}{l}\left(1-e^{-\frac{g}{k^{j+1}}}\right)^l\left(e^{-\frac{g}{k^{j+1}}}\right)^{k-1-l}+\mathcal{O}(\frac{1}{N}) \tag{4.7}$$

Proof. For a given node consider the possible vertices connecting to the $k-1$ subtrees at level j. A forwarding state for a particular vertex will be required, if and only if a receiver exists in the corresponding subtree. Being member of a particular subtree with root at level j is equivalent to carrying a prefix of length $j+1$, its probability was given in equation 4.5.

For any individual selection of l among the $k-1$ vertices, the probability of attaining these l forwarding states is approximated by $\left(1-e^{-\frac{g}{k^{j+1}}}\right)^{l}\left(e^{-\frac{g}{k^{j+1}}}\right)^{k-1-l}$. Adding all possible orderings proves the theorem. □

The multicast forwarding table of a node contains the entries for neighbors at all levels of the prefix tree. With the help of theorem 4.4 we are now able to compute the mean value of the table sizes along with its standard deviation:

Corollary 4.2 *Denote by $MFT(g)$ the multicast forwarding table size of a node in an overlay participating in the BIDIR-SAM multicast with g group members. Then*

$$< MFT(g) > = \sum_{j=1}^{h}(k-1)\left(1-e^{-\frac{g}{k^j}}\right) \tag{4.8}$$

$$\sigma(MFT(g)) = \left(\sum_{j=1}^{h}(k-1)\left(1-e^{-\frac{g}{k^j}}\right)\left(e^{-\frac{g}{k^j}}\right)\right)^{1/2} \tag{4.9}$$

Proof. According to the binomial distribution 4.7, the average number of table entries for a given subtree level j is given by $(k-1)\left(1-e^{-\frac{g}{k^{j+1}}}\right)$. The average total number of states evaluates as the sum of the averages over all levels $(0 \ldots h-1)$, which proves equation 4.8.

The variance of the table distribution at level j is given by $(k-1)\left(1-e^{-\frac{g}{k^{j+1}}}\right)\left(e^{-\frac{g}{k^{j+1}}}\right)$. It follows from the independence assumption in prefix selection that the total variance can be calculated as the sum over the conditional variances per level. Taking the square root yields the standard deviation 4.9. □

Unfortunately there are no closed expressions for the above results and we are unable to find a valid asymptotic expansion. The mean function is plotted in figure 4.2. Table entries remain significantly below its upper bound given in theorem 4.3, reproducing nicely the logarithmic dependency on g. The growth with the prefix alphabet size k remains sublinear.

Replication Load

In this section we want to quantify the replication load of the multicast data distribution. Its maximal value is defined by the number of forwarding table entries and comes into effect with a destination prefix of zero length. Routing from zero prefixes occurs only at the multicast source and leads to the immediate implication of theorem 4.3:

Corollary 4.3 *The multicast replication load for any overlay node remains less or equal to* $\log_2(g)(k-1)$.

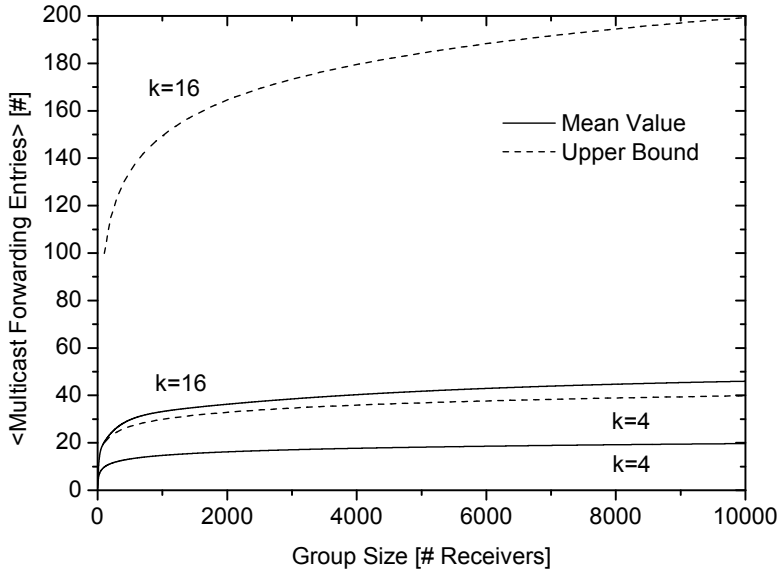


Figure 4.2: Mean Entries and Upper Bounds of the Multicast Forwarding Tables as a Function of Receiver Numbers for Alphabets of $k = 4$ and $k = 16$.

Following the line of the above argument, the replication load at the source is described by the full multicast forwarding tables and estimated by the quantities of corollary 4.2. In the general case, multicast forwarding occurs in combination with a destination prefix of length j, which rules out all table entries of shorter prefix length. Consequently, replication of packets with a destination prefix of length j is only performed for those table entries of prefix longer than j, which immediately yields the following estimates from corollary 4.2:

Corollary 4.4 *Denote by $RPL(j,g)$ the multicast replication load at a node in an overlay participating in the BIDIR-SAM multicast with prefix length j. Then*

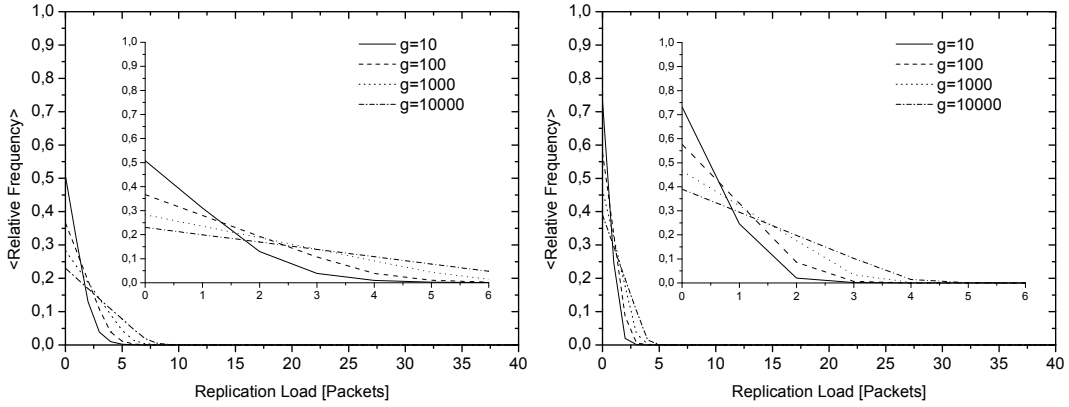
$$< RPL(j,g) > \;=\; \sum_{i=j}^{h}(k-1)\left(1-e^{-\frac{g}{k^i}}\right) \tag{4.10}$$

$$\sigma(RPL(j,g)) \;=\; \left(\sum_{i=j}^{h}(k-1)\left(1-e^{-\frac{g}{k^i}}\right)\left(e^{-\frac{g}{k^i}}\right)\right)^{1/2} \tag{4.11}$$

Characteristic distributions of the replication load are drawn in figure 4.3, representing a fair balance up until 0,6 child nodes for larger networks. As compared with the prefix flooding, gradients are less pronounced at low values and show reduced steepness with increasing group size. Smaller alphabets noticeably smoothen the distributions, which suggests k to serve as a tuning parameter of the multicast distribution tree.

Signaling Load

Signaling in the BIDIR-SAM scheme consists of the JOIN and LEAVE messages, which are flooded to context-specific subtrees of the overlay. In this section we want to calculate the



(a) $k = 4$ (b) $k = 16$

Figure 4.3: Normalized Distributions of the Mean Replication Load for Different Group Sizes g and Prefix Alphabets k.

distribution of the prefix flooding along with the expected number of flooded nodes. Due to the symmetry of these operations, the following analysis is restricted to JOIN.

Consider an established group G of g receivers in the overlay network. A node newly joining group G will change the group members to $g+1$ in distributing its JOIN to the smallest subtree containing its own ID and at least one previous receiver. The probability $P(j, g)$ that a JOIN injection occurs at level j, or at a subtree of height $h - j$, is equal to the probability that one of the previous g group members shares the prefix of length j with the newly joining node, but none does with the extended prefix of length $j + 1$. Hence using equation 4.5, we derived

Theorem 4.5 *The probability $P(j, g)$ for distributing a BIDIR-SAM* JOIN *or* LEAVE *message within a prefix tree at injection level j reads*

$$P(j,g) = \left(1 - e^{-\frac{g}{k^j}}\right) e^{-\frac{g}{k^{j+1}}} + \mathcal{O}(\frac{1}{N}), \tag{4.12}$$

where g is the number of group members prior to signaling.

From the distribution 4.12, expressions for the expected size of the flooded subtree can be deduced, as well as the expected number of nodes therein.

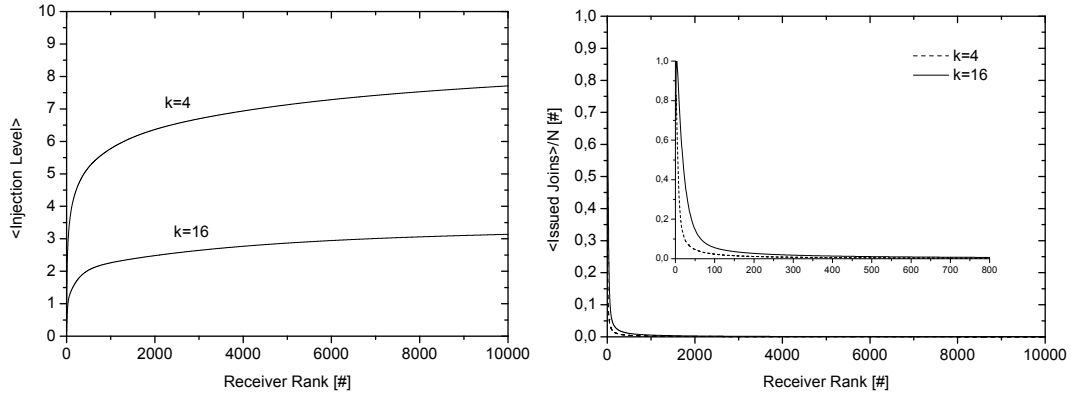
Corollary 4.5 *The expected injection level of the prefix tree for BIDIR-SAM* JOIN *or* LEAVE *signaling is given by*

$$\sum_{j=0}^{h} j \left(1 - e^{-\frac{g}{k^j}}\right) e^{-\frac{g}{k^{j+1}}}, \tag{4.13}$$

while the expected number of flooded nodes is well approximated by

$$N \left\{ (1 - e^{-g}) e^{-\frac{g}{k}} + \frac{k}{g(k+1)\ln k} \left(\left(e^{-\frac{g}{k^{h+1}}} - e^{-\frac{g}{k}} \right)(k+1) + e^{-\frac{g(k+1)}{k}} - e^{-\frac{g}{k^{h+1}}(k+1)} \right) \right\}, \tag{4.14}$$

where g is the number of group members prior to signaling.



(a) Injection Level (b) Effective Joins per Receiver

Figure 4.4: Mean Injection Level and Normalized Message Numbers of the Multicast Join/Leave Signaling as a Function of Receiver Rank for Alphabets of $k = 4$ and $k = 16$.

Proof. Given the probability distribution $P(j, g)$ of BIDIR-SAM signaling, the mean injection level immediately evaluates to $\sum_{j=0}^{h} jP(j,g) = \sum_{j=0}^{h} j \left(1 - e^{-\frac{g}{k^j}}\right) e^{-\frac{g}{k^{j+1}}}$.

To derive the mean value of flooded nodes, we first refer to the Binomial distribution 4.2 of nodes populating a prefix subtree of height $h - j$. Hence the conditional expectation of nodes below a given prefix level j equals $N(\frac{1}{k^j})$. The unconditional expectation of nodes flooded with signaling messages then calculates as follows:

$$\begin{aligned}
&\sum_{j=0}^{h} N\left(\frac{1}{k^j}\right) P(j,g) \\
&= N \sum_{j=0}^{h} \left(\frac{1}{k^j}\right) \left(1 - e^{-\frac{g}{k^j}}\right) e^{-\frac{g}{k^{j+1}}} + \mathcal{O}(\frac{1}{N} \\
&\approx N \left\{ (1 - e^{-g}) e^{-\frac{g}{k}} + \int_{j=0}^{h} \left(\frac{1}{k^j}\right) \left(1 - e^{-\frac{g}{k^j}}\right) e^{-\frac{g}{k^{j+1}}} dj \right\} \\
&= N \left\{ (1 - e^{-g}) e^{-\frac{g}{k}} + \frac{k}{g(k+1)\ln k} \left(\left(e^{-\frac{g}{k^{h+1}}} - e^{-\frac{g}{k}} \right) (k+1) + e^{-\frac{g(k+1)}{k}} - e^{-\frac{g}{k^{h+1}}(k+1)} \right) \right\},
\end{aligned}$$

where the integral was obtained from the Euler summation formula and solved using the substitution $x = \frac{1}{k^j}$. □

The results of corollary 4.5 are displayed in figure 4.4 as functions of the joining receiver rank. The essentially complementary quantities admit a logarithmic growths in the injection level, i.e., the level of descend within the prefix tree, on the one hand, and a strong exponential decay in the expected number of flooded nodes on the other. The mean number of messages issued for Join/Leave signaling reduces to below 10 in network sizes above 1.000. These results, which are well reproduced in the simulations, remain compatible with the low signaling costs of Scribe. Signaling expenses decrease almost linearly with k. Again, k serves as a tuning parameter acting in the same direction as for the replication load.

Hop Count

The prefix routing of the BIDIR-SAM multicast scheme coincides with the prefix flooding described in section 3.2. Multicast receivers are selected from overlay nodes at random without inducing bias. Consequently, the overlay hop count distribution of packets reaching multicast receivers is identical to the prefix flooding case and we immediately inherit the general estimates from theorem 3.5:

Theorem 4.6 *Any multicast receiver in an overlay of N receivers that performs a prefix routing using an alphabet of $k \geq 2$ digits will receive a packet after at most $\log_2(N)$ hops. In the presence of Pastry overlay routing, the number of hops attained on average equals $\log_{2^b}(N)$ with $k = 2^b$.*

In a typical overlay multicast setup, a group G will admit a number of g receivers, which is small compared to the total number N of overlay nodes. N in turn falls short with respect to the prefix address space, why sparsity of receivers in the distribution tree becomes more important. Compliant with the model outlined above, we thus want to derive a hop count distribution that represents sparsely scattered receivers in a prefix tree more closely than the homogenous branching model of section 3.3.1.

On the path from the source to the receivers, a multicast packet traverses an overlay hop, whenever the distribution tree branches at the corresponding prefix $\mathcal{C}$. Taking the branching rate given in equation 4.3, the corresponding recurrence relation of the hop count frequency can be written as

$$\begin{aligned} f_{h,k,N}(j) &= f_{h-1,k,N}(j) + \left(1 - \left(1 - \frac{1}{k^j}\right)^{N-1}\right) \cdot (k-1) \cdot f_{h-1,k,N}(j-1) \qquad (4.15) \\ &\quad \text{with } f_{1,k,N}(0) = 1, f_{1,k,N}(1) = \left(1 - \left(1 - k^{-1}\right)^{N-1}\right)(k-1). \end{aligned}$$

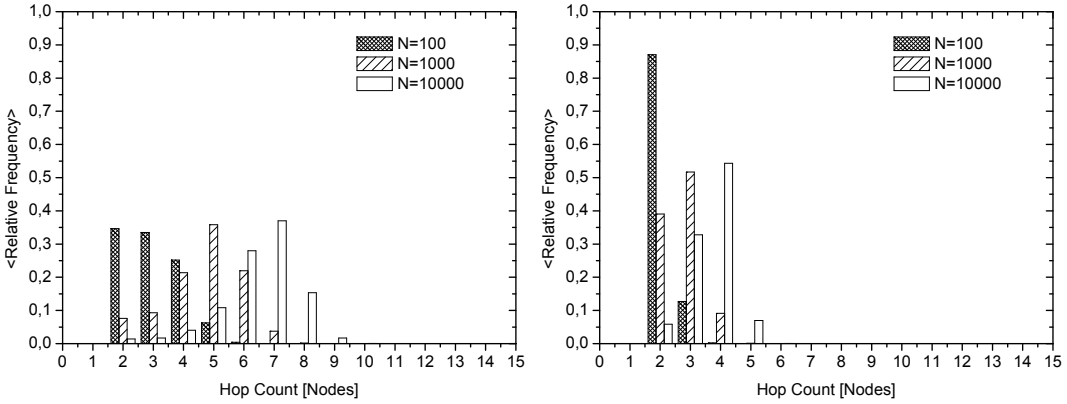
Solving the recursion leads to

Theorem 4.7 *The hop count frequency $f_{h,k,N}$ attained at prefix routing on N overlay nodes with independent uniformly distributed identifiers is given by*

$$f_{h,k,N}(j) = \binom{h}{j} \cdot \prod_{i=0}^{j} \left(1 - \left(1 - k^{-i}\right)^{N-1}\right) \cdot (k-1)^j. \qquad (4.16)$$

Proof. In the general model outlined above, we have derived that an independent uniform distribution in key space generates branching probabilities as defined in equation 4.3. Thus it remains to be shown that $f_{h,k,N}(j)$ satisfies the rate equation 4.15:

$$\begin{aligned} f_{h,k,N}(j) &= \binom{h}{j} \cdot \prod_{i=0}^{j} \left(1 - \left(1 - k^{-i}\right)^{N-1}\right) \cdot (k-1)^j \\ &= \binom{h-1}{j} \cdot \prod_{i=0}^{j} \left(1 - \left(1 - k^{-i}\right)^{N-1}\right) \cdot (k-1)^j \\ &\quad + \left(1 - (1 - k^{-j})^{N-1}\right)(k-1) \cdot \binom{h-1}{j-1} \cdot \prod_{i=0}^{j-1} \left(1 - \left(1 - k^{-i}\right)^{N-1}\right) \cdot (k-1)^{j-1} \\ &= f_{h-1,k,N}(j) + \left(1 - \left(1 - \frac{1}{k^j}\right)^{N-1}\right) \cdot (k-1) \cdot f_{h-1,k,N}(j-1), \end{aligned}$$



(a) $k = 4$ (b) $k = 16$

Figure 4.5: Normalized Hop Count Distributions for $k = 4$ and $k = 16$.

which proves the theorem. □

The hop count frequency $f_{h,k,N}(j)$ is plotted in figure 4.5 in normalized form. These (also numerically) delicate distributions are in significantly better agreement with the simulation values as the previous results in the prefix flooding model, cf. figure 3.6. Mean and width of the distributions grow as k decreases, acting in opposite direction of the branching properties investigated above. A BIDIR-SAM instance optimizing replication and signaling load by using a small prefix alphabet will encounter a moderate increase of routing hops in packet delivery.

Multicast data may be forwarded by group members, as well as by uninvolved overlay nodes. If we restrict our consideration to the members of a given group G involved in forwarding, the probability of branching at a prefix of length j changes to expression 4.5, and the hop count recurrence relation 4.15 turns into

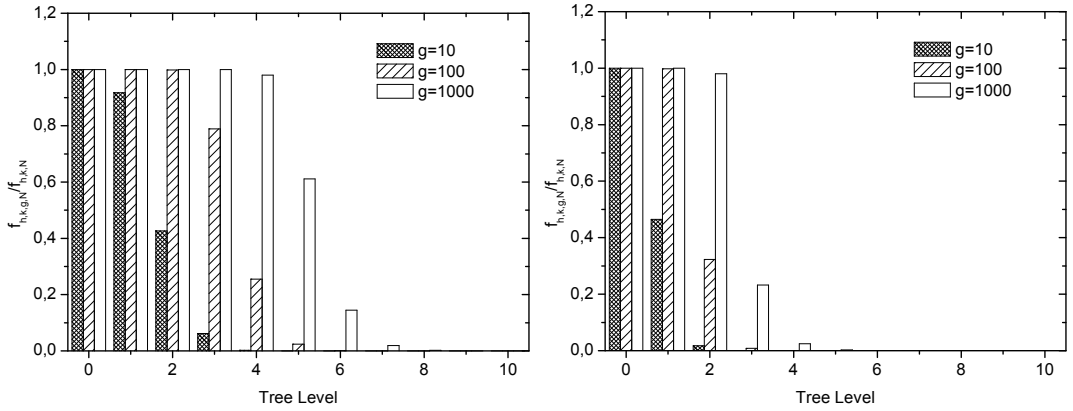
$$\begin{aligned} f_{h,k,g,N}(j) &= f_{h-1,k,g,N}(j) + \left(1 - e^{-\frac{g}{k^j}}\right) \cdot (k-1) \cdot f_{h-1,k,g,N}(j-1) + \mathcal{O}\left(\frac{1}{N}\right) \quad (4.17) \\ &\text{with } f_{1,k,g,N}(0) = 1,\ f_{1,k,g,N}(1) = \left(1 - e^{-\frac{g}{k}}\right)(k-1). \end{aligned}$$

Using the identical line of arguments as in the proof of theorem 4.7 yields

Theorem 4.8 *The hop count frequency $f_{h,k,g,N}$ of traversing g multicast receivers by prefix routing in an overlay network of N nodes with independent uniformly distributed identifiers is given by*

$$f_{h,k,g,N}(j) = \binom{h}{j} \cdot \prod_{i=0}^{j} \left(1 - e^{-\frac{g}{k^i}}\right) \cdot (k-1)^j + \mathcal{O}\left(\frac{1}{N}\right). \quad (4.18)$$

The ratio of forwarders, which are already receivers, over all routing nodes is of particular interest, as this may serve as a measure of fairness, while at the same time multicast forwarding and delivery coincide.



(a) $k = 4$ (b) $k = 16$

Figure 4.6: Fraction of Multicast Group Members among Forwarders in a Network of $N = 10.000$ Nodes for $k = 4$ and $k = 16$.

Corollary 4.6 *The fraction of multicast receivers acting as data forwarder in the prefix routing scheme evaluates to*

$$\frac{f_{h,k,g,N}}{f_{h,k,N}}(j) = \prod_{i=0}^{j} \frac{\left(1 - e^{-\frac{g}{k^i}}\right)}{1 - (1 - k^{-i})^{N-1}} = \prod_{i=0}^{j} \frac{1 - k^i + e^{-\frac{g}{k^i}}(k^i - 1)}{1 - k^i + k^i(1 - k^{-i})^N} \tag{4.19}$$

The portion of multicast forwarders, which are group members themselves, is shown in figure 4.6. Receivers reached within a small number of hops experience a high probability of traversing only receivers, while packets traveling on longer paths are more likely to utilize noninvolved forwarders. Larger group sizes and smaller prefix alphabets reduce this degree of 'unfairness', so that in a setup of $g = 1000$, $k = 4$, $N = 10.000$ clearly 50% of the packets are delivered on average for up to 5 hops solely performed via multicast group members.

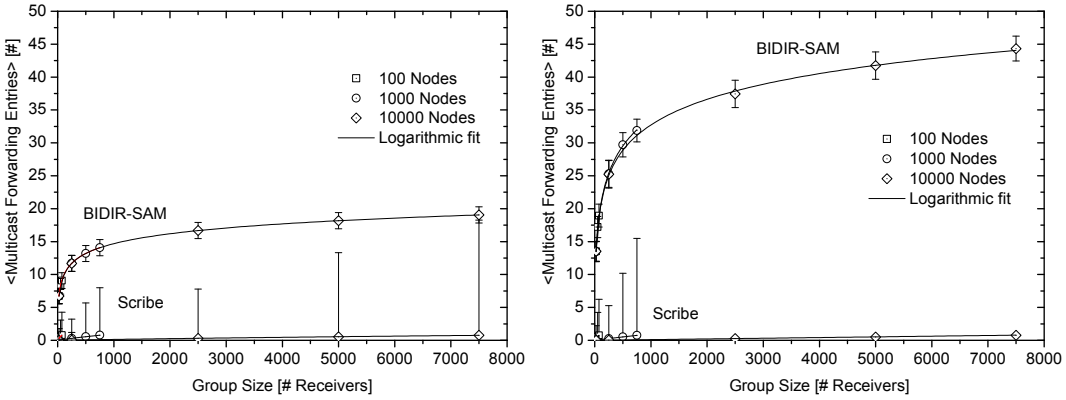
4.3.2 Simulation Results

In this section we will analyze the performance of BIDIR-SAM in comparison with a rendezvous point-based overlay multicast approach. To stay consistent with the prefix flooding (cf. section 3.3.2), for the latter, Scribe is chosen in its standard implementation [22]. Unless denoted otherwise, BIDIR-SAM is likewise used in its standard, unoptimized version. BIDIR-SAM as well as Scribe are implemented based on the key-based routing implementation Pastry. The multicast simulation starts after a proactive routing maintenance has filled complete Pastry routing tables. The costs for this additional routing maintenance are low, as incomplete tables rarely occur.[3]

The simulations are performed on the same network simulator platform as the prefix flooding, OMNeT++ 3.3 [99], extended by the OverSim-20080416 [10] package.

The general simulation setup corresponds to our analysis of the prefix flooding (cf. section 3.3.2): Pastry is configured to its original version with a key length of 128 bit and a varying prefix alphabet size. To concentrate on structural insights in the multicast routing protocols, we

[3] In a typical Pastry routing table, many entries remain empty due to non-existent node keys. This can be a priorily conclude from optimized probing as described in section 5.4.3.



(a) $k = 4$ (b) $k = 16$

Figure 4.7: Mean Multicast Forwarding Entries per Overlay Node for Prefix Alphabet Sizes k and Varying Overlay Dimensions

select again the Simple model of OverSim [9] as underlying network. We will analyze reliability aspects in future work, thus, we neglect any churn. For additional reasons we refer to section 3.3.2.

The simulations are conducted for a small, medium and large overlay of 100, 1.000 and 10.000 nodes.[4] Among all peers, one multicast source is chosen uniformly distributed, which sends its data to a multicast group selected with equal weights. Receivers are also picked up uniformly distributed, but distinct from the source. Group sizes vary from 25 %, 50 % to 75 % subscription ratio. Each scenario is sampled with the same parameter settings until it is converged. We average the results over all samples with the same settings. Again, this is detailed out in section 3.3.2.

The implementation of BIDIR-SAM has been tested by simulating the broadcast scenario based on an all peer subscription. The outcomes confirm with our results for the prefix flooding. Further on, repeated manual checks have been performed for small networks.

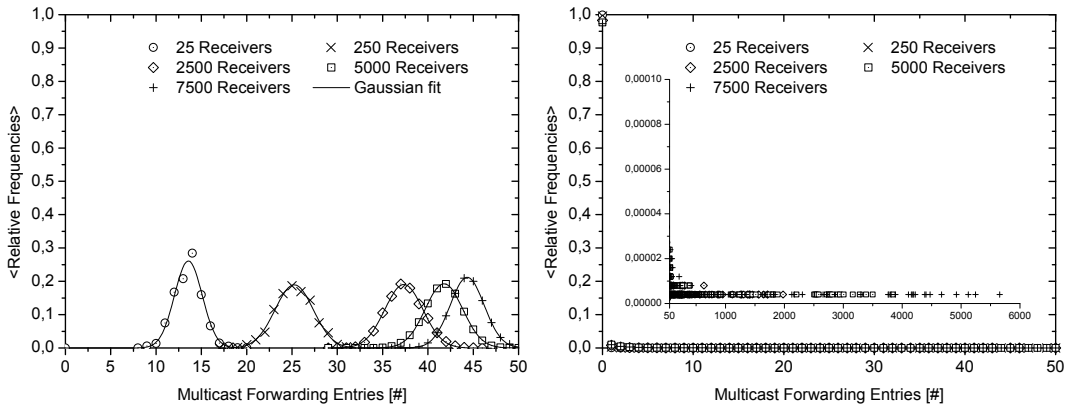
Multicast Forwarding Table Size

The average multicast forwarding table size is visualized in figure 4.7 as a function of the number of receivers for different network sizes. Both schemes clearly scale independently of the overlay dimension due to the local view of multicast forwarders.

Focusing on mean values, Scribe outperforms BIDIR-SAM as the average number of entries grows only marginally with the group size and remains below 5. However, all BIDIR-SAM tables increase with strict logarithmic bounds with the number of receivers, which complies with the scaling properties of the underlying DHT. It is worth noting that the additional entries in BIDIR-SAM provide inherent redundancy as distributed prefixes cover multiple peers.

Although the average number of tables entries in Scribe is almost constant, the fluctuation per node is significant. As indicated by the error bars in figure 4.7, the standard deviation may be larger by one order of magnitude than the average value. Maximal values for Scribe range up to 5600 entries for large overlays with a high receiver subscription ratio. Figure 4.8 reveals that

[4] In general, an overlay consisting of 10 nodes represents an unusual deployment scenario and was omitted in this analysis.



(a) BIDIR-SAM (b) Scribe

Figure 4.8: Distribution of Multicast Forwarding Entries per Overlay Node for Prefix Alphabet Size $k = 16$ and Varying Number of Receivers, Cut at 50 Entries With a Detail View for Scribe

the distribution of group states is extremely wide in Scribe. Almost all peers keep a multicast forwarding table without entries, but some dedicated overlay nodes maintain single states for up to 80 % of the receivers. In contrast, BIDIR-SAM nicely fits the normal distribution and consequently fills its routing tables in a balanced mode.

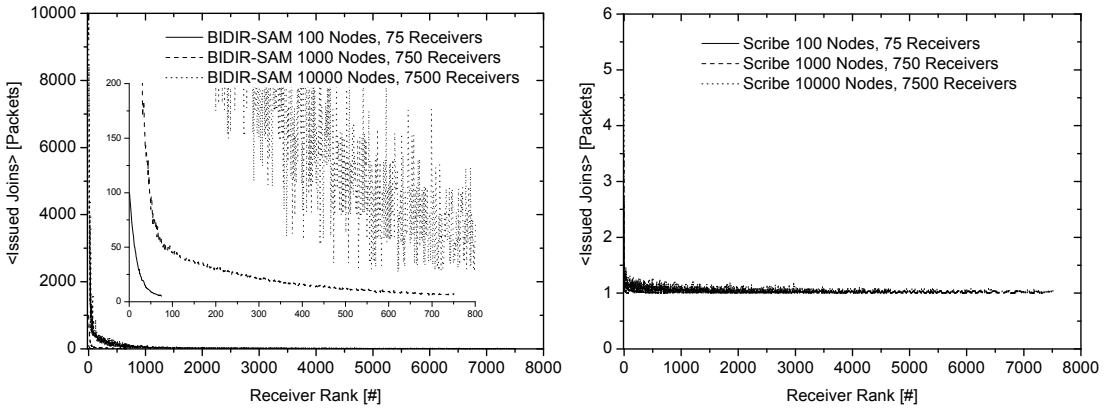
Signaling Load

Figure 4.9 displays the average signaling load depending on the receivers ranked by their subscription order. The number of issued joins is noticeably higher in BIDIR-SAM as in Scribe, which is also indicated by the different axis scales. This behavior reflects directly the different underlying group management algorithms and corresponds to our analysis of the multicast forwarding table size. In BIDIR-SAM, each new receiver floods a prefix subtree of different height, whereas Scribe submits single subscriptions on unicast paths towards the rendezvous point.

In BIDIR-SAM, the first overlay multicast listener needs to inform all peers about its subscription. The signaling load then decays exponentially with the number of receivers. For larger multicast groups with a receiver to overall overlay node ratio of more than 50 % BIDIR-SAM approximates asymptotically the signaling load of Scribe. Any BIDIR-SAM peer, however, owns at this time a richer routing table regarding the overall prefix tree. While Scribe states construct a single shared tree, which may break into incoherent parts, whenever intermediate states are lost, BIDIR-SAM distributes its states to all nodes resulting in redundant source-specific trees at each node.

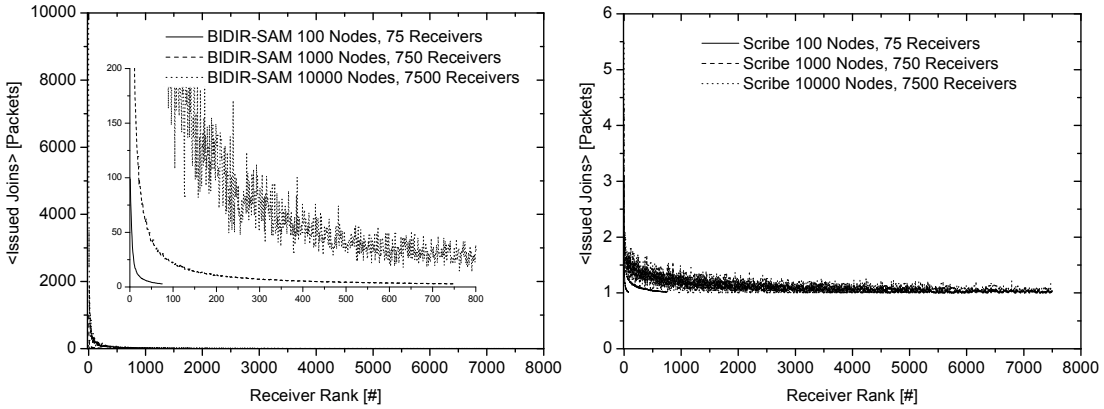
It is worth noting, that the standard deviation in BIDIR-SAM is one order of magnitude below the group size for the first 30 to 60 receivers and drops to three orders of magnitude below for larger group sizes. This reflects the case that receiver peers may be located in a prefix vicinity, which consequently results in a small subtree to flood.

The number of issued joins can be tuned by adjusting the prefix alphabet size (cf. figure 4.9(c) and 4.9(d)). Using a lower k accelerates BIDIR-SAM convergence to a comparable number of subscription messages with respect to Scribe. Scribe shows an opposite effect and increases slightly with the submitted joins as paths to the rendezvous point enlarge.



(a) BIDIR-SAM, $k = 16$

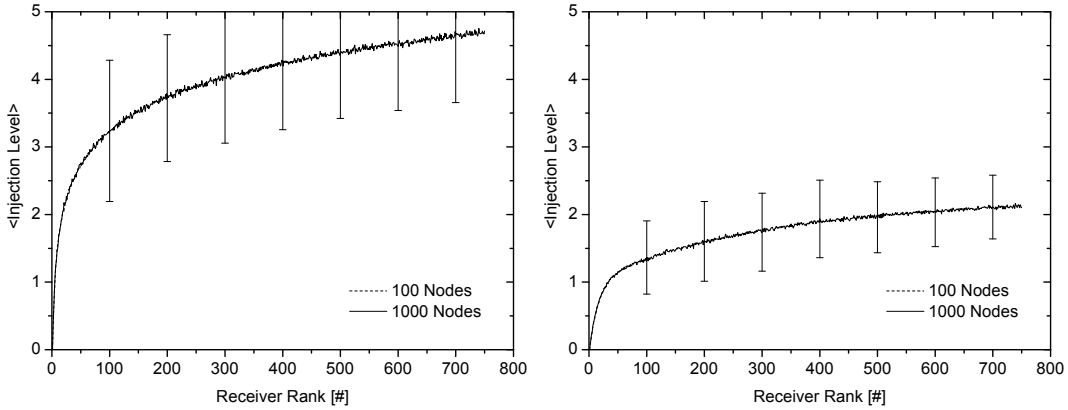
(b) Scribe, $k = 16$



(c) BIDIR-SAM, $k = 4$

(d) Scribe, $k = 4$

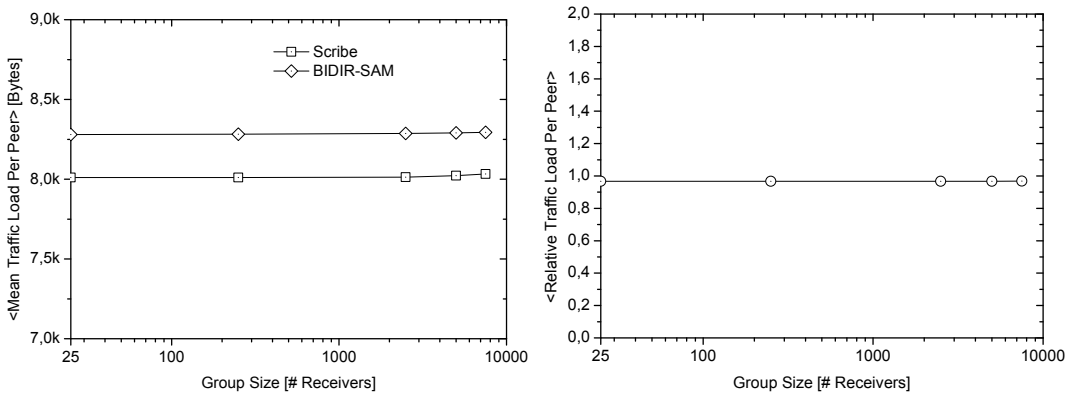
Figure 4.9: Effective Joins per Receiver



(a) $k = 4$

(b) $k = 16$

Figure 4.10: Mean Join Injection Level for BIDIR-SAM With Selected Error Bars for Prefix Alphabet Sizes k



(a) Absolute UDP-Traffic per Peer L

(b) $L_{\text{Scribe}}/L_{\text{Prefix Flooding}}$

Figure 4.11: The Mean UDP Traffic Volume per Peer in BIDIR-SAM and Scribe for Overlays With 10.000 Nodes

Join Injection Level

The average injection level for joins in BIDIR-SAM is shown in figure 4.10. Due to the nature of prefix trees, the distance to the root grows logarithmically with an increasing number of receivers. The measurements can be controlled by the prefix alphabet size and are independent of the network size.

The latter observation can be explained by the self-similarity of prefix trees. Reducing the number of overall peers downsizes the maximal tree height. Nevertheless, the branching structure in the prefix space persists. Thus, the paths traversed in subsequent joins will be shortened, but injections still occurs at the same level. This highlights the uniform construction process of the multicast distribution tree.

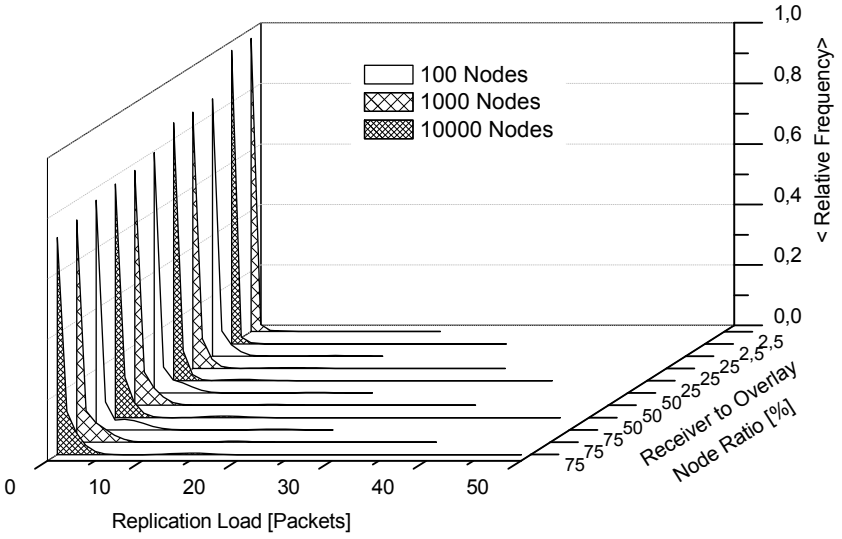
Traffic Load

The average UDP traffic per peer is visualized in figure 4.11 for a fixed overlay with 10.000 nodes.[5] In general, the traffic load per peer achieves almost constant values independent of the group size, nicely reflecting the multicast nature of packet replication (cf. figure 4.11(a)). The traffic load of BIDIR-SAM is negligibly higher than in Scribe, which results from increased signaling load. It is worth noting, that the metric focuses on the mean traffic load per peer. Thus, a high packet replication load for single peers in Scribe, as we will observe next, will be averaged over all group members. We omitted standard deviations bars for ease of readability. However, in Scribe fluctuations are significantly higher as in BIDIR-SAM, which is consistent with our next observation.

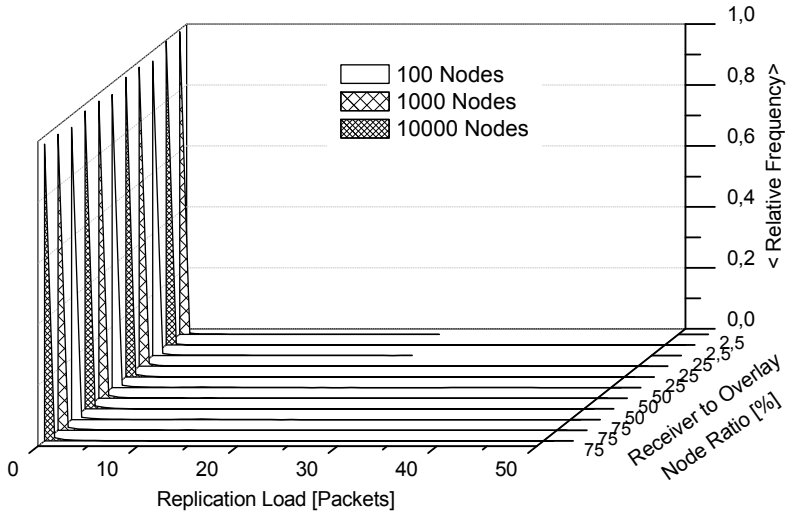
Replication Load

The distributions of the replication load for different network sizes and receiver populations are displayed in figure 4.12. The corresponding mean values and standard deviations are shown in table 4.1. Both schemes exhibit a sharp peak for low replication values and decay exponentially. The overall shape of the distribution depends mainly on the receiver population, which can

[5] As mentioned in section 3.3.2, a varying network size will basically change the KBR overhead.



(a) BIDIR-SAM



(b) Scribe

Figure 4.12: Packet Replication Distributions for a Varying Receiver to Peer Ratio, $k = 16$

	BIDIR-SAM, $k = 16$					
	$N = 100$		$N = 1.000$		$N = 10.000$	
R	$< X >$	σ_X	$< X >$	σ_X	$< X >$	σ_X
25%	0.36	1.48	0.36	1.56	0.36	1.53
50%	0.60	1.98	0.61	2.14	0.61	2.17
75%	0.81	2.32	0.82	2.48	0.82	2.58

	Scribe, $k = 16$					
	$N = 100$		$N = 1.000$		$N = 10.000$	
R	$< X >$	σ_X	$< X >$	σ_X	$< X >$	σ_X
25%	0.27	1.96	0.26	4.99	0.26	14.11
50%	0.52	3.72	0.51	9.68	0.51	27.93
75%	0.76	5.48	0.76	14.72	0.76	41.72

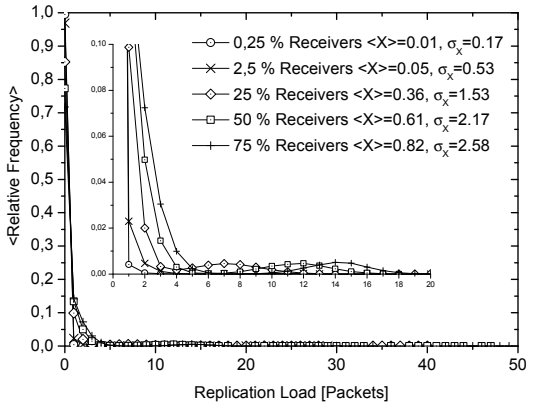
Table 4.1: The Mean $< X >$ and Standard Deviation σ_X for the results in figure 4.12. R represents the multicast listener subscription ratio and N the overlay size.

clearly be seen from comparing the average values for different overlay sizes: BIDIR-SAM as well as Scribe attain a constant average packet replication load for varying overlay dimensions and a fixed receiver to peer ratio. However, the distributions differ in detail.

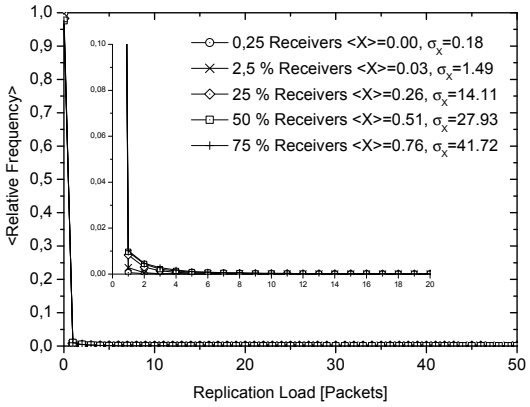
The mean variation increases linearly for Scribe and negligibly for BIDIR-SAM with the overlay network size. This result coincides with our observations for the broadcast scenario (cf. section 3.3.2) and previous conclusions can be adopted.

A detailed view of the packet replication for a varying number of receivers in a fixed size overlay is given in figure 4.13. The asymptotical growth of the packet replication in Scribe also remarkably depends on the group size. Additional multicast listeners, thus, increase the maximal replication load. This indicates a tendency that additional receivers construct branches meeting the maximal load replicator, which further implies that a single peer is responsible to forward multicast data to almost all group members. In contrast to this, BIDIR-SAM balances the load. The branching factor k shifts weights of higher replications, which results in a lower maximal load (cf. figure 4.14), but in a slightly increased load per peer and longer paths. The reduced k smooths the tail of BIDIR-SAM (cf. figure 4.14(c)), as branches are populated more densely and replications occur as multiples of $k - 1$.

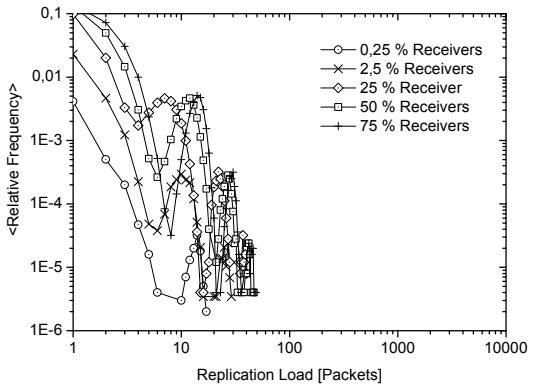
Figure 4.13(e) and 4.13(f) visualize the locations of the packet replications inside the multicast distribution tree. The main forwarding load is performed at the source (tree level 0) in BIDIR-SAM and by the rendezvous point (tree level 1) in Scribe. Surprisingly, Scribe lacks on visible replications for tree level 4-6, but continues on level 7. The reason for this is that level 4-6 of the tree represents longer one-way branches, which appears less likely in BIDIR-SAM.



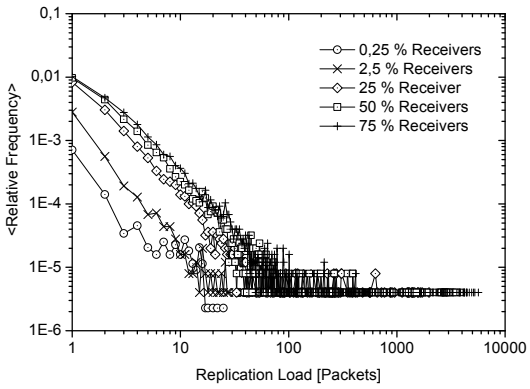
(a) BIDIR-SAM



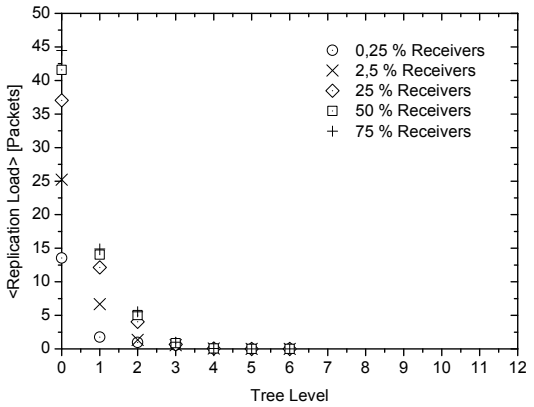
(b) Scribe



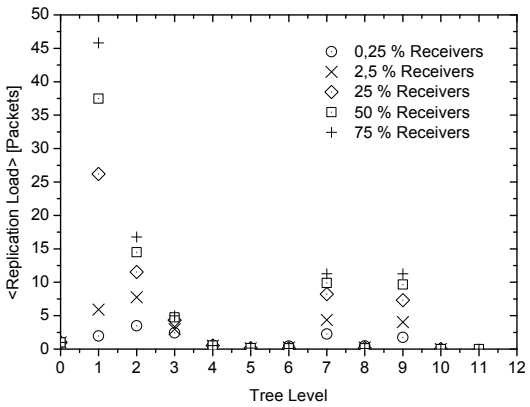
(c) Detail: Tail for BIDIR-SAM



(d) Detail: Tail for Scribe

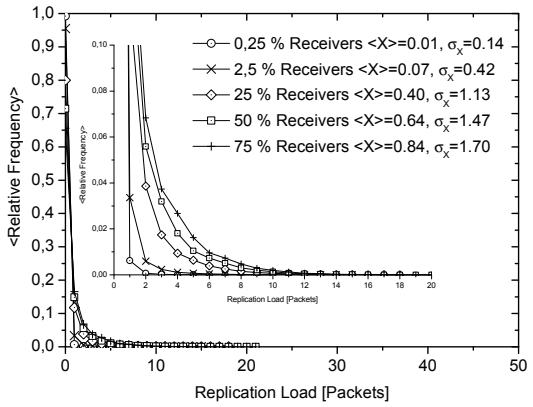


(e) Locations of Replications within the BIDIR-SAM Distribution Tree

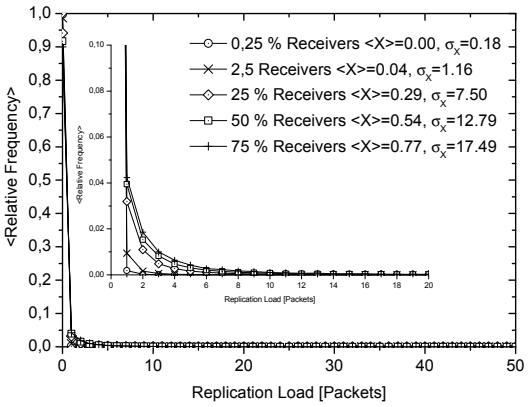


(f) Locations of Replications within the Scribe Distribution Tree

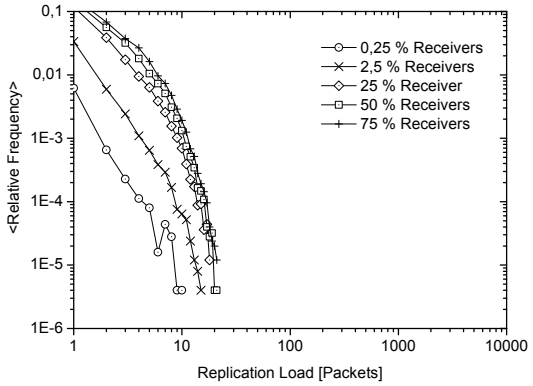
Figure 4.13: Distribution of Packet Replication Comparing BIDIR-SAM with Scribe for a Varying Ratio of Receivers to Peers Using a Fixed Key Length of 128 and $k = 16$ in a 10.000 Node Overlay



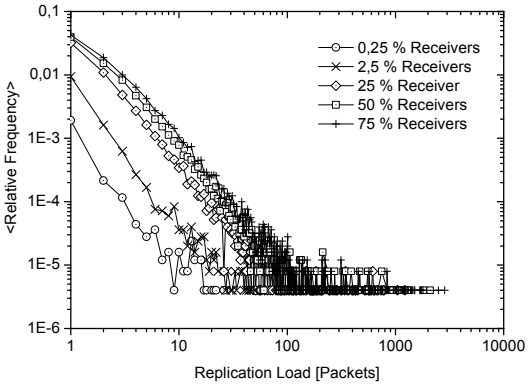
(a) BIDIR-SAM



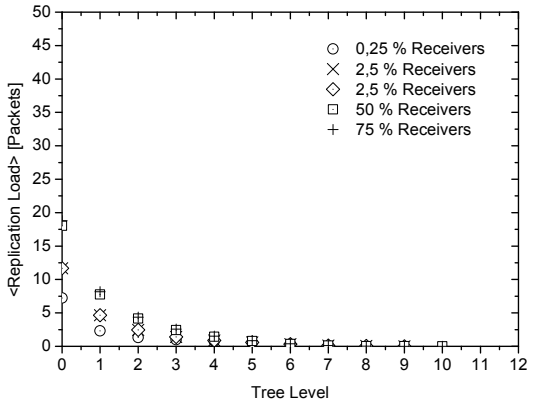
(b) Scribe



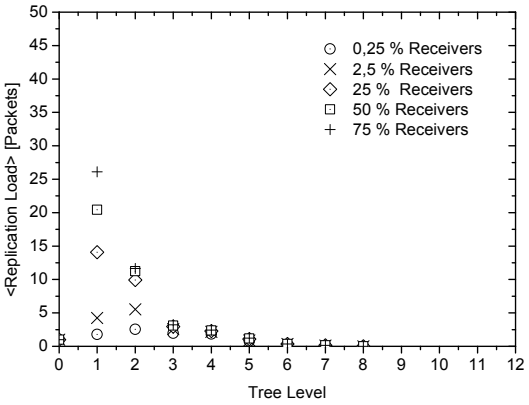
(c) Detail: Tail for BIDIR-SAM



(d) Detail: Tail for Scribe

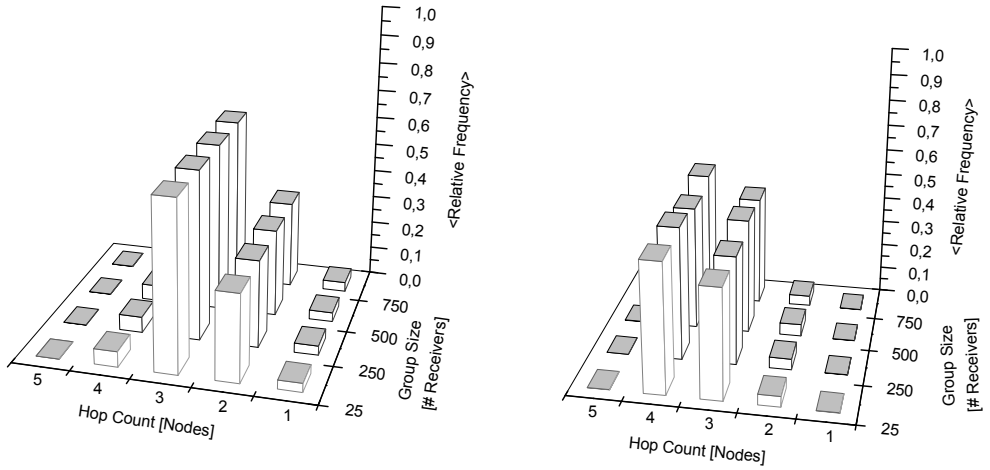


(e) Locations of Replications within the BIDIR-SAM Distribution Tree



(f) Locations of Replications within the Scribe Distribution Tree

Figure 4.14: Distribution of Packet Replication Comparing BIDIR-SAM with Scribe for a Varying Ratio of Receivers to Peers Using a Fixed Key Length of 128 and $k = 4$ in a 10.000 Node Overlay



(a) BIDIR-SAM $N = 1.000$ (b) Scribe $N = 1.000$

Figure 4.15: Hop Count Distribution for an Overlay of Size N and Different Numbers of Receivers, $k = 16$

Hop Count

The mean hop count distribution for an overlay with 1000 nodes and a varying number of receivers is plotted in figure 4.15.[6] Both schemes show that the path lengths in the multicast distribution trees are uncorrelated with the number of multicast listeners. This is reasonable as the shortest forwarding paths are created in the network of all overlay peers. For this reason, we continue the discussion focused only on the minimal receiver set in our scenario.

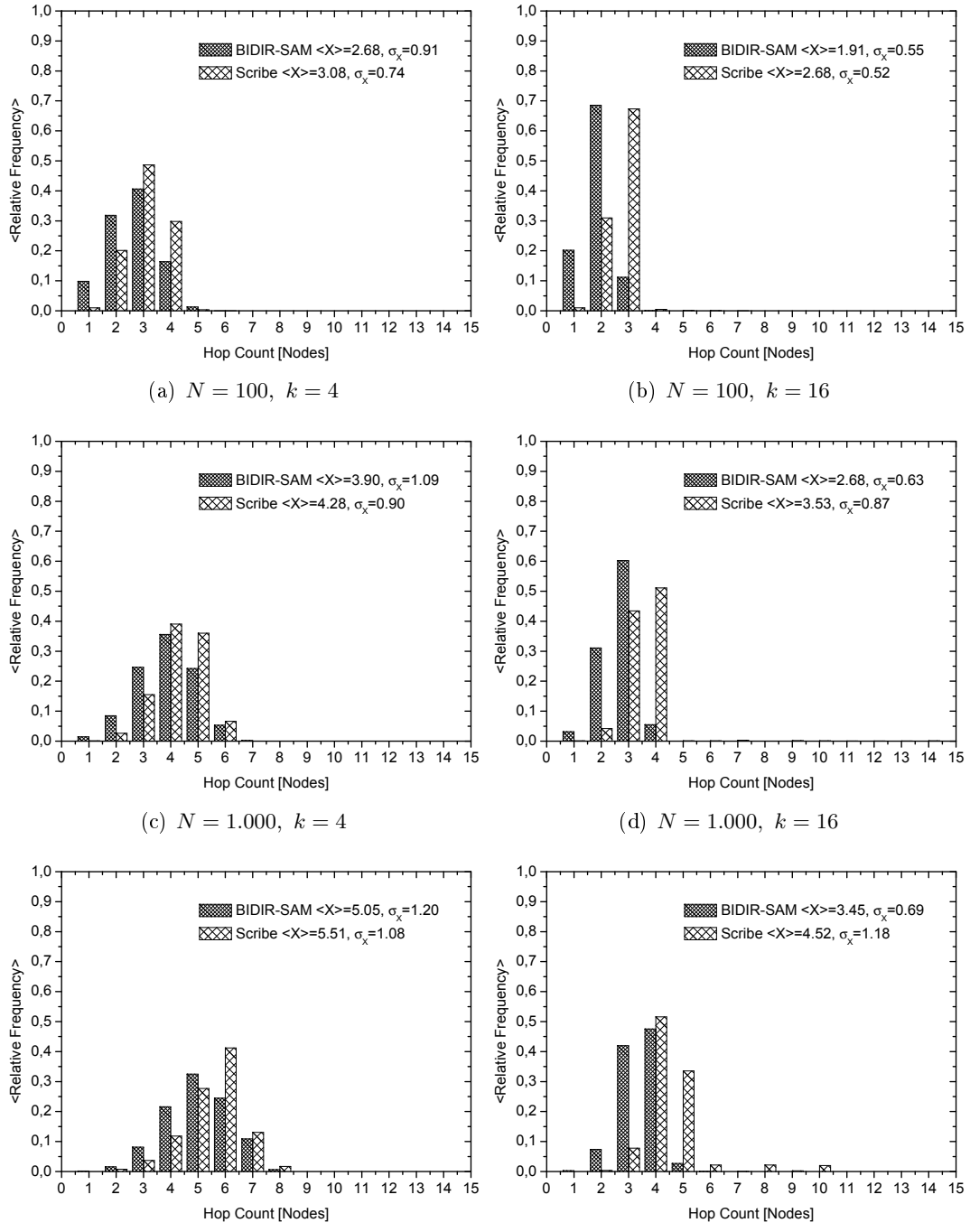
Figure 4.16 visualize the hop count distribution for different overlay network sizes. BIDIR-SAM as well as Scribe exhibit a logarithmically increasing path length, which results from the underlying Pastry prefix tree. Recalling our previous observation, the results nicely coincide with the broadcast scenario studied in section 3.3.2 (cf. figure 3.9). This includes in particular the elongated path by at least one hop in Scribe due to use of a rendezvous point approach.

A smaller prefix alphabet increases the height of the constructed multicast distribution tree and creates longer paths (cf. figure 4.16(a), 4.16(c) and 4.16(e)). In contrast to Scribe, the average hop count changes more significantly in BIDIR-SAM, but the difference to $k = 16$ remains below one hop. The standard deviation grows almost by a constant in BIDIR-SAM, but linearly in Scribe. Both schemes show higher weights for longer paths. Similar behavior could be observed for other values of k.

Forwarder Fairness

The mean fraction of multicast receivers acting as data forwarder is shown in figure 4.17 for different overlay network sizes. Both schemes attain a fairly equal ratio of listeners distributing data. This measurement scales almost independently of the overlay network size and grows linearly with the receiver population inside the overlay.

[6] The qualitative behavior holds for different overlay sizes, which are omitted.

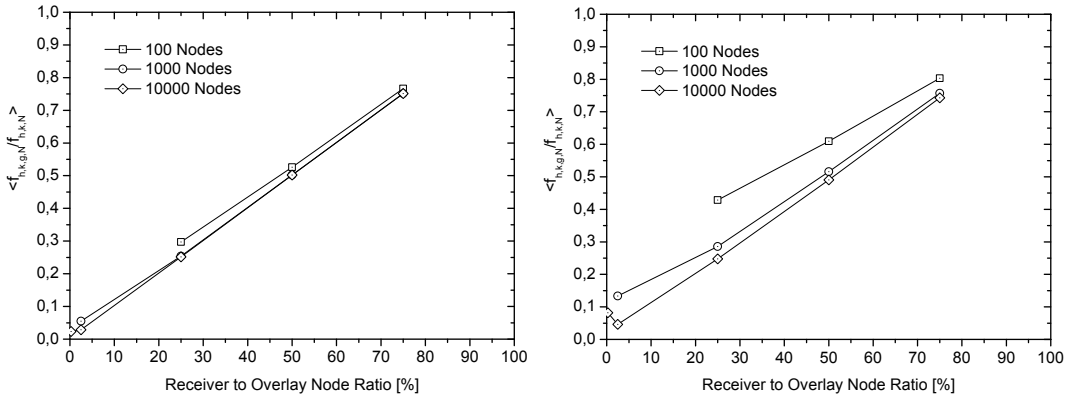


(a) $N = 100,\ k = 4$ (b) $N = 100,\ k = 16$

(c) $N = 1.000,\ k = 4$ (d) $N = 1.000,\ k = 16$

(e) $N = 10.000,\ k = 4$ (f) $N = 10.000,\ k = 16$

Figure 4.16: Hop Count Distribution for an Overlay of Size N and a Varying Prefix Alphabet Size k for 25 Receivers



(a) BIDIR-SAM (b) Scribe

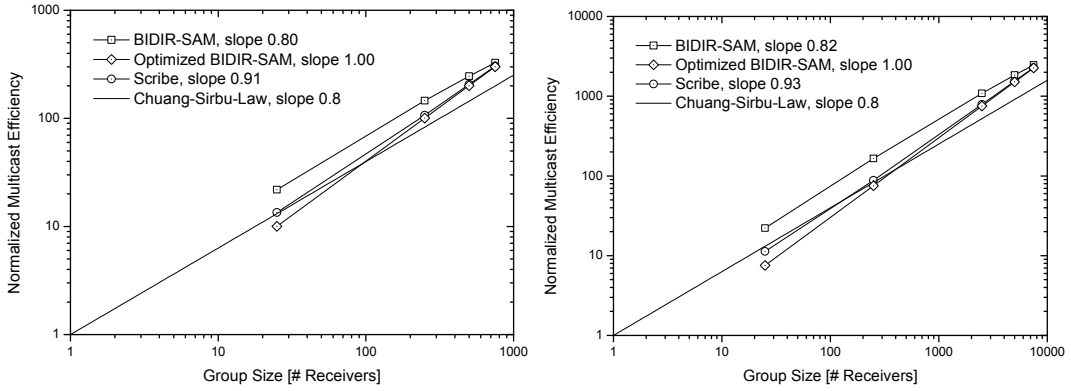
Figure 4.17: Fraction of Multicast Receivers Acting as Data Forwarder for Different Overlay Sizes, $k = 16$

Multicast Efficiency

The normalized multicast efficiency is plotted in figure 4.18 for different overlay sizes. For comparison, we also show the normalized multicast efficiency with respect to the analysis by Chuang and Sirbu [26]. Using a standard prefix alphabet with 16 digits, the absolute efficiency values of BIDIR-SAM and Scribe are higher than observed by Chuang and Sirbu for native multicast. However, this metric reflects the scaling behaviour of multicast protocols with growing group sizes. The slope, which represents the scaling factor, is calculated based on a linear fit. It can clearly be seen, that the basic BIDIR-SAM increases with equal rate as native multicast. In contrast, Scribe exhibits a higher scaling factor. This indicates that Scribe paths are constructed less efficiently. The observation coincides with our previous results, which show a high, unicast-like replication load for Scribe around a single node.

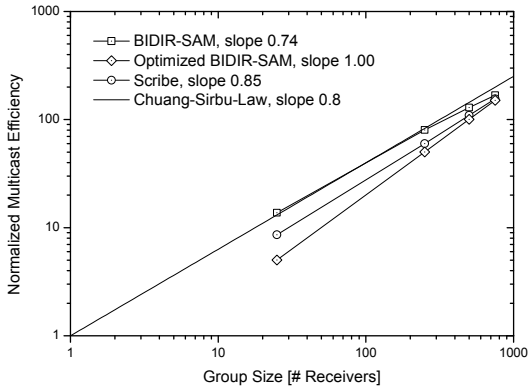
The optimized BIDIR-SAM attains a constant slope of 1 as all traversed edges correspond to receiver links. Surprisingly, this appears to be less efficient than current native multicast protocols, even though the optimized BIDIR-SAM traverses the minimal number of edges. This counterintuitive observation results from the Internet-guided picture for the metric, that a significant number of inner vertices represents only forwarders. This does not hold for the optimized BIDIR-SAM, which nevertheless attains the highest efficiency.

Reducing the prefix alphabet size, increases the absolute normalized multicast efficiency, as multicast paths will be longer and receivers are more likely to co-located on a path. Thus, the multicast protocols benefit from higher link re-use, which reduces the scaling factor slightly.

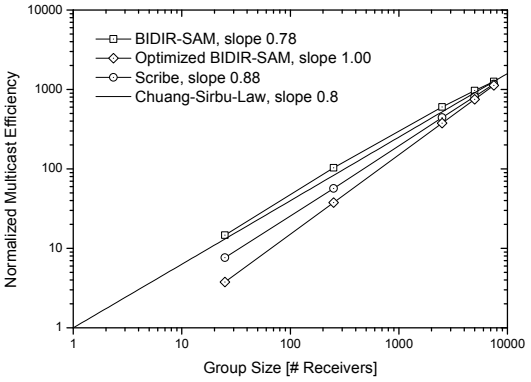


(a) $N = 1.000$, $k = 16$

(b) $N = 10.000$, $k = 16$



(c) $N = 1.000$, $k = 4$



(d) $N = 10.000$, $k = 4$

Figure 4.18: Normalized Multicast Efficiency for Different Overlay Sizes N and Prefix Alphabets k

4.4 Related Work

Derived from structured peer–to–peer routing, a collection of group communication services has been developed, with the aim of seamless deployability as application layer or overlay multicast [105, 48, 53]. Among the most popular approaches are multicast on CAN [78], Bayeux [108] as derived from Tapestry and Scribe [22] or SplitStream [21], which inherit their distributed indexing from Pastry.

Approaches to multicast distribution in the overlay essentially branch in two algorithmic directions. The first uses DHTs to generate a structured sub-overlay network of group members, which thereafter is flooded with multicast packets. This mechanism underlies multicast on CAN. CAN requires a bootstrapping mechanism to create this mini overlay, and instantiates a full new structured network afterwards. From the BIDIR-SAM perspective, the equivalent procedure would employ prefix flooding subsequent to setting up a sub-overlay for the specific group. Note that such schemes require a groupwise sub-overlay instantiation.

The second class of algorithms erects a shared or source specific distribution tree within the original full overlay, which is flooded thereafter. In a shared tree approach, these schemes are used in Scribe and SplitStream, where one or several rendezvous nodes are chosen from group key ownership. To the best of our knowledge, the only scheme that follows a source specific distribution model on top of a structured overlay is Bayeux. The creation of a group proceeds from hashing a group identifier, which thereafter is used as the name of a trivial file placed at the source node. Using Tapestry location services, the source root of a session announces that dummy document into the entire network. Thereby clients will learn about the group and source ID tuple to perform source-specific subscriptions. Any subscription is routed to the source, which acts as a centralized group controller including a complete receiver tracking. In proceeding along this line, Bayeux admits a state information growth of linear scaling in the number of receivers. Hence it suffers from severe scalability restrictions.

DHT–based multicast performance has been thoroughly studied in [23] with the comparative focus on tree–based and flooding approaches built onto CAN and Pastry (cf. also section 3.4). The separate construction of mini-overlays per group as needed for a selective flooding showed to incur significant overhead. In addition, flooding was found to be outperformed by forwarding along trees, where a shared group tree combined with proximity-aware routing as in Scribe could minimize the overlay delay penalty down to a factor of two.

Many overlay multicast concepts concurrently exist for unstructured peer-to-peer approaches [48]. Operating at a lower algorithmic complexity, but significantly higher coordinative signalling efforts, performance characteristics for unstructured schemes differ. While DHT-based schemes are close to optimal with respect to message overhead and forwarding efficiency, they tend to create unbalanced distribution trees as an outcome of structured routing rules. Multicast tree balance and performance comparing structured and unstructured schemes have been explored in [14]. Focusing on Scribe and SplitStream, and in agreement with our results, the authors identified a highly unbalanced forwarding load at inner tree nodes along with a large fluctuation of delays sustained at receivers. The latter where found to accumulate in SplitStream to mainly intolerable jitter values. In addition, conventional structured tree-based schemes are highly vulnerable to failures along the paths. A single failure along a spanning tree can result in loosing a message for a whole branch. An important value of BIDIR-SAM lies in its inherent tree redundancy, which effortless allows for a scalable redundant data dissemination without

noticeable performance penalties.

For the sake of completeness, we mention that probabilistic, gossip-based protocols [13, 61, 56, 101] form an alternate approach to the large-scale content distribution problem. By increasing scalability through reducing coordinative information, those algorithms attempt to optimize the likelyhood of a uniformly correct packet delivery.

Only recently, a more fundamental debate on multicast efficiency has come life. Though obvious, network efficiency gained from multicast data distribution has not been quantified for a long time, leaving providers with vague expectations for the outcome of multicast service provisioning. Starting in 1998, multicast distribution trees have been thoroughly studied with regards to network efficiency. Grounded on empirical observations on the IP layer, Chuang and Sirbu [26] proposed a scaling power–law for the total number $L_N(g)$ of links in a multicast shortest path tree with g receivers of the form

$$L_N(g) \approx < L_U > g^{\alpha},$$

where $< L_U >$ represents the average number of unicast hops taken by a message between uniformly chosen nodes in the corresponding network of N nodes. The authors consistently identified the scale factor to attain the independent constant $\alpha = 0.8$. The validity of such universal, heavy–tailed distribution suggests that multicast shortest path trees are of self–similar nature with many nodes of small, but few of higher degrees. Consequently, trees would rather be shaped tall than wide. Providers thus could count on a relative gain in network resource consumption, which uniformly scales in group size as $g^{0.2}$.

Subsequent empirical and analytical work in [71, 98, 24, 4, 51] has debated the applicability of the Chuang and Sirbu scaling law. Its consequences for multicast mobility has been analyzed in [87, 103]. Van Mieghem et al. [98] proved that the proposed power law cannot hold in general, but is indeed a valid approximation for moderate receiver numbers and the current Internet size $N = 10^5$ core nodes.

Multicast cost efficiency has also been studied on the overlay [35]. Fahmy and Kwon examine overlay tree structures experimentally and in theory, the latter using complete k-ary trees with receivers placed at leaf nodes and inner vertices. In general, no clear indication of a power law could be identified in the paper. However, the authors conclude that a power exponent of $\alpha = 0.9$ becomes visible for a small number of multicast group members g in the overlay.

4.5 Discussion

In this chapter we have presented and analyzed BIDIR-SAM, a novel overlay multicast approach, which enables any peer to distribute multicast data directly into a multicast group. Using a logical prefix overlay, BIDIR-SAM peers autonomously construct a bi-directional, shared distribution trees, which disseminates data according to source-specific shortest paths. There is no need for dedicated, infrastructure entities such as rendezvous point among the overlay nodes. The protocol costs in signaling and fowarding are strictly predictable and scale logarithmically with the network and group size, as has been shown in theoretical analysis and in simulations. Thereby, and to the best of our knowledge, BIDIR-SAM is the only structured multicast scheme, which distributes data on source-specific shortest path trees at logarithmic costs, and as well the only solution, which utilizes shortest path trees within a shared tree model.

BIDIR-SAM also uniquely defines a DHT-based multicast approach, which builds bi-directional distribution trees without the assistance of any bootstrap node or rendezvous point. The performance properties are evaluated based on simulations and a theoretical analysis, which quantifies measurements almost in exact agreement with the empirical calculations.

We compared our approach with Scribe, an implementation for a rendezvous point-based overlay multicast scheme. BIDIR-SAM admits superior performance in overal data distribution and scaling behaviour, when scaling towards very large networks and multicast groups. Scribe, which erects a single shared tree, outperforms BIDIR-SAM on *average* with respect to the tree construction costs for small size groups, but obligates dedicated overlay nodes with unbound, tremendous storage and forwarding load. As a result, Scribe performance values fluctuate on a large scale, leading in particular to high jitter values at the receiver nodes.

Large-Scale vs. Small Groups

Our analysis reveals that BIDIR-SAM is best suited for large-scale multicast groups. BIDIR-SAM packet distribution metrics and overall resource requirements scale evenly as logarithmic functions of the group size, while most performance values of Scribe fluctuate on a scale linearly growing with group participants. In contrast to Scribe, BIDIR-SAM has to flood prefix listener subscriptions to prefix (sub-)trees, whose size depend on the receiver population. While the first join message is distributed to all overlay nodes, the costs decay exponentially with further multicast listeners arriving, such that groups exceeding 500 receivers in large overlays closely approach the very low costs of Scribe. Thus, for application scenarios with a stable base number of receivers, BIDIR-SAM exhibits appropriate costs, even while most of the multicast listeners may change.

Multicast state updates directly influence multicast forwarding table sizes. In general, the average number of entries per peer is higher in BIDIR-SAM than in Scribe. Nevertheless, BIDIR SAM table sizes exhibit a strict logarithmic bound, while storage in Scribe may grow linearly. Individual Scribe peers frequently store almost all multicast forwarding states. This linear growth fails to meet the scaling characteristics of structured overlay networks. For large multicast groups, Scribe peers will be overloaded. In contrast, BIDIR-SAM distributes states fairly among all peers in small and in large groups.

As suggested by Castro *et al.*, the state distribution for Scribe can be improved by adding a balance mechanism of assisting replicators. This workaround implies a higher signaling load and may cause routing loops [22]. The proposed scheme re-arranges the forwarding tables at the reception of state updates, but may destabilize the forwarding tables. In contrast, BIDIR-SAM instantaneously creates balanced routing tables of appropriately scaling sizes.

Additionally, BIDIR-SAM outperforms Scribe with respect to the forwarding costs. A BIDIR-SAM multicast sender can control the maximal load it imposes onto the distribution infrastructure, which is a simple but effective QoS instrument. In particular, a sender-initiated forwarding load declines exponentially in BIDIR-SAM, thus being at hand as an a priori load estimate, which may be used for flow control. In contrast, the distribution of the replication load in Scribe is heavy-tailed. The rendezvous point in Scribe distributes the multicast data to a significant number of receivers. This load even increases in multiple source scenarios: Each source will send its data to the rendezvous point. In contrast, the highest load in BIDIR-SAM is located at the source, while replications steadily decrease as the prefix tree is descended. Based on the structural protocol properties it follows that in case of multiple sources, the source-specific

distribution model of BIDIR-SAM will balance the load automatically.

BIDIR-SAM always operates on a single virtual distribution tree, which is collectively known at peers. This common forwarding instruction leads to a coherent overlay routing performance among all peers, and is of particular importance in synchronized multisource scenarios. The latter become vital in mobility or load sharing scenarios. As already discussed in section 4.2.2, this prevents BIDIR-SAM from admitting intolerable jitter values under source variations as have been observed for SplitStream.

Redundancy and Reliability

The common major weakness of tree-based structured multicast approaches debated in literature and practice lies in a limited reliability and an increased vulnerability with respect to node or link failures. It is one of the major strength of BIDIR-SAM to overcome this deficit. On the price of an enhanced initial signaling load, a wider distribution of multicast state information is achieved among peers. Actually, each peer carries a self-consistent view of the distribution tree using a state table of logarithmic size. In particular, each peer is enabled to initiate or perpetuate the distribution of any packet it receives. This inherent redundancy allows for a drop out of nodes, without affecting multicast forwarding. In section 4.2.2 we have already discussed that a combination of network coding and a relay-oriented forwarding strategy can add r-redundancy effortless and without additional signaling to BIDIR-SAM.

On the contrary, only a few selected Scribe peers hold forwarding entries. If they disappear or suffer from disturbances, the multicast distribution tree collapses.

The reliability of BIDIR-SAM will be enforced by storing only prefixes instead of full destination keys. Each prefix covers a number of interchangeable overlay nodes. The reliability, thus, will be supplementary enhanced by the underlying KBR. Relying on individual distribution paths, Scribe does not benefit from this flexibility and will cut multicast branches, whenever an intermediate forwarder is not accessible. The latter behavior will be particularly disruptive at the RP.

Overall Performance

The performance of BIDIR-SAM is uniformly and strictly predictable over all peers, whereas Scribe produces an unfair, irregularly fluctuating load at forwarders. BIDIR-SAM constructs shorter paths and creates lower replication loads, which remain quite stable with growing group sizes. Operating on forward-oriented, prefix-defined paths, BIDIR-SAM not only complies with asymmetric links and hop alterations, but takes higher-than-average advantage of proximity selections at the KBR layer.

BIDIR-SAM can nicely be tuned by the prefix alphabet parameter. A smaller prefix alphabet directly smooths the branching, which in turn reduces signaling and replication load per peer. All deviations from mean values thereby remain small. This overall balancing effect faces the path lengths as its only counter-measure. They increase logarithmically with the alphabet size. In contrast to this, the branching parameter has only marginal effects on the performance of Scribe.

5 Design of an Overlay Group Communication Architecture

5.1 Concept of a Common API for Structured Overlay Routing

A structured overlay network consists of three functional groups: a routing scheme, a set of services and the applications. The routing, based on a decentralized key approach, is responsible for locating peers associated with specific key ranges. Such routing algorithm is independent of the applications built upon it. Services like group communication, failover redundancy, etc. supplement the structured overlay and can be developed independently of both, the underlying overlay routing and the application. A well designed protocol architecture should separately account for these components and offer pluggable modular building blocks.

Modeling routing, services and applications in a self consistent way leads to a layered architecture, which will attain flexibly interchangeable modules from common interface definitions between all components. Flexibility emerges from two perspectives. Firstly, the development process may be simplified as modules can be reused. This is important in particular in the context of new emerging technologies like structured overlay networks, which then can be analyzed more consistently and timely in different systems like simulators and real-world setups. Secondly, deployed modules can be combined from tailored units. In the scenario of structured P2P networks, an application can be composed of overlay implementations best suited for current requirements on performance and service needs. This may not only reduce complexity, but also diminish maintenance overhead from unwanted services.

5.1.1 The Dabek Model

The first ideas towards a layered architecture with a common API for structured overlays has been presented in [27]. The concept is known as the Dabek model and has been implemented in numerous simulators [68, 62, 15, 10] and DHT stacks [32]. The basis of the Dabek model is a unified overlay routing interface.

Structured overlay networks have been originally inspired by distributed indexing. Recently, they inspired even new routing schemes directly on top of the link layer [18]. The decentralized routing mechanism of P2P protocols serves as a virtualized network layer. Dabek *et al.* observed that many peer-to-peer services and applications are built upon such a key-based routing (KBR) module, which can locate peers for multiple purposes. Based there on, the authors suggest a compound P2P layer consisting of three tiers: Tier 0 represents the fundament of overlay communication, the KBR layer. Tier 1 implements higher level abstractions, complemented by tier 2 for end user applications and further, higher level services.

Dabek's model focuses on tier 0. The meaning of tier 1 and 2 has not been fully elaborated. Both tiers may accommodate services with direct access to the KBR layer. Reusable abstractions

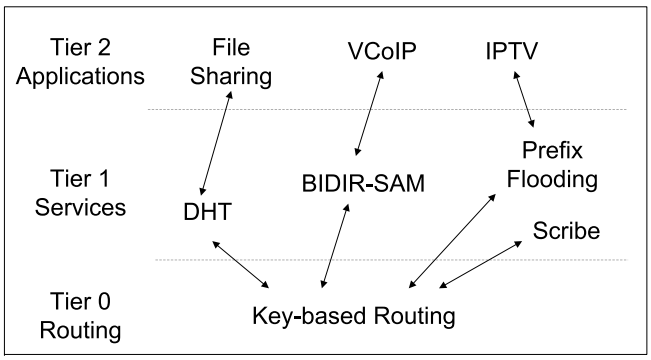


Figure 5.1: A Compound P2P Layer According to the Dabek Model [27] With Typical Application Classes. Interactions Between the Layer Components are Highlighted by Arrows.

for specific purposes are dedicated to reside on tier 1. The DHT abstraction[1] is an example for this This component provides a store-and-retrieval key management service on the overlay for applications like file sharing. On top of the DHT abstraction, further services may be available, such as an additional caching to be used by 'end-user' applications or specific store and retrieval functions. This concept of stacked services may actually result in additional tiers, not foreseen by the model. To avoid such complicacies, we assume that tiers 0 to 2 of the Dabek model reflect a routing, a service and an application layer (cf. figure 5.1).

The Common Key-based Routing API

The key-based routing layer provides the option to forward data along overlay hops. With respect to the concepts of layered architectures, an implementation may delegate messages to the KBR component that performs the routing. Hence, all application data will be encapsulated in overlay messages and delivered by the KBR. However, this may be inefficient, because it can cause double encapsulation, at the service and the KBR layer. Additionally, a tier 1 service may follow a different routing strategy than the overlay paths. Thus, a common API should remain open with respect to adaptable message forwarding, as well as KBR state access.

The KBR API by Dabek *et al.* consists of two functional groups. The first part provides function calls for sending and receiving data messages above tier 0, and also for controlling the overlay routing path for messages initiated by the service or application. These are subsumed as message routing. The second part allows access to the KBR routing states at peers.

The message routing calls offer all required primitives to perform an overlay routing configurable by higher layers. They are summarized in table 5.1 with respect to the implementation at the specific layers. In the following, we describe the use of these different functions. Starting at a source peer that calls `route`, data will be forwarded towards the (destination) key by the KBR. This request can be supplemented by a predefined first hop, the hint. At all intermediate peers between source and destination, the KBR implementation invokes `forward` at the tier above. This upcall informs the application about its intention to forward the message for key `K` to the `nextHopNode`. All passed parameters can be modified. Thus, tier 1 and 2 applications can

[1] Although, DHTs are equipped with a KBR, the aim of distributed hash tables is the management of key-value-pairs. This is a conceptually different task compared to the KBR.

Tier	Message Routing
1-2	`forward(key↔K, msg↔M, nodehandle↔nextHopNode)` `deliver(key→K, msg→M)`
0	`route(key→K, msg→M, nodehandle→hint)`

Table 5.1: The KBR Message Routing API Calls Implemented at the Corresponding Layers [27]

Tier	State Access
1-2	`update(nodehandle→n, bool→joined)`

0	`nodehandle [] local_lookup(key→K, int→num, boolean→safe)`
	`nodehandle [] neighborSet(int→num)`
	`nodehandle [] replicaSet(key→k, int→max_rank)`
	`boolean range(nodehandle→N, rank→r, key↔lkey,key←rkey)`

Table 5.2: The KBR State Access API Calls Implemented at the Corresponding Layers [27]

effectively override the default KBR (overlay) routing behavior by changing the `nextHopNode`. Finally, the message will be provided to the application at the peer responsible for `K` via `deliver`.

For a detailed description of the state access functions, we refer to [27]. An overview is given in table 5.2. These calls are limited to *local* operations and *do not* involve communication with other nodes. Applications can invoke these function from higher tiers to inquire on the local peer routing states. Nevertheless, local lookups will fail, whenever the requested information is not locally available. To facilitate a global peer access, the message routing functions can be used.

The routing as well as the state access functions require a common `key` parameter that corresponds to the overlay address. Consequently, all applications using one of the primitives must be aware of the hash function in use.

Limitations of the KBR-API

The generic approach of the KBR-API does not provide information about the actual overlay routing protocol or implementation-specific parameters, like the key length in KBR or the dimension in CAN. Such meta information are of interest for services, which implement cross layering or operate adaptive to the underlying KBR. An example are routing services that are derived from the local state information, but want to construct their own forwarding tables. Using the common API, these services can retrieve generic destination values, but are not enabled to reconstruct the underlying routing structure.

To make the protocol parameters visible to upper tiers, the API needs extensions. One possible concept would include dynamic maps to present the KBR specific configurations in the form of key value pairs. The corresponding schemes can be implemented by information bases similar to Management Information Bases (MIBs). The implementation of this approach requires only a getter call. The actual parameter set is then defined by the protocol instance.

In using such a rich, compound P2P layer concept, existing key-based routing implementations can be enhanced by new services without changing the KBR component. For this purpose, a new service is only required to implement the common KBR calls. However, more complex services such as overlay multicast, need to provide an own common API towards applications.

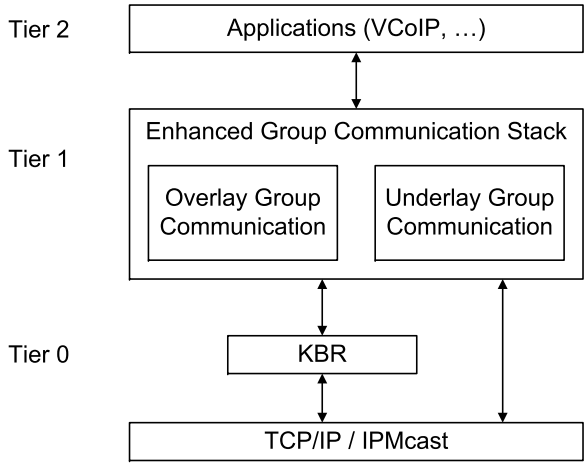


Figure 5.2: Generic Stack Architecture for Cooperated Underlay & Overlay Multicast

5.2 A Middleware for Structured P2P Group Communication

P2P group communication is strongly driven by the lack of infrastructure support for multicast. Although overlay multicast (OLM) is mainly used as a supplement for expanding native multicast regions, its mediator role between application and network layer may shape and enhance group communication. This is one reason, why an OLM scheme should be implemented in a flexible, common group communication framework, which simultaneously accounts for specifics of the overlay and underlay. The core architecture for an enhanced group communication stack is shown in figure 5.2.

The main task of an OLM component is the distribution of data according to the host group model. Destined for a group address, messages are replicated along an existing (virtual) routing infrastructure similar as in native networks. On the unicast side, routing in structured overlays can be provided by the KBR layer and the access may be decoupled from a specific implementation using the common primitives. Thus, structured overlay group communication requires at least the KBR layer.

The KBR layer connects peers to a unicast network. Multicast domains will be established, when overlay nodes form a group. Group creation and maintenance is operated within the OLM middleware. A special case is broadcast, which inherently floods all nodes of the network. This may happen in contrast to the establishment of selected routing paths. Hence, multicast and broadcast follow different distribution schemes, and require a different API towards the application. A modular OLM stack should account for these conceptual different challenges and solutions.

Besides flexibility, an OLM design should also be guided by a compatibility principle with respect to native network services. On the one hand, the API calls, e.g., joining and leaving groups, should be compliant to well-known functions, such that application developers are not distracted. On the other hand, the OLM middleware should provide an interface, which allows transmission of data to both, overlay and native multicast networks.

Overlay group communication differs from native IP networks with respect to addressing. An IP stack is fed with the correct network address type to perform routing, whereas the key-based routing layer maps an arbitrary identifier to the deployed key space, which commonly

is not invertible and does not allow to recover original IDs. Applications need to regain those identifiers explicitly, and it may be an advantage for the overlay routing to be aware of the application addresses, as well. Based on application addresses, the overlay scheme may for instance aggregate or scope groups. Thus, the design of an OLM middleware should preserve common group functions, but also include support for specific aspects of the layers involved, and remain open for additional functionalities arising from structured overlay networks.

5.2.1 Current State of the Art

The Dabek model proposes the multicast abstraction CAST [27]. The idea is to provide overlay services in a generic multicast module. CAST consists of a set of interfaces for group management and data distribution (`join(groupId)`, `leave(groupId)`, `multicast/anycast(msg, groupId)`), as well as a basic multicast routing on top of the KBR layer. Routing within CAST is built upon a dedicated tree management and forwarding scheme, which is similar to Scribe. Calling the `join` function initiates a subscription message routed towards the hash of the group id, employing the KBR `route`. At all intermediate peers, the upcall `forward` is invoked to establish corresponding multicast states in CAST. On top of CAST, the authors have foreseen further multicast implementations, e.g., Scribe or Bayeux.

This approach of a universal routing protocol as part of the middleware layer may conflict with the forwarding strategy pursued by services above CAST. A simple example can be identified in the difference between Scribe [22] and Bayeux [109]. While Scribe creates a rendezvous point-based shared tree according to reverse path forwarding, Bayeux sets up source-specific states along the path from the source to the new receiver. Thus for Bayeux, CAST would establish multicast states in the opposite direction. Indeed, the idea of providing the P2P layer with a generic multicast routing logic is valuable for applications agnostic to routing services, but fails in general for OLM modules. Group specific routing forms the core component of overlay multicast, which may change by approach and domain-specific demand. In this sense, a service abstraction should only provide an interface definition, but not a routing logic.

Another application layer multicast (ALM) middleware architecture including a wrapper API is currently presented in [60]. The authors assume that an ALM protocol consists of the following parts: group management, topology management and traffic management, which can be adopted by different ALM protocols. For the latter, they propose an API for interoperability and transparency. The API calls support the selection of different transport protocols, as well as a native networks and an overlay transmission mode.

The motivation of the suggested API is to provide a unique interface for supporting structured and unstructured ALM protocols. As mentioned by the authors, the requirements slightly differ which is reflected in the API design. Unstructured group management introduces functions unknown in the context of structured overlay multicast, an external group management for example. Many unstructured multicast schemes rely on central management and provide a global view on the group structure. In general, the simultaneous support of centralized and decentralized approaches poses a severe challenge to a common middleware. Actually, such a cooperative deployment scenario is less likely than KBR protocols jointly operating with network layer multicast, as both are fully distributed and explicitly neglect a server infrastructure.

The authors in [66] present a middleware for unstructured application layer multicast only. They decompose the ALM component in several functional units, e.g., a metric estimator or a logic net to maintain and optimize the overlay network. The proposed API does not account for a

transparent overlay and underlay group communication and consists only of simple receive/send calls.

5.2.2 Current Challenges

Current concepts for the implementation of structured application layer multicast [27, 66, 60] deal with the modularization of the OLM/ALM component and a corresponding API definition. The APIs presented either focus on a common tree construction interface or a direct adoption of native group management calls. However, the concepts do not account for two further important aspects:

(a) Which type of addresses may join the application?

(b) How does the OLM API provide dedicated broadcast?

Commonly, a structured overlay network does not restrict the address space to any specific type. Each address will be handled as a string and hashed to the same identifier space without further syntactical or semantical processing. Nevertheless, the overlay may require special addresses for group communication to predefine a subset of group members like a broadcast address. Applications operate in different contexts and denote communication parties with respect to a domain-specific namespace. Special addresses should be available in all namespaces to allow for its continuous use.

Dedicated broadcast can be offered by a structured OLM API, after all applications have joined a specific "all-hosts" multicast group. This obviously does not meet native broadcast, which works without active receiver subscriptions. The problem can easily be fixed by distinguishing broadcast and multicast data. Supplementing the API with an additional broadcast interface may be one simple solution. However, broadcast and multicast operate on the same overlay which may results in key collisions while hashing identical identifiers, since there is no reserved address space. Consequently, the OLM middleware should foresee a specific, but well-known broadcast address, which can also be used to identify broadcast and multicast data on the same channel.

There are to natural options to guarantee that broadcast addresses used by different applications are always mapped on the same overlay key. The API may define a broadcast address which belongs to a specific context, but is obligatory for all applications. Application programmers then have to use this specific address and need to account for context switches, if regular group communication is based on a different namespace.

Alternatively, broadcast addresses should be embedded in every namespace. These multiple, dedicated addresses can then be mapped by the middleware to a common identifier, which does not conflict with multicast addresses. For this reason, an OLM should be aware of namespaces, and each namespace should include a unique broadcast address. This can be implemented, e.g., by using the natural 255.255.255.255 for IPv4 or the link-local all-nodes address for the IPv6 namespace. Application layer addresses like SIP URLs can reserve the asterisk for broadcast identification.

Based on the distinction of broadcast and multicast data, a group communication framework can identify data and follow the distribution algorithms implemented. OLM approaches, which do not support a dedicated broadcast scheme, may assure an all-node reachability by an automatic pre-subscription via multicast.

Identifying addresses may enhance overlay group communication in contrast to native multicast. A typical example may be news channels, e.g., sam@irtf.org and mobopts@irtf.org which fall into a combined group *@irtf.org. Based on a corresponding aggregation, a user subscribes only once, instead of joining each channel of an organisation. Such group aggregation can be offered, if an arbitrary namespace includes a broadcast address and the OLM component is aware of the namespace definition to distinguish addresses semantically. In the described example, the OLM middleware would identify the user and host parts of the compound namespace and initiate a partial broadcast to all members of irtf.org.[2]

5.2.3 A Common Network Stack for Group Communication

The design goals of an application layer multicast service for structured overlay networks are twofold. On the one hand, the *architecture* for the OLM component itself needs to be defined along with its placement in the global system. On the other hand, a generic *API* for each of the interchangeable modules must be identified.

There is a common sense in the literature to modularize P2P components, even though the perspectives differ. The Dabek model introduces a common 3 tier peer-to-peer stack with a generic routing layer for structured overlays, while contributions in the field of ALM protocols focus more on decomposing the ALM service. In the following we describe a generic architecture and its main building blocks. These components can also consist of sub-components, which depend on implementation details.

Architectural Overview

Overlay multicast supplements nodes without a global multicast connectivity with a wide-area group communication service. Thus it is important to provide a transparent (virtual) network stack to application developers beyond the P2P context. This may include enhanced group communication services like group aggregation in namespaces.

In the following, we describe the architectural components required for such an enhanced group communication and the corresponding API. The overall architecture is displayed in figure 5.3. The group communication stack consists of a middleware, underlay and overlay multicast modules. The middleware manages data exchange between the application and group services. Depending on availability and application requests, it creates a transparent overlay or a native network communication channel. In addition to common multicast interfaces for applications, the middleware provides a service API reflecting group communication states.

Overlay data will be handled by the broadcast or multicast implementation, depending on the destination address in use. Since broadcast will be delivered without explicit subscription, it is only the multicast implementation that internally provides join and leave calls.

The OLM protocols operate on overlay unicast communication. For this reason they are connected with the key-based routing layer via the common API (see section 5.1.1). The KBR can be used twofold: On the one hand, it may operate as a transmission layer delivering data to overlay peers. On the other hand, it provides group protocol implementations with unicast rout-

[2] This group aggregation does not follow the common multicast paradigm, but can for instance be naturally implemented in CAN which we only sketch: Each part of the namespace is separately hashed and corresponds to a CAN dimension. Equal addresses result in equal CAN coordinates. Flooding the selected namespace, then corresponds to data dissemination in the selected dimension.

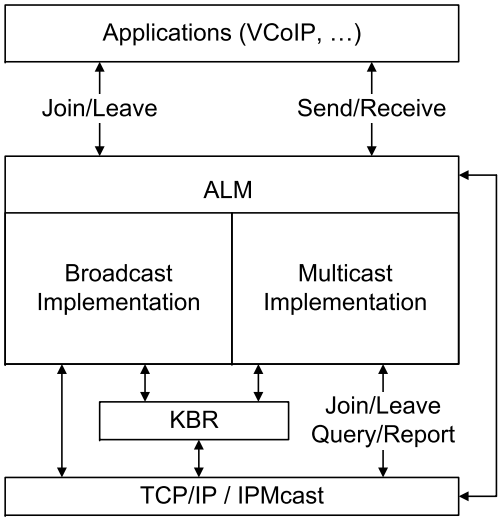


Figure 5.3: An Application Layer Multicast Middleware Embedded in a P2P Stack

ing states. In the latter case, overlay multicast and broadcast data need not use the KBR layer for transmission, but may sent data directly to peers. Therefore the multicast and broadcast components need to provide an interface to the IP layer, as well.

An API Proposal

The overlay routing is based on hashed keys. As the applications are unaware of any overlay specifics, the mapping of the destination address to the key space should be performed on the ALM layer as already suggested by Dabek *et al.* Nevertheless, it is important to preserve the original address, because an application may receive streams directed to different multicast addresses from different receivers via the same communication channel. The IP address noted in the IP header cannot be used, as the root of a key may change due to volatile peers.

As an overlay is used in a specific context, the identifiers selected by the applications are likely to belong to the same namespace. Thus, we suggest to pre-initialize the communication channel between application and group stack with the corresponding context to simplify the API calls.

The destination group address configured by the application is a common application layer or network address and is denoted by `address`. In contrast, overlay IDs are identified as `key`. The application can choose, if it sends the data to the underlay or overlay. We denote the corresponding data type `mode`, which also allows to leave the decision at the group communication stack, if the mode is unspecified. In the following, we will describe the API calls used between the group communication stack and the application.

At first, we explain calls to function for sending and receiving multicast data, thus, reflecting typical source and receiver instances.

`init(namespace→n)` This call is implemented by the multicast middleware to set the communication channel between application and stack up. It preinitializes the namespace, which then can be used for all further calls.

`void join(address→a, mode→m)` The join call is implemented by the ALM stack. It initiates a group subscription. Depending on the mode, this may result in an IGMP/MLD join, if the address equals a valid IP multicast address. The address of joins towards the overlay will be pre-processed by the middleware to implement, e.g., group aggregation and broadcast. The middleware creates a corresponding overlay key.

`void leave(address→a, mode→m)` This downcall is implemented by the middleware and results in an unsubscription for the given address.

`void send(address→a, mode→m, message→msg)` This function is invoked at the middleware to sent group data. If the overlay parameter has been configured, the middleware decides to forward the data supplied with the corresponding key to the broadcast or multicast module based on the destination address.

`void receive(address→a, message→msg)` This upcall is implemented by the application and delivers overlay and underlay messages received at the node. The address represents the destination used by the source application instance.

To request multicast states, we define the following group service API:

`nodehandle [] groupSet(mode→m)` This operation returns all registered multicast groups. The information can be provided by group management or routing protocols. The return values distinguish between sender and listener states.

`nodehandle [] neighborSet(mode→m)` This function can be invoked at the middleware to get the set of multicast routing neighbors.

`bool designatedHost(address→a)` This function is implemented by the middleware and returns true, if the host has the role of a designated forwarder or querier. Such an information is provided by almost all multicast protocols to handle packet duplication, if multiple multicast instances serve on the same subnet.

`address updateListener(mode→m)` This upcall is invoked to inform a group service about a change of listener states for a group. This is the result of receiver new subscriptions or leaves. The group service may call `groupSet` to get updated information.

`address updateSender(mode→m)` This upcall is implemented by the middleware to inform the application about source state changes. Analog to the `updateListener` case, the group service may call thereupon `groupSet`.

In the next section we describe a deployment use case for such a modular group communication stack.

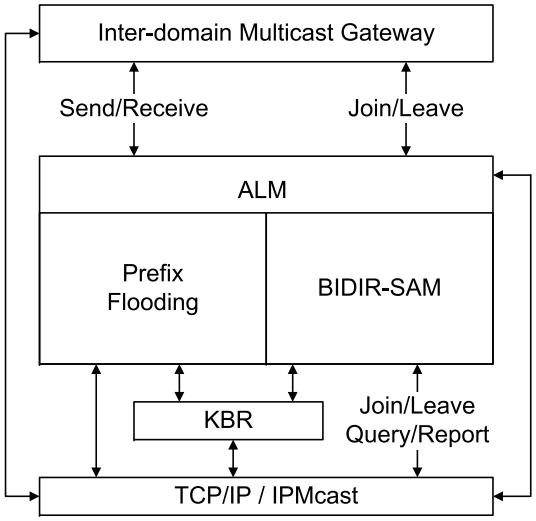


Figure 5.4: The Hybrid Shared Tree Network Stack Highlighting the Overlay Components

5.3 Design of the Hybrid Shared Tree Architecture

In this section we will describe the design of the Hybrid Shared Tree (HST) architecture, which is composed of a prefix-based P2P group communication scheme and a relay agent forwarding data between native and overlay multicast. The core component of the HST is the Inter-domain Multicast Gateway (cf. section 2.4), an application which interacts with native and overlay multicast (OLM) components via the enhanced group communication stack and thereby enables relaying (cf. figure 5.4).

5.3.1 The Inter-domain Multicast Gateway

The Inter-domain Multicast Gateway (IMG) transparently forwards multicast data between the overlay and the native network. This gateway will participate in multicast traffic originating from its attached network, which it will forward into the overlay according to the distributed multicast receiver domains of this group. It will also advertise group membership and receive data according to any subscription from its IP multicast domain. With respect to an easy deployment, the IMG should account for current multicast techniques.

The IMG represents a transition point between overlay and underlay. In this role it translates between the different protocols. A structured overlay multicast protocol does not provide any explicit group management to discover the presence of underlay receivers, since applications use direct API calls on the host. An IMG, which may represent a complete multicast domain consisting of multiple receivers and sources, acts in this sense as a proxy. It aggregates and then delegates underlay states to the overlay routing, as well as data originating from underlay sources to the overlay routing.

General Protocol Description

The multicast overlay represents the routing backbone, connecting multiple multicast domains. Hence, the construction and destruction of distribution branches will be triggered by the underlay, which includes receivers, sources, both or none of them. The IMG will be informed about

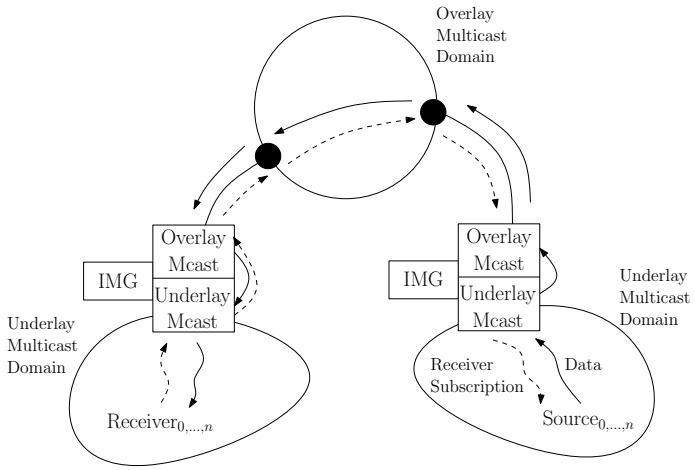


Figure 5.5: Schematic View of General IMG Scenarios

listener and sender activities from the native network by its group communication stack.

For simplicity, we describe the general protocol procedure in the case, that an IMG is placed in a receiver domain, which consists of zero or more multicast listeners, and an IMG serving a source domain with zero or more senders (cf. figure 5.5).[3]

For treating new multicast parties on the underlay, the IMG operates as follows:

Multicast Source: If the IMG learns about new underlay sources, it immediately sends a corresponding `join` to the underlay group management to receive its traffic. Thus, the IMG holds the data independent of receivers in other multicast domains. This data will be `sent` internally to the overlay routing instance, which distributes the message with respect to its forwarding states. If there is no subscription in the overlay, the data will be discarded by the routing protocol and not transmitted into the overlay.

In contrast, an update about new sources propagated on the overlay takes no effect, as joins are initiated only based on underlay subscriptions.

Multicast Receiver: For each receiver subscribing to a group as the first member in the underlay network, the IMG invokes a `join`, processed by the middleware and delegated to the overlay multicast routing protocol. The OLM instance initiates an overlay subscription. Data from the source domain can be transmitted without additional signaling in the underlay, as data is held at the corresponding IMG. Arriving overlay data in the receiver domain will be forwarded transparently by the middleware to the IMG, which distributes it via `send` into the native network.

If the IMG recognizes that the last receiver in the underlay network left the multicast group, it has to send a `leave` message into the overlay to cut multicast branches.

Several IMGs can be deployed in one multicast domain for redundancy and load sharing. Thus, it is important to prevent loops between underlay and overlay. Overlay multicast protocols in general do not implement coordinating mechanisms, since underlay distribution is

[3] For domains including both, senders and receivers, the IMG behavior can be derived by combining the separate functions.

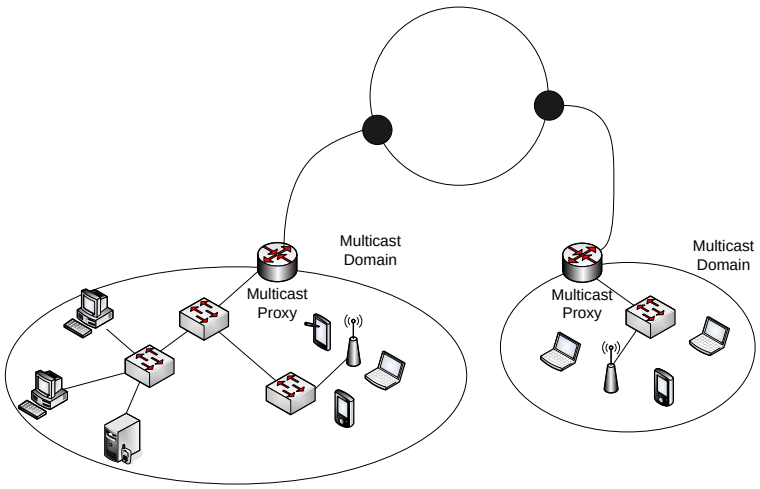


Figure 5.6: Two Small Size Multicast Domains Connected via an Overlay

performed outside the scope of these protocols. In contrast, common underlay multicast protocols are confronted with data redundancy. They have implemented appropriate forwarder election mechanisms. For this reason, the IMGs should be aware of one another based on an underlay assert signaling, which is assisted by the `designatedHost` call. If the status of the `designatedHost` is false, the IMG does not send data towards overlay and underlay.

In the following we describe how the protocol concepts can be integrated in current multicast scenarios. We distinguish between two cases: small size multicast domains without a routing infrastructure, and large size domains, which are served by native multicast forwarders.

5.3.2 Connecting Small Size Domains

A small size multicast domain consists of one IP network. Two of them are visualized in figure 5.6. Native multicast data communication is supported by group management and routing protocols. Group management is implemented in IP by IGMP/MLD [19, 100].[4] Multicast receivers send MLD (un-)subscriptions to a standardized group address, such that (potential) routers can track the presence of multicast listeners. Implementing MLD signaling represents the minimal requirement for multicast enabled devices and can be assumed in any multicast domain. The router part of MLD allows to monitor domain-wide group members by a query report scheme. The IMG operates in the MLD router part.

Packets destined to a multicast group address may be broadcasted in switched ethernet domains or selectively forwarded based on MLD snooping operated by domain switches. An MLD snooping-enabled device should transmit group membership messages and multicast data only to routers and subscribed receivers, i.e., it implements multicast on layer 2 [25].

Although MLD snooping is not standardized, its deployment is common practice [25]. However, an MLD node running the router part sends periodically group membership queries. This allows not only for learning about active receivers, but also for preventing the suppression of MLD signaling and source data by layer 2 devices.

Based on the `groupSet` call, the IMG requests the MLD state table, which provides infor-

[4] In the following we will only refer to MLDv2, because both terms can be used vice versa, as IGMPv3 implements the same scheme for IPv4.

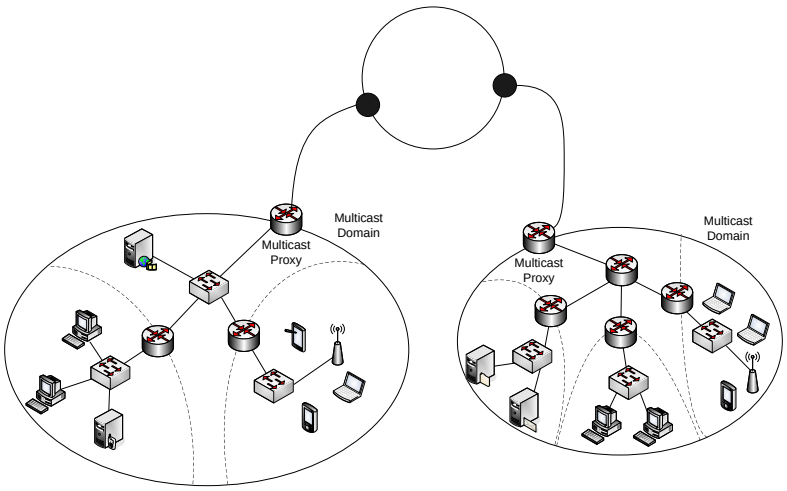


Figure 5.7: Two Large Size Multicast Domains Covering Multiple Layer 3 Networks (Dashed Lines)

mation about active listeners. In combination with the `update` call, the IMG will be informed about the first and last multicast receiver. The IMG then initiates `join` and `leave` calls towards the overlay according to the general protocol description.

To avoid transmission loops in the case of multiple IMGs connecting to the same domain, one of the relays has to be selected as forwarder. A common autonomous approach independent of end systems is a selection based on lowest IP addresses. For this reason, nodes periodically send generic protocol advertisements to the local domain and listen to notifications of other systems. A node does not establish forwarding states, when it receives advertisements with a lower IP address than its own. This is implemented in the MLD router part to avoid multiple group queries and data duplications. Thus, only the IMG with lowest IP address acts as designated relay between the overlay and a MLD domain.

Interconnecting multiple domains without deploying a multicast routing protocol is specified in [38]. The IMG operates in concordance with this standard, which is solely based on IGMP/MLD.

Running the IMG as MLD proxy easily allows to connect small multicast networks, which are not neighboring. On the one hand, this approach reduces deployment complexity as the IMG can be placed all over the multicast domain without obligation to maintain a multicast routing protocol. On the other hand, this scheme limits its scope by the layer 3 region, because MLD signaling is not forwarded across routers.

5.3.3 Connecting Large Size Domains

Connecting multicast islands by an IMG MLD proxy architecture requires a layer 2 access in each local LAN. It does not scale to establish a separate proxy in any layer 2 domain of a corresponding larger network. In addition, most larger network domains have established a local host-group routing which provides domain-wide multicast, but fails on global multicast connectivity. In such cases, an IMG should be incorporated into the local routing infrastructure to interconnect larger native multicast islands (cf. figure 5.7).

Like in the proxy scenarios, a hybrid multicast gateway must be aware of all groups inside

a multicast domain to initiate corresponding states in the overlay. In contrast to link-local domains, which can solely be monitored by a group membership protocol, group states are distributed in routed multicast sites. Hence, an IMG requires an interface to the routing infrastructure, where subscriptions occur. In general, this depends on the multicast routing protocol deployed. In rendezvous point (RP) schemes like PIM-SM, all receiver subscriptions and source data will be registered at the RP. Flooding schemes like DVMRP, however, distribute the information across all neighboring routers.

In the following, we sketch methods to integrate the IMG in a selection of multicast routing architectures.

DVMRP In the Distance Vector Multicast Routing Protocol (DVMRP) [104, 74] source data is flooded covering all multicast domain routers. If no group member is present at the subnets of a router, this router sends a prune message to its DVMRP neighbors to cut the multicast distribution branch. DVMRP routers maintain information of source group pairs to prevent incorrect data propagation and to incorporate new multicast branches. New receivers are integrated by sending graft messages towards the multicast source. This activates new forwarding states at intermediate routers. Thus, an arbitrary DVMRP router will not be informed about new receivers, but will learn about new sources immediately.

The concept of DVMRP does not provide any central multicast instance. Thus, the IMG can be placed anywhere inside the multicast region, but requires a DVMRP neighbor connectivity. The group communication stack used by the IMG is enhanced by a DVMRP implementation. New sources in the underlay will be advertised based on the DVMRP flooding mechanism and received by the IMG's DVMRP instance, which informs the group communication stack middleware. The middleware delegates the call to the relay application. The relay agent initiates a corresponding join in the native network and forwards the received source data towards the overlay routing protocol. Depending on the group states, the data will be distributed to overlay peers.

DVMRP establishes source specific multicast trees. Therefore, a graft message is only visible for DVMRP routers on the path from the new receiver subnet to the source, but in general not for the IMG. To overcome this problem, there are two approaches:

(1) In the case of smaller DVMRP domains, an IMG can be installed as proxy (cf. section 5.3.2) in each local subnet with respect to our general protocol description. This is an appropriate option, as the deployment of DVMRP is limited. To avoid loops induced by forwarding overlay multicast data also via DVMRP, the IMG has to send the data with a time to live of one hop.

(2) In larger DVMRP networks, a single, domain-wide IMG should be deployed. This requires to flood the data of multicast senders in the overlay as well as in the underlay. Hence, an IMG has to initiate an all-group join to the overlay using the namespace extension of the API. Each IMG is initially required to forward the received overlay data to the underlay, independent of native multicast receivers. Subsequent prunes may limit unwanted data distribution thereafter.

PIM-SM The Protocol Independent Multicast Sparse Mode (PIM-SM) [39] establishes rendezvous points (RP). These entities receive listener and source subscriptions of a domain. To be

continuously updated, an IMG has to be co-located with a RP. Whenever PIM register messages are received, the PIM routing instance must signal a new multicast source towards the stack. Subsequently, the IMG joins the group and a shared tree between the RP and the sources will be established, which may change to a source specific tree after a sufficient number of data has been delivered. Source traffic will be forwarded to the RP based on the IMG join, even if there are no further receivers in the native multicast domain.

Designated routers of a PIM-domain send receiver subscriptions towards the PIM-SM RP. The reception of such messages invokes an update call at the IMG, which initiates a join towards the overlay routing protocol. Overlay multicast data arriving at the IMG will then transparently be forwarded in the underlay network and distributed through the RP instance.

PIM-SSM PIM Source Specific Multicast (PIM-SSM) is defined as part of PIM-SM and admits source specific joins (S, G) according to the source specific host group model [47]. Based on the unicast source address S, a multicast subscription can be forwarded directly to the multicast sender. Hence, a multicast distribution tree can be established without the assistance of a rendezvous point.

Sources are not advertised within a PIM-SSM domain. Consequently, an IMG cannot anticipate the local join inside a sender domain and deliver a priori the multicast data to the overlay instance. If an IMG of a receiver domain initiates a group subscription via the overlay routing protocol, relaying multicast data fails, as data are not available at the overlay instance.

The IMG instance of the receiver domain, thus, has to locate the IMG instance of the source domain to trigger the corresponding join. In the sense of PIM-SSM, the signaling should not be flooded in underlay and overlay.

One solution could be to intercept the subscription at both, source and receiver sites: To monitor multicast receiver subscriptions in the underlay, the IMG is placed on path towards the source, e.g., at a domain border router. This router intercepts join messages and extracts the unicast source address S, initializing an IMG specific join to S via regular unicast. Multicast data arriving at the IMG of the sender domain can be distributed via the overlay.

Discovering the IMG of a multicast sender domain may be implemented analogously to AMT [94] by anycast. Consequently, the source address S of the group (S, G) should be built based on an anycast prefix. The corresponding IMG anycast address for a source domain is then derived from the prefix of S.

BIDIR-PIM Bidirectional PIM [42] is a variant of PIM-SM. In contrast to PIM-SM, the protocol pre-establishes bidirectional shared trees per group, connecting multicast sources and receivers. The rendezvous points are virtualized in BIDIR-PIM as an address to identify on-tree directions (up and down). However, routers with the best link towards the (virtualized) rendezvous point address are selected as designated forwarders for a link-local domain and represent the actual distribution tree.

The IMG should be placed at the RP-link, where the rendezvous point address is located. As source data in either cases will be transmitted to the rendezvous point address, the BIDIR-PIM instance of the IMG receives the data and can signal new senders towards the stack.

The first receiver subscription for a new group within a BIDIR-PIM domain needs to be transmitted to the RP to establish the first branching point. Using the update invocation, an IMG will thereby be informed about group requests from its domain, which are then delegated

to the overlay.

5.4 Design of the Prefix Flooding & BIDIR-SAM

The bidirectional scalable adaptive multicast protocol (BIDIR-SAM) (cf. section 4.2) and the prefix flooding (cf. section 3.2) are overlay group communication protocols used on top of a prefix-based KBR layer. The KBR layer serves as overlay unicast routing information base. It should provide underlay proximity selection and is represented by Pastry in our implementation.

The overlay packet transmissions can be implemented using the KBR substrate or by a direct TCP/IP socket connection to the destination. The first of these two options results in double encapsulation and additional internal stack calls, which should be avoided to reduce packet size and load inside the structured overlay. Thus, data messages as well as control messages are sent directly via the TCP/IP layer. As both protocols may operate separately, they open different TCP/IP sockets using a predefined TCP port number.

To find the responsible peer for a given key, the key's root, the KBR routing table will be searched by the `local_lookup` call of the common API by Dabek *et al.* [27] (cf. section 5.1.1). This ensures the latest valid entry for the routing decision of BIDIR-SAM and the prefix flooding, and does not cause additional signaling overhead. However, both protocols need awareness of the key space to internally 'rebuild' the Pastry routing table. The parameters are configurable in Pastry and available to BIDIR-SAM and the prefix flooding.

It is sufficient to use the key data type provided by the KBR as data structure for the prefix key. The generated key has to agree with the prefix digits and carry the prefix length. Each overlay key corresponds to exactly one entry in the Pastry routing table. Thus, applying the KBR lookup on such a key will determine the unique position inside the Pastry routing table and return a valid peer.

Both schemes comply with the proposed API in section 5.2.3. However, as the prefix flooding operates stateless and in particular without subscriptions triggered by applications, it does not require any group management and provides only `send` and `receive` calls.

5.4.1 Tree Construction & Tree Maintenance for BIDIR-SAM

Each BIDIR-SAM peer creates a prefix neighbor table maintained by group management functions and used by the routing scheme. This table contains per group entries including the corresponding destination prefixes and refresh timers. Entries will be added according to the BIDIR-SAM Join and deleted based on the BIDIR-SAM Leave algorithm (see section 4.2. If a refresh timer expires, the maintenance routine is invoked to identify invalid neighbors.

For the construction and maintenance signaling we re-use parts of the tree life cycle message scheme presented in [17] to keep messaging overlay agnostic. This generic ALM overlay definition includes tree creation and maintenance messages, which are exchanged between the BIDIR-SAM peers to update multicast forwarding states.

At first, we will describe the semantic of the join and leave message:

Join Message(PeerId k, GroupId g) is sent to inform prefix neighbors about a new downstream peer k for group g. The reception of this message will create a new entry in the prefix neighbor list.

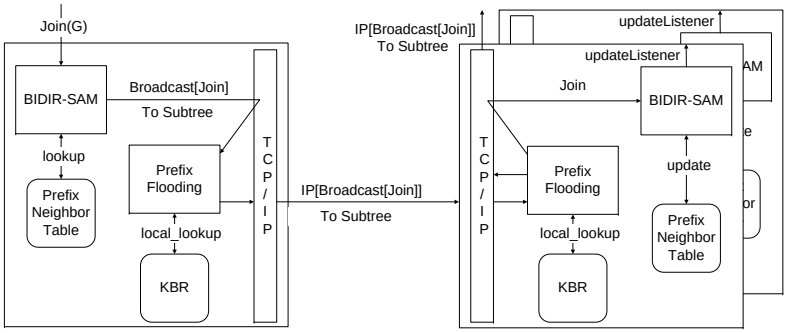


Figure 5.8: Schematic View of the Join Call Procedure

Leave Message(PeerId k, GroupId g) is used to inform prefix neighbors that the downstream peer k does not belong to the group g anymore. The reception of this message will delete the corresponding entry in the prefix neighbor list.

If the `join` call is invoked (cf. figure 5.8) by an application on top of BIDIR-SAM, the multicast group address will be mapped to an overlay group id by the hash function. In the next step, the prefix neighbor table will be checked for entries. With respect to BIDIR-SAM JOIN (cf. section 4.2), the prefix neighbors will be calculated. As the join message has to be broadcasted in the corresponding prefix subtree, BIDIR-SAM passes the packet locally to the prefix flooding. The prefix flooding initiates the broadcast dissemination procedure for the given prefix length. Underlay addresses required for forwarding will be resolved by the KBR `local_lookup`. Now, the join message can be sent directly to the prefix neighbors.

The reception of a join message will invoke in the `updateListener` call at the group communication stack to inform applications about a state change. Corresponding operations will be performed on reception of a `leave` call at the group communication stack.

The multicast states of BIDIR-SAM can be inquired by the `groupSet` and `neighborSet` functions. The first call returns all entries of the prefix neighbor table, which may be `NULL` in the absence of receivers. The latter call returns overlay unicast prefix neighbors, i.e., all valid prefixes included in the Pastry routing table.

Although, the tree construction algorithm of BIDIR-SAM fills and purges the prefix neighbor table correctly, entries may be invalidated due to silent disappearance of receivers from the multicast overlay. To keep the multicast routing table updated, BIDIR-SAM is equipped with a maintenance routine. The idea is the following soft state approach: When the refresh timer expires for a prefix neighbor, a peer sends a group query to it. If the receiving peer is not a multicast listener and has no neighbor entry for the corresponding subtree, it sends a leave message to the subtree originating the group query, otherwise it transmits a group report. Invalid entries can be deleted based on the leave message or a timeout. In contrast, a group report will reset the refresh timer and keep the entry.

For the maintenance mechanism, we extend the messages in [17] as they only provide a heartbeat announcement:

Group Query(PeerId k, GroupId g) will be sent from k to the expired prefix neighbor to inquire for existing downstream neighbors. If no group report is received after a specific time, the entry will be deleted.

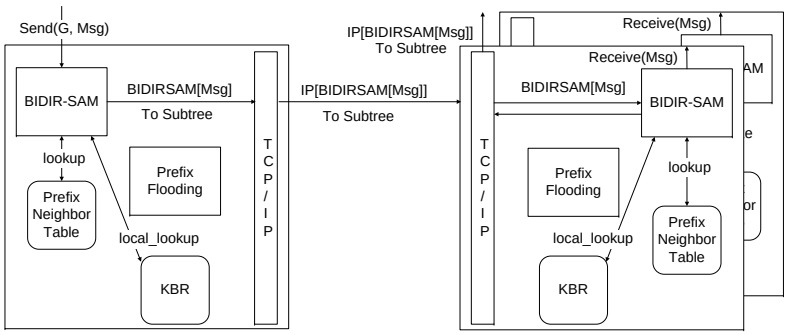


Figure 5.9: Schematic View of the Send Call Procedure for BIDIR-SAM

Group Report(PeerId k, GroupId g) will be transmitted, whenever a group query has been received at the peer k and k's prefix neighbor table contains entries larger than or equal the destination prefix. On reception of a group report, the inquirer will update the refresh timer.

5.4.2 Data Distribution for Broadcast and Multicast

Multicast or broadcast data created by the applications are encapsulated in an overlay multicast or broadcast data frame carrying overlay specific information. The data message frame contains the destination prefix length, and a sequence number to identify packet duplication or facilitates further services like ordering, reliability and prevention of replay attacks. For BIDIR-SAM it also carries the overlay group address to select the correct forwarding states.

BIDIR-SAM Data Message(Int k, GroupId g, Int seq) is used to encapsulate the application data destined for group g. The message is forwarded to the TCP/IP layer and transmitted to the prefix neighbor. Based on the prefix length k, the destination prefix will be reconstructed.

Broadcast Data Message(Int k, Int seq) encapsulates application data, which are to be transmitted to all overlay peers. Based on the prefix length k, the destination prefix will be reconstructed.

Invocation of `send` in BIDIR-SAM

When the multicast `send` call is invoked (cf. figure 5.9), BIDIR-SAM checks for its local prefix neighbors according to the group address hash. Without entries, the packet is silently discarded. Each existing prefix is resolved by the KBR state access function `local_lookup`, which returns a destination node handle including the IP address. The data obtained from the application above BIDIR-SAM will be encapsulated in a BIDIR-SAM data message and sent via TCP to the prefix neighbor. The BIDIR-SAM recipient will encapsulate the message and deliver the data via `receive` to the application, if the peer is a receiver. According to its entries in the prefix neighbor table and the BIDIR-SAM multicast routing, the message is duplicated and forwarded to further neighbors.

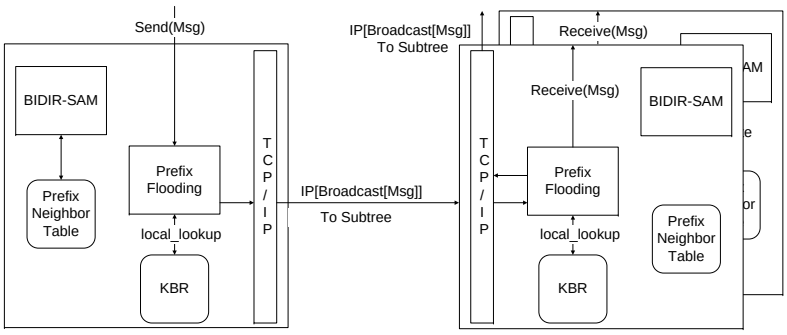


Figure 5.10: Schematic View of the Send Call Procedure for the Prefix Flooding

Invocation of `send` in the Prefix Flooding

In contrast to BIDIR-SAM, the broadcast `send` function does not require a forwarding state table as the distribution tree can be resolved of the KBR (cf. figure 5.10). Thus, it extracts all prefix neighbor keys, which can be derived from the Pastry routing table layout. The prefix flooding then iterates over the Pastry prefix table via `local_lookup` to get the corresponding underlay addresses. The encapsulated data message is passed to the TCP/IP socket and distributed to all available prefix neighbors. The receiving peer forwards the data towards the application via the `receive` call and distributes it to other prefix neighbors.

5.4.3 Routing Maintenance on Pastry

BIDIR-SAM and the prefix flooding require a complete KBR routing table to ensure correct message distribution. As Pastry does not guarantee this condition, we propose a proactive maintenance mechanism, which periodically discovers overlay peers for empty routing table entries.

The idea is the following: The routing maintenance iterates via `local_lookup` over the Pastry routing table to find an empty entry. For such an entry, the routing maintenance initiates the `route` call of the common KBR API for routing a maintenance message. This sends the maintenance message to the key's root. The receiver will reply with its ID to the requesting peer. Based on the answer, the maintenance peer verifies, whether the key's root ID matches the empty entry. If so, the overlay node will be merged into the Pastry routing table, otherwise ignored.

As an overlay node is responsible for a set of keys, which may cover multiple routing table entries of a peer, some empty entries can be omitted: We assume that the requested entry $\mathcal{P}$ is empty. Receiving an answer from node $\mathcal{P}'$, the maintenance peer maps $\mathcal{P}'$ to its appropriate position in the Pastry routing table and proceeds on the following entry. This skips all empty columns between $\mathcal{P}$ and $\mathcal{P}'$.

For the routing maintenance routine we propose the messages:

Routing Maintenance Request(PeerId k) will be sent from k via the KBR to an overlay key which matches an empty entry in the Pastry routing table.

Routing Maintenance Reply(PeerId k) will be sent from k on the reception of a routing maintenance request message.

6 Implementation

6.1 Introduction

Aside from theoretical analysis, protocols can be evaluated based on real-world experiments, emulations and simulations. Analyzing new structured P2P networks in a hands-on way is a difficult problem, as they are designed for a large number of participants. Many relevant protocol effects will only become visible for network sizes varying over several orders of magnitude.

Real-world large-scale experiments require a wide testbed. There are public platforms available, which support researchers with a large number of accessible end devices. One of the most popular project is PlanetLab admitting more than 800 nodes. However, due to frequent system overuse, further platform-specific problem are induced. As stated by its inventors is not suitable for reproducible experiments. Rhea *et al.* [80] show, that experiments conducted on PlanetLab may be biased, because systems are overloaded and slow. As we are at the beginning of the analysis of our protocols, it is important to obtain a clean, fundamental understanding. Thus, we decided to avoid such 'sources of irritation', and concentrate on a controlled environment. Real-world experiments on PlanetLab may be compared to our initial evaluations in future experiments.

Network emulators represent a compromise between real-world deployment and simulations. They are made to run on one or few hosts, using internals port as interprocess communication. There are few developments around in the context of peer-to-peer, e.g., Overlay Weaver [90, 91]. However, they are usually dedicated to specialized tasks, i.e., P2P evaluations, and do not allow an easy incorporation of additional node instances or protocols. As we want to evaluate our protocol on top of Internet topologies in further experiments, we have settled for a rich simulator platform.

The selection of a simulator for structured P2P networks should account for all the requirements above: feasibility of simulations with at least ten thousand nodes, availability of general, non-P2P protocols and an interface to real-world networks. Additionally, for the analysis of our protocols, the simulator needs to support Pastry as well as Scribe. With respect to a possible code re-use of our implementation, the P2P API should implement the Dabek model (cf. section 5.1.1).

In many cases, current structured P2P simulators are limited to a few thousands of nodes [68] or do not support multicast. Recent developments like PeerfactSim [54] have a strong focus on realistic link delays for the underlying topology, but provide only a peer-to-peer environment without a common network layer. An appropriate choice fulfilling the most important of our requirements is the recently launched OverSim [10]. OverSim is implemented on the the well-known generic network simulator OMNeT++ [99].

6.2 A P2P Simulation Framework: OverSim

OverSim [10] is an open-source overlay network simulation framework for OMNeT++ [99], developed at the University of Karlsruhe. OMNeT++ is discrete event simulation system, programmed and extendable in C++. OverSim includes the following features [9]:

- a set of structured and unstructured P2P protocols
- a layer-based architecture conforming to the Dabek model
- three underlying network models
- application layer multicast support[1]
- scalability up to 100.000 nodes in very simple networks
- an injection option for real-world network traffic.

However, the simulator includes some drawbacks. Up until now, there is no support to include externally modeled or real-world topology data sets for setting up realistic underlay topologies. It is likewise not possible to create multiple overlay services on the same P2P middleware layer.

In the following, we will describe the fundamental classes of OverSim representing the base for our own developments.

Class Concept

OverSim includes a nested peer-to-peer layer, which consists of an overlay tier and three tiers designed for applications and services (cf. figure 6.1). This concept is reflected by its classes. Overlay protocols are based on the `BaseOverlay` class and applications thereupon on the `BaseApp` class. `BaseApp` provides new classes equipped with the common API by Dabek *et al.* (cf. section 5.1.1) and interfaces to the RPC and UDP layer. KBR calls invoke corresponding functions of the `BaseOverlay` class, which have to be implemented by the KBR/DHT protocols. However, KBR message routing calls are only available on tier 1. This limitation prevents services from being implemented on an arbitrary tier if the routing API is required.

Overlay services and applications will be placed on tiers 1-3 within the OverSim P2P stack. Although each module can include multiple submodules, the current design of OverSim does not allow to derive several classes from the `BaseApp` that will be located on one tier.

6.3 Implementation in OverSim

In this section, we will sketch our implementation of the prefix flooding and BIDIR-SAM 6.2. With respect to future updates, the implementation is guided by the attempt to leave original OverSim classes unmodified as far as possible. To discover memory leaks, we debugged our software using Valgrind [96].

OverSim does not fully allow to deploy several protocols on one P2P layer, because inter-layer communication does not provide virtualized ports. Recalling the concepts of our P2P group communication stack, multiple protocols, i.e., broadcast and multicast should reside on one tier.

[1] Actually, application layer multicast was not available at the beginning of this work, but was provided later.

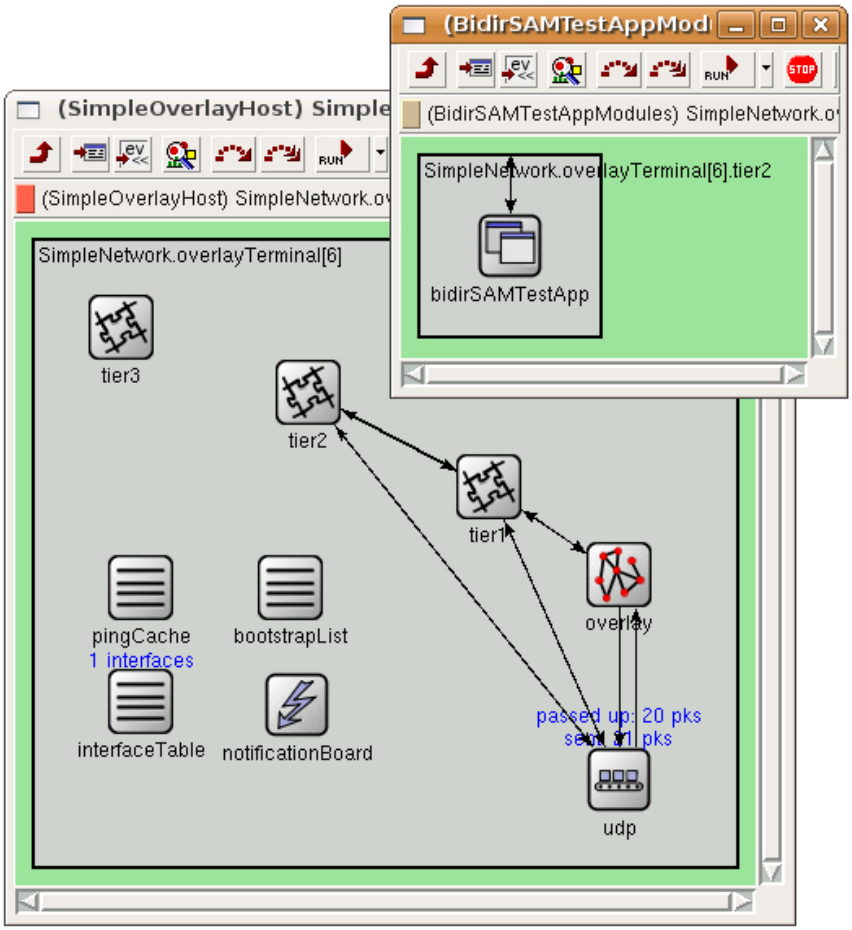


Figure 6.1: The Network Stack for a Simple OverSim Node Highlighting an Application on Tier 2 of the P2P Stack

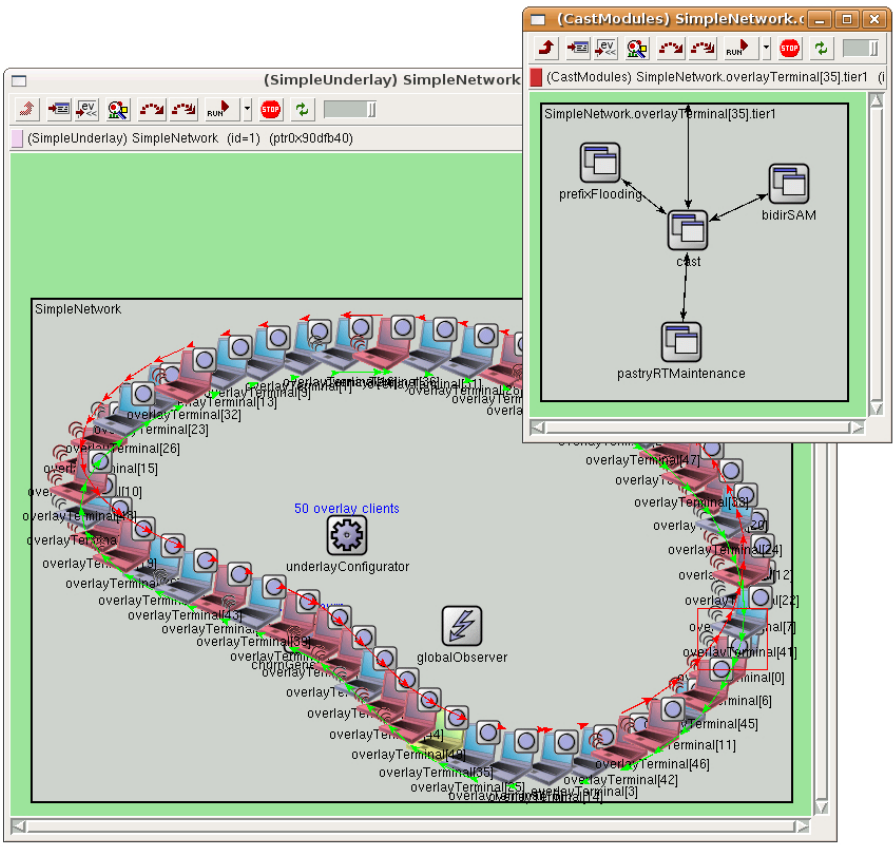


Figure 6.2: A BIDIR-SAM Network With 50 Peers Highlighting the Multicast Stack. Multicast Receivers and Source are Colored in Red and Yellow Respectively.

Additionally, the proactive routing maintenance should not be intermingled with existing Pastry code base. For this purpose, our module is implemented as an additional service on top of the overlay. Thus, we designed a multiplexer delegating messages between tier 1 and further layers.

In addition to the pure protocol implemention, a test application is required to issue multicast traffic, as well as a global observer to conduct the experiment. The multicast observer module controls the experiment, i.e., generates the multicast group address, selects source and receivers, initiates data transmission via the test application and collects statistical data of experimental outcome. The multicast test application performs the group join/leave and send/receive of multicast data via the multicast API.

6.3.1 Prefix Flooding

The prefix flooding class implements a message handler, which broadcasts all packets received from upper applications to overlay nodes. We have to distinguish between source initiated broadcast and forwarding of received packets. In the first case, the broadcast originates from level 0 of the distribution tree. In the second case, a destination prefix is received and forwarded to the calculated prefix neighbors. Additionally, the packet will be delivered to the application. The distribution algorithm is equal for both cases. The corresponding code snippet is shown in listing 6.1.

The prefix neighbor will be calculated by a helper routine (`createPrefixKey`) based on the current row and column, the local overlay ID and the Pastry key space parameters. The lookup call inspects the KBR, i.e., Pastry routing table and resolves the IP address of the destination prefix. Finally, the message will be encapsulated in an internal stack message to send the message out via the network layer.

```
1 void PastryBroadcast::distributeBroadcast(PastryBroadcastMessage* msg, uint
    row) {
2   //[...]
3   //Iterate over the Pastry table rows, start at destination prefix
4   for (uint i = row; i < pastryKeyLength; ++i) {
5     //For each column in the Pastry routing table
6     for(uint j = 0; j < pastryKeyBase; ++j) {
7       //Create prefix neighbor
8       prefix = PrefixKey::createPrefixKey(ownKey, i, j, pastryKeyLength,
            bitsPerDigit);
9       //Resolve prefix via KBR API
10      nodeVector = callLocalLookup(prefix, 1, 1, false);
11      //Ignore empty RT entries
12      if(nodeVector->size() == 1 && (!((*nodeVector)[0].isUnspecified()))) {
13        PastryBroadcastMessage *msgCopy = (PastryBroadcastMessage *) msg->dup
              (); //Create Broadcast message
14        msgCopy->setPrefixlength((i+1)*bitsPerDigit);
15        //[...]
16        //Encapsulate message for internal cast forwarding (multiplexing)
17        CastInternalUdpMsg* internmsg = new CastInternalUdpMsg();
18        internmsg->setDstIP((*nodeVector)[0].ip);
19        //[...]
20        send(internmsg,"from_bcast"); //Send message to Cast/transport layer
21      }
22      delete nodeVector;
23    }
24  }
25 }
```

Listing 6.1: Snippet of the Prefix Flooding Forwarding Algorithm

6.3.2 BIDIR-SAM

In contrast to the prefix flooding, BIDIR-SAM requires group membership management. This includes join/leave functions and the multicast fowarding table. The latter should be efficient to lookup prefix neighbors for a multicast group. As the forwading algorithm operates similar to the prefix flooding, we only focus on the group management in the following.

The multicast forwarding table is implemented as map data structure (cf. listing 6.2). It consists of the group address as identifier and an ordered multiset of destination prefixes. Finding destination prefixes for given group is performed in $O(\log n)$. As the multicast set is ordered, the forwarding algorithm iterates directly over the data structure and stops immediately at insufficient prefix lengths. The average complexity also equals $O(\log n)$.

```
1 //BidirSAM.h
2 //[...]
3 typedef map<OverlayKey,BidirSAMListeners> GroupList;
```

```
4 GroupList groupList;
5 //BidirSAMListeners.h
6 class BidirSAMListeners {
7 public:
8   OverlayKey groupIdentifier;
9   BidirSAMListeners(OverlayKey groupId);
10  void addListeners(PrefixKey k);
11  void removeListeners(PrefixKey k);
12  PrefixKey getLongestPrefix();
13  multiset<PrefixKey>::iterator getListenersBegin();
14 //[...]
```

Listing 6.2: Snippet of the Multicast Forwarding Table Declaration

The multicast group membership is twofold. For simplicity we only focus on the join: On the one hand, receiver subscriptions initiated by the local application will be injected in the correct prefix subtrees. On the other hand, received joins have to be propagated towards the leaves. Both functions use the prefix flooding module as shown for the join injection in listing 6.3.

```
1 void BidirSAM::subscribeToGroup(OverlayKey groupId, BidirSAMCtrlInfo*
    ctrlInfo) {
2   SAMJoinMessage* samjoinMsg = new SAMJoinMessage();
3   samjoinMsg->setGroupId(groupId);
4   //[...]
5   PastryBroadcastMessage* bcastMsg = new PastryBroadcastMessage();
6   //[...]
7   // Insert group into grouplist, if unknown
8   groupInserter = groupList.insert(make_pair(groupId, BidirSAMListeners(
      groupId)));
9   BidirSAMListeners& listeners = groupInserter.first->second;
10  listeners.setSubscriptionStatus(true); // Set subscription status
11  //Inject BIDIR-SAM join
12  //First subscribed router to this group
13  if(listeners.empty()) {
14    //Initiate flooding
15    bcastMsg->setPrefixlength(0);
16    bcastMsg->encapsulate(samjoinMsg);
17    //[...]
18    sendDirect(bcastMsg,0,bcastModule,"direct_in",-1);
19  }
20  //Not first subscribed router to this group
21  else {
22    //Initiate partial flooding => select LCP
23    multiset<PrefixKey>::iterator iLongestPrefix = listeners.
        getListenersBegin();
24    //[...]
25    //Start broadcast at routing table row
26    bcastMsg->setPrefixlength((longestPrefix.getOverlayKey()).
        sharedPrefixLength(thisNode.key));
27    bcastMsg->encapsulate(samjoinMsg);
28    sendDirect(bcastMsg,0,bcastModule,"direct_in",-1);
29  }
30 }
```

Listing 6.3: Snippet of the Join Injection Implementation

6.4 Analytical Calculations

For ease of use, our analytical calculations have mainly been untertaken using the data analysis and graphing program Origin 8 [69]. This software includes a programming language that supports ANSI C and can directly access internal Origin data sets. However, general numerical data types of C (as well as C++) are incapable of processing the very large numbers required to evaluate the expressions in common.

Although prefix flooding and BIDIR-SAM measures evaluate on a small scale, subexpressions of the analytical functions can grow tremendously due to powers. Final values are frequently obtained by quotients or difference cancelations, both of which are numerically sensitive. To avoid numerical artifacts, we calculated values based on the arbitrary precision arithmetic library Apfloat [7]. Apfloat is available for C++ and Java. In contrast to the standard arbitrary precision data types of Java, this library provides various mathematical functions that are important for our calculations, e.g., powers of real numbers.

7 Conclusion & Outlook

The Internet uniquely offers the service of distributing data in a multicast host group model. Nevertheless, this fundamental service still suffers from a state of deployment too restrictive to allow for global dissemination of group communication services. This thesis is concerned with the design and the analysis of a structured multicast overlay solution, which may augment the Internet core and facilitate globally available & seamlessly integrated multicast network services of general use.

7.1 Achievements of this Work

As the major contribution of this work, we have proposed and analyzed BIDIR-SAM, the first structured overlay multicast scheme based on bi-directional shared prefix trees. Data distribution in BIDIR-SAM is guided along a logarithmically scalable source specific shortest path tree, even though it follows the general Any Source Multicast Model scheme. We presented a detailed concept for its deployment in a hybrid multicast architecture, combining IP and Layer 2 group communication services at edge networks with a structured overlay backbone on top of the Internet core. The correctness of our algorithms has been proven. The analysis is justified by extensive theoretical and simulation-based results.

In our hybrid approach, unlike in conventional mono-layer solutions, the well-adopted native multicast in enterprise domains is complemented by scalable, robust and transparent transmission services on structured overlays. Resting upon this newly developed routing scheme, the overlay will allow operators to deploy segregated, individually configurable multicast services with rigorously predictable system load, while leaving the inter-domain Internet unicast backbone untouched. Furthermore a shared per group forwarding decouples group state establishment from the data plane, which gives rise to an option of transparent, scalable support for mobile group communication.

In detail, we have started with the broadcast scenario and presented the prefix flooding approach (cf. chapter 3). This service on top of a key-based routing disseminates data at predictable costs to all overlay members. We pursued a comprehensive discussion of our results with respect to large-scale group communication. This included an elaboration on the problem of building distribution trees from reverse pathes, while asymmetric routes are present in overlay networks.

In chapter 4 we introduced and evaluated BIDIR-SAM, the overlay multicast approach to scalable, prefix tree dissemination.. This approach is well suitable for large-scale groups. Our analytical work granted general insight into multicast forwarding on k-ary prefix trees. This outcome may be of further value for adjusting real-world deployment. Providers for example, which use BIDIR-SAM in a hybrid environment, are thereby enabled to pre-calculate network costs. Our simulations have been nicely explained by this theoretical analysis.

BIDIR-SAM establishes a virtual shared distribution tree in prefix space, which is jointly

known at routing peers. This collective knowledge offers a variety of additional functions, out of which a scalable degree of reliability appears as most important. Without additional signaling, but exploiting network coding, the overlay multicast scheme can assure a k-level redundancy in data and routing paths. Similar strategies can be used to achieve a variable load sharing to serve data intensive streams, e.g., in IPTV applications.

In chapter 5 we discussed the design of a generic group communication architecture combining overlay and native multicast. We extend current structured P2P middleware approaches by a universal multicast API, integrated into a group communication network stack. Further on, the integration and implementation of the hybrid shared tree architecture for common Internet multicast routing protocols has been detailed out. These results can be adopted for other hybrid protocols, as well.

Finally, we implemented our protocols in the well-known P2P simulator OverSim, which was discussed in chapter 6. This implementation covers the full protocol feature set and can be used to extend this simulation framework by a DHT-based broadcast mechanism and the new BIDIR-SAM multicast approach.

We would like to conclude that the initial idea of employing a prefix-guided bi-directional shared distribution tree led to a promising group communication scheme, which – as we believe – will demonstrate its value in practice.

7.2 Future Tasks

This thesis gives results for overlay group communication and its deployment. Nevertheless, during this work new questions opened on the one hand, which will be exciting to answer, and some problems are unfortunately left open due to time and compute capacity restrictions, on the other hand.

Additional Analytical and Simulation Analysis

For the prefix flooding and BIDIR-SAM, the most important next step leads to the protocol analysis based on real-world topologies provided by CAIDA [55] and DIMES [89]. Such an integration in the OverSim simulator has been omitted as it was not foreseen in the available implementation and colleagues from the Athens University of Economics and Business currently prepare corresponding add-ons. Real-world topologies and link delays represent the fundament to gaining insight in proximity neighbor effects that reduce overlay stretch and stress.

Additionally, the analysis should include new topology models, like the dK-graphs [64], which allow to perform evaluations on a downsized network still preserving characteristic graph properties [63]. This reduction is desirable with respect to memory and computation restrictions on current computers. It would be interesting to see the sensitivity of our metrics on the underlying topology.

Another topic for BIDIR-SAM is offered by the more complex evaluation of our proposed optimizations. They have only been sketched and should be fully implemented in the existing OverSim modules. Then, the simulation scenario should be enhanced for multiple multicast sources and different groups. Further on, we will include experiments on reliability with appropriate churn models representing varying application scenarios.

Beyond simulations or practical experiments we will refine our analytical work. Although the

current outcome is quite expressive, simpler terms may be found and should also account for our optimizations.

Real-World Deployment

The subsequent steps in the context of our group communication architecture would be a real-world implementation of an underlay and overlay aware stack and its usage in an application benefiting from global multicast support. The latter is foreseen in cooperation with our video conferencing partner [28]. The transparent group communication stack can be enriched by name space support, which should be detailed out for DHTs in general.

The stack is a prerequisite to set up an IMG. With respect to a real deployment, the location of the IMG within the network should be defined more precisely. Such positioning questions can most likely be solved with the help of service placement theory.

It is also valuable to think about an adaptation of our prefix-based distribution algorithms to wireless sensor networks (WSNs). Commonly, structured overlay networks as well as WSNs focus on the self-organization of its members. A multicast approach may facilitate WSNs with data aggregation, which enhances transmission capacities and reduces power consumption. Our prefix-based scheme appears promising with its large-scale logarithmic performance behaviour.

Applicability to Multicast Mobility

Key-based routing provides a virtualized network layer. Decoupling hash keys from IP addresses e.g., by using location independent identifiers, will provide applications with a routing layer transparent under IP address changes. Hence, any overlay multicast protocol will preserve distribution trees under mobility in contrast to native multicast. In addition, BIDIR-SAM includes a flexibility as it creates bi-directional shared trees, which decouple group and state management from the forwarding plane. Multicast transmission will thus be location-transparent. Any multicast source can submit packets from any location to the same prefix distribution tree without the need of mobility-related signaling or assisting agents.

This property will likewise hold in hybrid Internet architectures, wherever intra-domain protocols reflect mobility transparency. Thus in combination with Bidir-PIM at edge domains, Hybrid Shared Tree will lead to a mobility-agnostic routing environment in the sense that listeners and senders may freely move on an inter-domain scale, while a mobility-unaware routing layer will equally enable multicast services. Senders can seamlessly transmit multicast data from any location, while listeners can benefit from uninterrupted services wherever they encounter previously established group reception.

Applicability to Internet Routing

A challenging question remains about a direct applicability of our solution to the Internet routing layer. Facing the ongoing debate on a clean slate replacement of the current Internet protocol by routing on a flat, structured address space, one may reflect on combining core ingredients of the key-based routing with the currently deployed Internet architecture.

Following these considerations, it springs into mind that the current Classless Inter-domain Routing – similar to Pastry – is based on prefix routing and aggregation. However, two major differences between DHT and Internet routing remain evident:

1. Keys, or IP addresses need not be active and in particular there is no key ownership or a key root.

2. The IPv4 address space is mainly unstructured, while there are some attempts to establish a structure of prefix aggregation in IPv6.

Thus, an immediate, unaltered transfer of the BIDIR-SAM prefix-directed multicast distribution is not applicable. Fortunately, there are strong correspondences, which in particular hold for IPv6.

Internet routers perform a prefix aggregation inherently, i.e., an upstream router will always know about the aggregation level it serves towards its downstream peers (for a certain address space). Accordingly, a multicast join (or leave) received from a downstream interface can be agglomerated with parallel joins to be tied to the prefix level in operation. Consequently, any router in the Internet will be able to identify its own prefix aggregation level as well as its immediate prefix neighbors. Assuming a clear, hierarchical address structure as foreseen in RFC 2374 [45], group management could proceed as in the BIDIR-SAM scheme. Unfortunately RFC 2374 has been obsoleted due to the persistent customer demand for provider independent addresses.[1] To cope with unstructured address organization, a router receiving a join for a specific multicast group needs to

- memorize the group and listener prefix received in a multicast forwarding table;
- identify the corresponding prefix level (according to the joining listener) it operates on;
- flood the join message under prefix aggregation to its remaining (downstream and upstream) interfaces, if the corresponding prefix has not been signaled before;
- memorize the join flooding state in a per interface table.

Subsequent joins need only to be communicated up to a level of aggregation, where they are uncovered by previously joining group members. In this way, a shared prefix tree will be erected throughout the Internet analogously to BIDIR-SAM.

It may be however undesirable to initiate an Internet-wide distribution tree for any multicast address. IPv6 therefore has foreseen scoping rules [44], but these only apply to local regions. To enable arbitrarily restricted prefix ranges of valid multicast dissemination, one could proceed as follows: On creation of the group, a uniformly covering prefix[2] could be chosen and embedded into the multicast group address in analogy to a rendezvous point address [85]. This prefix would then be interpreted as the root of the prefix tree, preventing state distribution beyond the region of interest for the group in common.

Multicast routing thereafter can proceed like in BIDIR-SAM on a bidirectional shared tree. Destination prefixes, which may be multiply required in topologies not following the prefix structure, are to be stored in a dedicate routing header. Routers will be equipped with a shared virtual multicast forwarding table in prefix space, which is downward directed and likewise allows for a forward route selection. However, upward routing must be separately foreseen as

[1] Nevertheless, the IPv6 address space remains structured according to RIRs and the RIRs itself leverage a structured address dissemination. Furthermore, address indirection approaches like LISP [36] try to regain provider-bound address hierarchies by splitting locators and identifiers.

[2] The prefix 2001:0638::/29 could for example be chosen to restrict a group to the DFN community.

follows: Any router receiving multicast data from a downlink with respect to the group tree root as encoded in the multicast address, will forward the traffic not only downwards according to its multicast forwarding tables, but also direct it in upward direction towards the embedded prefix, which forms the tree root.

Following this scheme, multicast distribution is bi-directionally flooded along a prefix tree spanning all receivers. Within a cleanly aggregated address space, such prefix routing will be strictly bound to provider borders and — if required — will cross peering links exactly once. All fundamental properties, especially redundancy and mobility options, are inherited from BIDIR-SAM.

Having sketched such solution, it appears promising and challenging at once to elaborate on the details of such a protocol, which attempts to combine traditional IP-layer and structured overlay routing techniques. Future work will unveil the feasibility of such out-of-the-box thinking.

Bibliography

[1] 3rd Generation Partnership Project, "Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2, 3GPP TS 23.228, Rel. 5 ff.,," 3rd Generation Partnership Project, Tech. Rep., 2002 – 2007.

[2] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*. New York: Dover Publications, 1964.

[3] A. Adams, J. Nicholas, and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)," IETF, RFC 3973, January 2005.

[4] C. Adjih, L. Georgiadis, P. Jacquet, and W. Szpankowski, "Multicast Tree Structure and the Power Law," *IEEE Transact. on Information Theory*, vol. 52, no. 4, pp. 1508–1521, 2006.

[5] L. Aguilar, "Datagram Routing for Internet Multicasting," in *Proceedings of SIGCOMM '84*. New York, NY, USA: ACM Press, 1984, pp. 58–63.

[6] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[7] "Apfloat – High Performance Arbitrary Precision Arithmetic," http://www.apfloat.org/, 2008.

[8] S. A. Baset, H. Schulzrinne, and M. Matuszewski, "Peer-to-Peer Protocol (P2PP)," P2PSIP Working Group, IETF Internet Draft – work in progress 01, November 2007.

[9] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proceedings of the 10th IEEE Global Internet Symposium*, M. Faloutsos *et al.*, Eds. Washington, DC, USA: IEEE Computer Society, 2007, pp. 79–84.

[10] I. Baumgart, B. Heep, S. Krause, and S. Mies, "The OverSim P2P Simulator," http://www.oversim.org, 2008.

[11] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)," IETF, RFC 3569, July 2003.

[12] E. W. Biersack, "Where is Multicast Today?" *Computer Communication Review*, vol. 35, no. 5, pp. 83–84, 2005.

[13] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu *et al.*, "Bimodal Multicast," *ACM Trans. Comput. Syst.*, vol. 17, no. 2, pp. 41–88, 1999.

[14] S. Birrer and F. E. Bustamante, "The feasibility of dht-based streaming multicast," in *MASCOTS '05: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 288–298.

[15] A. Brown and M. Kolberg, "Tools for Peer-to-Peer Network Simulations," P2PRG, IRTF Internet Draft – work in progress 0, January 2006.

[16] J. Buford, "Hybrid Overlay Multicast Framework," SAM RG, IRTF Internet Draft – work in progress 2, February 2008.

[17] J. Buford, "SAM Overlay Protocol," SAM RG, IRTF Internet Draft – work in progress 1, March 2008.

[18] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual Ring Routing: Network Routing Inspired by DHTs," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 351–362, 2006, proceedings of the SIGCOMM conference 2006.

[19] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," IETF, RFC 3376, October 2002.

[20] M. Castro, M. Costa, and A. Rowstron, "Debunking some myths about structured and unstructured overlays," in *NSDI'05: Proceedings of the 2nd Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005, pp. 85–98.

[21] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. I. T. Rowstron *et al.*, "SplitStream: High-Bandwidth Content Distribution in Cooperative Environments," in *Peer-to-Peer Systems II. Second International Workshop, IPTPS 2003 Berkeley, CA, USA, February 21-22, 2003 Revised Papers*, ser. LNCS, M. F. Kaashoek and I. Stoica, Eds., vol. 2735. Berlin Heidelberg: Springer–Verlag, 2003, pp. 292–303.

[22] 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.

[23] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer *et al.*, "An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer Overlays," in *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*, vol. 2. Washington, DC, USA: IEEE Computer Society, 2003, pp. 1510–1520.

[24] R. C. Chalmers and K. C. Almeroth, "On the topology of multicast trees," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 153–165, 2003.

[25] M. Christensen, K. Kimball, and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches," IETF, RFC 4541, May 2006.

[26] J. C. I. Chuang and M. A. Sirbu, "Pricing Multicast Communication: A Cost-Based Approach," *Telecommunication Systems*, vol. 17, no. 3, pp. 281–297, 2001, presented at the INET'98, Geneva, Switzerland, July 1998.

[27] F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers*, ser. LNCS, M. F. Kaashoek and I. Stoica, Eds., vol. 2735. Berlin Heidelberg: Springer–Verlag, 2003, pp. 33–44.

[28] "daviko – digital audio video communication," http://www.daviko.com, 2008.

[29] S. E. Deering, "Host Extensions for IP Multicasting," IETF, RFC 1112, Aug. 1989.

[30] S. E. Deering, "Multicast Routing in a Datagram Internetwork," Ph.D. dissertation, Stanford University, Stanford, CA, USA, December 1991.

[31] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network Magazine*, vol. 14, no. 1, pp. 78–88, 2000.

[32] P. Druschel *et al.*, "FreePastry," http://freepastry.rice.edu/FreePastry/, 2008.

[33] S. El-Ansary, L. A. Alima, P. Brand, and S. Haridi, "Efficient Broadcast in Structured P2P Networks," in *Peer–to–Peer Systems II. Second International Workshop, IPTPS 2003 Berkeley, CA, USA, February 21-22, 2003 Revised Papers*, ser. LNCS, F. Kaashoek and I. Stoica, Eds. Berlin Heidelberg: Springer–Verlag, 2003, vol. 2735, pp. 304–314.

[34] ETSI EN 302 304, "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)," ETSI, European Standard (Telecommunications series), November 2004.

[35] S. Fahmy and M. Kwon, "Characterizing Overlay Multicast Networks and Their Costs," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 373–386, 2007.

[36] D. Farinacci, V. Fuller, D. Oran, and D. Meyer, "Locator/ID Separation Protocol (LISP)," IETF, Internet Draft – work in progress 07, April 2008.

[37] A. Feldmann, "Internet Clean-Slate Design: What and Why?" *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 59–64, 2007.

[38] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")," IETF, RFC 4605, August 2006.

[39] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," IETF, RFC 4601, August 2006.

[40] A. Garyfalos and K. Almeroth, "A Flexible Overlay Architecture for Mobile IPv6 Multicast," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 11, pp. 2194–2205, November 2005.

[41] A. Ghodsi, L. O. Alima, S. El-Ansary, A. Ghodsi, P. Brand *et al.*, "Self-Correcting Broadcast in Distributed Hash Tables," in *Proceedings of 15th International Conference Parallel and Distributed Computing and Systems (PDCS'03)*, T. Gonzalez, Ed. Calgary, AB, Canada: ACTA Press, 2003.

[42] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)," IETF, RFC 5015, October 2007.

[43] M. Handley, I. Kouvelas, T. Speakman, and L. Vicisano, "Bi–directional Protocol Independent Multicast (BIDIR-PIM)," IETF, Internet Draft – work in progress 09, February 2007.

[44] R. M. Hinden and S. E. Deering, "IP Version 6 Addressing Architecture," IETF, RFC 4291, February 2006.

[45] R. M. Hinden, M. O'Dell, and S. E. Deering, "An IPv6 Aggregatable Global Unicast Address Format," IETF, RFC 2374, July 1998.

[46] H. Holbrook and B. Cain, "Source-Specific Multicast for IP," IETF, RFC 4607, August 2006.

[47] H. Holbrook, B. Cain, and B. Haberman, "Using IGMPv3 and MLDv2 for Source-Specific Multicast," IETF, RFC 4604, August 2006.

[48] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A Survey of Application-Layer Multicast Protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.

[49] IEEE 802.16 Working Group, "802.16e-2005 and IEEE Std 802.16-2004/Cor1-2005: Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1," IEEE, New York, IEEE standard, February 2006.

[50] T. Inoue and R. Kurebayashi, "Evaluating the Impact of Tunneling on Multicast Efficiency," in *2005 Asia-Pacific Conference on Communications.* Piscataway, NJ, USA: IEEE Press, October 2005, pp. 254–258.

[51] M. Janic and P. Van Mieghem, "On properties of multicast routing trees," *Int. J. Commun. Syst.*, vol. 19, no. 1, pp. 95–114, 2006.

[52] D. B. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," IETF, RFC 3775, June 2004.

[53] K. Katrinis and M. May, "Application–Layer Multicast," in *Peer–to–Peer Systems and Applications*, ser. LNCS, R. Steinmetz and K. Wehrle, Eds. Berlin Heidelberg: Springer–Verlag, 2005, vol. 3485, ch. 11, pp. 157–170.

[54] S. Kaune, A. Kovacevic, K. Graffi, and K. Pussep, "PeerfactSim.KOM – Peer-to-Peer Simulator," http://www.peerfact.org/, 2008.

[55] kc claffy *et al.*, "CAIDA," http://www.caida.org, 2008.

[56] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh, "Probabilistic Reliable Dissemination in Large-Scale Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 3, pp. 248–258, 2003.

[57] J. Li, K. Sollins, and D.-Y. Lim, "Implementing Aggregation and Broadcast over Distributed Hash Tables," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 1, pp. 81–92, 2005.

[58] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[59] W. Li, S. Chen, P. Zhou, X. Li, and Y. Li, "An Efficient Broadcast Algorithm in Distributed Hash Table Under Churn," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCom'07)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1929–1932.

[60] B. Lim and K. Ettikan, "ALM API for Topology Management and Network Layer Transparent Multimedia Transport," individual, IRTF Internet Draft – work in progress 0, January 2008.

[61] M.-J. Lin and K. Marzullo, "Directional Gossip: Gossip in a Wide Area Network," in *EDCC-3: Proceedings of the Third European Dependable Computing Conference on Dependable Computing*, ser. Lecture Notes in Computer Science, vol. 1667. London, UK: Springer-Verlag, 1999, pp. 364–379.

[62] P. G. López, C. Pairot, R. Mondéjar, J. P. Ahulló, H. Tejedor *et al.*, "PlanetSim: A New Overlay Network Simulation Framework," in *Proceedings 4th International Workshop on Software Engineering and Middleware (SEM 2004). Revised Selected Papers*, ser. LNCS, T. Gschwind and C. Mascolo, Eds., vol. 3437. Berlin Heidelberg: Springer–Verlag, 2005, pp. 123–136.

[63] P. Mahadevan, C. Hubble, D. Krioukov, B. Huffaker, and A. Vahdat, "Orbis: Rescaling Degree Correlations to Generate Annotated Internet Topologies," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '07)*. New York, NY, USA: ACM, 2007, pp. 325–336.

[64] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "Systematic topology analysis and generation using degree correlations," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '06)*. New York, NY, USA: ACM, 2006, pp. 135–146.

[65] P. Merz and K. Gorunova, "Efficient broadcast in P2P Grids," in *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)*, vol. 1. Washington, DC, USA: IEEE Computer Society, 2005, pp. 237–242.

[66] N. Mimura, K. Nakauchi, H. Morikawa, and T. Aoyama, "RelayCast: A Middleware for Application-level Multicast Services," in *Proceedings of the 3st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '03)*, S. Matsuoka and Y. Ishikawa, Eds. Washington, DC, USA: IEEE Computer Society, 2003, pp. 434–441.

[67] T. Munzner, E. Hoffman, K. Claffy, and B. Fenner, "Visualizing the Global Topology of the MBone," in *Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 85–92.

[68] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman *et al.*, "The State of Peer-to-Peer Simulators and Simulations," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 95–98, 2007.

[69] "OriginLab – Data Analysis and Graphing Software," http://www.originlab.com/, 2008.

[70] V. Paxson, "End-to-End Routing Behavior in the Internet," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 601–615, 1997.

[71] G. Phillips, S. Shenker, and H. Tangmunarunkit, "Scaling of multicast trees: comments on the chuang-sirbu scaling law," in *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*. New York, NY, USA: ACM Press, 1999, pp. 41–51.

[72] C. Plaxton, R. Rajaraman, and A. Richa, "Accessing Nearby Copies of Replicated Objects in a Distributed Environment," in *Proc. of 9th ACM Sympos. on parallel Algor. and Arch. (SPAA)*. New York, NY, USA: ACM Press, June 1997, pp. 311–330.

[73] H. Pucha, S. M. Das, and Y. C. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks," in *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*. Washington, DC, USA: IEEE Computer Society, December 2004, pp. 163–173.

[74] T. Pusateri, "Distance Vector Multicast Routing Protocol," IETF, Internet Draft – work in progress (expired) 11, October 2003.

[75] Y. Qiao and F. E. Bustamante, "Structured and unstructured overlays under the microscope: A measurement-based view of two P2P systems that people use," in *Annual Tech '06: Proceedings of the ANnual Technical Conference on USENIX*. Berkeley, CA, USA: USENIX Association, 2006, pp. 341–355.

[76] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting IP Multicast," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '06)*. New York, NY, USA: ACM Press, 2006, pp. 15–26.

[77] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2001, pp. 161–172.

[78] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," in *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings*, ser. LNCS, J. Crowcroft and M. Hofmann, Eds., vol. 2233. London, UK: Springer–Verlag, 2001, pp. 14–29.

[79] S. P. Ratnasamy, "A Scalable Content-Addressable Network," Ph.D. dissertation, University of California, Berkeley, October 2002.

[80] S. Rhea, B.-G. Chun, J. Kubiatowicz, and S. Shenker, "Fixing the Embarrassing Slowness of OpenDHT on PlanetLab," in *Proceedings of the 2nd conference on Real, Large Distributed Systems (WORLDS'05)*, vol. 2. Berkeley, CA, USA: USENIX Association, 2005, pp. 1–6.

[81] I. Romdhani, M. Kellil, H.-Y. Lach, A. Bouabdallah, and H. Bettahar, "IP Mobile Multicast: Challenges and Solutions," *IEEE Comm. Surveys & Tutorials*, vol. 6, no. 1, pp. 18–41, 2004.

[82] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001, pp. 329–350.

[83] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The Design of a Large-Scale and Event Notification Infrastructure," in *Networked Group Communication. Third International COST264 Workshop, NGC 2001. Proceedings*, ser. LNCS, J. Crowcroft and M. Hofmann, Eds., vol. 2233. Berlin Heidelberg: Springer–Verlag, 2001, pp. 30–43.

[84] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End Arguments in System Design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, Nov. 1984.

[85] P. Savola and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address," IETF, RFC 3956, November 2004.

[86] T. C. Schmidt and M. Wählisch, "Predictive versus Reactive — Analysis of Handover Performance and Its Implications on IPv6 and Multicast Mobility," *Telecommunication Systems*, vol. 30, no. 1/2/3, pp. 123–142, November 2005.

[87] T. C. Schmidt and M. Wählisch, "Morphing Distribution Trees – On the Evolution of Multicast States under Mobility and an Adaptive Routing Scheme for Mobile SSM Sources," *Telecommunication Systems*, vol. 33, no. 1–3, pp. 131–154, December 2006. [Online]. Available: http://dx.doi.org/10.1007/s11235-006-9010-4

[88] T. C. Schmidt and M. Wählisch, "Multicast Mobility in MIPv6: Problem Statement and Brief Survey," MobOpts, IRTF Internet Draft – work in progress 03, February 2008, previous track: draft-schmidt-mobopts-mmcastv6-ps-02.txt. [Online]. Available: http://tools.ietf.org/html/draft-irtf-mobopts-mmcastv6-ps

[89] Y. Shavitt *et al.*, "The DIMES project," http://www.netdimes.org, 2008.

[90] K. Shudo *et al.*, "Overlay Weaver: An Overlay Construction Toolkit," http://overlayweaver.sourceforge.net/, 2008.

[91] K. Shudo, Y. Tanaka, and S. Sekiguchi, "Overlay Weaver: An Overlay Construction Toolkit," *Computer Communications*, vol. 31, no. 2, pp. 402–412, 2008, special issue on foundations of peer-to-peer computing.

[92] R. Steinmetz and K. Wehrle, Eds., *Peer-to-Peer Systems and Applications*, ser. LNCS. Berlin Heidelberg: Springer-Verlag, 2005, vol. 3485.

[93] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2001, pp. 149–160.

[94] D. Thaler, M. Talwar, A. Aggarwal, L. Vicisano, and T. Pusateri, "Automatic IP Multicast Without Explicit Tunnels (AMT)," IETF, Internet Draft – work in progress 9, June 2008.

[95] D. Thaler, "Border Gateway Multicast Protocol (BGMP): Protocol Specification," IETF, RFC 3913, September 2004.

[96] "Valgrind," http://www.valgrind.org/, 2008.

[97] P. Van Mieghem, *Performance Analysis of Communications Networks and Systems*. Cambridge, New York: Cambridge University Press, 2006.

[98] P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad, "On the Efficiency of Multicast," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 719–732, 2001.

[99] A. Varga *et al.*, "The OMNeT++ discrete event simulation system," http://www.omnetpp.org, 2008.

[100] R. Vida and L. H. M. K. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6," IETF, RFC 3810, June 2004.

[101] S. Voulgaris and M. van Steen, "Hybrid Dissemination: Adding Determinism to Probabilistic Multicasting in Large-Scale P2P Systems," in *Middleware 2007*, ser. Lecture Notes in Computer Science, R. Cerqueira and R. Campell, Eds., vol. 4834. Berlin Heidelberg: Springer-Verlag, November 2007, pp. 389–409.

[102] M. Wählisch and T. C. Schmidt, "Between Underlay and Overlay: On Deployable, Efficient, Mobility-agnostic Group Communication Services," *Internet Research*, vol. 17, no. 5, pp. 519–534, 2007, selected papers from the TERENA networking conference 2007.

[103] M. Wählisch and T. C. Schmidt, "Exploring the Routing Complexity of Mobile Multicast – A Semi-empirical Study," in *Proceedings of 2007 ACM CoNEXT Conference. Student Workshop*, S. Banerjee, R. Karrer, and A. Sridharan, Eds. New York: ACM, December 2007, extended abstract. [Online]. Available: http://www.sigcomm.org/co-next2007/papers

[104] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol," IETF, RFC 1075, November 1998.

[105] H. Yu and J. Buford, "Advanced Topics in Peer-to-Peer Overlay Multicast," in *Encyclopedia of Wireless and Mobile Communications*, B. Fuhrt, Ed. Boston, MA, USA: CRC Press, 2008.

[106] T. Zahn and J. Schiller, "MADPastry: A DHT Substrate for Practicably Sized MANETs," in *Proc. of 5th Workshop on Applications and Services in Wireless Networks (ASWN 2005)*, Paris, France, June 2005.

[107] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph *et al.*, "Tapestry: A Resilient Global-Scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, January 2004.

[108] 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.

[109] 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 *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '01)*, J. Nieh and H. Schulzrinne, Eds. New York, NY, USA: ACM, 2001, pp. 11–20.