Reducing Memory Requirements for Routing Protocols

Elliott M. Karpilovsky

A Dissertation
Presented to the Faculty
Of Princeton University
In Candidacy for the Degree
Of Doctor of Philosophy

Recommended for Acceptance
By the Department of
Computer Science
Advisor: Jennifer Rexford

September 2009

Abstract

Due to the rapid growth of the Internet, routers require increasing amounts of memory to forward packets to their destinations. Lack of sufficient memory can cause routers to crash, reject new routing information, or enter into an indeterminate state, as current protocols and systems do not deal gracefully with memory exhaustion. Current “solutions” are often ineffective. Techniques such as over-provisioning memory are expensive, since networks may have thousands of routers needing upgrades; moreover, over-provisioning may be infeasible in cases where the routers are not physically accessible (*e.g.*, routers on satellites). Other methods, such as restricting the set of routes that can be learned, are impractical; such restrictions must be loose to allow sufficient connectivity to other domains (which may be needed when primary routes fail and backup routes are required). Moreover, routers typically have two different memory structures that are at risk of overflow: a Routing Information Base (RIB) for tracking available paths, and a Forwarding Information Base (FIB) for moving packets toward their destinations.

Our study measures the deployment and characteristics of routing protocols and proposes new mechanisms to curtail associated burden. We design two different ways to reduce the two different router memory requirements, and additionally investigate how router memory requirements might change in the future, due to emerging protocols. First, we examine how to reduce RIB table size in our work known as Forgetful Routing. We explore a space-time trade-off between memory needed to store the RIB and time needed to compute a forwarding path from among all possible paths. We demonstrate how RIB memory can potentially be reduced by a factor of 3 or more. Second, we reduce the size of the FIB with a Memory Management System (MMS). The MMS can modify routing behavior such that memory is saved without introducing aberrant behavior (*i.e.*, without routing loops). The Memory Management System allows operators to adjust forwarding behavior in a controlled manner.

The system can also operate in a transparent mode, where no forwarding behavior is changed but memory is still reduced. The MMS can reduce memory usage up to a factor of 3. Finally, we examine the growth trends of emerging protocols such as IPv6 and multicast; IPv6 is being deployed to solve the address shortage problem, and multicast's one-to-many model has become a popular means to distribute content to many receivers, such as with IPTV. By measuring these protocols, we better understand where they are headed, and thus better understand future routing requirements.

Acknowledgements

My advisor, Jennifer Rexford, played a key role during my time at Princeton University. She guided me during some research projects and directed me in others. Overall, she significantly influenced my work. Jen also provided post-graduate career help.

Others who contributed to the thesis research include: Lee Breslau, Aman Shaikh, Dan Pei, Kobus van der Merwe, Shubho Sen, Alexandre Gerber, and Matt Caesar. No matter the sizes, their contributions played a role in the formation of my thesis.

Among the aforementioned, the spotlight shines on Lee Breslau, Aman Shaikh, and Dan Pei. Their help, contributions, and efforts were appreciated. I extend my thanks to them.

This thesis might look a bit different if not for Nicholas A. Piro, *a.k.a.* Dr. Piro. I am grateful for his suggestion to use Adobe Illustrator to properly format the dissertation's graphs. He also provided a few useful LaTeX commands that were appreciated.

I thank Rebecca Flint for proofreading a part of the thesis, and I thank other people for also providing feedback.

I thank all the pets that I have had the pleasure of knowing over all these years. They have brought me much happiness, and I hope that I have been able to reciprocate the same joy. I thank Woolly, Twichy, Tiger, and Ratty. Even though I was too young to remember possible interactions with them, I also thank Tsar and Prince. I am deeply sorry if any names are misspelled.

I thank my Mom for helping me with the spelling of pet names and for telling me about Prince. There was serious effort to try to spell their names correctly. These thanks are independent from the ones that are possibly generated by the next paragraph.

Acknowledging family and friends is a tough matter. Who do I acknowledge? What kind of acknowledgements do I give? As such, this space is reserved for the future. If I decide that someone should have a specific acknowledgement or an extra acknowledgement, it should be considered as if it was written here.

For those that I have forgotten but would acknowledge if I remembered, I reserve this space to hold their appropriate acknowledgement.

I thank Square-Enix for providing me with much entertainment during this time and other instances.

I extend gratitude to the funding agencies. I received assistance through NSF grant CNS-0519885, HSARPA/ARO (Grant W911NF-05-1-0417), HSARPA grant 1756303, and a URP grant from Cisco Systems. I am grateful for the support.

Finally, I thank you, the reader. This thesis represents the culmination of my research at Princeton University. I wish I could have spent more time revising it, but there were deadlines. Although you may find imperfections, I hope that you nonetheless derive satisfaction from the dissertation.

Contents

Chapter 1

Prologue

Without a doubt, the Internet is one of the defining inventions of the 20th century. By providing a common mechanism for machines in completely different parts of the world (or even outer space) to communicate with each other, the Internet has revolutionized the way computers interact with each other. However, the popularity of the Internet has led to the concern that it could, figuratively speaking, collapse under its own weight. In order to maintain its rapid growth, "better" routers are needed, where better often means faster, with more memory.

The memory resource problem is surprisingly complex. For example, a doubling in the number of reachable Internet addresses does not necessarily result in a doubling of the amount of memory needed. This phenomenon is due to the fact that as the number of machines on the Internet grows, there are associated increases in both the number of addresses *and* the number of paths that can transit traffic between end-points. Both impact router memory growth.

Understanding how these factors affect memory resources can be difficult. When viewing the Internet as a "cloud", where machines plug in (or connect wirelessly) and somehow manage to successfully send data back and forth, the aforementioned issues are invisible. However, the resource problem becomes more apparent after looking

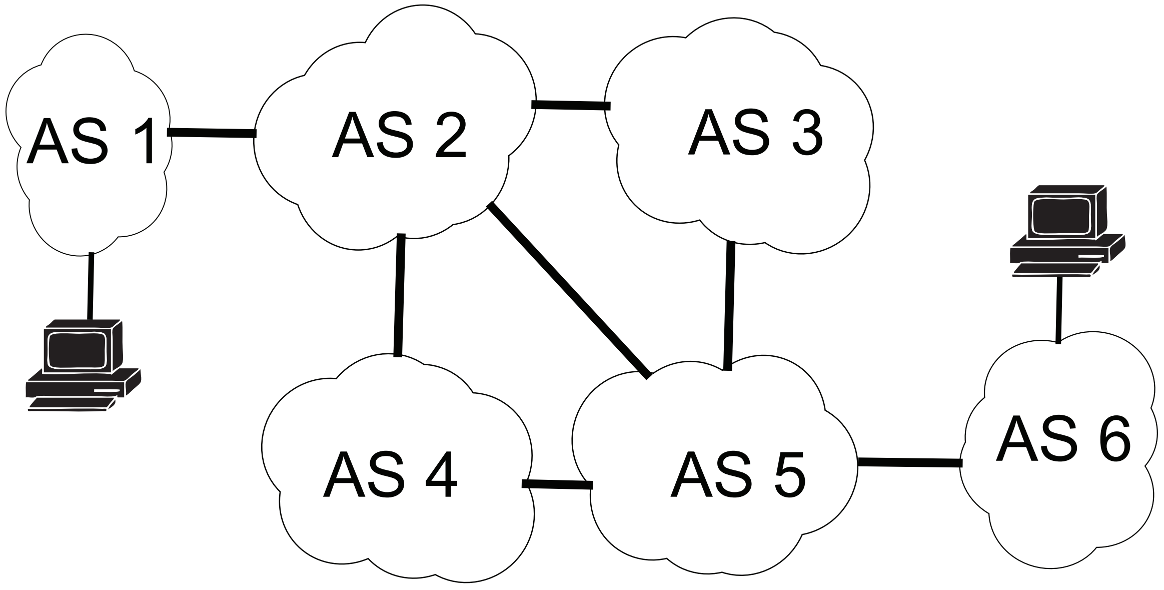


Figure 1.1: An example of how ASes interconnect. The two computers can potentially communicate over the AS paths (1, 2, 3, 5, 6), (1, 2, 4, 5, 6), or (1, 2, 5, 6). However, business agreements or other external factors may cause an AS to prefer one neighbor over another (or even to ignore a neighbor altogether) during routing.

at the Internet as a collection of Autonomous Systems (ASes). Each AS represents a network operated by a single entity, such as an Internet service provider (ISP) or university (hence why these clouds are called autonomous). The way in which these ASes interconnect determine how data is sent across the Internet, as shown in Figure 1.1. When an AS receives packets of data, it determines if the destination is located on its network. If so, it routes the traffic internally in its network. Otherwise, it routes the traffic to an appropriate neighboring AS; the neighboring AS repeats the same steps until the traffic finally reaches its destination.

The resource problem exists for two primary reasons. First, as more and more machines connect to the Internet, each AS must remember the locations of all these machines (otherwise, it will not know where to send the information). Second, as increasing numbers of devices connect to the Internet, more ASes may arise to manage them. Each AS may need to further interconnect to ensure it has a wide diversity of paths for each destination. Thus, more information about the different paths may need to be maintained.

A *router* level view of the Internet better illustrates the specifics of the problem. Examining how an individual router tracks reachability to destinations, as well as maintains backup routes in the event of a networking failure, reveals the precise nature of the resource exhaustion problem. Moreover, understanding the problem in detail is necessary to understand how it can be mitigated.

1.1 A Router Level View of the Internet

Routers are the fundamental building blocks of today's Internet. They are specialized machines with multiple incoming and outgoing interfaces (usually provided by line cards that plug into the system), where each interface is connected to another machine's interface via a link (usually a cable). Figure 1.2 shows the design of a typical high-end router. Routers aggregate addresses into sets known as *prefixes*, which are primitives used both internally and for communicating with other machines. Each router has three primary methods of interacting with the world around it (also known as the *planes* that a router operates within): talking to other routers (the *control plane*), receiving configuration information from network operators (the *management plane*), and forwarding packets (the *data plane*).

As previously mentioned, routers operate over prefixes (rather than individual addresses) when storing and transmitting routing information. A prefix is a compact representation of a set of addresses. For example, in the IPv4 protocol, the prefix 1.2.0.0/16 represents all IPv4 addresses whose first 16 bits start with 1.2. The reason for using prefixes (over individual addresses) is done for practical reasons. ISPs and other organizations are assigned contiguous blocks from Regional Internet Registries (RIRs). These registries collectively own the IP address space and are responsible for delegating it to others. Without some form of aggregation, a router would need memory that is proportional in size to all addresses in use. Billions of routing table entries would be needed as of today, which would make routing infeasible, given current technology. However, with prefixes, the number of entries is reduced by orders of magnitude. However, prefixes suffer the drawback that they complicate forwarding decisions, as there may be two prefixes that overlap but use different outgoing interfaces. For example, the prefix 1.0.0.0/8 may forward traffic to one interface, and 1.2.0.0/16 may forward traffic to a different interface. In the event of ambiguities, the *more-specific prefix* (that is, the prefix that specifies more bits) is given preference.

To disseminate information about prefixes, a router talks to its neighbors via control messages that it sends through the control plane. These messages come in across the line cards and are sent to the processor; such messages usually describe the discovery of new routing paths (and various characteristics of those paths, such as the lists of ASes they traverse) and the removal of old paths. These changes are stored in a data structure known as the *Routing Information Base* (RIB) in main memory.

In addition to control messages, operators may interact with the routers (either remotely or through a local terminal) and give them special instructions on how to operate. Such instructions are also sent to the processor and, depending on the specific router implementation, possibly effect changes in the RIB. At this point, the router has all necessary information to actually forward packets.

The router will then contact each line card and provide it with specific forwarding information, used for the forwarding plane. From the control and management planes, the RIB now contains a list of possible routes for each Internet destination; a single route per destination among all possibilities is chosen and installed in the FIB. When packets now come into the router that are not control or management

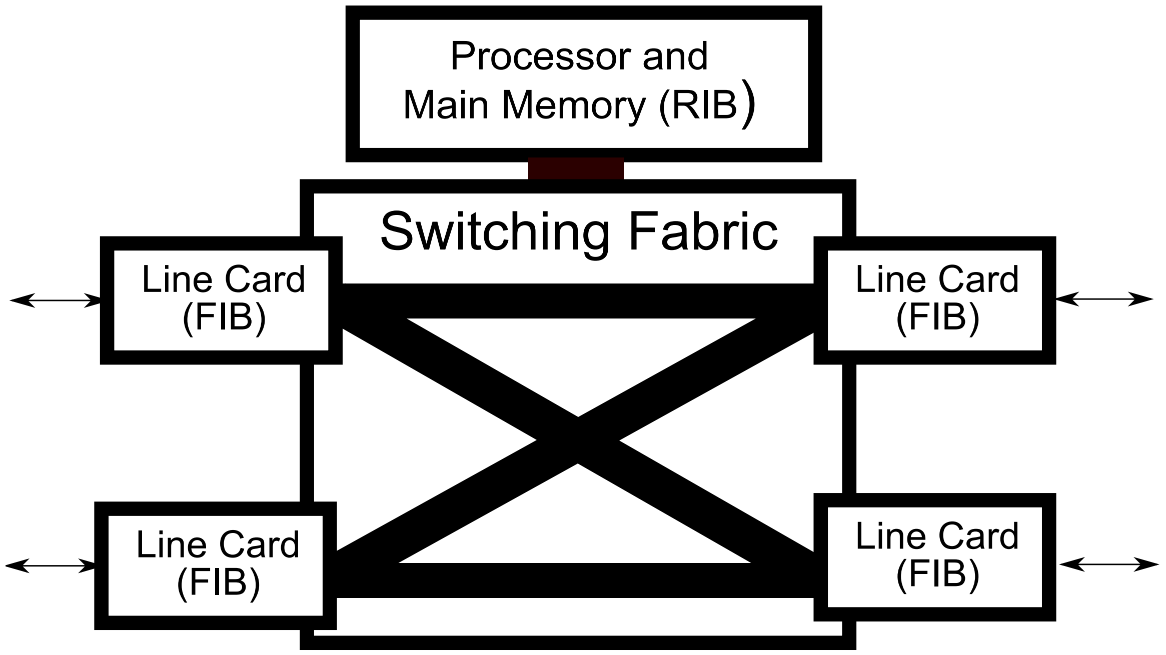


Figure 1.2: Modern routers have three main components: a processing unit, a switching fabric, and a series of line cards.

packets, the associated line card will perform a lookup and determine the appropriate outgoing interface. Packets are then moved across the switching fabric from ingresses to egresses.

The RIB and the FIB are crucial components of any router. The RIB determines how a router sends traffic to a destination, and the FIB performs the actual forwarding. Because the RIB and the FIB have different goals, they store different information and have different requirements. For example, for a set of destinations, the RIB will store all known possible paths to it. Information about each path, known as attributes, is also kept. These attributes can be used for a multitude of purposes, such as approximating latency, load balancing traffic across a network, or determining which routes will generate the most revenue. As such, the RIB needs sufficient memory to capture 1.) all the destinations, 2.) all the routes to each destination, and 3.) all the attributes for each route. The FIB, on the other hand, matches IP addresses to outgoing links. As such, it does not care about path attributes or alternate paths

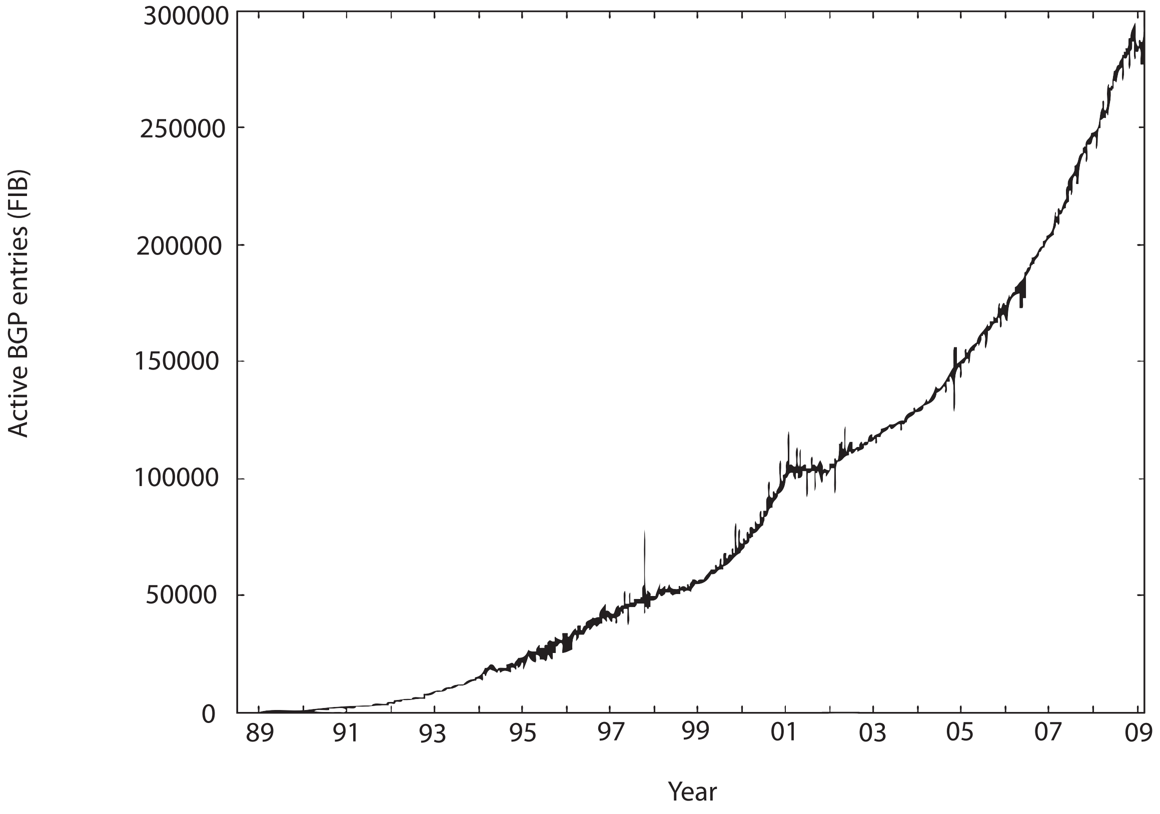


Figure 1.3: Growth of the FIB. "BGP entries" refers to the number of prefixes in use. Graph taken from the CIDR report [84].

to a destination – that is the RIB's responsibility. However, once the RIB informs it of a routing decision, the FIB must be able to perform this match at very high speeds (in order to keep up with the vast amount of traffic that ISPs see). Thus, the FIB needs memory that is 1.) extremely fast, and 2.) large enough to associate all destinations with outgoing links.

1.2 Memory Exhaustion: The Irony of Success

The widespread success of the Internet has led to an explosion in terms of the number of prefixes that can be reached and to the number of ways each prefix can be reached. As such, memory requirements for both the RIB and the FIB have skyrocketed. For example, Figure 1.3 shows increases in a FIB data structure, which is due to prefix growth (note that prefix growth also affects the RIB). From 1989 to 1993, the Internet was still nascent, but signs of exponential growth were present. Fearing a shortage of addresses, a new addressing scheme known as CIDR [70] was introduced in an attempt to more efficiently aggregate addresses into prefixes. CIDR proved to only be temporarily effective, reducing growth to linear rates between 1993 and 1998; soon afterward, rapid growth resumed. Other than a brief hiatus between 2001 and 2002 (probably due to fallout from the dot-com burst), the number of different prefixes that are reachable has been growing at an approximately exponential rate. Possible reasons for such an increase include address fragmentation (caused by load balancing and customers moving to different ISPs while retaining their old IP addresses) and increases in the total amount of IP address space used (as more and more devices connect to the Internet). Note that the RIB suffers from additional growth factors separate from the FIB, such as multi-homing (where a network connects to multiple ISPs to improve uptime) [43], which can increase the number of alternate paths.

The RIB and FIB are essential for the proper functioning of a router; if memory is exhausted, these data structures (and the router) will not be able to operate correctly. A previous study has shown that during such duress, a router may engage in a number of bad activities, such as crashing, rejecting new routes, or continuously closing and opening connections with neighboring routes [48]. Moreover, memory usage often spikes during events known as "route leaks." A route leak is an attack (either accidental or intentional) on neighboring routers that can completely fill up memory and cause memory exhaustion. Such route leaks have been responsible for several major Internet outages, including the massive outage on April 25th, 1997 (known as the AS7007 event [39]) and the WorldCom outage of October 3rd, 2002 [123].

Finally, the growth of new protocols, such as IPv6 and multicast, only exacerbate the problem. Such protocols run on top of routers and are currently offered as services to customers. As of today, their usage is little, and the associated memory they use is small. However, it is speculated that these protocols may soon become widespread, changing the landscape of routing and driving memory requirements even higher.

1.3 Current "Solutions" are Not Enough

There have been many solutions proposed to combat memory inflation, but none have proven to be panaceas. Such propositions generally fall into two categories: changing router configuration, or changing routing protocols.

Router configuration changes include all things that an operator can do to a router. Such solutions often assume that today's routing protocols and routers are, for the most part, fixed and cannot be easily changed. As such, these solutions are very pragmatic in nature, asking, "what can we do with networks as they exist today?" These solutions have the benefit of being immediately deployable and usable. However, they also provide the least amount of benefit; they typically do not significantly reduce memory consumption, providing few safeguards against the possibility of memory overflow. These solutions range from simply upgrading more memory, to configuring routers to limit the number of routes they accept, to physically altering a network to prevent total route propagation.

In each instance, problems such as cost, feasibility, and manageability have prevented these solutions from becoming successful. For example:

- *Memory upgrades* are often infeasible because 1.) the cost is considerable, considering that thousands of routers may exist in a network, and that router memory is much more expensive than commodity RAM, 2.) routers have physical and design limitations that prevent them from upgrading beyond a fixed amount of memory, and 3.) in the case of routers in outer space, such as routers on satellites, installing the memory may not be an option.

- *Route filters* can be installed to discard routes that do not meet certain criteria. In fact, the Regional Internet Registries (RIR) publish guidelines for the maximum prefix lengths for various parts of the IP address space [82]. However, these guidelines are merely suggestions, and many ISPs ignore them. This creates a catch-22, where ASes do not filter based on RIR guidelines since so many ASes violate them, leading ASes to feel little pressure to obey them. Additionally, a hard limit on the number of prefixes accepted from each neighbor can be imposed. However, unless extremely conservative limits are imposed, the router remains vulnerable to learning too many routes across all of its neighbors [115].

- *Route reflectors* can be deployed by operators [38]. These machines alter the basic structure of their networks to prevent route table growth, through their deployment at strategically located positions. Route reflectors act as internal accumulation points, which collect routing updates from a subset of border routers, and only advertise the most preferred route to a subset of their neighbors. Unfortunately, the use of route reflectors introduces a set of problems. They can induce persistent forwarding loops and oscillations if deployed improperly [37]. They require additional work for network operators to maintain, as they must be reconfigured to match changes in the underlying network topology. While route reflectors reduce memory usage, they do not reduce the number of prefixes in the routing table.

On the other end of the spectrum, the research community has investigated clean slate designs to reduce the total amount of memory needed by routers. In such designs, the constraints of today's real routing systems are ignored. These ideas focus on *protocol change*; that is, they seek to alter the underlying mechanisms in use on today's Internet. For example, architectures such as CRIO [137] and compact routing [131] introduce "controlled deflection" into the network architecture; routes in these systems typically experience minor path inflation (where the forwarding paths are not as short as they could be), but allow for significant memory savings under these conditions. Moreover, work such as the LISP architecture [62], which separates the notions of identity from location, could aggregate routes better than current methods today. However, while clean slate designs are quite nice and may serve useful in the future, they do not help with the problems of today. Service providers are reluctant to deploy such solutions because they typically require the cooperation of other organizations before there is benefit. Coupled with the cost of implementing such changes, ASes face a bootstrapping problem: until others deploy such changes, there is no incentive for an individual AS to modify its network.

In addition to these problems, there is growing concern about how emerging protocols, such as IPv6 and multicast, will change the routing landscape. Because these protocols are not yet widespread, operators are not even sure how they are currently being used. It is extremely difficult, if not impossible, to gauge potential impact with so little information known about them.

1.4 Contributions of the Thesis

Given the deficiencies of previous solutions, this thesis approaches these problems with a focus on *backwards compatibility.* A backwards compatible solution is one that will inter-operate with other networks, even if no other networks deploy the solution. As such, the inter-AS protocols are not changed. However, protocol *implementations* can be changed to perform the same tasks in more efficient ways. These solutions give extra emphasis to practicality. For example, applying a software upgrade to every router on the Internet is infeasible, but applying a software upgrade to every router in an Autonomous System can be feasible since, by definition, a single point of authority controls the network. The goal of this thesis is to answer the question, "what can a single AS do to reduce its memory consumption while remaining inter-operable with its neighbors?”

Given everything that has already been done, it is clear that to appropriately attack this problem, a two pronged approach must be used. First, for existing and widespread protocols, both backwards compatible and feasibly implementable solutions must be devised. Second, emerging protocols must be studied so that researchers can better understand how they are evolving and how they will impact routing in the future.

This thesis takes this two pronged approach. The focus is on solutions that operators can feasibly deploy on their networks. Rather than trying to change routing protocols, we examine the memory usage of such protocols and *change their implementation*. Our goal is to develop new algorithms to push today’s networks closer to their optimal operating efficiency. Additionally, we measure and characterize upcoming protocols. We study data traffic to understand how people are using these protocols. By doing so, we can better understand trends that we see and suggest possible mechanisms for containing their memory growth when they become widespread.

In the first part of the thesis, we explore how to reduce both the RIB and FIB while remaining interoperable with current protocols. We developed a scheme known as Forgetful Routing that allows routers to distributively share their RIBs, reducing redundant entries and significantly saving memory. We then describe a separate scheme called the Memory Management System (MMS) that saves FIB memory by coalescing prefixes that have the same forwarding address. The MMS can also be configured to allow suboptimal routes in exchange for greater memory reduction. These schemes, optimized for today’s routing protocols, can be used to fight the problem of memory exhaustion.

In the second part of the thesis, we quantify the IPv6 and multicast protocols, characterizing their usage patterns. These protocols are very important because they have the potential to significantly increase memory requirements on routers. However, because they have not been widely deployed and so little is known about them, it is difficult to try and develop methods for reducing their memory usage. In fact, because so little is known about them, it is necessary to characterize the behavior of these protocols, in order to allow such research to proceed. We study them through measurements taken from a tier 1 ISP and publicly available sources, and give our findings. As of the present time, we cannot draw significant conclusions about how they will change routing and how we can combat future memory growth – however, our study sheds light on their deployment and current usage, and (for multicast) suggests how some of its behavior could be optimized to consume less memory.

By addressing memory growth for routers through incrementally deployable solutions, we provide network operators with solutions to the problems they currently face. The RIB and the FIB are two primary router resources that can suffer from memory exhaustion, and our solutions are aimed at these data structures. Moreover, by investigating upcoming protocols, we can provide a starting point for future research concerning memory reduction. Our goals are to provide feasible, deployable solutions for the problems of today while keeping an eye to the problems of tomorrow.

Chapter 2

Using Forgetful Routing to Control RIB Table Size

The successful delivery of traffic through the Internet depends on the smooth operation of the routing protocols running in and between thousands of Autonomous Systems. The responsibility for stitching these disparate ASes together into a single, coherent network falls to the Border Gateway Protocol, the Internet's interdomain routing protocol. BGP enables routers to learn paths through other ASes to reach remote destination address blocks, or prefixes. A router stores the BGP routes it learns for each destination prefix in RIB, and selects a single "best" route for forwarding data traffic; all other routes are "alternates," used when the primary path becomes unavailable.

Running the Border Gateway Protocol (BGP), the Internet's interdomain routing protocol, consumes a large amount of memory. A BGP-speaking router typically stores one or more routes, each with multiple attributes, for many prefixes. When the router does not have enough memory to store a new route, it may crash or enter into other unspecified behavior, causing serious disruptions for the data traffic.

The focus of this chapter is RIB memory. The chapter proposes and discusses a new mechanism for routers to handle memory limitations without modifying the underlying routing protocol. Upon running out of memory, the router simply discards information about some alternate routes, and requests a "refresh" from its neighbors later if necessary. An optimal offline algorithm is presented that decides which alternate routes to evict, which is used to evaluate the trade-off between RIB memory size and refresh overhead using a large BGP message trace. Based on these promising results, efficient online algorithms are designed and evaluated that achieve most of the performance benefits.

Section 2.2 presents a formalism for describing BGP, providing the background knowledge needed to understand Forgetful Routing; this leads directly into a discussion of Forgetful Routing itself. Section 2.3 analyzes the trade-off between memory savings and refreshes by evaluating an optimal, offline algorithm over a large BGP message trace. Section 2.4 proposes and evaluates several efficient, online algorithms for deciding which alternate routes to evict when the RIB memory is full. In Section 2.5 we examine which ASes would most benefit from deploying Forgetful Routing. We discuss related work in Section 2.6, and conclude the chapter in Section 2.7.

Forgetful Routing has been previously published in the proceedings of the CoNEXT 2006 conference, under the title *Using Forgetful Routing to Control BGP Table Size.*

2.1 Introduction

2.1.1 Memory Limits and Current Workarounds

Given the relatively low cost of memory, and the fact that even desktop PCs often have hundreds of megabytes of main memory, the notion that routers would encounter memory limits may seem surprising. Even taking into account that today's routers must store BGP routes for many prefixes, and growing [43, 84], the amount of space needed seems small. For example, a router with 10 neighbors, with each neighbor announcing a route for each prefix, with 300,000 prefixes, would need approximately 240 megabytes of memory (assuming 80 bytes to store routing entry information). So why the concern?

The problem with these back-of-the-envelope calculations is that they're a *lower bound* on memory, and in practice more memory is needed for various reasons. First, the number of prefixes is sometimes much higher, such as when configuration errors or malicious attacks trigger route leaks, where a router receives BGP announcements for address blocks that are not normally visible. As examples: on April 25th, 1997, AS7007 leaked 23,000 routes, causing enough instability to create massive Internet outages [39]; on October 3rd, 2002, 20% of WorldCom's customers lost connectivity due to a configuration error that "...propagated more route-broadcasts than the affected routers could handle" [123]; on December 24, 2004, AS9121 leaked over 100,000 prefixes [115], *etc.* From 1994 to 2004, there have been more than 60 threads about route leaks on the North American Network Operators' Group (NANOG) mailing list alone [64]. Second, a router may learn multiple BGP routes for a prefix, especially as ASes increasingly connect to the Internet in multiple locations for better fault tolerance and more flexible load balancing. Third, operating system upgrades generally provide new features and consume more memory, adding to the problem. Fourth, data structure overhead and other implementation details consume additional space. Considering all these factors, the BGP routing table can grow quite large, up to a gigabyte in size, with the risk that an unexpected route leak may drive the memory requirements significantly higher.

There are many partial solutions to the memory problem, but no panaceas, such as:

- *Adding more memory.* RIB memory is much more expensive than conventional SDRAM, often between one to two orders of magnitude more in price. Moreover, determining how much memory to add is very challenging. Thus, upgrading the memory for every router in a large AS to some "acceptable" level is quite expensive. In other cases, such as routers deployed in a satellite network, adding memory may be impossible.

- *Using secondary storage.* Swapping parts of the RIB to secondary storage may seem appealing, but many routers do not have disk drives; network operators are reluctant to rely on disks, as they have relatively high failure rates [44]. Even when secondary storage is available, excellent virtual-memory techniques need to be used to prevent thrashing.

- *Using compression.* Applying compression techniques to the RIB data may yield some memory savings, at the expense of computational overhead in handling new update messages. Since routers need to respond quickly to routing changes, any compression scheme would need to be simplistic and operate only over local chunks of data, severely constraining any savings.

- *Filtering routes.* Route filters can be installed to discard routes that do not meet certain criteria. In fact, the Regional Internet Registries (RIR) publish guidelines for the maximum prefix lengths for various parts of the IP address space [82]. However, these guidelines are merely suggestions, and many ISPs ignore them. This creates a catch-22, where ASes do not filter based on RIR guidelines since so many ASes violate them, leading ASes to feel little pressure to obey them.

- *Enforcing prefix limits.* By imposing a hard limit on the number of prefixes accepted from each neighbor, and tearing down the BGP session when the number is exceeded, a router can avoid dedicating too much memory to a single neighbor. However, unless extremely conservative limits are imposed, the router remains vulnerable to learning too many routes across all of its neighbors [115].

In sum, existing technologies that control BGP table growth are either only applicable in specialized circumstances or are generally ineffective.

2.1.2 Practical Constraints on Extending BGP

Unfortunately, devising solutions to BGP's memory problem is not an easy feat. Since BGP is the glue that holds the disparate parts of the Internet together, having a "flag day" to replace BGP with a new protocol is infeasible. Any solution must be incrementally deployable, where one AS can upgrade the software on its routers even if other ASes do not. In addition, upgrading the software needs to offer clear benefits to the early adopters, rather than relying on a large-scale deployment before any memory savings are realized. We argue that a good, practical solution should have the following three properties:

- *Backwards compatibility with existing systems.* In today's Internet, an AS must assume that its neighbors will speak BGP. As such, any new scheme must remain backwards compatible with BGP. Changes to BGP's route selection and advertisement process must prevent forwarding loops and unexpected loss of connectivity.

- *Memory savings for ASes that deploy the solution.* Since interdomain routing is tied to the dynamics of businesses, economic incentives drive the deployment of BGP modifications. Upgrading all the routers in an AS is time consuming and expensive. An AS that deploys the new software should see memory savings, even if no other ASes have deployed the solution.

- *No significant increase in routing convergence delay.* The scheme must not significantly affect the time it takes for routing changes to propagate across a network, also known as convergence delay. During convergence, multiple routers recalculate their routing tables in response to a topology or policy change, and transient forwarding loops or black holes (where packets are lost) may occur. An increase in convergence delay translates into additional time when data traffic may be lost or delayed.

2.1.3 Our Contribution

With these constraints in mind, we propose a new approach that we dub *Forgetful Routing*. To avoid exceeding the available memory, a forgetful router can selectively discard one or more alternate routes. If a discarded route is needed sometime in the future, the router requests a "refresh" from the neighbor responsible for announcing the route.

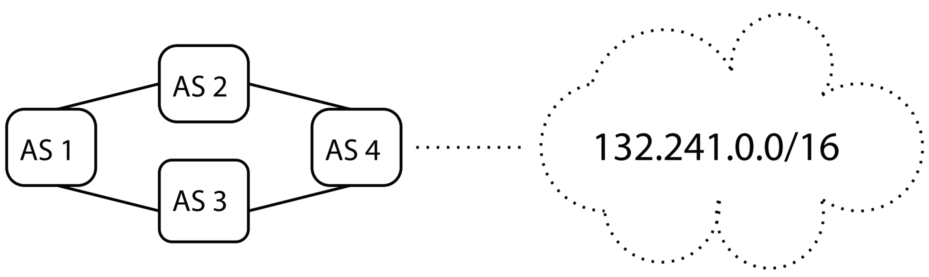
Our solution exploits several important aspects of the BGP routing system. First, *alternate routes are not needed while the primary route is in use.* This enables our scheme to offer significant memory savings by potentially discarding all alternate routes. If a router has, on average, n routes per prefix, it can reduce its total memory usage by a factor n. Second, *every alternate route is some neighbor's best route.* Thus, every forgotten route is available for re-transmission later. This allows a router to always re-construct its original routing table when needed and thus select the same best routes as conventional BGP. Third, *BGP's route-refresh feature can be used to trigger a refresh from a neighbor.* BGP's route-refresh capability [50] has been a standard for many years and is already deployed in many large ASes. Although designed for a different purpose, it can be used to ask a router to resend BGP announcements. While it would need modification to only send information pertinent to a single prefix, the current feature helps to lower the barrier to deploying Forgetful Routing.

Moreover, our solution does not require any changes to the BGP protocol – only software upgrades on the routers themselves; in addition, one AS could deploy Forgetful Routing even if other ASes do not. Even in the rare case where none of its neighbors support the route-refresh feature, an AS can use Forgetful Routing to reduce the amount of memory consumed by internally learned routes (i.e., in internal BGP).

Forgetful Routing introduces a trade-off between memory size and refresh overhead, leading to a sort of a "cache replacement" problem. Our goal is to create an efficient eviction algorithm that minimizes the "miss rate," *i.e.*, the likelihood that the best route for a prefix does not already reside in the routing table and will trigger a route refresh. We first present an optimal offline algorithm that assumes perfect knowledge of the future arrivals of BGP update messages. Experiments applying the optimal algorithm to BGP message traces show that Forgetful Routing can achieve substantial reductions in memory usage. The analysis of the measurement data also provides important insights into the characteristics of BGP update dynamics. We capitalize on these observations in creating two efficient online algorithms that evict the least attractive alternate routes for prefixes that have not changed their best routes for the longest time. Efficient data structures enable a forgetful router to make eviction decisions in constant time, and our experiments show that the online algorithms perform well over actual BGP data, reducing the memory footprint with a moderate number of refresh operations.

2.2 Forgetful Routing

Describing all aspects of BGP would require an entire book [127], and many of these details are irrelevant to memory management. We first present a simple formalism that captures how BGP-speaking routers handle update messages and select routes; this formalism is used throughout the chapter as a reference model. Then, we apply this model to explain Forgetful Routing and introduce the cache replacement problem that is the focus of the rest of the chapter.



AS	RIB
AS 1	(132.241.0.0/16, AS3, 1 $\rightarrow$ 3 $\rightarrow$ 4 $\rightarrow$ ...) (132.241.0.0/16, AS2, 1 $\rightarrow$ 2 $\rightarrow$ 4 $\rightarrow$ ...)
AS 2	(132.241.0.0/16, AS4, 2 $\rightarrow$ 4 $\rightarrow$ ...)
AS 3	(132.241.0.0/16, AS4, 3 $\rightarrow$ 4 $\rightarrow$ ...)
AS 4	(132.241.0.0/16, ..., 4 $\rightarrow$ ...)

Figure 2.1: A hypothetical network where AS 1 can route to 132.241.0.0/16. After AS 4 announces a route to the prefix to its neighbors, AS 2 and AS 3 will be able to route to it. They, in turn, generate announcements for AS 1. RIB entries for this prefix are shown using the (p, n, r) notation. Note that AS 1 can always re-derive its RIB entries by issuing route refreshes to its neighbors.

2.2.1 An Abstract Model of BGP

Initially, a router establishes a BGP session with each neighbor. Each router then shares information about the best routes through update messages. After learning new information from its neighbors, the router checks if better routes exist, and if so, uses them. Upon changing its best route, the router must send an update message to any neighbors that previously received an announcement for the replaced route, indicating that the old route is no longer in use; this withdraw is then followed by an announcement of the new route[1]. If connectivity to a prefix is lost, only a withdraw message is sent.

All the information about routes is stored in the Routing Information Base. A route is a three-tuple (p, n, r) containing a prefix, a neighbor that advertised the prefix, and some route attributes (*e.g.*, the number of hops required to reach the

[1]In practice, an announcement is an implicit withdrawal of the previous route, requiring only one update message.

```
while session exists:
   (m, (p, n, r)) = get_message()
   oldbest = m_best(p, RIB)
   if m == ``ANNOUNCE'':


      RIB = RIB + (p, n, r)
   if m == ``WITHDRAW'':
      RIB = RIB - (p, n, r)
   newbest = m_best(p, RIB)
   if oldbest != newbest:
      generate_withdraw(oldbest)


      generate_announce(newbest)
      route_data_using(newbest)
```

(a) Regular Router Pseudocode

```
while session exists:
   (m, (p, n, r)) = get_message()
   oldbest = m_best(p, RIB)
   if m == ``ANNOUNCE'':
      while no memory available:
         evict_route(RIB)
      RIB = RIB + (p, n, r)
   if m == ``WITHDRAW'':
      RIB = RIB - (p, n, r)
   newbest = m_best(p, RIB)
   if oldbest != newbest:
      generate_withdraw(oldbest)
      if is_evicted(newbest):
         refresh_route(newbest)
      generate_announce(newbest)
      route_data_using(newbest)
```

(b) Forgetful Router Pseudocode

Figure 2.2: Pseudocode describing how a regular router and how a forgetful router would operate. These two code pieces are space-aligned to help highlight the differences.

destination, preference values, *etc.*). The RIB is a set of such routes. By default, we assume that for every prefix p, the RIB contains a null route to that destination, *i.e.,* the entry $(p, \emptyset, \emptyset)$, signifying that the router cannot or does not want to route to it.

An *announcement* is an update message of the form:

$$(``ANNOUNCE'', (p, n, r))$$

that tells the router that neighbor n currently uses route r to reach prefix p. The neighbor n can only advertise its best routes, and must be using the route r to reach p. A *withdraw* is an update message of the form:

$$(``WITHDRAW'', (p, n, r))$$

indicating that neighbor n can no longer reach prefix p using its previously announced route. The RIB entry corresponding to (p, n, r) is removed. See Figure 2.1 for an example of how the RIB becomes populated.

Once the RIB becomes populated with entries, the router must decide how to route data to all accessible prefixes. The decision is calculated by using a total ordering[2] over each R_p, where R_p is the subset of RIB entries that route to prefix p. When picking a "best" route for each prefix, BGP relies on a combination of custom routing policies and the BGP Decision Process [52], a universally agreed-upon ranking function. An example of a routing policy may be, "prefer routes learned from neighbor p over neighbor q." The BGP Decision Process is a multi-step procedure over the route attributes: *e.g.*, prefer routes with fewer AS hops to ones that have more; if a tie exists, prefer routes that were learned earlier than ones learned later; if another tie exists, prefer routes learned from ASes with a lower ID number.

In general, it is possible to think of these routing policies and the BGP decision process as being represented by a single metric that compares all the routes. This metric uses all the route attributes, along with these rules and policies, to rank them in the same order as in BGP. We define $m_{\texttt{best}}$ to be the metric that picks the routes to use for forwarding traffic to prefix p:

$$\begin{aligned} m_{\texttt{best}} : R_p &\rightarrow (p, n, r) \\ m_{\texttt{best}}(R_p) &\in R_p \\ \forall e \in R_p,\ e &\preceq m_{\texttt{best}}(R_p) \end{aligned}$$

[2]In practice, some ASes use the Multiple-Exit Discriminator (MED) attribute in a way that prevents the formation of a total ordering [54]. This leads to other problems, such as non-deterministic routing decisions and protocol oscillation. In our work, we assume that MED is not used, or is configured to produce a deterministic outcome. When this assumption does not hold, the simplest solution is to store all routes for a prefix if any route has the MED set.

The first two equations state that $m_{\texttt{best}}$ operates over R_p and always picks a RIB entry from R_p; the last one describes how the "best" RIB entry has the highest rank among all entries.

When the session closes, the router withdraws all routes advertised by the neighbor, re-computes its new set of best routes, and sends updates as needed. As new routes are learned and old ones are forgotten, the set of best routes will change over time. Each time a best route changes, update messages are generated for appropriate neighbors as long as the BGP session continues. See Figure 2.2(a) for pseudocode describing how an ideal router acts.

2.2.2 Evicting and Refreshing Alternate Routes

Routers have fixed amounts of memory and can only store a limited number of RIB entries. However, the BGP specification does not define how the protocol should behave when a router runs out of memory. Furthermore, BGP is memory intensive. It is *prefix-based*, and although this is more efficient than enumerating IP addresses, there are many prefixes. Moreover, BGP is a *path-vector* protocol, forcing routers to store the entire AS path for each router. In addition, BGP is a *policy-based* protocol, with numerous route attributes that influence the selection and propagation of routes. Finally, BGP is an *incremental protocol*: upon receiving an update message, a router must compare the new information with all previously learned routes to select the new best route; information cannot be arbitrarily discarded, since it may be needed at a later time.

We thus propose the following behavior whenever a router experiences a dearth of free memory: pick a (p, n, r) that is currently an alternate route and *evict* extraneous information from r that is ignored by $m_{\texttt{best}}$. We define an evictor ev with the following properties:

$$\begin{aligned} ev : (p, n, r) &\rightarrow (p, n, r') \\ \texttt{sizeof}\ r' &\leq \texttt{sizeof}\ r \end{aligned}$$

These equations tell us that *ev* modifies the additional routing information to possibly consume less memory. Moreover, modifying our RIB such that one of the entries has evicted information does not change the BGP decision process. See Figure 2.3 for an example RIB entry and how it could be compressed.

The fact that BGP's routing data can be compressed without affecting its decision process is non-intuitive. For example, although the decision process favors routes with fewer hops than those with more hops, BGP stores the entire AS path instead of the length. BGP must do this so when it advertises a route, loop detection can occur. Thus, while routes are not announced, they can be safely compressed; when they are in use, all their original routing information must be retrieved and sent to all neighbors.

Because the modified route contains enough information to allow $m_{\texttt{best}}$ to rank it, the routes chosen by this scheme always match the routes chosen by an equivalent, regular router. However, if a modified route suddenly becomes "best," it cannot be announced until the complete non-metric routing information is retrieved, since it might be relevant to other routers. Retrieving this route will require a *refresh*. Since routers only advertise their best routes to their neighbors, every alternate route is always some neighbor's best route, so the additional routing information will be retrievable.

In order to decide which alternate routes to evict, we define an eviction mechanism $m_{\texttt{evict}}$:

$$\begin{aligned} m_{\texttt{evict}} : R &\rightarrow (p, n, r) \\ m_{\texttt{evict}}(R) &\in R \\ \forall p, m_{\texttt{evict}}(R) &\neq m_{\texttt{best}}(p, R) \end{aligned}$$

These equations state that the eviction mechanism can choose any route in the RIB, as long as it is an alternate and as long as it has not already had some of its routing information evicted. Pseudocode used to describe a router's behavior with this extension is provided in Figure 2.2(b).

It is important to note that *ev* does not have to keep the metric relevant information. For example, *ev* could simply delete all additional routing information. This would force the router to issue a set of refreshes when it could not identify the new best route. In fact, there is an interesting trade-off between ambiguity introduced to the metric and potential savings. While we do not explore this trade-off, it holds promise for future research.

2.2.3 Properties of Forgetful Routing

Forgetful Routing has three important properties: it does not alter the BGP decision process, it is incrementally deployable, and we expect it to add minimal convergence delay.

Because compressed RIB entries keep enough information to allow ranking, and because evicted information is retrievable, a forgetful router acts exactly like a regular router after convergence. Although a network of forgetful routers will act differently than a network of regular routers during route exploration and route changes (due to the fact that different routes may be announced at different times, due to delays

```
PREFIX:         4.21.252.0/23
FROM:           194.85.4.55 AS3277
TIME:           2004-12-31 20:07:56
TYPE:           MSG_TABLE_DUMP/AFI_IP
VIEW: 0         SEQUENCE: 440
STATUS:         1
ORIGINATED:     Fri Dec 31 06:26:51 2004
AS_PATH:        3277 13062 20764 701 6389 8063 19198
NEXT_HOP:       194.85.4.55
COMMUNITIES:    3277:13062 3277:65301 3277:65307 20764:3000
                20764:3011 20764:3020 20764:3022
```

Figure 2.3: A BGP RIB entry from a Route Views table dump. All values other than "prefix" and "from" are represented by r in our abstract model. Note that while the path attributes would take at least 50 bytes to encode, the metric-relevant attributes listed here (time and AS path length) could be encoded in about 5 bytes.

introduced by refreshes), the two systems will always converge to the same steady state solution.

Moreover, it is possible for an Autonomous System to deploy Forgetful Routing and obtain significant memory savings even if no other AS uses it. Since BGP already has a route-refresh option [50], and since Forgetful Routing's only assumption about its neighbors is that they support a form of refreshing a route, there are fewer barriers to deploying the system. However, it is important to note that the route-refresh message triggers all prefixes to be re-advertised. This introduces a considerable amount of overhead for a single prefix needing a refresh. However, a cooperative route filtering capability [49] has been suggested that would allow individual prefixes to be refreshed.

In addition, we believe that Forgetful Routing should not typically add significant delay to the propagation of routes. This property is best illustrated through the following example. Imagine a network with three routers, R1, R2, and R3, that can all route to prefix P. R1 depends on R2 to reach P, R2 depends on R3, and R3 has some mechanism to reach P; in other words, there is a chain of dependency. Now imagine that all alternates are discarded and R3 loses its primary route. While R3 is refreshing, it simultaneously sends a withdraw message to R2, allowing R2 to start issuing its own refreshes *in parallel*. Likewise, R2 performs the same actions, allowing R1 to start issuing its own refreshes as well. By the time R3 has recomputed its best route and sent it out, R2 will have finished performing its refreshes and can propagate the new routing information instantly. The same holds true for R1. The net effect in this situation is that the expected, additional end-to-end delay increases by the time of resolving a single refresh. Given that typical convergence delay on the Internet is on the order of minutes, and given that a refresh may be processed in milliseconds, we feel that the additional delay is negligible. However, we have also been able to formulate scenarios where the convergence time increases proportionally to the length of the dependency chain. Such instances require precise timing of network events, though, in order to trigger this effect. While there may be other scenarios out there that we have not considered, our thought experiments lead us to believe that Forgetful Routing will probably impact the convergence time in a minimal way.

2.3 Optimal Offline Algorithm

Choosing an eviction policy for forgetful routing is a specialized instance of the cache replacement problem. Since an eviction policy will depend highly on the RIB entries and update messages seen, it is important to evaluate different policies and their performance. In fact, some baseline is necessary even before evaluation of policies can begin.

In order to investigate the trade-off curve and to obtain a gold standard for comparing other routing policies, we developed an offline algorithm that, given a RIB dump (*i.e.*, a set of (p, n, r) in the RIB at a specific time) and a stream of update messages, determines the best possible trade-off between available memory and fre-

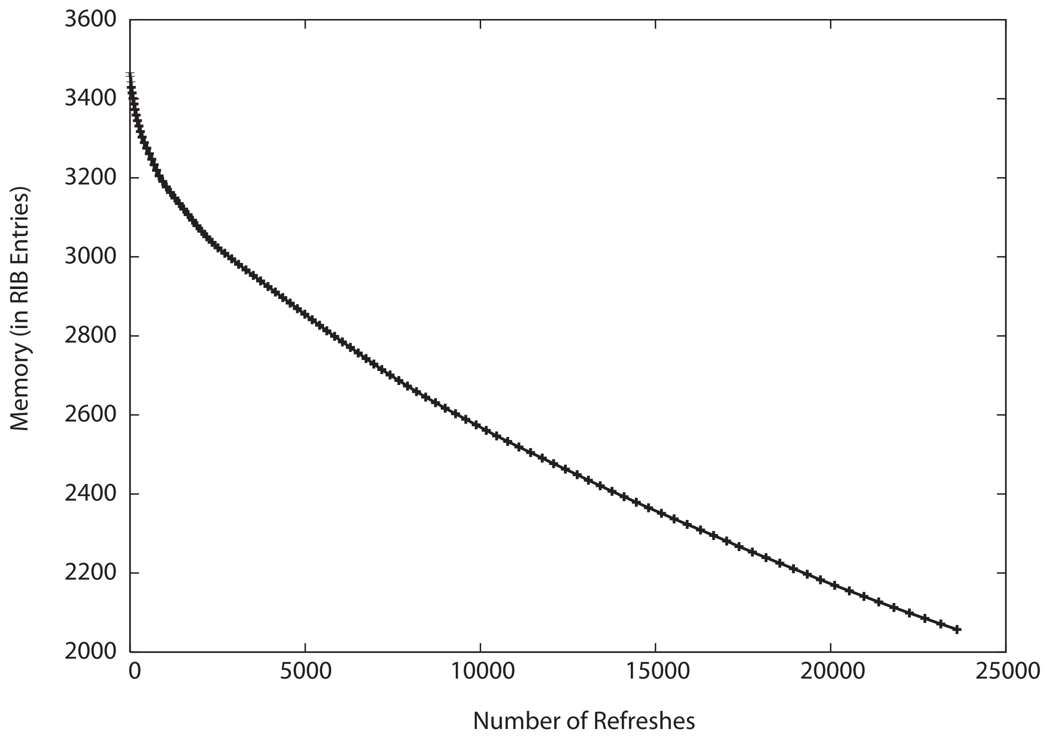


Figure 2.4: The optimal trade-off curve from January 1, 2005 to July 1, 2005, with number of refreshes issued as a function of memory.

quency of refreshes. For simplicity, we assume that every eviction frees an entire RIB entry, and that all RIB entries consume the same amount of memory. The algorithm is offline because in order to calculate the theoretical best trade-off, it must have foresight of the future to influence decisions in the past.

2.3.1 Optimal Eviction Policy

The first step of the optimal algorithm computes the set usage times of all routes. That is, each route is annotated with the set of times that it will be selected as "best." This is calculated by performing a pass through an initial RIB dump and a stream of update messages, maintaining a hash table that maps from RIB entries to lists, where each RIB entry has its associated list updated whenever it is chosen as best.

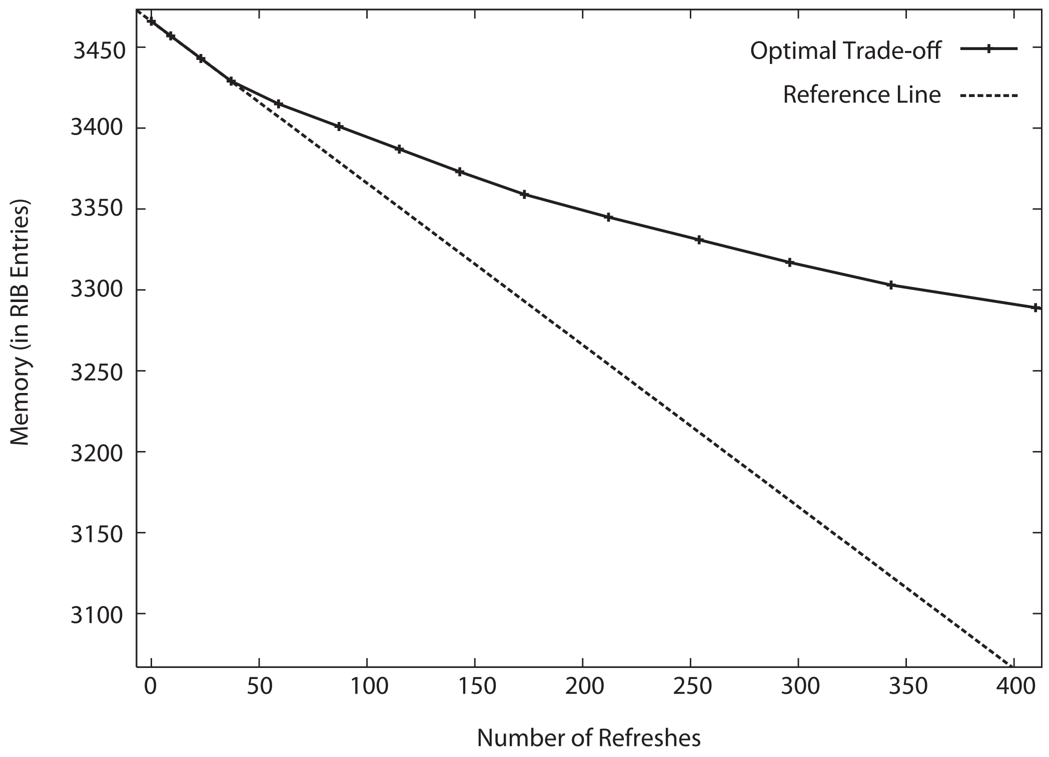


Figure 2.5: Close-up of the optimal trade-off curve. The dotted line has a slope of -1 and is meant as reference.

A simulated router is used for determining usage for each RIB entry e:

$$\texttt{use}(e) = \{t_1, t_2, \ldots, t_n\}$$

Element t_i represents the $i^{\texttt{th}}$ time that $m_{\texttt{best}}$ chose the RIB entry e.

Once this calculation is complete, the eviction policy is straightforward: always choose the alternate route that will be used last, *i.e.*, furthest in the future. Given that the current time is t when an eviction is about to occur, the route that will be needed furthest in the future can be calculated and $m_{\texttt{evict}}$ can be implemented via the following statements:

$$\texttt{nextuse}(e,t) = \begin{cases} 0 & \texttt{if } e \texttt{ is best} \\ \min_{t_i > t} \texttt{use}(e) & \texttt{if } \exists t_i > t \\ \infty & \texttt{otherwise} \end{cases}$$

$$m_{\texttt{evict}}(R) = \texttt{maxarg}_{e \in R}[\texttt{nextuse}(e,t)]$$

The function *nextuse* returns the earliest time after t when the route will be used. Note that 0 is returned if e is currently best (ensuring the algorithm picks an alternate route), and that routes that are never needed are given a time of infinity, making them the best eviction candidates.

A proof of optimality has been omitted due to length constraints, but the intuition behind the proof can be described concisely. By always choosing a route that is needed furthest in the future, we can guarantee that we cannot do any worse than if we picked any other route. This is because any route that is needed in the future will cause an eventual refresh and, in the interim time, lowers the total amount of memory needed. Given that routes are independent of each other in terms of their usage, it is not possible to pick a route that will be needed *earlier*, and yet somehow causes other routes to refresh *less* often; the route that is needed furthest in the future provides the longest interval of memory relief, allowing other routes to remain in memory longer before eviction.

Implementing this algorithm naïvely is easy: perform a pass through the RIB dump and update stream to calculate a list of usage times for every route, and use this information to evict routes. Implementing this scheme efficiently, however, is much more complex. Without going into details, lazy evaluation of evictions can be used to achieve the same results without as much computation; that is, instead of calculating which route needs to be evicted at the very moment when memory capacity has been reached, the simulation can note that a route was forgotten and later deduce the correct one by monitoring usage of alternate routes.

2.3.2 Evaluation on BGP Message Traces

The data set we use was obtained from Route Views [106] and consisted of a BGP table dump on January 1st, 2005, and all BGP update information for six consecutive months. Route Views was chosen primarily because it is publicly available, while ISP feeds are proprietary and difficult to obtain. There were approximately 270,000 different prefixes announced over that period. We randomly sampled 1% of them and their associated updates for our analysis. Our sampling is due to the fact that these simulations take significant computational time – on the order of years when making multiple scans through six month's worth of update messages.

It is important to note that the data obtained from Route Views is not typical. Route Views connects to many more neighbors than most routers and thus has many more alternate routes, upwards of 40 per prefix. Rather than filter the Route Views data to represent a "typical" AS, we included all feeds to capture the dynamics of Forgetful Routing, as well as quantify how much memory savings are possible. Thus, although our estimates of the amount of memory saved may be optimistic, Route Views allows us to examine the general impact of forgetful routing. We leave analysis of the gains typical ASes would see to Section 2.5.

Our results show that many alternate routes are never needed and can be safely discarded without causing refreshes. In the best possible case, an average of one alternate route per prefix is sufficient and will not cause any refreshes. The algorithm's trade-off curve can be seen in Figure 2.3. The first important item to note is the reduction in BGP table size. At zero refreshes, the amount of memory needed never exceeds 3,500 RIB entries, whereas a full table would require approximately

68,000 entries. In other words, if the Route Views router had foresight and sufficient additional computing power, it could reduce its memory footprint size by 95%.

The second important item to note is the initial one-to-one trade-off between memory size and refreshes, as seen in Figure 2.3. Initially, lowering the total amount of memory by one unit causes one refresh. In this part of the curve, the alternate routes that are evicted in these cases belong to very stable prefixes, where alternates are rarely used. After total memory has decreased by about 200 units, freeing one unit of memory results in multiple refreshes. This occurs when we have evicted all the alternate routes for stable prefixes and we must start evicting routes for unstable ones. Since these alternate routes will be refreshed into memory much sooner, more refreshes will be needed in the same period of time.

Even at a memory size of 2048 memory units, the number of required refreshes is surprisingly small, given that sixteen million RIB entries were added and removed for this six-month interval through update messages. Furthermore, this memory footprint is 3% of the original size. All of these items demonstrate several important points: 1.) that tremendous memory savings are possible without significant increases in bandwidth, 2.) that the majority of alternate routes are not used, and 3.) that the best way to minimize refreshes is to keep the alternate routes for unstable prefixes "in the cache."

2.4 Online Algorithms

Although the results of the offline analysis are promising, the algorithm is not feasible in practice; foresight of the future and infinite computing power are not luxuries afforded to today's routers. In this section, we devise efficient online algorithms that approximate the optimal results. We evaluate these online algorithms in two parts. First, we examine all algorithms under the assumptions that RIB entries have uniform

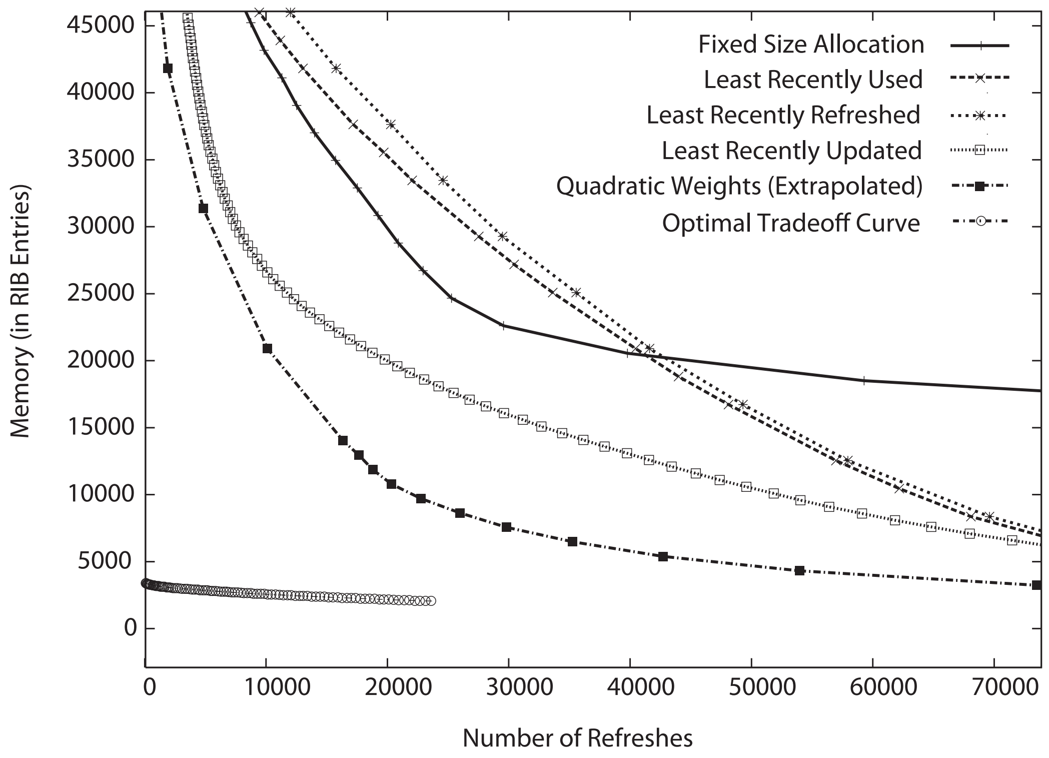


Figure 2.6: The trade-off between refreshes and number of RIB entries needed for various online algorithms. 2684 prefixes were used for these calculations (randomly sampled out of the 268480 total prefixes seen over a six month period).

size, that each eviction frees one RIB entry of space, and that memory overhead from additional data structures is negligible; this yields raw performance information for direct comparison with the optimal offline algorithm. Then, we re-examine the algorithms under realistic assumptions.

2.4.1 Least Recently Refreshed and Updated

When first devising an online algorithm, it is important to see whether a very simple scheme can achieve most of the savings. As such, our first strategy was constant-size allocation, where each prefix is given a fixed amount of memory in advance. By varying the amount of dedicated memory per prefix, and evicting the lowest-ranked routes for each prefix when memory is tight, the memory-bandwidth trade-off can be explored. We implemented such a scheme and found it to be too simplistic; prefixes

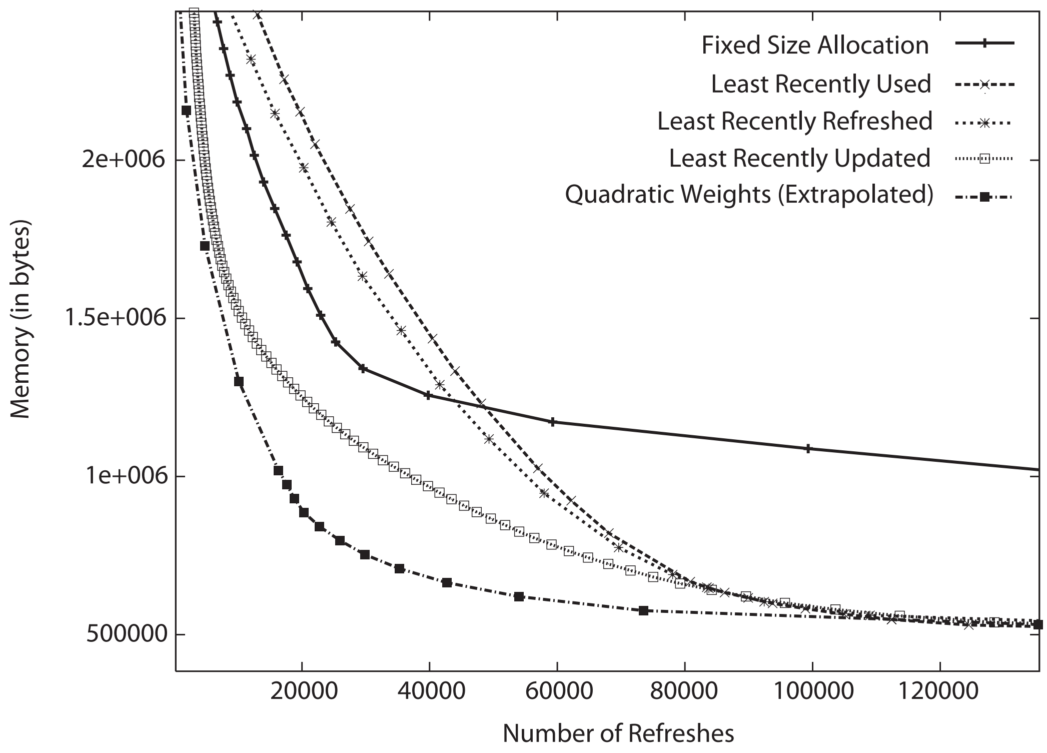


Figure 2.7: The tradeoff between refreshes and memory capacity for various online algorithms. 2684 prefixes were used for these calculations (randomly sampled out of the 268480 total prefixes seen over a six month period).

with very stable primary routes are allocated too much space, while unstable prefixes are allocated too little space, leading to frequent refreshes.

Our next strategy was to evaluate the canonical algorithm for cache replacement problems: Least Recently Used (LRU). Our LRU algorithm uses a doubly-linked list, where each cell has a RIB entry. RIB entries are added to the back (and removed when their associated withdraw is received), and are picked for eviction from the front. Thus, assuming four bytes per pointer with two pointers per linked list cell, all decisions can be made in constant time with $8n_r$ bytes of memory overhead, where n_r is the number of RIB entries. Unfortunately, $8n_r$ overhead is rather steep, especially considering our goal is to *save* memory, not consume it!

In order to achieve most of the benefits of LRU without the memory overhead, some approximation is needed. We thus devised a variant called Least Recently

Refreshed (LRR). Under LRR, we always evict the least desirable route of the prefix that has not needed a refresh in the longest amount of time. In doing so, we can replace our doubly-linked list of *RIB entries* with a doubly-linked list of *prefixes*. Prefixes are put on the back of the list when they have just been discovered by the router and when they are just refreshed. This enables all operations to complete in constant time, while reducing memory overhead to $8n_p$, where n_p is the number of prefixes.

In addition, a variant of LRR called *Least Recently Updated* (LRUp) was implemented. This variant uses the time since the last update message as the ranking metric, rather than time since last refresh. It uses all the same memory structures as the LRR algorithm and the same computational operations.

The results of fixed-size allocation, LRU, and LRR can be seen in Figure 2.4. Although fixed size allocation performs reasonably well when a large amount of memory is available, it quickly begins to fail as memory is driven lower and lower. From the figure, it is clear that simple schemes cannot capture the dynamics of routes and their refresh needs. Both LRU and LRR, on the other hand, perform competitively, within an order of magnitude of the theoretical best. Although LRR's data structures require much less memory, the performance is nearly identical to LRU. For a small sacrifice in the number of refreshes, a significant reduction in overhead can be achieved.

Surprisingly, LRUp performs much better than LRU or LRR. The reason behind this phenomenon lies in the fact that any update activity is usually a sign of route instability. By exploiting this fact, LRUp can anticipate which prefixes will need more alternate routes available in memory. Still, a sizable gap remains between the LRUp scheme and the optimal offline algorithm. Although some gap is inevitable, because the offline algorithm has foresight of the future, we wanted to explore whether a more sophisticated eviction policy could improve the effectiveness of forgetful routing.

2.4.2 Quadratic Weights

Although LRU, LRR, and LRUp perform well, they rely on just a single variable to rank routes for eviction. Moreover, none of these schemes consider the *number* of alternate routes available. Often, when a relatively stable prefix has a routing change, the router switches from the primary route to the first or second alternate route [118]. Keeping one or two alternate routes in the RIB, even for the relatively stable prefixes, can help reduce the number of refreshes. At the expense of additional computational complexity, an online algorithm can account for both of these factors: the time since the last refresh and the number of alternate routes.

In doing so, we devise another online algorithm which we call quadratic weights (QW). Instead of using time to order routes, we calculate a "goodness" value for each prefix p, according to the formula $n * (n - 1) * t$, where n is the number of routes to p and t is the time since the last refresh. Note that other multiplicative factors could be used, such as linear or cubic. Depending on the dynamics of the network, such factors may overlook or overstress the importance of alternate routes.[3] After a factor is chosen, routes are ordered by goodness and then appropriately evicted.

Although the quadratic-weights algorithm only consumes $8n_p$ bytes of memory in additional data structures, it has $O(n_p)$ computational overhead. Since each prefix's t value is constantly changing (even if no corresponding update messages have been received), all prefixes must have their goodness values re-computed during an eviction.

The results of this algorithm can be seen in figure 2.4. Due to the computational overhead, the quadratic weights algorithm was calculated over a 0.1% sampling and then extrapolated for our 1% sampling. We believe this extrapolation to be fairly accurate, as extrapolating the LRU algorithm from the same 0.1% sampling back to a 1% sampling resulted in an almost perfect fit. Quadratic Weights is better than

[3]Our choice of a multiplicative, quadratic factor of `n(n-1)` came from theoretical analysis that made simplifying assumptions about the types of updates seen and their frequency.

any of the online algorithms, but not considerably; in particular, it is somewhat close to the LRUp algorithm. Given that LRUp is very simple to implement in practice, it is unclear that if a QW approximation algorithm would be desirable, as it would most likely be more complicated.

2.4.3 Memory Usage

In order to understand how these algorithms would operate in practice, we extrapolated total memory usage for each online algorithm. Drawing on analysis of memory usage on several commercial routers, we assumed an average size of 45 bytes per RIB entry. Any memory overhead from the algorithms' data structures was taken into account. Additionally, one byte per RIB entry of overhead was added for booking which routes were forgotten. Lastly, we assumed that evicting a route leaves behind four bytes of metric-relevant data.[4].

The results can be seen in Figure 2.4. For reference, a regular router would need to consume approximately 353 megabytes of space to store all of the routes, or 3.53 megabytes at a 1% sampling.

It is interesting to note that all the algorithms hardly deviate from their relative positions. Moreover, the memory savings are still quite substantial. Approximately 10% of the memory savings is lost due to overhead costs, most of which stem from the one additional byte per RIB entry and the four bytes of metric-relevant data left behind. If one bit was used instead of one byte (to mark whether the routing data was represented in a compact format or not) and fewer bytes of metric-relevant data were used, this gap could be closed even further. Exploring the trade-off of throwing out some metric data (and thus making it possible to have a set of possible next best

[4]Of these four bytes, one can be allocated for AS path length, one for ranking the time the route was learned against other routes for this prefix, and two for compactly representing the other BGP metric variables

routes) and the increase in refreshes that would be required is an area of potential future work.

2.5 Expected Memory Savings

Given that Forgetful Routing's ability to save memory relies on network topology, it is important to investigate how much savings real routers might see. Moreover, observing how these gains vary by AS provides insight into the types of systems that would see the most benefit from our scheme.

2.5.1 Challenges for Realistic Evaluation

Ideally, we would like to obtain the BGP message streams sent to a router and use them to quantify the dynamics of Forgetful Routing. By sampling these feeds for routers from ISPs, business customers, university networks, *etc.*, we could build an extremely accurate picture of how Forgetful Routing would affect a wide range of BGP speakers, both in terms of memory and in terms of refreshes. Unfortunately, it is very difficult to obtain accurate information about real-world memory savings. Knowing how ASes' connect and what messages they receive reveals valuable business information. For this reason, companies do not publish their feeds nor do they make them easily accessible.

While we do have Route Views as a source, it is not representative of any typical AS. This is because it connects to over forty neighbors and receives full routes (i.e., routes for all destination prefixes) from many of them. While we can use Route Views to evaluate different algorithms against each other, we cannot use it to evaluate expected gains. Since obtaining dynamic feeds is out of the question, we cannot deduce real world estimates of the refresh rates of Forgetful Routing. However, static dumps of various RIBs are available, allowing us to quantify memory savings. First,

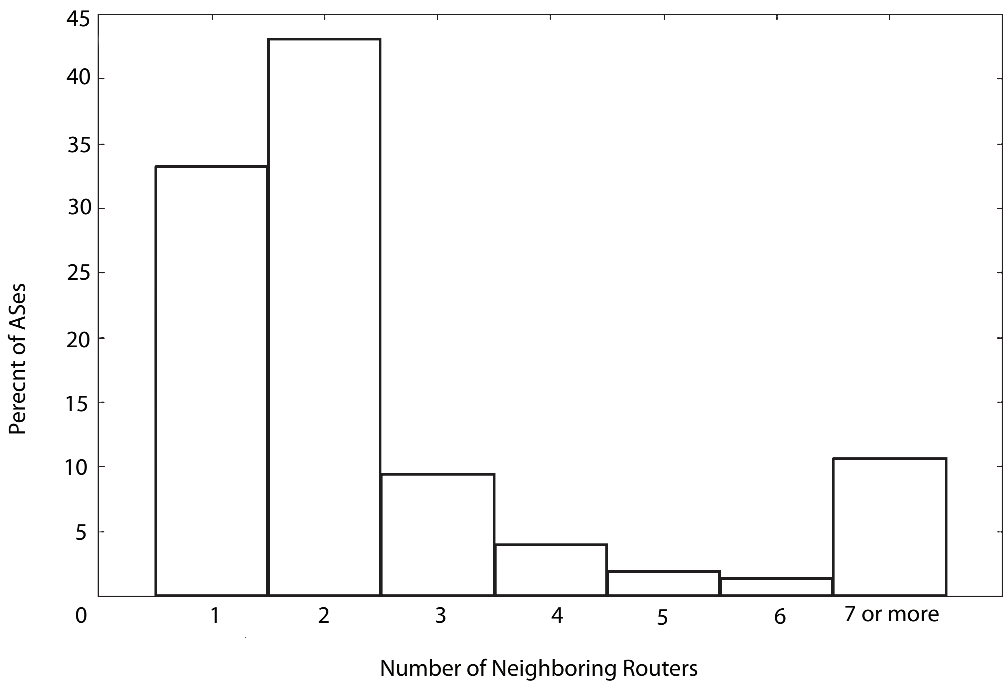


Figure 2.8: Percent of ASes with a given number of neighboring ASes.

we analyzed the RIB of the border router that connects Princeton University to the rest of the Internet. The Princeton campus network connects to four other networks. Based on a dump from April 6th, 2006, we observed that, on average, the Princeton network has 2.4 alternate routes per prefix. Thus, a memory savings of nearly 70% would be achievable for this router.

Next, we looked at BGP data from looking glass servers. A looking glass server is a router that additionally runs a publicly available interface (such as a website or a telnet connection) permitting anyone to query routing information from it. The results of analyzing five looking glass servers [108] can be found in Table 2.1. For each of these ASes except AS 7018, twenty prefixes were randomly sampled and the average number of RIB entries calculated. For AS 7018, the first 20,000 prefixes in its RIB table were used to compute its average. All of these ASes represent domains with high connectivity. Unfortunately, looking glass servers are not very accurate indicators of what real routers' RIB tables look like. Because BGP information has business value, as describes how different competing organizations form alliances with each other, looking glass servers tend to omit many RIB entries. For example, AS7018, which is AT&T, is reported to have an average of 1.2 available routes per prefix, a number which is absurdly low for a tier-1 provider.

2.5.2 Inference of BGP RIB Sizes

Looking at such a small number of static dumps is very limiting. Without a wider breadth of routers' RIBs, it is difficult to quantify what typical gains an AS might see. However, we can infer other routers' RIBs without direct access to them. If we know the Internet's topology, and if we know where prefixes originate, and if we know the routing policies used between ASes, we can simulate the propagation of routing information across all the unknown ASes' routers and calculate what their RIBs look like.

In order to infer Internet topology, all that is needed is a RIB dump from any well-connected router. By looking at all AS paths for all routes in the dump, one can deduce that there exists a link between each adjacent pair of ASes (*e.g.*, if a router receives a route advertisement that has an AS path of (41, 65, 45), then 41 and 65 connect to each other, as do 65 and 45). Doing so results in a conservative count—there may be some links that were not represented in AS paths, depending on the reference router's isolation from other networks. Thus, any Internet topology inference based on this scheme will, if anything, undercount various ASes' RIBs in a simulation, as well as the number of alternate routes available to prefixes.

Calculating where prefixes originate is a rather simple task. For each RIB entry in a router dump, observe the prefix that it routes to as well as the last AS in the AS path. By definition of BGP's behavior, this AS will be the originator. Problems that may arise include undercounting the number of prefixes available. Since these prefixes and their origins are derived from a RIB dump, a router that is partitioned off from a significant section of the Internet may not know where certain prefixes originate, or may not know that they even exist. Likewise with Internet topology, any inference based on this scheme will undercount the number of RIB entries in a simulation, but will not affect the count of alternate routes available to known prefixes.

Inferring an AS's routing policies is difficult. By observing sets of AS paths, one can examine how they constrain the possible relationships between organizations and attempt to model them. A scheme that infers extremely liberal routing policies will overcount the number of alternate routes per prefix, while a scheme that infers extremely conservative routing policies will undercount the number of alternates.

To derive routing policies, we used the Gao inference algorithm [72] on a BGP RIB dump obtained from Route Views [106] on February 10th, 2005. The Gao algorithm defines three different types of business relationships: provider-customer, peer-peer, and sibling-sibling. Provider-customer relationships represent one AS purchasing connectivity from another AS. Providers almost always advertise all their routing information to their customers, as they typically charge their customers for traffic that flows between them. Peer-peer relationships represent two competing organizations that connect to each other in order to achieve greater reachability. Peers generally only advertise their customer-learned routes and do not re-advertise routes they learn from other peers. By doing so, their customers have great reachability (and thus more traffic, generating more revenue), and they never need to transit traffic meant for another peer (which would consume network resources without generating revenue). Sibling-sibling relationships indicate that the two ASes are actually controlled by the same organization, either through mergers or buyouts. In some sense, siblings should be treated as one large AS.

The results from our analysis can be seen in Figure 2.8. From this figure we can identify three different categories of Autonomous Systems:

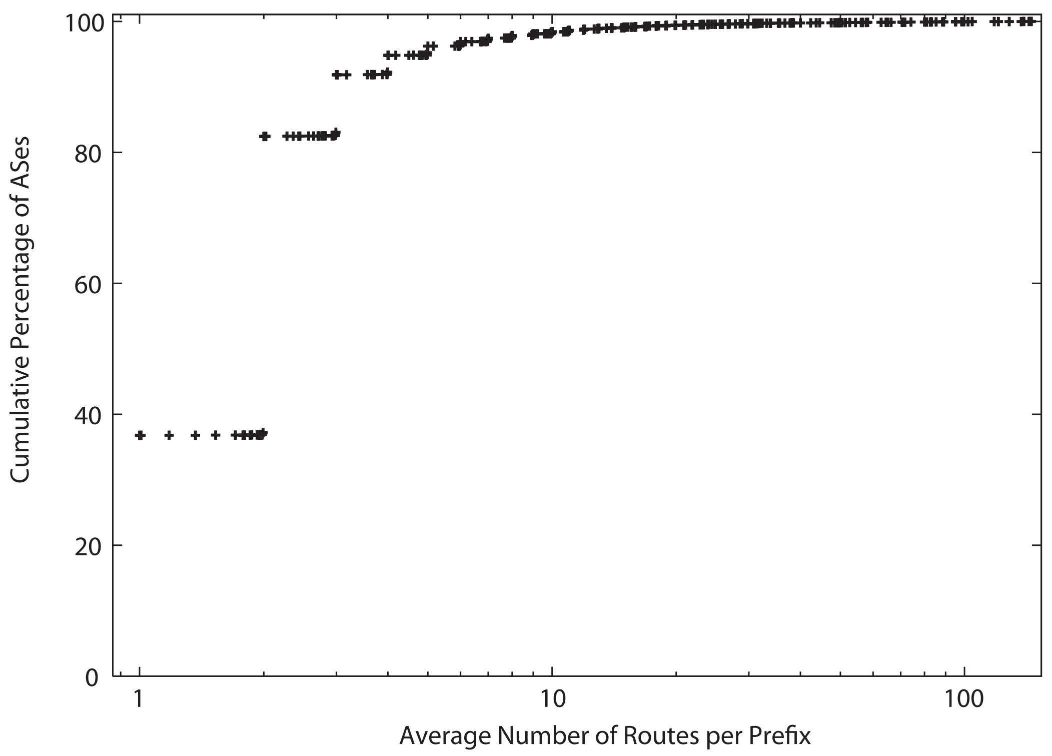


Figure 2.9: Percent of ASes with a given average number of routes per prefix.

- Single-homed ASes are Autonomous Systems that purchase their Internet connectivity from a single, upstream provider. Based on our data, about 30% of all ASes fall into this category. Because they only have one provider, they never have alternate routes. For these systems, Forgetful Routing offers no benefits.

- Multi-homed ASes are Autonomous Systems that purchase their Internet connectivity from two or more upstream providers. Approximately 40% of all Autonomous Systems belongs to this labeling, including the Princeton network mentioned earlier. The upstream providers are used either as emergency backups or for load balancing. Multi-homed ASes are thus candidates for Forgetful Routing.

- Providers are Autonomous Systems that have excellent connectivity and typically sell their reachability to single-homed or multi-homed customers. Some of these providers may themselves be customers of larger ISPs. Approximately

30% of all ASes fall into this category. For these systems, Forgetful Routing could offer an order of magnitude of memory saved, or more.

Additionally, by analyzing the business relationships between ASes, we inferred the number of alternate routes available to each Autonomous System for each prefix. Using the business relationship data obtained from the Gao algorithm, along with the BGP RIB data from Route Views to determine prefix origins, we simulated BGP announcements over the derived topology. We assumed that providers always advertise information to their customers, that customers do not advertise information to their providers (unless they are originating a prefix), that peers only advertise their customer learned routes, and that sibling ASes advertise all their information to each other. Figure 2.9 shows the results. This CDF plot has three key areas:

- *The No-Savings Zone.* The first area contains all the single-homed ASes mentioned earlier, which never have any alternate routes. This explains the large vertical offset of 30% in the graph. We also see that there are a few Autonomous Systems that connect to multiple ASes, but have virtually no alternate routes to choose from.

- *The Savings Zone.* The second area consists of the large spike from 2 to 20 on the x-axis. The immediate jump at 2 represents multi-homed ASes, which have alternate routing information for backup connectivity. Onward until 20 we see a spectrum of values, representing other multi-homed ASes and providers.

- *The Overestimated Zone.* After 20 we see that there are a small number of ASes which appear to have extreme numbers of alternate routes. This part of the graph, from 20 onward on the x-axis, is actually an artifact of our simplistic assumptions about routing policies. Many of these ASes in this region of the graph connect to peering points, where hundreds of other organizations also connect. Using the RIPE [13] Whois database, we discovered that many of

AS	Looking Glass
AS 3333	9.0
AS 6395	1.7
AS 6461	5.7
AS 7018	1.2
AS 9607	4.0

Table 2.1: A comparison of the number of RIB entries per prefix.

these peers have extremely strict import and export policies to avoid the RIB table explosion as seen in the graph.

2.5.3 Discussion of Potential Savings

While the gains can vary from as low as nothing to as high as an order of magnitude, it is important to evaluate these gains in the context of the likelihood of deploying Forgetful Routing. *Single-homed stub ASes* only connect to one upstream provider and receive all routes from their provider; since they have no alternates, they derive no benefit from Forgetful Routing and would thus not deploy it. *Multi-homed stub ASes*, like Princeton, generally have low connectivity, but receive almost all routing information from every connection. Thus, these ASes may see a reduction in RIB size from 33% to 75%. These ASes are likely to have older routers with smaller amounts of fixed memory. Moreover, since their ISPs would most likely have the route-refresh capability, they could deploy Forgetful Routing and obtain these memory savings.

It is also important to note that, for stub networks, Forgetful Routing has no side-effects. Performance is unchanged, as well as convergence delay, as stubs never re-transmit routing information. For these networks, Forgetful Routing can truly be transparently deployed.

Transit providers generally have high connectivity, but do not always receive all routing information from every connection. Our analysis estimates that providers could see a reduction in RIB size of 75% to 95%. Moreover, if Forgetful Routing is deployed, incentives will then exist for increasing connectivity further, allowing for greater reliability without significant increases to memory usage. In fact, many of those ASes who peer with 100+ neighbors, but currently use extremely strict filters to avoid RIB table explosion, could use Forgetful Routing to regain the lost connectivity at a mere fraction of the memory cost.

It is important to note that providers may see additional gains not presented in this analysis if they run Forgetful Routing *inside* the AS. Many ASes use a popular variant of BGP called *iBGP* to internally store routes learned from their neighbors. Under iBGP, every router within a single AS has a BGP session with every other router and all information is shared between all routers, providing a global view of the network. However, iBGP's memory requirements grow very quickly; for n routers in a network, each one would need up to n times more memory. Several solutions have been proposed to scale back on the burdens iBGP introduces, with the most notable being route reflectors and AS Confederations [38, 132]. Route reflectors operate by dividing routers into two subsets: route reflectors and route reflector clients. Clients connect to reflectors and rely on them to re-advertise their routes, as well as to learn new routes [127]. However, reflectors make networks more vulnerable to point failures and can cause inconsistent routing decisions, persistent loops, and route oscillations [36, 61]. AS Confederations partition an AS into sub-ASes, where each sub-AS maintains a full mesh within itself, with BGP sessions the between border routers of each sub-AS. However, not only do AS Confederations make networks less robust, they can also lead to instances of sub-optimal routing [61]. With Forgetful Routing, it could be possible to maintain full iBGP connectivity without inflating the memory requirements.

The most important insight from our analysis can be summed up in one statement: having n neighbors does not always imply a factor of n savings. The type of network, such as multi-homed stub, provider, *etc.*, greatly affects the total amount of savings.

However, in most cases, our results indicate that significant savings are possible. It is difficult to say whether our estimates for memory savings are high or low. On the one hand, our topology is almost certainly incomplete, undercounting total savings. On the other hand, our inferred routing policies are liberal, overcounting total savings. Because of the difficult nature of inference, covert policies and routing data, *etc.*, this area is one of potential future work.

2.6 Related Work

CRIO [137] is a mechanism which uses IP tunneling to reduce the memory needed by BGP significantly, at the expense of longer paths. It operates by having tier-1 ISPs announce "virtual prefixes" that are very large (*e.g.*, /8's), and using tunnels to directly connect routers as they forward packets. However, CRIO requires a globally deployed system to operate and changes the way BGP functions. Our method is incrementally deployable on a per-router basis and can be implemented using the BGP route-refresh option and route filters. Additionally, CRIO sometimes reduces path diversity and increases path lengths, while our mechanism does not reduce the resilience and natural efficiency of BGP.

A similar endeavor is the atomized routing project by CAIDA [42]. Atomized routing groups prefixes together into units called 'atoms' if they share the same AS path in all their routes through any router in a network. These atoms are globally computed and used to route packets. Studies have found that up to 78% of memory can be saved using this technique. However, atomized routing has the same issues as tunneled-BGP—it is not incrementally deployable and it requires modification to BGP.

Draves *et al.* worked on a similar problem of reducing the Forwarding Information Base (FIB) size of routers [60]. The FIB stores forwarding information for each prefix and is usually installed on fast line cards, unlike the main memory used by the RIB. Draves *et al.* proposed an algorithm that constructs optimal trees to store this information, guaranteeing maximal FIB savings. This work, however, is vastly different from ours. It focuses on an entirely different memory system where all information is local and operates using prefix aggregation. Unlike the FIB, the RIB's information may need to propagate to others and thus cannot be aggregated.

Research on shifting the burden of memory and computation to centralized servers has been proposed by Caesar et al. in the development of a Routing Control Platform (RCP) [46]. The RCP acts as a central server that collects BGP data from all neighboring routes in the same AS. The RCP then has access to global information and can make global routing decisions. It sends forwarding information back to each router, moving the RIB memory burden and computation to a single machine. While this can allow for different compression mechanisms (such as storing all unique AS path tags across all routers in a single table, eliminating duplicates), such schemes have not been fully developed yet. It is also important to note that a system such as an RCP may run into memory problems itself if it attempts to store all routes for an entire AS. Thus, a system such as the RCP may benefit from schemes like Forgetful Routing that are modified for RCP to RCP communication.

Finally, earlier work on the EIGRP protocol [4] is related to Forgetful Routing. EIGRP uses distance-vector routing along with a query/reply system to ensure loop-free routing. For every route, each router stores each neighbor's metric associated with that route. The "best" route chosen is the one with the lowest metric. Whenever that route becomes unavailable, the router checks if an alternate exists with the same metric, and if so, uses it. Otherwise, it propagates a query message to all its neighbors and recomputes its best route after it has heard back from all of them. Although EIGRP has a similar "refresh" mechanism like forgetful routing, it has two distinct differences. First, since EIGRP is a distance-vector protocol, it suffers from the "count to infinity" problem that must be resolved by setting a maximum hop count. Second, unlike forgetful routing's refresh mechanism, EIGRP's refreshes are executed *in series* rather than *in parallel*, resulting in convergence delays that increase linearly with network size.

2.7 Conclusion

Today's routers are susceptible to a wide range of memory problems. When a routing table overflows, the router may enter unspecified behavior, including freezing, rejecting new routes, or entering into an infinite loop. It is not always possible to upgrade router memory, due either to costs or limited access to the machines. Furthermore, best common practices used by network operators only partially protect routers from memory overflow, as they must often sacrifice safety for operability.

Our proposal of forgetful routing offers a unique solution to the problem without modification to BGP. As memory is needed, alternate routes are forgotten and requested back as needed. Since best routes are never forgotten, and an alternate route is some neighbor's best route, the route refresh option in BGP can be used to help implement this mechanism. An offline analysis of BGP data shows that there is much memory that can be saved; most alternate routes are never used. This analysis motivated the development of an online algorithm that runs in constant time for each message and approximates the optimal solution well. We believe that our scheme can significantly improve the scalability and robustness of IP routers in the future.

Chapter 3

Practical Network-wide Compression of FIB Tables

The tremendous growth of the Internet has also led to increased FIB memory requirements, resulting in increased costs. In this chapter, we push the boundaries of current backwards compatible solutions and demonstrate that they can be much more efficient and deployable than previously thought. Building on previous work, we develop a *Memory Management System* (MMS) that reduces the number of FIB entries by up to 65%, offers greater flexibility in route selection, and runs in real time. The MMS can be deployed locally on each router or centrally on a route server. The system can operate transparently, without requiring re-configuration in other ASes. Our memory manager extends router lifetimes up to six years, given current prefix growth trends.

The rest of the chapter proceeds as follows. Section 3.2 surveys Internet routing, the kinds of state kept at routers, and challenges in reducing that state. Section 3.3 covers our research addressing these issues, along with a description of the MMS deployed in local mode. We then discuss the setup of an AS-wide deployment of an MMS in Section 3.4. Section 3.5 presents results using a simulated MMS over real traces from tier 1 ISP routers. We discuss related work in Section 3.6 and conclude in Section 3.7.

3.1 Introduction

3.1.1 Router Memory and its Problems

The continued growth of the Internet over the last decade has resulted in larger state requirements for IP routers. As Autonomous Systems (ASes) have become further inter-connected, the number of possible routes has grown, causing the memory requirements to likewise increase. Moreover, these requirements are getting worse, due to increased deaggregation (advertising more-specific routes) arising from load balancing and security concerns [43, 126], the fact that routers run multiple routing protocols simultaneously (each with their own routing state), and increasing demand for Virtual Private Networks, which require multiple routing tables.

This memory growth is problematic for two reasons: safety concerns and increased costs. Routing protocols typically rely on adjacent routers maintaining synchronized state. When memory is exhausted, this synchrony breaks down, causing routing protocols to behave incorrectly [48]. In addition, increased IP table size requires more router memory, leading to additional expenses.

As previously mentioned, backwards compatible solutions are more likely to be deployed in practice than clean slate designs. Among the backwards compatible solutions, one piece of research stands out: Optimal Route Table Construction, also known as ORTC [60]. The ORTC algorithm operates only on FIB memory, taking a FIB as input and produces a FIB as output. It guarantees that the output FIB has the exact same forwarding behavior as the input, and that the output FIB has a provably minimal number of entries. Experimental tests have shown that it can reduce the number of FIB entries by up to 50%. Despite this benefit, ORTC has not been adopted in practice, as it suffers from several major drawbacks. First, it is computationally expensive (the original implementation takes approximately 500 milliseconds to run); if many routing updates are received each second, the algorithm cannot keep up with the workload. Moreover, it is inflexible; it must always produce an output that forwards exactly the same as the input. However, there may be times when even a "compressed" FIB will not fit in memory. In this case, it may be preferable to alter forwarding behavior if it can allow further compression. If these two problems were fixed, ORTC could be a useful building block in a larger system that managed memory.

3.1.2 The MMS as a Complete Solution

The focus of our research is to improve the ORTC algorithm to the point of where it is feasible to run in practice, and then to extend it to a generic *memory management system* (MMS) to manage the memory usage in the routers of an ISP's network. We apply several techniques to greatly boost the speed of the algorithm. Moreover, the MMS provides multiple levels of compression, allowing for a trade-off between unaltered routing and "maximal memory compression."

The MMS can be deployed either *locally* on each router or in a central system that monitors and compresses all routers in the *AS-wide* network. In local deployment, each router locally performs the operations of an MMS over its own routing state. This enables our system to run in a completely distributed fashion. However, this does have some drawbacks. It requires router software upgrades and possible hardware upgrades (if CPU power is lacking). Moreover, because each router has a local view of the network and acts independently, there are limitations to the potential amount of memory reduction.

To circumvent these problems, the MMS can also be deployed in an AS-wide setting, where it runs on a set of logically-centralized servers that collectively assume responsibility for the routing interaction of an AS with neighboring ASes [46]. The MMS receives routing updates from neighboring ASes, preprocesses these updates before sending routes to routers within the MMS-enabled network, and communicates selected routes to neighboring ASes. Neighboring ASes can be configured to send updates directly to the MMS, rather than to the border routers. If neighboring ASes do not wish to perform any re-configuration, border routers can act as proxies and forward messages between the ASes and the MMS. Not only does this deployment provide correctness and extra compression, but the logically-centralized approach allows for additional amortization techniques to be applied.

3.1.3 The Benefits of the MMS

Our design has several benefits:

Flexibility: By default, MMS operates in a transparent fashion, with absolutely no changes to the way routes are chosen and packets are forwarded. In this "transparent mode" an external observer cannot tell the MMS has even been deployed. In such a situation, the MMS can still provide about a 50% reduction in router memory across the entire network, without altering forwarding behavior. If more memory savings are desired, the MMS can shift paths to attain additional memory reduction, up to 65%. However, routes selected for forwarding may differ from the "transparent" case. We provide algorithms to automatically perform a small set of routing changes that increase compressibility without operator involvement. It is important to note that even if paths are shifted, the system remains inter-operable with routing protocols and does not introduce any routing loops.

Reduced Operational Cost: The MMS can simplify capacity planning and extends the lifetimes of older routers. We demonstrate this through experimental results conducted within a large tier-1 ISP backbone: memory usage can be reduced between 50% to 65%, the rate of increase of table growth is decreased by a factor of 2.2, and variation in table size is reduced by a factor of 2.6 (reducing variability increases the accuracy of future provisioning predictions). Given current levels of routing table growth [2], these reductions can be expected to increase lifetimes of routers needing immediate replacement by up to 6 years. Moreover, since the MMS can operate in the form of a logically-centralized cluster (or a small redundant set of clusters), it can form a single location where resources may be upgraded, reducing expenses associated with field deployment.

Safety: Routers near their memory limits can use the MMS to increase the amount of available resources. This improves resilience to misconfigurations in neighboring networks. Moreover, given that our compression techniques perform better with increased levels of deaggregation, our approach could enable interdomain routing on fully-deaggregated /24 prefixes, which has benefits in terms of routing flexibility and mitigating hijacking attacks.

Incrementally Deployable: A single ISP can deploy an MMS while maintaining interoperability with existing protocols and Autonomous Systems. Moreover, in AS-wide deployment mode, our MMS design requires no changes to existing router hardware. The MMS communicates routes to border routers using internal BGP (iBGP) sessions, and maintains external BGP (eBGP) sessions to neighboring domains on behalf of its border routers. Furthermore, this deployment may proceed in an incremental fashion, by having the MMS only control a limited subset of routers within the ISP. Finally, the MMS can be deployed and operate at its full capability without cooperation from neighboring ISPs.

3.2 Memory Saving Approaches and Limitations

The primary goal of the MMS is to reduce router memory usage within an ISP. To do this reduction, the MMS performs *route coalescing*, i.e., replacing groups of

```
TIME:               2007-12-23 09:51:11
TYPE:               BGP4MP/BGP4MP_MESSAGE AFI_IP
FROM:               12.0.1.63
TO:                 128.223.51.102
BGP PACKET TYPE:    UPDATE
ORIGIN:             Incomplete
AS_PATH:            7018 8151
NEXT_HOP:           12.0.1.63
COMMUNITIES:        7018:2000
ANNOUNCED:          200.94.228.0/22
ANNOUNCED:          200.94.224.0/22
```

Figure 3.1: BGP update message from Route Views on Dec 23, 2007, announcing that two prefixes are reachable through the same next-hop.

routes sharing the same next-hop with smaller, equivalent sets. Although this seems like a simple procedure, several operational complexities of ISPs make this process quite complex. In this section we describe the challenges in route coalescing through several examples. We show that naïve approaches can introduce inconsistencies in packet forwarding, and we motivate why our design decisions are necessary.

3.2.1 Routing across ISPs

BGP propagates routes for prefixes, which denote a collection of host addresses immediately adjacent in the IP namespace. Prefixes specify reachability on multiple levels of granularity, creating ambiguity in reachability information. For example, a route to 12.0.0.0/8 could have a next hop of 1.1.1.1, while a route to 12.0.0.0/9 could use 2.2.2.2. To eliminate this ambiguity, routers select the longest matching prefix when there are multiple choices. However, longest prefix matching significantly complicates aggregation, *i.e.*, the ability to take two prefixes with the same next hop information and combine them into a single, larger prefix. An example of such a complication with aggregation is shown in Figure 3.2. To avoid introducing such difficult-to-predict side effects, ISPs are constrained in the types of aggregation they can perform.

Although ISPs cannot aggregate advertised routes (RIB), they can aggregate forwarding entries (FIB). As previously shown, even if two prefixes have the same next-

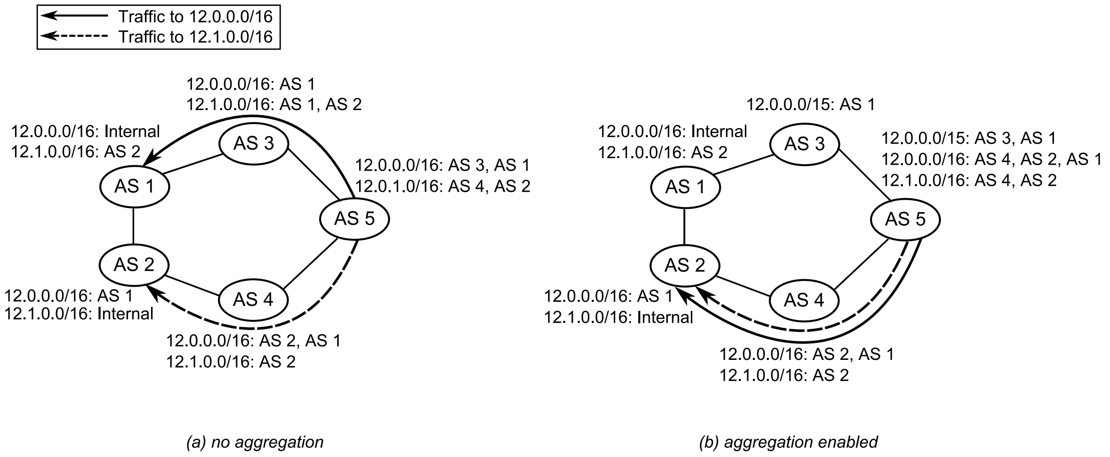


Figure 3.2: Aggregation can have unintended consequences: When no ASes perform aggregation, AS 5 can route traffic to 12.1.0.0/16 via AS 3, and traffic to 12.0.0.0/16 to AS 4. However, if AS 3 decides to aggregate 12.1.0.0/16 and 12.0.0.0/16 into 12.0.0.0/15, AS 5 can no longer use the route via AS3. The reason is that all of 12.0.0.0/15 is covered by more specific prefixes that are reachable via alternate exit points, and Internet routing always prefers more-specific prefixes.

hop, an ISP cannot announce an aggregate route, as it causes problems for other ASes. However, in the case of forwarding, there are no negative effects from such aggregation. Aggregating FIB entries is completely transparent to other routers; they still see the same router forwarding packets in the same manner. Moreover, if we choose routes from the RIB that have the same next-hop, we can aggregate these entries in the FIB. In other words, our choices of routes in the RIB will determine the compressibility of the FIB.

To summarize, *Autonomous Systems cannot advertise compressed routes.* While forwarding entries can be coalesced, routing entries cannot.

3.2.2 Routing within an ISP

ISP networks earn revenue by providing transit service, *i.e.*, by forwarding traffic between their neighbors. Hence, ISPs must share reachability information received from one neighbor with the others. This is often done by establishing BGP sessions between border routers (when BGP is run within an ISP, it is often referred to as iBGP). Internal reachability between border routers is provided by an intra-domain routing protocol such as OSPF [107] or IS-IS [110]. iBGP sessions are sometimes established in a full-mesh configuration, where each border router maintains a session to *every* other border router. However, since routers must maintain routing state separately for each iBGP session, full-mesh configurations have very large RIB memory requirements. For example, if there are n border routers, then each border router may need to store and maintain $n-1$ internal routes for each of the hundreds of thousands of prefixes in the routing table.

To circumvent this problem, larger networks often deploy route reflectors [38] at strategic locations within their network. Route reflectors act as internal accumulation points, which collect routing updates from a subset of border routers, and only advertise the most preferred route to their iBGP neighbors; as such, border routers only receive the most preferred routes from their associated route reflectors. Unfortunately, the use of route reflectors introduces a set of problems. They can induce persistent forwarding loops and oscillations if deployed improperly [37]. They require additional work for network operators to maintain, as they must be reconfigured to match changes in the underlying network topology. While route reflectors reduce memory usage, they do not reduce the number of prefixes in the routing table. Hence route reflectors do not reduce the size of the router's *forwarding* table (which is commonly stored in expensive, fast memory).

3.2.3 Router-Level Routing

Routers are logically divided into a *control plane* and a *data plane*. The goal of the control plane is to compute the set of routes the router should use locally, and of these, which should be advertised to neighboring routers. The goal of the data plane is to forward data packets, by selecting from a set of next-hops computed by the control plane. Routers maintain two key data structures corresponding to each of these two planes: a Routing Information Base (RIB) for the control plane, and a Forwarding Information Base (FIB) for the data plane.

The *RIB* stores the set of routes available to forward packets. The RIB is stored as a map from the set of advertised prefixes to the next hop which may be used to reach that prefix, along with additional state associated with the route (the AS-path, cost metrics, attributes, etc). Routers typically need to store several routes per prefix. This is done so that if the currently-used route fails, the router may use an alternative route through a different neighbor to circumvent the failure.

Unfortunately, when routers run out of memory, they can continuously reboot, crash, or begin behaving incorrectly [48]. Reducing RIB memory is quite difficult. RIB entries contain routing information that may be vital when primary links fail and backup routes are needed. Moreover, routing information is often exchanged between routers and used to determine forwarding paths. As such, care must be taken when attempting to reduce RIB memory – data cannot be simply discarded.

The other memory structure, the *FIB*, stores the set of routes which will be used to forward packets to individual prefixes. The FIB must perform forwarding lookups very fast, since it must forward data packets at high rates. Hence FIBs are typically implemented in fast memory with low access times, such as SRAM or TCAM. The contents of the FIB are populated by the RIB. Newer routers typically update the FIB immediately after RIB changes.

There are two restrictions regarding FIB memory reduction. First, if the router advertises it will forward packets a certain way, it must actually forward packets in that manner. Prefixes can be coalesced if such actions do not change the forwarding behavior advertised by the router. In other words, routing decisions in the RIB must be honored by forwarding decisions the FIB. Figure 3.3 provides an example. Second, FIB reduction techniques must be extremely fast. If an algorithm is too slow, a router may not be able to stop an ever growing backlog of updates that need processing.

3.3 The MMS in Local Deployment

There are fundamental problems with trying to compress routes: prefixes cannot be coalesced when announced, FIB compression is limited by RIB decisions, compression algorithms must be fast, *etc.* In this section, we discuss how our Memory Management System can circumvent some of these problems when deployed in “local mode” on individual routers. We demonstrate how the MMS can provide flexibility in aggregation for the FIB without introducing network problems. Moreover, we show how techniques such as parallelization and incremental computation can be used to significantly speed-up the ORTC algorithm, which is used as a building block for the MMS. The local mode MMS also serves as a basis for the AS-wide MMS, which is discussed in the next section.

Although the MMS can be used to reduce FIB memory consumption, the RIB cannot be easily compressed in “local mode.” A router may need backup routes and may need to advertise this information to other routers. As such, we focus on FIB compression. Later, during the discussion of “AS-wide mode,” we demonstrate how the RIB can be compressed.

3.3.1 A Fast FIB Compression Implementation

Draves *et al.* [60] previously proposed an Optimal Routing Table Construction (ORTC) algorithm, which takes a routing table as input, and computes the provably smallest [1] routing table that performs forwarding in an equivalent manner. Figure 3.4 outlines their algorithm, which assumes a binary tree representation known as a trie data structure. ORTC works by making three passes over the trie, in steps known as *normalization*, *prevalent hop set calculation*, and *next-hop selection*. The authors of [60] provide several optimizations to speed up this computation. They also extend the algorithm to deal with multiple next-hops per prefix and default routes.

[1]With respect to the number of prefix/next-hop pairs.

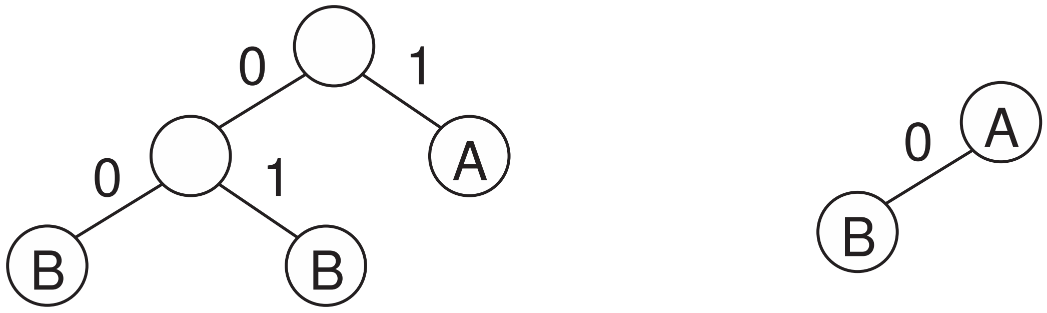


Figure 3.3: Example of prefix coalescing over tries, *i.e.*, binary tree representations of forwarding tables. The uncompressed FIB on the left represents prefix 128.0.0.0/1 mapping to next-hop A, and prefixes 0.0.0.0/2 and 64.0.0.0/2 mapping to B. The FIB trie on the right forwards in the same manner, although it only has two prefixes: 0.0.0.0/0 mapping to A and 0.0.0.0/1 mapping to B.

Unfortunately, even with optimizations, ORTC is too slow to use in real time. While the authors were able to optimize run time down to several hundred milliseconds for the smaller routing tables that existed when their paper was published, these run times remain too slow. Core routers commonly process tens of updates per second and bursts up to thousands of updates per second [106, 24]. We leverage the techniques of parallelization and incremental updates to augment this algorithm, speeding it up so it can be used with the MMS.

Parallelization

Parallel algorithms are becoming increasingly important as chip manufacturers move to multi-core designs. Conventional wisdom is now to double the number of cores on a chip with each silicon generation, while the speed of each core grows much more slowly or remains constant [33]. Such trends in processor design can be exploited to help compression algorithms keep pace with the increased computational load associated with the growth and churn of Internet routing tables.

There has been substantial previous work on parallel algorithms for graph structures [116]. Our design is loosely based on these techniques and consists of two stages. In the first stage, all nodes associated with /8 prefixes are added to a queue. When a thread becomes available to perform work, it selects a node from the head of the queue, and performs compression on the sub-trie rooted at that node. To ensure correctness, it is important that no other threads concurrently process any nodes in that sub-trie. As a result, a thread locks all descendants of that node. In the second stage, a single thread performs the rest of the remaining compression for the nodes that have not been processed. Note that the second stage could be parallelized as well to further decrease computational time.

Incremental Computation

The ability to incrementally update data structures is crucial for speed. The benefit of an incremental approach is that changes to a single prefix do not require recomputing the router's entire FIB. However, with ORTC compression, this is no longer true – changing a single prefix may trigger other routes to become coalesced (or to deaggregate). The naïve way to deal with this would be to rerun ORTC after every received update. However, doing this would be wasteful, as the vast majority of routes would not change after a particular update is received. Furthermore, some updates do not require any recomputation (for example, an update that removes a route that is not used by any routers in the network).

To improve processing time, traditional ORTC works by periodically processing batches of updates at fixed intervals. However, such an approach increases the time needed before a router can respond to a change in the network. To deal with this, we developed an incremental algorithm (Figure 3.5), that only processes the portion of the ORTC trie that is affected by the received update. When a routing update for a prefix is received, the algorithm looks up the corresponding node and recurses up and down the trie, stopping whenever it determines that no more changes are needed.

For example, consider a trie with two (prefix, next-hop) pairs: $(0.0.0.0/0, 1.1.1.1)$, and $(0.0.0.0/1, 1.1.1.1)$. This trie can be compressed to a single (prefix, next-hop) entry: $(0.0.0.0/0, 1.1.1.1)$. Now consider the announcement of a new route: $(128.0.0.0/1, 2.2.2.2)$. Adding this new route does not change forwarding behavior of 0.0.0.0/1. The forwarding behavior does change for 0.0.0.0/0, though, and it (along with new nodes) will need to be re-evaluated for compression gains. 0.0.0.0/1, however, can be unaware of any such computations, as long as 0.0.0.0/0 still "covers" it by forwarding toward 0.0.0.0. The new compressed trie, $(0.0.0.0/0, 1.1.1.1)$, $(255.0.0.0/1, 2.2.2.2)$, is optimal and does not require a full computation.

While parallelization and incremental updates could be combined, it is unclear how well they would work together. Parallel algorithms work best with large sets of data that can be processed independently, while the incremental algorithm attempts to operate over small sets of nodes that may have inter-dependencies. As such, the overhead from thread locks may outweigh the amount of possible parallelization savings.

Thus, we consider these techniques to be complimentary and useful for different situations. For example, if a router suddenly switched compression on, a parallelized full computation would be faster than the incremental algorithm (since a full computation is needed anyway, and a parallelized version has additional opportunity for speedups). However, if a router is simply processing updates received in normal BGP communication, the incremental version would most likely be the fastest algorithm to use.

3.3.2 Selecting Routes to Improve Compression

Given this implementation, there is still a problem of flexibility. As previously mentioned, the RIB is responsible for all routing decisions. The RIB uses the BGP decision process (Figure 3.6), which provides a way to select a single route for forwarding from among many possible choices. It uses a series of rules to pick routes. Each rule eliminates some set of routes, and rules are applied until a single route

```
Maximally compressed binary trie t:

// Normalization: all nodes to have 0 or 2 children.
for node N in t in preorder traversal:
   if N has one child:
      create missing child for N
      child inherits N.rib_info

// Prevalent hop calculation: find the set of
// maximally aggregated next-hops.
for node N in t in postorder traversal:
   if N has no children:
      N.prev_set = {N.rib_info}
   else:
      N.prev_set is intersection of its children's prev_sets
      if N.prev_set == ∅:
         N.prev_set is the union of its children's prev_sets

// Next-hop selection.
for node N in t in preorder traversal:
   if N is root of t:
      N.next_hop = arbitrary element of N.prev_set
   else:
      clst = closest ancestor of N with non-NULL next-hop
      if clst.next_hop ∈ N.prev_set:
         N.next_hop = NULL
      else:
         N.next_hop = arbitrary elem in N.prev_set
```

Figure 3.4: Pseudo-code for the ORTC algorithm. Each node represents a different prefix. `rib_info` represents the chosen route for a prefix (as dictated by the RIB). NULL next-hop indicates no FIB entry needed for that prefix.

remains. The process is designed around business goals, such as maximizing revenue (through local preference settings), attempting to minimize latency (through shortest AS paths), load balancing (through IGP metrics), and so on. The last rule is an exception, though, and exists simply to pick a "winner" if there are multiple routes at the last stage.

The BGP decision process is very important because it dictates the routes that will be installed into the FIB (and limits the amount of memory reduction that is possible). The FIB blindly accepts the results of the BGP decision process, even if

```
Incrementally update a trie t with an update u from neighboring
router u.neighbor:

// Update the node with the new routing information.
N = node in t associated with u.prefix
N.rib_info -= {old next_hop of u.neighbor}
if u is an announcement:   N.rib_info += {u.next_hop}

// Normalize all affected children of N. Same as normalization in original
// ORTC, except if a node is not modified by normalization, do not
// normalize the children of that node.
mod_normalize(sub-trie rooted at N)

// Calculate the prevalent hop set. Same as the prevalent hop calculation
// in original ORTC, except if a node N has no children, set
// N.prev_set = N.rib_info
mod_calc_prev_set(sub-trie rooted at N)

// Normalize all affected ancestors of N. Ancestors are processed in
// ascending order. If a node was not modified by normalization, do not
// normalize its ancestors. Update ‘highest’ variable to reference the
// highest ancestor normalized.
highest = N
mod_ancestor_normalize(N, highest)

// Compute new next-hops as needed. Same as next-hop selection in original
// ORTC, except if the next_hop of a node is unchanged, do not process that
// node’s children.
mod_select_next_hop(sub-trie rooted at highest)
```

Figure 3.5: Pseudo-code for the incremental update algorithm. **rib_info** now represents the set of routes passed to a prefix from the RIB.

there are alternative routes that are *nearly as good* and provide better compression. If a router allowed for more flexibility in route selection, it could achieve more memory gain.

The MMS allows for such flexibility. Instead of having the RIB decide on a single route per prefix, the MMS allows the RIB to decide on a set of routes that are acceptable for use. Based on all these sets per prefix, the MMS can now pick routes that coalesce. In particular, an operator configures the MMS with a *threshold level*. The threshold level determines how many steps of the BGP decision process to execute. All routes that are equally good at a particular level are considered possible routes for the FIB. A route coalescing algorithm is then computed over these possibilities. For example, a “level 0” setting would not run any steps of the decision process, and use all possible routes in the coalescing algorithm. A “level 1” setting would select all routes that remain after applying step 1 of the decision process; a “level 2” setting would select all routes that remain after applying steps 1 and steps 2; and so on.

It is important to note that such flexibility requires the use of tunnels between border routers. Without tunnels, packets may be forwarded in a different manner than expected. For example, consider the network depicted in Figure 3.7. Routers A and B both use their external links to reach 1.2.0.0/16. It is possible for router D to choose A for forwarding to this prefix, while router E choose B. However, both D and E must go through C. If C decides to forward traffic to 1.2.0.0/16 through A, then router E's choice is invalidated. Since BGP specifies a single next-hop for a given prefix, this problem is unavoidable. To overcome it, tunnels between border routers can be used. Tunnels have the additional benefit of freeing memory in the core of the network. Such “BGP-free cores” have been deployed in practice (*e.g.*, through MPLS [121] tunnels) and are feasible to implement.

Flexibility in route selection may cause divergence from the original forwarding behavior; such deviation may be tolerable in a number of circumstances. Higher levels of compressibility may be achieved given. Moreover, the amount of divergence can be tweaked, offering more divergence and savings in some situations and less divergence and savings in others. In addition, this approach can be used as a fallback mechanism if the level 7 (*i.e.*, no divergence) compressed routing table size would exceed router capacity.

However, deviating from the BGP decision process can lead to routing loops or oscillation. These problems can occur at either the inter-AS level or intra-AS level.

$$\begin{array}{rl} \textit{Router Invariant} & \left\{\begin{array}{l} 1.\ \texttt{Highest local pref} \\ 2.\ \texttt{Shortest AS path} \\ 3.\ \texttt{Lowest origin type} \\ 4.\ \texttt{Lowest MED} \end{array}\right. \\ \textit{Router Specific} & \left\{\begin{array}{l} 5.\ \texttt{eBGP over iBGP learned} \\ 6.\ \texttt{Lowest IGP metric} \\ 7.\ \texttt{Lowest router ID} \end{array}\right. \end{array}$$

Figure 3.6: The BGP Decision Process.

Luckily, due to the relationships that exist between Autonomous Systems, such problems can be avoided at the inter-AS level as long as step 1 of the BGP decision process is always applied. This is because step 1 is primarily used by ISPs to encode relationships, with customers often receiving higher local preference values than peers, and peers receiving higher local preference values over providers. As long as people are routing according to economic incentives, loops and oscillations should not happen [73].

However, intra-AS loops and oscillations can occur if we are not careful and routers act independently. For example, imagine two routers in a network that both have external routes to the same set of prefixes, as well as routes to each other. If enough BGP decision steps are ignored, router A might decide that it should simply forward everything to B. Likewise, B might decide to forward everything to A, resulting in a loop. To solve this problem, each MMS should be configured with a threshold value of at least 5 (eBGP preferred over iBGP). This configuration prevents loops because, due to the dynamics of iBGP, iBGP learned routes always point to a router that has an eBGP learned route. Thus, all routers in a network fall into one of two cases:

- *The router has at least one "equally good" eBGP-learned route.* In this case, the MMS forces the router to pick an external route, preventing intra-AS problems.

- *The router's set of "equally good" routes are all iBGP-learned.* Using any of these routes will send packets to a router that has at least one eBGP-learned

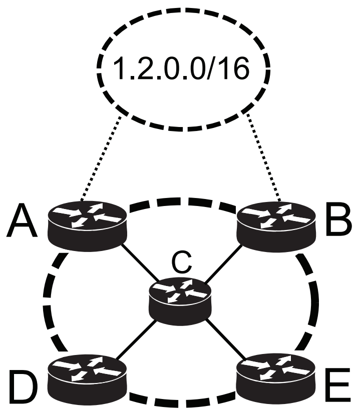


Figure 3.7: An example network with four border routers and one internal router. Dashed lines represent AS boundaries, solid lines indicate links, and dotted lines represent paths to the AS that owns 1.2.0.0/16.

route. Once these packets arrive at these border routers with eBGP routes, they are sent off the network, as in the previous case.

In summary, as long as the threshold level drops no lower than 5, both inter- and intra-AS problems can be avoided. Moreover, the increased flexibility can lead to better compression.

3.4 AS-Wide Deployment of the MMS

Alternatively, the MMS scheme may be deployed in a centralized fashion. The overall architecture of an AS-wide MMS is shown in Figure 3.8. It can be implemented through a logically-centralized architecture which offloads memory management functionality to a small set of servers. These servers are completely responsible for disseminating routing information to routers within the ISP. This design draws on previous work on the Routing Control Platform (RCP) [46]. The MMS directly maintains peering sessions with neighboring ASes, offloading the responsibility from its associated border routers. The Memory Management System has a network-wide view and knows the routing preferences of all border routers. Thus, the MMS can locally maintain a routing table on behalf of each BGP-speaking router in the network. The

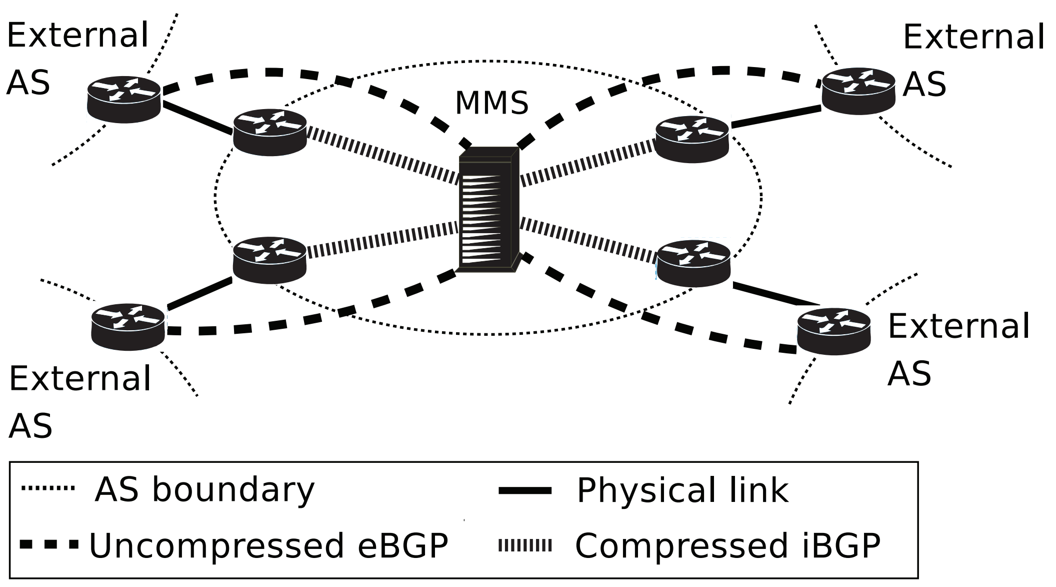


Figure 3.8: Solid lines represent physical connections, while dotted lines represent virtual connections. Border routers at other autonomous systems speak directly to the MMS, which is permissible as BGP is run over the TCP/IP protocol. The MMS then sends coalesced information to its own routers for forwarding.

MMS can compress the routes and send the compressed information to the border routers (while sending the uncompressed information to other autonomous systems).

This approach has several benefits. First, our centralized approach offloads computation from routers, freeing up computational resources for other protocols or for speeding convergence. Second, our approach requires minimal changes to existing routers. Third, common computations across routing tables could be amortized to yield further computational saving.

In a centralized deployment mode, the MMS should be replicated to improve fault tolerance. We use an approach similar to the RCP, having one server act as a primary in charge of distributing routes throughout the network, with the rest of servers acting as failovers.

3.4.1 Compressing FIB Entries

In AS-wide deployment mode, the MMS can obtain all the FIB compression benefits from the local deployment mode. In its simplest setting, the AS-wide MMS can run an instance of a local MMS for each router, performing all the computation on behalf of the routers. Moreover, because the AS-wide MMS has a complete view of the network, it can avoid the problem of routing loops caused by incomplete routing information. The MMS can dictate all forwarding decisions such that no routing loops occur.

In addition, the MMS can leverage an *amortization* speed boost when the threshold level is set below 5. Below level 5, all routers with the same routing information will make the exact same decisions regarding "equally good" routing sets. This phenomenon occurs because the first four BGP decision process steps are always the same for every router, if given the same set of routes. However, not every router will have the same set. For example, if router 1.1.1.1 has an eBGP learned route r with next-hop 2.2.2.2 and advertises it, all other routers will see r as having next-hop 1.1.1.1. However, if routers share similar sets, the computations can be amortized.

To efficiently compute the FIBs from the RIBs in an amortized fashion, the MMS first compute sets of routes that are equally good according to the first N steps of the BGP decision process, where N is the threshold level. All routers in the network must select a route from this set. A smaller computation is then done to further select routes on a per-router basis (that deviate from the 'common' case). This approach consists of two separate stages:

Stage 1, compute common FIB: First, the MMS computes a compressed FIB that all routers in the network share. In particular, the MMS logically creates a virtual *internal* router, which receives all routes from every border router in the network. The MMS then constructs a compressed FIB for this router.

Threshold Level	FIB Entries	% of original size
1	110890	35.6%
2	119150	38.3%
7	130842	42.0%
Uncompressed	311466	100.0%

Table 3.1: Tier-1 results for a single router on a single day. Results were similar for other routers in the network.

Stage 2, compute router-specific differences: At first glance, it appears that every router in the network should use the common FIB computed in stage one. However, this is not the case. For example, consider a network that picks next-hop 1.1.1.1 for prefix p. If 1.1.1.1 is a border-router in the network, then everyone can route successfully *except for 1.1.1.1.* Since 1.1.1.1's forwarding table would state that the next-hop is 1.1.1.1, the router would forward packets to itself. To avoid this scenario, the MMS computes (on behalf of 1.1.1.1) which one of 1.1.1.1's outgoing links is best suited for forwarding traffic to p, and sends that information to 1.1.1.1.

3.4.2 Compressing RIB Entries

The AS-wide MMS has the opportunity to reduce the amount of redundant routing state in a network. For example, every time a route is announced and propagated, it may be stored on every router that receives it. Individually, each router may not be able to remove RIB entries, since it may need to transmit the information to ASes; thus, reducing the redundancy may be difficult. However, the MMS can act as a central database to store all such routes. Only one copy of the route need be stored in this case.

Moreover, if the AS-wide MMS is responsible for routing advertisement, prefixes can be coalesced and supernetted for *both* the RIB and the FIB. Since routers are no longer advertising information, they can compress their RIBs through the same mechanism that FIBs are compressed. Attributes can be completely stripped (except for prefix and next-hop information), as the MMS would retain an original copy. As such, the amount of RIB memory needed on border routers can be reduced dramatically.

In the AS-wide deployment mode, the MMS offers greater FIB compressibility and more opportunity to reduce RIB memory requirements.

3.5 Evaluation

Data used to evaluate the MMS comes from a tier-1 ISP's BGP feeds from January of 2008 to June of 2008. These feeds are input into a simulation of the MMS. The data represent the actual loads seen by that ISP's border routers, and thus these numbers reflect a system running under "tier-1 stress."

3.5.1 Compression Ratio

Figure 3.1 shows compression achieved across a router within the ISP. Here, the compression techniques were run over a routing table snapshot that was collected on June 1, 2008. The compression gains of this routing table snapshot were compared with other routers in the network; no significance difference was seen. Moreover, the router's compression gains were studied over a two month period from April 15, 2008 to June 15, 2008; no significant variance was seen. As such, the data from Figure 3.1 can be considered representative. With Level 7 compression, routing table size is 42% of its original size, without causing any changes in forwarding behavior. Lower levels increase compression.

Route Views traces indicate that the number of more-specific prefixes is increasing at a faster rate than less-specific prefixes, as shown in Figure 3.9. For example, between January 1, 2002 and June 1, 2008, /24s were the fastest growing segment, in terms of number of prefixes.

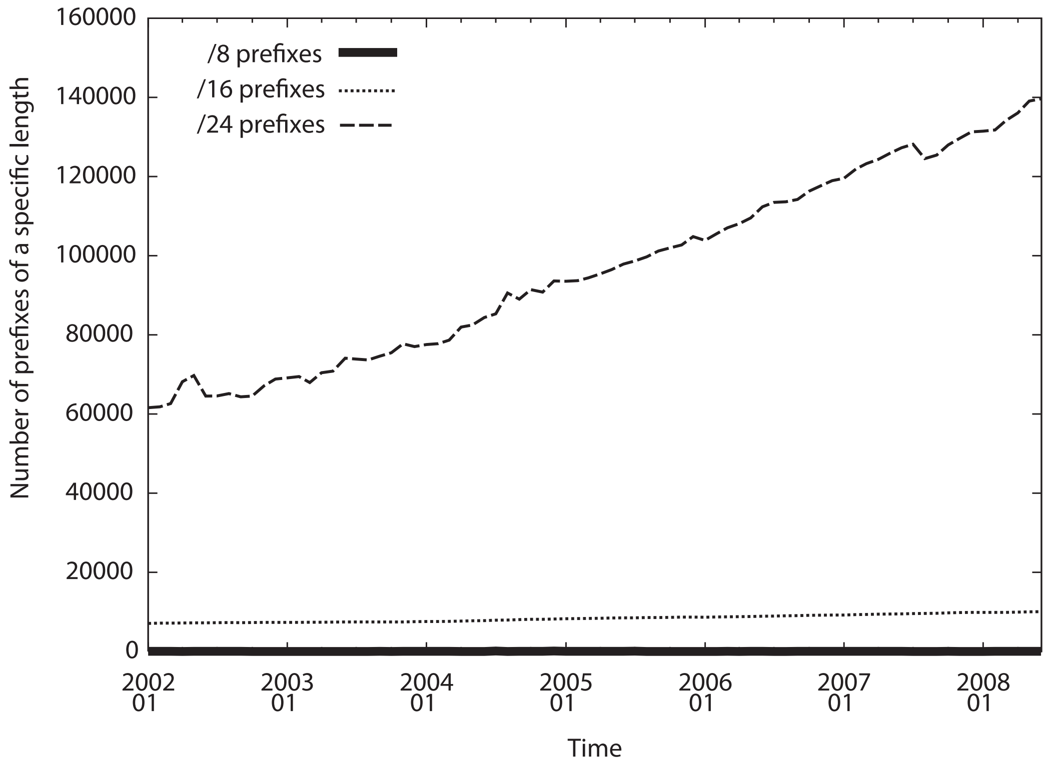


Figure 3.9: Growth rate of three different prefix lengths, as observed from Route Views.

3.5.2 Runtime

Figure 3.10 shows the speed up results from parallelization. In this experiment, a single threaded version of ORTC was run, and timing information was recorded for processing each node. These results were fed to a simulator that simulated a multithreaded version of the ORTC algorithm. The simulator used the results from the single threaded run to estimate the time that each thread spends when it processes a node. Based on these results, significant speed up can occur. However, after about 20 threads, speed up becomes negligible. It is important to note that a speedup of up to a factor of 8 (with approximately 20 threads) is significant.

Figure 3.11 demonstrates the benefits from incremental computation, *i.e.*, only recomputing the portion of the routing table that is affected by a received update. The figure shows a time-series plot of update processing time, for both the incremental

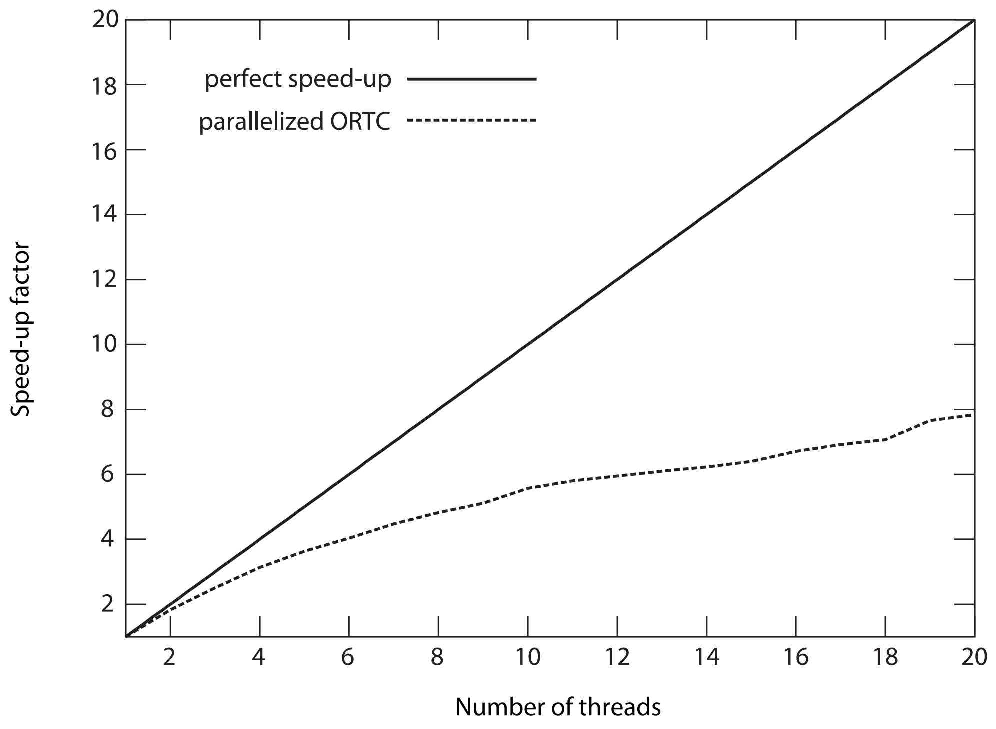


Figure 3.10: Parallelization and its effects on speed up.

algorithm and the traditional non-incremental ORTC algorithm. The incremental computation significantly improves update processing time, both in terms of absolute magnitude, as well as in terms of absolute variance. For example, over the one month period from June 15, 2008, to July 15, 2008, the computation time decreased by a factor of 9344 on average.

3.6 Related Work

Improving network scalability by reducing memory usage has been widely studied in previous work. Hierarchical routing [107, 90], landmark routing [133, 68], and geographic routing [98] embed topological information in addresses, so as to reduce the number of routes required to be stored at routers. Alternatively, DHTs [129] work by reducing the number of routes maintained by each participant in the system.

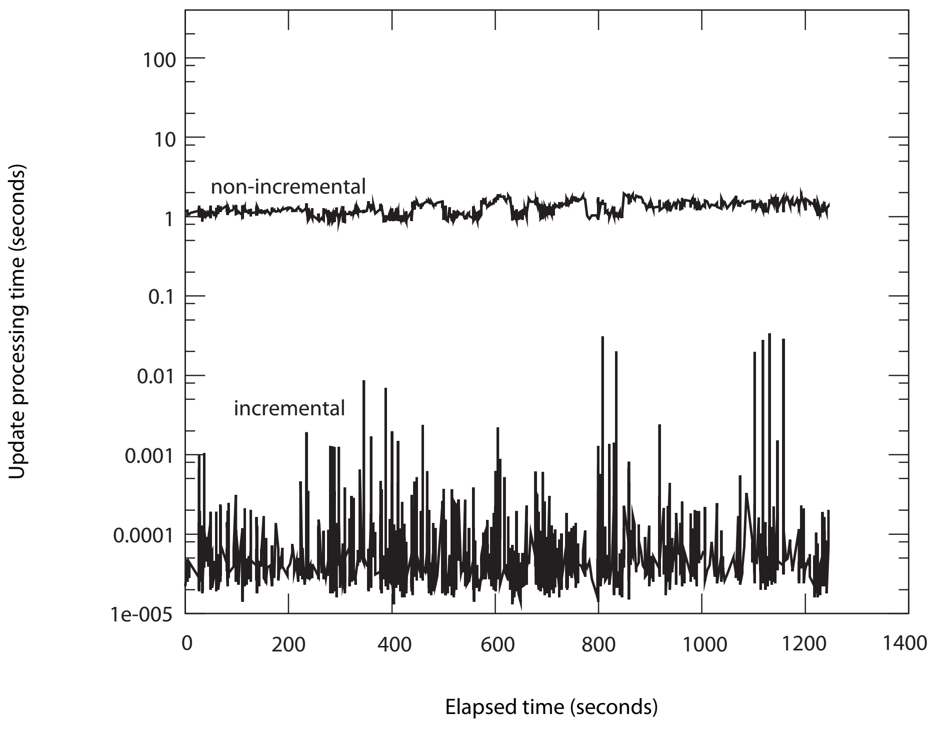


Figure 3.11: Incremental computation speed up.

These techniques are more formally studied by *compact routing* [131], which provides theoretical bounds on the amount of memory that can be saved for a given degree of suboptimal routing. Commonly, such work focuses on minimizing routing table size or control overhead while bounding path inflation. The MMS architecture differs from previous work in these areas, in that it aims to operate within the confines of existing IP routing protocols, rather than replacing them.

One practical method that can be used today (leveraging existing technologies) is to use MPLS tunneling in an ISP's core, while deploying route reflectors to exchange routes amongst the edge routers. While this helps the memory requirements for both border routers and ISP's cores, it does require the deployment of route reflectors. As previously mentioned in Section 3.2, route reflectors have their own set of problems and limitations.

There has been other recent research in reducing memory consumption while remaining backwards compatible. The ViAggre [34] work demonstrates how to routers can be reconfigured to store a fraction of the routing table. The memory on each router can be significantly reduced, although the route a packet takes may be suboptimal. Another piece of work known as Route Relaying [96] demonstrated a similar technique in the VPN setting, where edge routers forward to traffic to a collection of "hub" nodes that store the full routing table. However, it is worth noting that such deflection techniques can interfere with traffic engineering. In contrast, the MMS can be configured to use IGP weights in the decision process, which are typically used for traffic engineering.

The Routing Group (RRG) has also explored scalability issues with respect to memory [14]. In particular, the work on Locator / ID splitting (LISP) has gathered attention, where the IP address space is divided into separate spaces for end-hosts and for organizations. Substantial memory savings are possible under this scheme [117]. However, this scheme has a deployment problem. A single ISP cannot deploy it and realize the savings unless at least one other ISP cooperates, due to its use of IP-in-IP tunneling. While it may be considered "incrementally deployable" from the perspective that it builds on top of existing infrastructure, it does require some coordination between ASes. As such, we consider this work complementary to the MMS.

There has also been work on several technologies that enable the MMS design. The Routing Control Platform (RCP) [46, 134] provided an architecture for logically centralizing route selection within an RCP. The prototypes in [46, 134, 136] demonstrated that this architecture can scale to the size of a large, tier-1 ISP. The RCP aimed to compute and distribute BGP routes to routers within an ISP, and did not aim to reduce table sizes at routers. However, the MMS algorithms may be deployed on top of an RCP-like infrastructure.

Other related work includes Verkaik *et. al.*'s BGP Atoms [42], and Draves *et. al.*'s Optimal Routing Table Constructor [60]. BGP Atoms can be used to reduce memory overhead by clustering prefixes based on policy, rather than supernets. Forgetful Routing enables routers to distributively share their RIBs, reducing redundancy in a network. The work by Draves *et. al.* served as a primary inspiration for our work. The algorithmic contributions, architecture, and deployment strategies used in the MMS can be viewed as a way to make ORTC practical in a modern-day network environment. The MMS study also measures compression benefits over a range of topologies and environments, including a large Tier-1 ISP.

3.7 Conclusions

Deploying an MMS within an ISP has several benefits. An MMS can prevent router memory requirements from exceeding capacity, as well as extend the lifetime of routers. Moreover, experimental results show substantial reduction of routers' FIBs. Reducing these requirements and safely preventing routers from becoming overloaded reduces the need to upgrade them as often, decreasing operational costs and administrative work. The MMS is designed to be practical and even amenable to partial deployment.

For future work, several items may be interesting to investigate. While the threshold levels are assumed to be fairly static, a fully-automated "adaptive mode" could be developed; the algorithm would automatically adjust the threshold level to stay within memory bounds while deviating from the BGP decision process as little as possible. Additional savings might be possible by developing protocols to perform memory management across ISPs. Finally, if memory is still scarce after compression, the memory management system could be used to selectively filter less popular routes to ensure that the most popular ones remain available.

Chapter 4

Quantifying the Extent of IPv6 Deployment

The previously mentioned solutions have been specific to BGP and were evaluated in the IPv4 Internet. Despite the dominance of these two protocols, there are indications that other protocols may soon challenge them. Unfortunately, because upcoming protocols are so new, it is very difficult to guess their eventual effects. Instead of extrapolating too far into the future, the next two chapters examine current deployment and usage of two emerging protocols. This information helps lay the groundwork for future research into these areas. In this chapter, we specifically look at the IPv6 protocol, designed to replace IPv4.

Our understanding of IPv6 deployment is surprisingly limited. In fact, it is not even clear *how* we should quantify IPv6 deployment. A variety of data is analyzed to characterize the penetration of IPv6. It shows that each analysis leads to somewhat different conclusions. For example: registry data shows IPv6 address allocations are growing rapidly, yet BGP table dumps indicate many addresses are either never announced or announced long after allocation; Netflow records from a tier-1 ISP show growth in native IPv6 traffic, but deeper analysis reveals most of the traffic is

DNS queries and ICMP packets; a more detailed inspection of tunneled IPv6 traffic uncovers many packets exchanged between IPv4-speaking hosts (e.g., to traverse NAT boxes). Overall, the study suggests that (from these vantage points) current IPv6 deployment appears somewhat experimental and that the growth of IPv6 allocations, routing announcements, and traffic volume probably indicate more operators and users are preparing themselves for the transition to IPv6.

The chapter is organized as follows. Section 4.2 examines IPv6 address block allocation and compares it against an analysis of address block announcements. Section 4.3 looks at both native and tunneled IPv6, analyzing the types of technologies used to enable IPv6 communication and the application mix. We discuss related work in Section 4.4, and conclude with Section 4.5.

The IPv6 study has been previously published in the proceedings of the Passive Active Measurement Conference 2009, under the title *Quantifying the Extent of IPv6 Deployment*.

4.1 Introduction

IPv4, the current Internet protocol, is showing its age. Addresses are becoming scarce, with estimates of exhaustion within the next several years [11]. People are looking toward IPv6, with its 2^{128} possible addresses, as the solution. While there has been pressure to deploy IPv6, NAT technologies have extended IPv4's life. Given the lack of urgency to upgrade, coupled with the administrative and financial overhead of becoming IPv6-enabled, it is difficult to say whether we have moved any closer to a day when IPv6 is so dominant that IPv4 can be "switched off."

Not only has IPv6 deployment has been slower than expected, but our understanding of it is surprisingly limited as well. Questions such as, "are organizations actually using IPv6," "what IPv6 transitional technologies are being used," and, "what applications are using IPv6" remain largely unanswered.

To answer these questions, we looked at a variety of data sources, ranging from regional Internet registry (RIR) logs to BGP dumps to Netflow records to packet header traces. Along the way, we found several "gotchas," where the surface level analysis implies a different conclusion from the in-depth analysis. For example:

- RIR data indicates that IPv6 prefixes are being allocated at near exponential rates, implying strong growth. However, longitudinal BGP analysis shows that nearly half of these allocated prefixes are never announced, and the remainder takes an average of 173 days to appear in the global routing system. In other words, many people are acting as IPv6 speculators but not deployers.

- Native IPv6 traffic analysis of the enterprise customers of a US tier-1 ISP shows considerable volume, yet most of the traffic is generated by DNS and ICMP; this indicates a dearth of real IPv6 applications.

- A reasonable amount of tunneled IPv6 traffic is already observed on a US broadband ISP (0.001% of total traffic). However, further analysis indicates that much traffic is between IPv4 clients, implying that IPv6 tunneling technologies are primarily used to circumvent NAT and firewall restrictions.

4.2 Allocation and Announcement of IPv6 Address Blocks

RIR and BGP data are important for understanding how IPv6 addresses are allocated and announced, respectively.

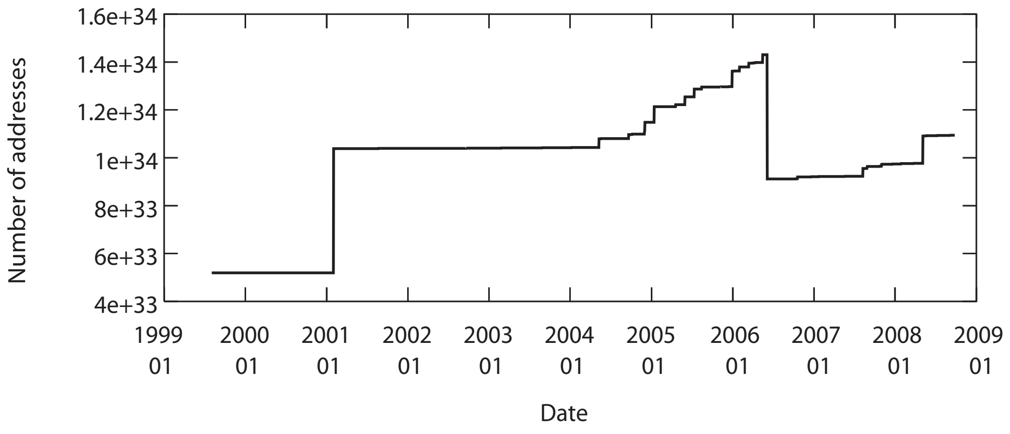


Figure 4.1: Number of IPv6 addresses allocated by RIRs. Each prefix is completely deaggregated.

4.2.1 Data Sources

RIR allocations are important because they indicate the number of institutions requesting blocks, as well the sizes being allocated. For our analysis of IPv6 allocations, we used the ARIN public FTP repository [1] that maintains information about the five regional registries responsible for allocating IPv6 address blocks: APNIC, RIPE, ARIN, LACNIC, and AFRINIC. Date ranges for the different repositories are: 1999-8-13 to 2008-9-25 (APNIC); 1999-8-12 to 2008-9-26 (RIPE); 1999-8-03 to 2008-9-23 (ARIN); 2003-1-10 to 2008-9-22 (LACNIC); and 2004-12-14 to 2008-9-23 (AFRINIC).

In order to analyze how address blocks are announced, we used the Route Views BGP data archives [106]. We collected routing table (RIB) snapshots at approximately 6 hour intervals from this web site. The BGP data obtained from Route Views starts on 2003-5-3 and ends on 2008-9-28.

4.2.2 Why Address Allocation Statistics are Misleading

Looking at the distribution of allocated prefixes, along with the total number of addresses allocated, seems like a reasonable method for quantifying the extent of IPv6 deployment; however, we find that using such information can be misleading.

First, one can incorrectly conclude that *IPv6 address allocations are very volatile*, as seen by the gigantic spike and dip in the curve. Figure 4.1 shows the total number of IPv6 addresses assigned by the RIRs. We count the number of allocated addresses by deaggregating all prefixes into their corresponding sets of IPv6 addresses and taking the union of all such sets. A couple points clearly stand out. On 2001-2-1, the number of addresses doubles, due to the allocation of `2002:/16` to the 6to4 transitional technology [28], which reserved a part of the IPv6 space for 6to4; since this is a reservation, it cannot be considered a true measure of IPv6 growth. Likewise, a gigantic drop is seen on 2006-6-6, due to the decommissioning of the 6Bone (`3FFE::/16`). Since the 6Bone was experimental and decommissioned once IPv6 growth was observed [78], it cannot be considered evidence of significant IPv6 constriction.

Second, one can incorrectly conclude that *IPv6 growth has plateaued*. The number of allocated addresses only grew by 20% from 2006-6-7 to 2008-9-26, nearly a 2.5 year period. However, growth is masked by a few extremely large prefixes that hide the allocation of smaller ones. As of 2008-9-28, of the 2774 allocated prefixes, 31 had a prefix length of 22 or shorter, compared to a median prefix length of 32. Since such large address blocks are allocations (*i.e.*, delegating responsibility of address assignment to local or national registries) as opposed to assignments, they are not true measures of growth. This delegation is the explanation for the plateau, and it is incorrect to draw conclusions about IPv6 growth based on it.

4.2.3 Drawing Correct Conclusions from the RIR Allocation

What information can be gleaned from the RIR data? After further analysis, we find that statistics concerning the number of prefix allocations provide insight into the deployment of IPv6.

year	allocations	mean	mode	1st quartile	median	3rd quartile
2005-3-3	1183	33.65	32	32	32	34
2006-3-3	1421	33.39	32	32	32	34
2007-3-3	1720	34.19	32	32	32	34
2008-3-3	2179	34.65	32	32	32	34

Table 4.1: Distribution of prefixes over time. Numbers are cumulative over time.

Why Analyzing Prefix Allocations is Better

Since looking at the number of allocated *addresses* is not particularly insightful, why would looking at the number of allocated *prefixes* be appropriate? Moreover, is it really fair to look at prefixes independent of their size, lumping the /64s with the /48s and /32s, treating them as all "equal"?

Table 4.2.3 shows the distribution of various prefix lengths as a function of year. As time passes, more and more /32 address blocks are present, and the statistics indicate that it is the favorite for recent allocations. In fact, as of 2008-9-28, more than 67% of all prefixes allocated were /32. We believe this heavy bias is due to RIR policy changes in 2004, which made obtaining /32s easier [23, 27, 16].

Since so many prefixes are the same size, analyzing allocated prefixes will be roughly fair (since most address blocks are the same size), and the results will not be skewed by the few "heavy hitters" seen before.

Unfortunately, the sub-allocations are not recorded in the RIRs. Thus, if a /32 is allocated to an organization, and that organization re-allocates it to others, only the first entry is recorded. Thus, our prefix allocation analysis can potentially underestimate the growth of IPv6.

Prefix Allocations Reveal Growth Trends

Figure 4.2 shows allocations of address blocks for the different registries. RIPE is clearly dominant, and shows extremely large growth for 2008. Likewise, ARIN allocations are also increasing at a very fast rate, causing it to surpass APNIC. APNIC

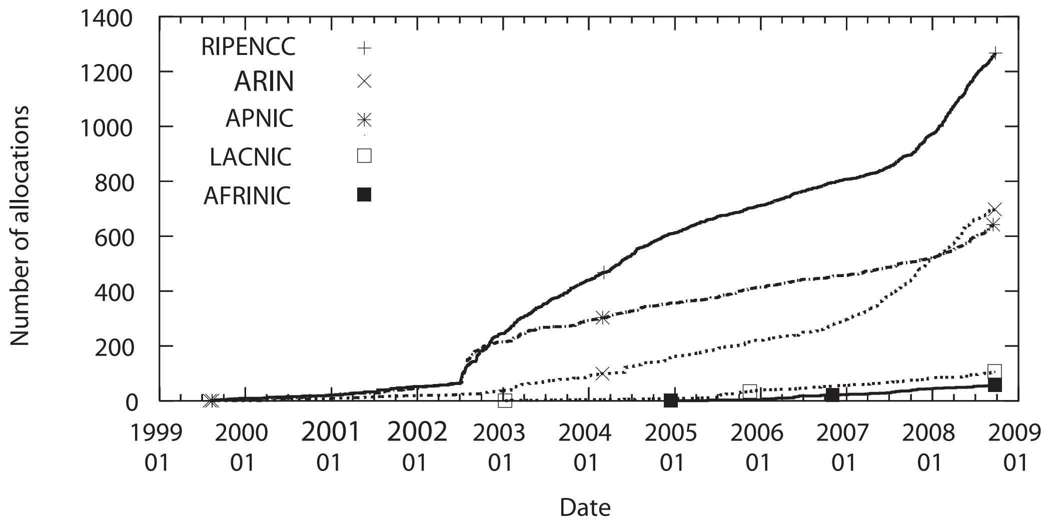


Figure 4.2: Growth of the individual registries.

has many allocations, but has been growing slowly, and only starts to show signs of acceleration toward the end of the study. LACNIC allocations remain approximately linear. While AFRINIC has very few allocations, it shows signs of acceleration. Cumulatively, there have been nearly 2800 address block allocations. Overall, it would appear as if IPv6 growth has been somewhat stagnant, increasing at a mostly linear rate, until recently.

One point on the graph requires explanation. In July of 2002, RIPENCC and APNIC experienced abnormal growth. Investigation revealed that on July 1st, 2002, RIPENCC and APNIC both instituted policy changes regarding IPv6 allocation [119]; this policy set new guidelines for how to allocate space, as well as guidelines for allocating to new organizations desiring IPv6 space. For example, it defined clear criteria for requesting IPv6 space, as well as a method for organizations to request additional allocations. As such, we believe that these policy changes are responsible for the sudden surge in allocations.

To summarize, IPv6 allocation has only recently started taking off; previous years had mostly linear growth, while current growth could possibly be exponential. How-

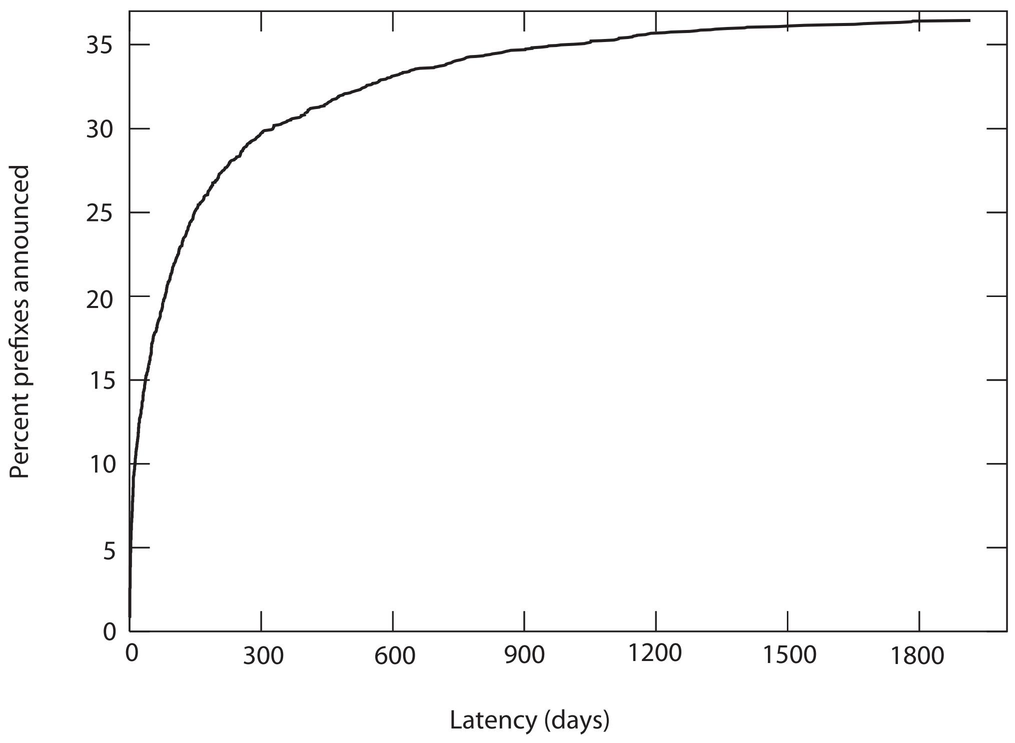


Figure 4.3: CDF of latency for the 36% of prefixes that eventually were announced in the BGP data and were analyzable.

ever, since we are only at the beginning of the "knee" in the curve, we should be careful in extrapolating too far into the future.

However, address allocations do not imply address usage. Is this allocation really a good measure of IPv6 deployment?

4.2.4 Prefixes are Allocated, but Often not Used

We turn to BGP routing data, which documents which IPv6 addresses are currently routable on the IPv6 Internet. Figure 4.3 shows how long it takes institutions to announce their address blocks (or any sub-block) after they've been allocated, given that they do eventually announce at some point. We call this "usage latency," and is defined as the difference in time between allocation and the prefix's first appearance in the BGP routing tables.

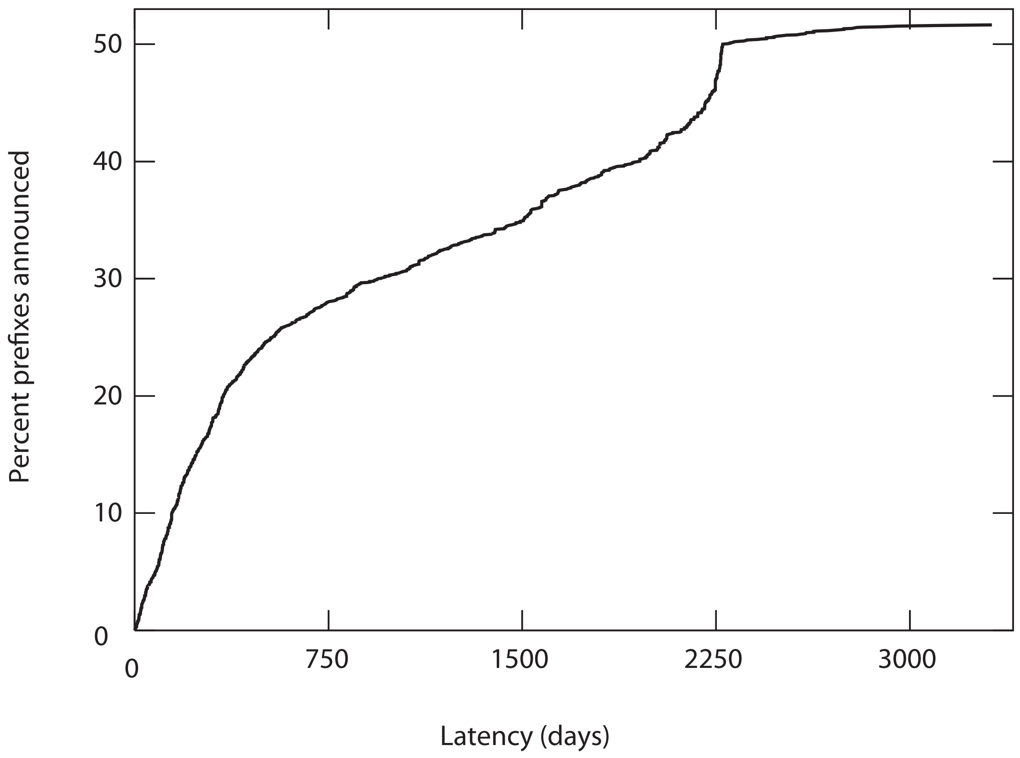


Figure 4.4: CDF of latency for the 52% of prefixes that never announced themselves in the BGP data and were analyzable.

There are a few points of note about Figure 4.4, which plots information about organizations that never announce their connectivity. In particular, *52% of all allocated prefixes never appears.* Measuring latency for these prefixes is impossible, since it is unknown when they will be used (if ever). Instead, we measure minimum latency, *i.e.*, the minimum amount of time before usage. We find that the average minimum latency is 957 days (which is an underestimate for the true latency!). The kink in the curve corresponds to the policy changes of APNIC and RIPENCC in July of 2002, as mentioned earlier; many addresses allocated during that surge were never used.

When computing true latency for the remaining prefixes, we run across a snag. Approximately 12% of all prefixes was allocated before 2003, when our BGP data begins. As such, it is impossible to accurately measure latency for these prefixes. We ignore such prefixes since they are a small minority.

For the remaining 36% of prefixes, latency averages 173 days. For comparison, we look to a study concerning usage latency in IPv4 in 2005 [105]. The study found that 87% of all prefixes was eventually announced in BGP, and the average latency for these prefixes was 52 days. Thus, there are fewer IPv6 users than their IPv4 counterpart, and they are also much slower to deploy.

Overall, there was some slight variation in latency between regions, but all regions were within a factor of 1.5 of each other. RIPE averaged 141 day latency. LACNIC's latency was 159 days, and AFRINIC's was 177 days. APNIC and ARIN were nearly identical, at 202 and 211 days, respectively.

4.3 Traffic Analysis in a US Tier-1 ISP

While the RIR and BGP data capture the rate of IPv6 adoption, this ignores three other aspects of IPv6 deployment – how people are transitioning their IPv4 networks to IPv6, how IPv6 addresses are actually assigned to individual machines, and what IPv6 applications are being used. Identifying transitional technologies helps us understand how IPv4 networks connect to the IPv6 world.

The upper 64-bits of an IPv6 address identify such mechanisms, as they each have different IP ranges. Observing addresses tells us how organizations assign IPv6 addresses to individual interfaces. Since the low order 64 bits are reserved entirely for host machines, we can use this to see how individual organizations number their devices. To analyze the application mix, we look at the signature *(source port, destination port, protocol)* to map it to an application name.

4.3.1 Data Sources

To analyze native IPv6 traffic, we use Netflow records collected from an IPv6 Internet gateway router in a US tier-1 ISP with 11 IPv6 BGP neighbors. These records were

name	**2008-4**	**2008-5**	**2008-6**	**2008-7**	**2008-8**	**2008-9**
Native IPv6	85.5%	90.5%	87.0%	74.2%	63.5%	75.2%
6to4	12.7%	7.1%	10.6%	23.4%	32.4%	20.8%
Teredo	1.7%	2.3%	2.4%	2.3%	4.1%	4.0%
Other	0.1%	0.1%	0.0%	0.0%	0.0%	0.0%

Table 4.2: Monthly averages of different types of IPv6 addresses, as seen in native IPv6 records.

collected from 2008-4-1 to 2008-9-26, and are taken from the business customers. To analyze tunneled traffic, we collected packet header traces from 2008-7-02 to 2008-8-31 at an access router servicing approximately 20,000 DSL subscribers (different from the business customers) in an ISP. In particular, we analyzed the IPv6 headers within Teredo tunnels. Teredo [21] is an IPv6 tunneling technology created by Microsoft to enable IPv6 communications for Windows users. Due to the prevalence of Windows among typical Internet users, we assume that most tunneled IPv6 traffic destined for these subscribers use Teredo.

Unfortunately, our records are not perfect, and have some holes. Note that 5th, 6th, 9th, 10th, and 11th of July are not analyzed for Netflow. Also note that for the tunneled traffic, data from 2008-7-10 to 2008-7-21, along with 2008-8-19, 2008-8-23, and 2008-8-26 are not included.

4.3.2 Identifying Transitional Technologies and Address Enumeration

We identify transitional technologies as follows. Teredo uses `2001:0000:` for the first 32 bits [21] of an IPv6 address, making it easily identifiable. 6to4, another popular encapsulating scheme, begin with `2002:`. Although other transitional schemes exist and can be identified (*e.g.*, ISATAP, automatic tunnels, *etc.*), they are quite rare in practice; as such, we lump them together as under the label "other".

name	**2008-7**	**2008-8**
Native IPv6	70.2%	44.2%
6to4	2.8%	4.8%
Teredo	26.7%	50.3%
Other	0.3%	0.7%

Table 4.3: Monthly averages of different types of IPv6 addresses, as seen in tunneled IPv6 headers.

To discover how organizations assign addresses to devices, we use the same methodology as presented in [103]. The types of enumeration are: Teredo (Teredo encodes options and routing information in the lower 64-bits), MAC address based (also called auto-configuration), low (only using the last 12 bits), wordy (using words that can be spelled in hexadecimal, like `BEEF`), privacy (all bits are randomly set, according to the IPv6 privacy specification [109]), v4 based (when a corresponding IPv4 address influences the choice of host address), and unidentified (for all others).

4.3.3 Transitional Technologies

The results of analyzing the IP address structure are presented in Table 4.2 and in Table 4.3.2. Most of the native IPv6 addresses of the tier-1 ISP tended to communicate with other native IPv6 addresses; approximately 80% of addresses fell into this category. 6to4 addresses were also significant, representing approximately 18% of addresses seen. Teredo addresses constituted approximately 2%, and the remaining technologies were almost negligible. These results also match those found for an analysis done in 2007 [103]. As an important note, the data sets used in our analysis are quite different from those in [103] (which included web server traffic, name server traffic, and traceroutes). Since we have a different vantage point and a different time-frame, yet have the same results, we believe that the technologies used by organizations remain unchanged for the past year.

From the tunneled perspective, we see that Teredo and native addresses are popular. Moreover, around 2008-8, a surge of Teredo-to-Teredo connections is seen.

name	2008-4	2008-5	2008-6	2008-7	2008-8	2008-9
IPv4 Based	49.5%	28.7%	19.3%	5.9%	20.2%	6.1%
Low	22.0%	29.9%	32.5%	36.5%	31.0%	34.8%
Auto-configured	18.6%	29.2%	33.5%	40.3%	31.2%	42.6%
Teredo	1.7%	2.3%	2.4%	2.3%	4.1%	4.0%
Wordy	0.2%	0.2%	0.4%	0.3%	0.8%	0.3%
Privacy	1.0%	1.0%	1.3%	1.7%	1.5%	1.5%
Other	7.0%	8.6%	10.5%	11.8%	11.2%	10.7%

Table 4.4: Monthly averages of assignment schemes seen for the native IPv6 records.

4.3.4 Assigning Addresses to Machines

In addition to looking at transitional technologies, we looked at the breakdown of IPv6 address assignment schemes. Table 4.4 demonstrates the ratios of various host configurations. A few interesting trends emerge. First, IPv4 based addresses decline sharply (although there is a spike in August that remains unexplained). Moreover, privacy extensions remain relatively unused, occupying a small percentage of all addresses (possibly because some operating systems do not enable privacy extensions by default).

4.3.5 Application Mix

Looking at the application breakdown yielded interesting results, as seen in Table 4.3.5. Expected traffic, like web and mail, was surprisingly low – usually between 1% to 8% for web and 1% and 2% for mail. We performed DNS reverse lookups on the few IPv6 addresses that used web protocols and found that popular sites include an IPv6 deployment and tunnel broker and a backbone network for universities. On average, about 85% of traffic is DNS queries and 8% ICMP messages. Overall, these results are quite surprising. We believe there are two possible reasons. One could be that people are mainly using probing applications over their IPv6 networks, and not actual applications. Another is that operating systems like Windows Vista will send an extra DNS request when IPv6 capabilities are turned on: one requesting the

name	**2008-4**	**2008-5**	**2008-6**	**2008-7**	**2008-8**	**2008-9**
DNS/Domain	75.5%	86.0%	87.9%	85.3%	88.8%	93.1%
ICMP	11.0%	10.2%	6.9%	6.5%	7.3%	5.2%
Web	8.3%	1.9%	1.3%	2.7%	0.8%	0.4%
Mail	1.3%	0.4%	1.0%	1.5%	0.4%	0.3%
Other	3.9%	1.5%	2.9%	4.1%	2.7%	1.0%

Table 4.5: Monthly averages of applications, as seen in native IPv6 records; percentages based on number of bytes.

IPv4 address and one requesting the IPv6 address [18]. Thus, the IPv6 interface may send and receive DNS queries but not traffic. Despite the potential inflation of DNS records in our data, there is still very little "real" traffic seen for IPv6. We believe that this demonstrates, for at least this tier-1 ISP, customers view IPv6 as experimental.

For Teredo tunneled traffic, application breakdown was also interesting. Table 4.3.5 shows that almost all traffic is unidentifiable UDP or TCP, indicating random port numbers. Given the vast quantity of unidentifiable traffic, and the rise of Teredo pairs, it is likely that these are P2P applications communicating with each other (as random port numbers are characteristic of P2P traffic). Indeed, some applications have turned to Teredo to solve the issue faced by end hosts that are limited by their NAT/firewall technologies when they try to initiate communications with each other; using the Teredo protocol, a client contacts a Teredo server, which acts as a broker agent between Teredo clients, aiding in NAT/firewall hole punching, as well as providing unique IPv6 addresses. Several P2P clients have implemented IPv6 support [29], such as uTorrent and Vuze (formerly Azureus); moreover, uTorrent has the ability to set up Teredo automatically [25]. To summarize, it appears as if considerable tunneled IPv6 traffic is a by-product of applications (such as P2P file-sharing) using Teredo as a mechanism to bypass local NATs and firewalls, simplifying the application developers' jobs.

name	**2008-7**	**2008-8**
Random TCP	30.6%	94.7%
Random UDP	67.3%	3.2%
Web	0.2%	0.03%
Other	1.9%	2.07%

Table 4.6: Monthly averages of applications as seen in tunneled IPv6 headers; percentages based on number of bytes.

4.4 Related Work

IPv6 topology has been investigated by CAIDA's scamper work [83], as well Hoerdt and Magoni's Network Cartographer [80]. Because we did not investigate this aspect of IPv6 deployment, we consider our work to be complementary to these studies.

Anomalous BGP behavior has been analyzed through Huston's automatically generated IPv6 reports [85]. These reports include information about routing instability, prefix aggregation, table sizes, and allocation sizes.

Testing the readiness of IPv6-enabled software occurred in February of 2008, when NANOG shut off IPv4 access from their meeting for one hour [125]. It resulted in a severe restriction of services, with end users often needing to re-configure their machines. It revealed that IPv6-enabling software is still somewhat user unfriendly [59]. We believe this work on *how an individual can use IPv6* to be complementary to our work on *how organizations are using IPv6*.

Regarding traffic analysis, Arbor Networks [88] found that IPv6 traffic is growing at the same rate as IPv4 traffic. Savola [122] analyzed 6to4 traffic and found much was experimental, and also noted a rise in P2P applications. Hei and Yamazaki [77] analyzed 6to4 traffic on a relay in Japan and found that TCP traffic dominated UDP, with a considerable amount of HTTP traffic (40% of total). Our work complements these studies because we analyze different data sources, and offer a new perspective by analyzing traffic from a tier-1 ISP.

Finally, David Malone's work on IPv6 addresses analyzed transitional technologies and the assignment of IPv6 addresses to machines [103]. He looked at the breakdown of types of IPv6 addresses (Teredo, 6to4, *etc.*), as well as the classification of the host part of IPv6 addresses. While we do repeat some of the same analysis (and use some of the same techniques), we believe there are key differences between our study and his. We cover broader ground by looking at more data sources: RIR allocations, BGP data, Netflow records, and packet header traces. We also perform additional analysis, such as address space allocation and latency.

4.5 Conclusion

While IPv6 is beginning to see larger deployments, it still has some significant barriers to overcome. IPv6 is still viewed as experimental by some, and often is deployed in counter-intuitive ways. By analyzing RIR and BGP data, it appears that many allocations are speculatory, and that autonomous systems wait significant amounts of time before actual announcement. Moreover, although IPv6 traffic is growing, our data from a US tier-1 ISP indicated that much of it is still DNS and ICMP packets, indicating a lack of true IPv6 applications from our vantage point; additionally, tunneled traffic analysis shows much of the communication is between IPv4 pairs, implying that applications like P2P file sharing are dominant.

Further work would include a longer study of these characteristics, as well as a topological study involving more end hosts. Moreover, it would be interesting to track operating system developments and their support for various transitional schemes, as well as native support, to better understand how this software shapes the future of IPv6.

Chapter 5

Characterizing Enterprise Multicast Traffic

Another upcoming protocol is multicast. IP multicast, after spending most of the last 20 years as the subject of research papers, protocol design efforts and limited experimental usage, is finally seeing significant deployment in production networks. The efficiency afforded by one-to-many network layer distribution is well-suited to such emerging applications as IPTV, file distribution, conferencing, and the dissemination of financial trading information. However, it is important to understand the behavior of these applications in order to see if network protocols are appropriately supporting them. In this chapter we undertake a study of enterprise multicast traffic as observed from the vantage point of a large VPN service provider. We query multicast usage information from provider edge routers for our analysis. To our knowledge, this is the first study of production multicast traffic. Our purpose is both to understand the characteristics of the traffic (in terms of flow duration, throughput, and receiver dynamics) and to understand whether the current mechanisms to deliver multicast to VPNs can be further improved. Our analysis reveals several classes of multicast traffic that have ample opportunity for further multicast optimization.

The rest of the chapter proceeds as follows. Section 5.2 discusses protocol specifics. Section 5.3 discusses how the data is collected and challenges of inference. Section 5.4 shows our results. We compare our work with related research in Section 5.5, and conclude in Section 5.6.

The multicast study will be published in the proceedings of the ACM SIGCOMM Workshop: Research on Enterprise Networking 2009, under the title *Multicast Redux: A First Look at Enterprise Multicast Traffic*.

5.1 Introduction

IP multicast [56, 58], which provides an efficient mechanism for the delivery of the same data from a single source to multiple receivers, was first proposed more than two decades ago. It was deployed experimentally on the MBone [111] in the early 1990s and was the subject of a significant amount of research into scalable and robust intra- and inter-domain routing protocols [57, 35, 135]. The MBone, an overlay used primarily to support audio and video conferencing between multicast-capable networks around the Internet, grew rapidly. However, the initial enthusiasm for multicast did not translate into widespread success. The MBone eventually declined, and more importantly, multicast was not adopted in any significant way by service providers.

The failure of multicast to achieve widespread adoption can be attributed to several technical and economic factors. When sources and receivers were in different domains, it was unclear how to appropriately charge for the service. Furthermore, the *de facto* multicast protocol, PIM [57], presented challenges of its own for inter-domain multicast, as it could create inter-provider dependencies. In particular, users in one domain could rely on infrastructure in other domains to carry strictly intra-domain traffic. Moreover, multicast protocols were new, quite complex, and difficult to manage. Finally, the lack of popular multicast applications translated to little interest for deployment.

After languishing for many years on the Internet, multicast is experiencing a resurgence in two main contexts. First, within service provider networks, it is being used to support IPTV applications which benefit from efficient one-to-many distribution. Second, multicast is being deployed in enterprise networks, where the aforementioned issues of management, billing and inter-provider dependencies, are mitigated. In enterprise networks, multicast is used to support a variety of applications, including file distribution, conferencing, and dissemination of financial trading information. Moreover, many enterprises connect their geographically disparate sites via provider-based VPNs. Hence, multicast is becoming increasingly available to VPN customers.

Since the use of multicast in production networks is a relatively recent phenomenon, little is known about its traffic characteristics. However, gaining such knowledge of traffic characteristics is important, as it can help researchers understand whether protocols in the network are adequately supporting the applications. Additionally, such knowledge is needed for proper network planning and provisioning. Moreover, today's multicast VPN services are new and complex (as described in Section 5.2), with many configuration options based on assumptions regarding usage, engineering guidelines, rules of thumb, and controlled testing. Behavior "in the wild" is poorly understood, yet is critical to uncovering possible issues and improvements regarding design, operation, capacity planning, provisioning, resource usage, and performance. A first step to resolving these issues requires an understanding of traffic in the multicast VPN service.

Studying enterprise traffic from the vantage point of a service provider presents both challenges and benefits. The provider does not have the same visibility into the enterprise traffic as would be possible within the enterprise network itself. This derives from the fact that the enterprise traffic is encapsulated as it crosses the provider network, thereby masking some relevant information. However, we believe that this challenge is far outweighed by the benefits of scale and diversity that a provider domain study affords. Specifically, rather than being limited to studying traffic within a single enterprise, we are able to study traffic in many different enterprise networks. Thus, our results are more general than they would be otherwise.

As such, we undertake a study of enterprise multicast traffic as observed from the vantage point of a tier 1 ISP that provides multicast service to some of its VPN customers. Our purpose is both to understand the characteristics of the traffic and how the VPN service can more efficiently deliver multicast traffic to its customers. By understanding how applications use multicast, we can better understand how to design our networks. To our knowledge, this is the first broad study of production multicast traffic. We collect usage data from provider edge routers that describe the activity of their associated multicast groups. We analyze multiple traffic characteristics, including flow duration, throughput, peak rates, and receiver dynamics. These statistics compactly describe every multicast session and provide information about the benefits (or lack thereof) that the provider network receives from multicast. We also apply clustering techniques to classify flows according to their behavior, allowing us to see if dominant usage patterns emerge.

5.2 Multicast VPN Overview

In this section we provide an overview of how multicast and Multicast VPNs (MVPNs) are supported by the ISP. A description of the specification upon which the MVPN service is based can be found in [120]. While MVPN service may evolve in the future, the description provided here represents current industry practice.

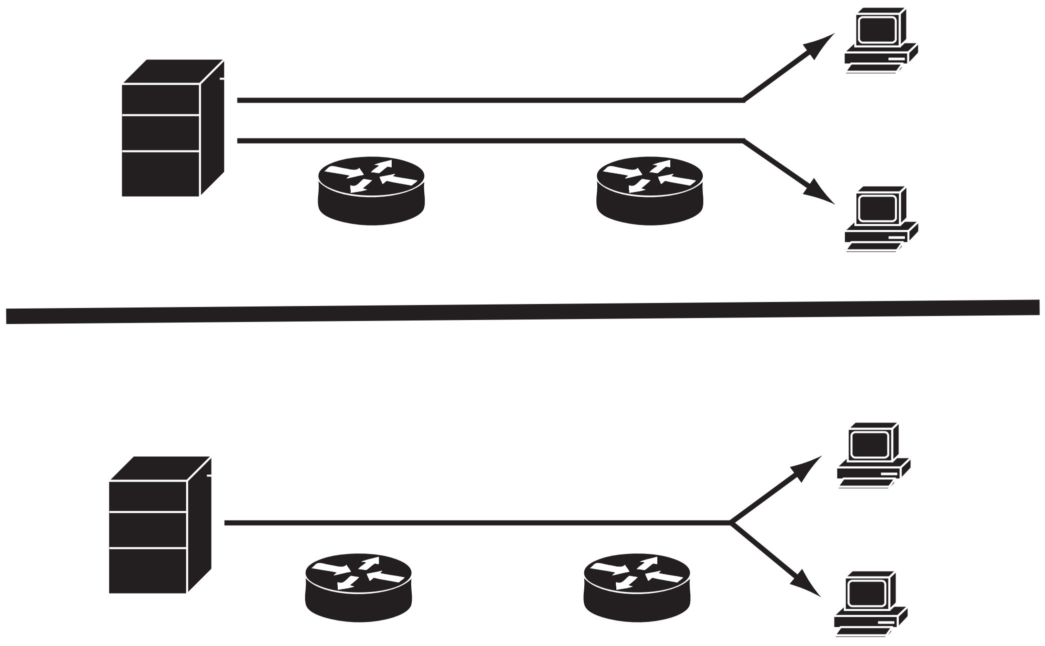


Figure 5.1: On top: a server sending data to clients using unicast. On bottom: efficient distribution with multicast.

5.2.1 IP Multicast

With traditional unicast, every connection has exactly two endpoints; multicast, on the other hand, replaces point-to-point delivery with the idea of *groups*. Multicast groups use specially designated IP addresses, known as multicast addresses, taken from the 224/4 address block [30]. Hosts can join or leave groups (on behalf of applications) at will. Packets sent to the group address are forwarded to all members.

The requirement to deliver the exact same data to multiple end hosts provides the opportunity to reduce network resource consumption. Multicast protocols create a distribution tree on the network topology for every group [1]. This tree reaches all group members. When packets are transmitted to the group, a single copy is sent along the initial branch from the source. When the tree branches, the router at the branch point replicates packets and sends separate copies along each branch.

[1]For simplicity we omit many details of multicast routing; some trees may be specific to an individual sender while others may be used by all senders to a group.

Figure 5.1 demonstrates this capability. As a result, the amount of bandwidth needed on shared path segments can be greatly reduced.

In general, every multicast tree requires a routing table entry on every router along the tree. Thus, the total amount of routing state can potentially be proportional to the number of multicast groups.

5.2.2 Multicast VPNs

In a VPN, the customer attaches to the provider network at one or more locations. At each such provider location, one or more Customer Edge (CE) routers attaches to a Provider Edge (PE) router, as depicted in Figure 5.2. The PE receives packets from attached CEs and transports them across the backbone to other PEs, which then deliver them to attached CEs. In the case of Multicast VPN, a customer multicast packet entering at an ingress PE, may be destined to receivers at multiple customer locations. Thus, the provider will be required to deliver the packet to one *or more* egress PEs. The customer multicast packet is encapsulated using GRE [63] and transported across the backbone to the PE routers using IP multicast. That is, the customer multicast packet is encapsulated in a provider multicast packet between the ingress and egress PEs.

Within the provider backbone, every VPN is assigned a unique multicast group, called a *Default MDT (Multicast Distribution Tree).* When a customer attaches itself to a set of PEs, those PEs join the associated multicast group. The Default MDT acts as a broadcast channel among the PEs and serves two purposes. First, customer domain multicast control messages are transmitted over the Default MDT. Second, customer multicast packets (*i.e.*, application traffic) are initially transmitted over the Default MDT. These multicast packets are delivered to all PEs to which the customers attach. However, some of these PEs may not have group members downstream of their attached CEs. Packets that reach such PEs are dropped.

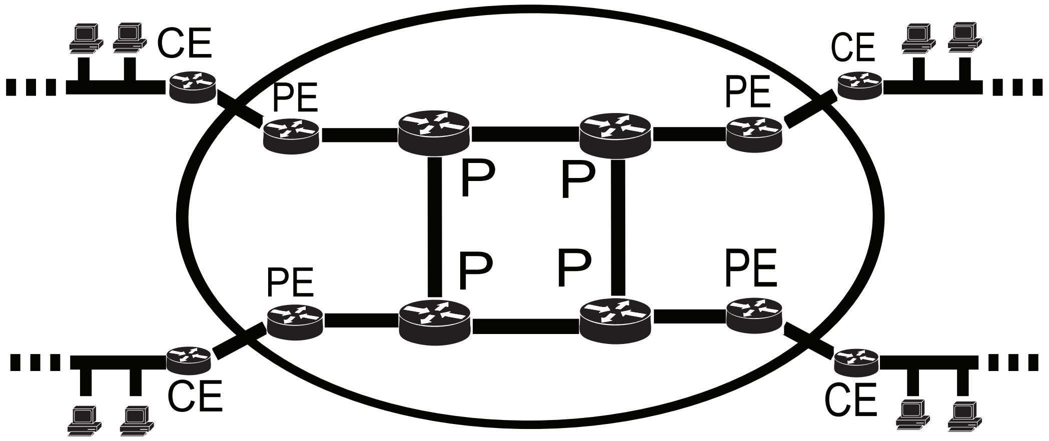


Figure 5.2: Setup of an example MVPN network. Circle encloses the provider network. Customer edge (CE) routers attach to provider edge (routers). The core provider (P) routers do not interact with customer routers directly, but transit traffic between PEs.

Since packets may be dropped at some PEs, this Default MDT mechanism has the potential to waste bandwidth in the provider network. To mitigate this problem, a second kind of multicast group, referred to as a *Data MDT*, is used in the backbone. When an ingress PE detects that the sending rate of a sender to a group exceeds a configured threshold (based on duration and throughput), that sender's traffic to the group is moved from the customer's Default MDT to a customer-specific Data MDT in the provider backbone. A PE will only join this announced Data MDT if there are receivers for the corresponding customer group downstream of its attached CEs. The Data MDT mechanism does not waste bandwidth; packets are only delivered to PEs that have downstream receivers. However, Data MDTs consume additional resources (in the form of routing table entries) in the provider network.

5.3 Methodology

In this study, we collect and analyze data about enterprise VPN multicast traffic seen on a large, tier-1 ISP. Along the way, we ran across several analysis issues. We disclose our challenges and solutions here.

5.3.1 Data Collection

Between January 4, 2009 and February 8, 2009, we monitored multicast traffic using a specially designed, lightweight poller. The poller contacts the PE routers that support enterprise VPN traffic using SNMP [76]. We mine customer provisioning information to configure the poller to understand the mapping between multicast groups and associated PEs [2].

We refer to PEs that inject traffic into the backbone as senders, and PEs that receive traffic from the backbone as receivers. *Note that in our analysis, "receivers" always refers to egress PEs.* These definitions arise from the fact that we study multicast traffic over a backbone network, with PEs on the boundary between the backbone and the customers.

Each PE tracks the amount of data sent to and received from each Default and Data MDT using a byte counter. Note that multicast groups are uniquely identified by an (S, G) tuple, where S is the IP address of the sender and G is the multicast group address. Due to the way in which multicast is configured in the provider backbone, data received on Default MDTs at egress PEs is reported on a per-group basis, rather than for each (S, G) pair. Data sent by ingress PEs is reported on a per-source basis. Given this setup, the data we collect is a series of records in the form of Figure 5.3.

[2]One multicast group is assigned per Default MDT per VPN, and one multicast group is assigned per Data MDT per VPN. However, a VPN may have multiple Data MDTs.

```
Timestamp:   1237114394
Group:       239.255.255.128
Source:      58.122.125.122
Receiver:    58.122.16.89
Counter:     16938168
Status:      Okay
```

Figure 5.3: Format of the multicast information received from the poller. The entries are synthesized for illustration.

We poll all the PEs at five minute intervals. While we would ideally like to poll at a smaller interval, we were limited by the operational constraints of a large backbone network. Routers were already supporting SNMP polling to support a wide variety of management functions, and we needed to limit the additional load placed on them. As a side note, not all successive polls were spaced exactly five minutes apart, due to variability in the polling process. For the time frame of our study, actual polling intervals ranged from 2.4 to 13.2 minutes. However, 99.7% of the poll intervals was between 4.8 minutes and 5.2 minutes. Thus, except for a few outliers, the polling is very consistent. Overall, we collected and analyzed approximately 88 million records.

The post-analysis of the polled results needs to handle two issues. As with any polling-based approach, a poll may not return the requested data object. This can occur for a variety of reasons. For example, SNMP requests or responses (and subsequent retransmissions) could be lost or the router could take too long to respond. In such cases, the request is considered to have "timed out" and values are not reported. Also, if configuration information is outdated, the poller may request information for a multicast group that is no longer a part of a particular PE router. However, in practice these problems occur infrequently. Approximately 99.3% of all polls was successful; 0.4% failed due to time outs and 0.3% failed due to outdated configuration information.

Second, in some cases we observed discrepancies (beyond what one would expect due to packet loss) between the amount of data reported by senders and receivers for a (S, G) pair. As a sanity check, we examined such discrepancies for a small sample of the Default and Data MDT data. For the Default MDT, senders and receivers usually agreed on the amount sent, although there was noticeable dissent as well. We analyzed these dissenting cases and found several possible explanations. In some of the cases, we identified the data as obviously incorrect (*e.g.*, sending rates faster than the line speed) which we believe were due to bugs in the way the data was reported. We also found that our polling method did not always accurately report data when a multicast group spans domains (as is the case for VPNs in the provider network that have presence on multiple continents). There were also cases in which we could not successfully identify the problem. When we could identify the cause of problems they often were associated with the amount of data reported received at egress PEs for Default MDTs. Thus, the analysis we can perform on the Default MDT is limited to aggregate analysis based on sending data at ingress PEs. However, the Data MDT had a higher rate of agreement. As such, we have general confidence in the accuracy of the data.

5.3.2 Default MDT Analysis

Our analysis for the Default MDT is further limited by several factors:

- Every PE router in a VPN is always attached to the Default MDT. Thus, it is impossible for us to study the dynamics of receivers that leave or join.
- Every PE router sends constant keep-alive messages (*i.e.*, PIM Hello messages) to the Default MDT. These messages are recorded by the PE routers as incoming data. Thus, every PE is always sending data, and it is difficult to tell when a true multicast flow starts or stops.
- Moderate to high bandwidth flows (which are of greater interest) are usually transmitted on the Data MDT.

The first problem is unavoidable, given our setup. The second problem can be solved using a threshold value, stating that flows that generate less than a certain amount of data per interval have ended. Since the keep-alive messages are fairly consistent in size and frequency, it is possible to model the background load generated by them. However, determining the threshold accurately is difficult, as variability in the messages raise the problem of accidentally filtering low bandwidth flows. Finally, since the Default MDT typically only carries low bandwidth flows, the total impact of it on the multicast network is limited. As such, we perform limited analysis on the Default MDT (avoiding any per-flow results) and focus most of our attention on the Data MDT.

5.3.3 Data MDT Analysis

For the Data MDT, we track flows (*a.k.a.* sessions) by observing the amount of data transmitted by the source for a given multicast group. A flow is defined by the Data MDT records associated with it, indicating its start and end times, as well as the amount of data sent and receiver dynamics. We calculate the amount of data sent by taking the difference between byte counters in successive sender records. We define *throughput* as the total amount of bytes sent averaged over the entire flow duration. We define *peak rate* as the maximum amount of bytes/second seen between any two consecutive polling intervals. In addition, we keep track of the number of receiver PEs joined to a particular group and record when a PE router joins or leaves the group. We also cluster flows to determine if dominant behavior patterns emerge. We employ a variant of the k-means algorithm [93] that applies a "simulated annealing"-style approach to the problem, along with a local search heuristic. We take into account the following characteristics: duration, throughput, peak rate, maximum number of receivers, and average number of receivers. Because such clustering algorithms usually assume equal variance for each characteristic, we normalize all characteristics to z-scores (*i.e.*, mean of 0 and standard deviation of 1) before clustering.

Although this analysis may appear straightforward, there are many corner cases. In particular, there are issues determining when a particular flow starts or stops, and determining the amount of data sent during a session. Some of these issues revolve around a condition we refer to as a *counter reset*, when a subsequent polling record either has a byte value of 0 or has a lower value than the previous record. These occur often in the data and may signify that a new flow has begun. However, interpreting what they actually mean in a particular situation is quite difficult. This problem is further complicated by the fact that some of the routers use a 32-bit byte counter which may overflow. Thus, when we see a counter value that is less than the previous one, did the session actually reset? Did the session end in-between a polling interval, with a new session starting up and taking its place? Did the counter overflow? Because of the difficulty in analyzing these cases, we adopted a simple rule to classify when the flows start and stop in this situation: *if the counter value is zero, a new flow has begun; otherwise, it is the same flow.* Note that in the case of a counter reset, instead of subtracting two intervals (to get the amount of data sent), we simply use the byte counter value of the last interval.

Another problem that arises is error messages (*e.g.*, SNMP time outs). Although they are rare, they can cause problems when analyzing when flows start and stop. If one sees a valid flow record, a sequence of error messages for the following intervals, and then another valid flow record, do we have two separate flows, or one longer flow? For the Data MDT analysis, we consider flows separated by error messages to be separate flows. Our analysis thus has a bias that reports more flows, each of shorter duration.

Another related issue is receiving a partial result from a query, *i.e.*, a router returns part of an answer but "times out" before returning all records. For simplicity, flow entries missing from the partial result are considered to be terminated. Although we did not quantify the number of such cases, we believe them to be sufficiently rare to not warrant serious investigation.

Up until this point, we have assumed that we have consecutive records to calculate quantities like duration, throughput, *etc.* However, for the Data MDT, it is possible that very short lived flows (less than approximately 10 minutes) may only have a single record associated with them. This behavior happens because of a temporary sending spike, which will boost a Default MDT multicast session into the Data MDT for a short period of time. Because we can only infer information from consecutive pairs of records, these single record sessions are difficult to analyze. As such, we label these flows as having 0 second duration with 0 kilobits/sec throughput and 0 kilobits/sec peak rate. This provides them with a unique identifier in our analysis.

5.4 Results

5.4.1 Default MDT Analysis

Figure 5.4 displays a range of the sending rates seen in the Default MDTs for each (S, G) pair per polling interval. Not all sending rates are shown. For example, there is a large step in the CDF at approximately 0.02 kilobits/sec, equivalent to approximately 800 bytes per 5 minute interval. We highly suspect that this value is related to the minimum amount of background PIM messages sent by every router in the network. Likewise, we observed a very small number of rates (less than 0.05%) in excess of 200 kilobits/sec. After manually inspecting several cases and finding them to not be in agreement with the receiving records, we removed them for this analysis.

There are several important aspects. First, the vast majority of the Default MDT rates are quite low, with approximately 99.5% of all rates less than 5 kilobits/sec. Second, we notice a very small number of flows that consume moderate bandwidth,

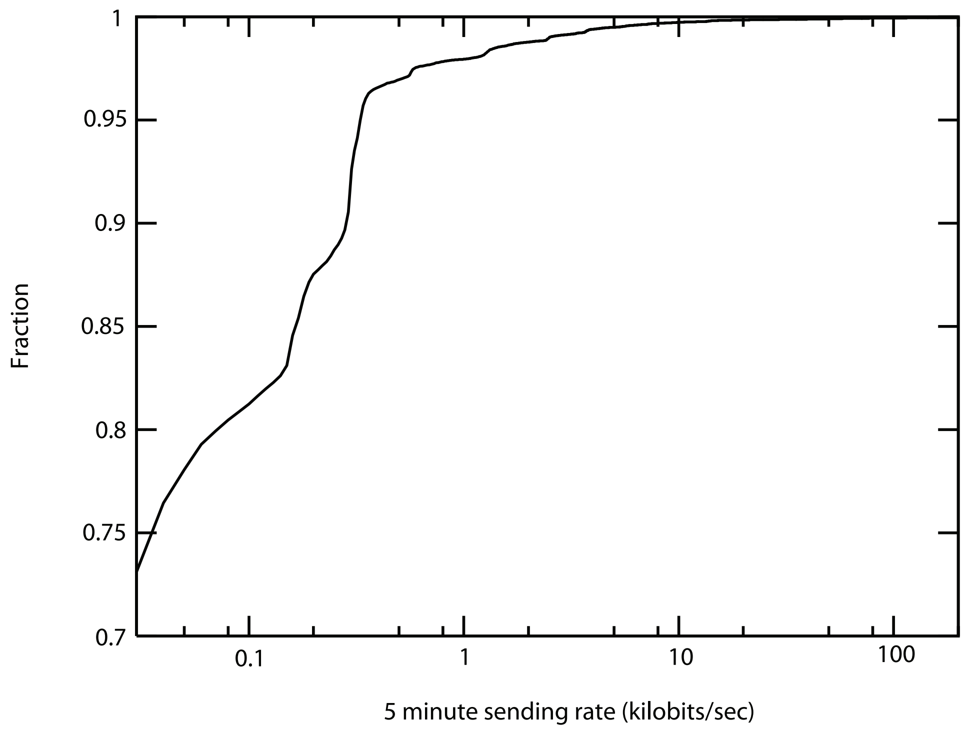


Figure 5.4: CDF of sending rates seen in the Default MDT between intervals.

surpassing 100 kilobits/sec. While it may seem odd for these "large" flows to be transmitted on the Default MDT (as they should be switched to the Data MDT), there are several reasons why we may see them here. A flow changes to the Data MDT only if it maintains a certain throughput for a given duration. Thus, it is possible that a very bursty flow may never meet the duration threshold, while still sending large amounts of data. Another possibility is that these rates represent flows that were eventually switched to the Data MDT; thus, these rates may reflect the short period of time before these flows were changed (and thus had their traffic registered in the Default MDT). It is also plausible that a router may be configured to never promote flows from the Default MDT to the Data MDT. In the first two cases, by reducing the duration threshold, these flows can properly be handled by the Data MDT, at the potential cost of increased routing state churn (since more entries will transition between the Default and Data MDTs and vice-versa).

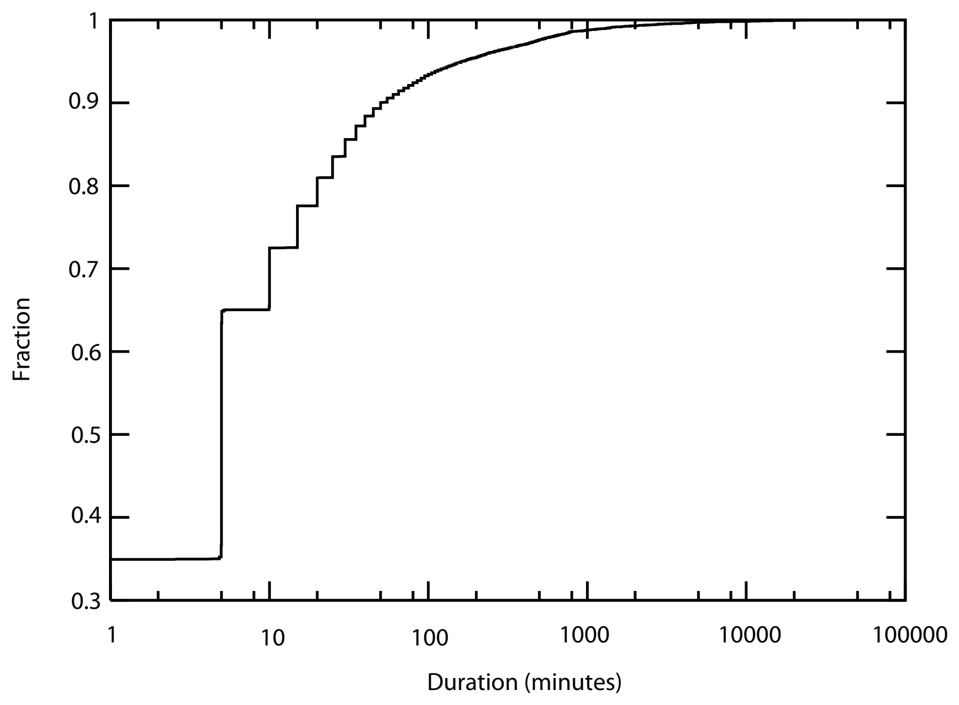


Figure 5.5: CDF of Data MDT session durations.

5.4.2 Data MDT Analysis

Figure 5.5 shows a CDF of flow durations. The stepwise nature arises from the fact that almost all poll intervals are five minutes. A significant portion of flows (approximately 70%) last 10 minutes or less. Ideally, such short lived flows, if sufficiently low bandwidth, would be kept entirely in the Default MDT to prevent an increase in routing state. Whether there exists a way to identify these flows and prevent them from moving to the Data MDT remains an open problem. While most flows are short-lived, a very small number lasts more than a week.

Figure 5.6 displays the average throughput seen in the Data MDT flows (as compared with Figure 5.4 for the Default MDT). Quite surprisingly, we see many flows (more than 70%) that send at less than 5 kilobits/sec. One reason for this phenomenon is that a large percentage of flows (36%) are so short-lived that they only have a single Data MDT record, and thus no accurate estimate for their throughput

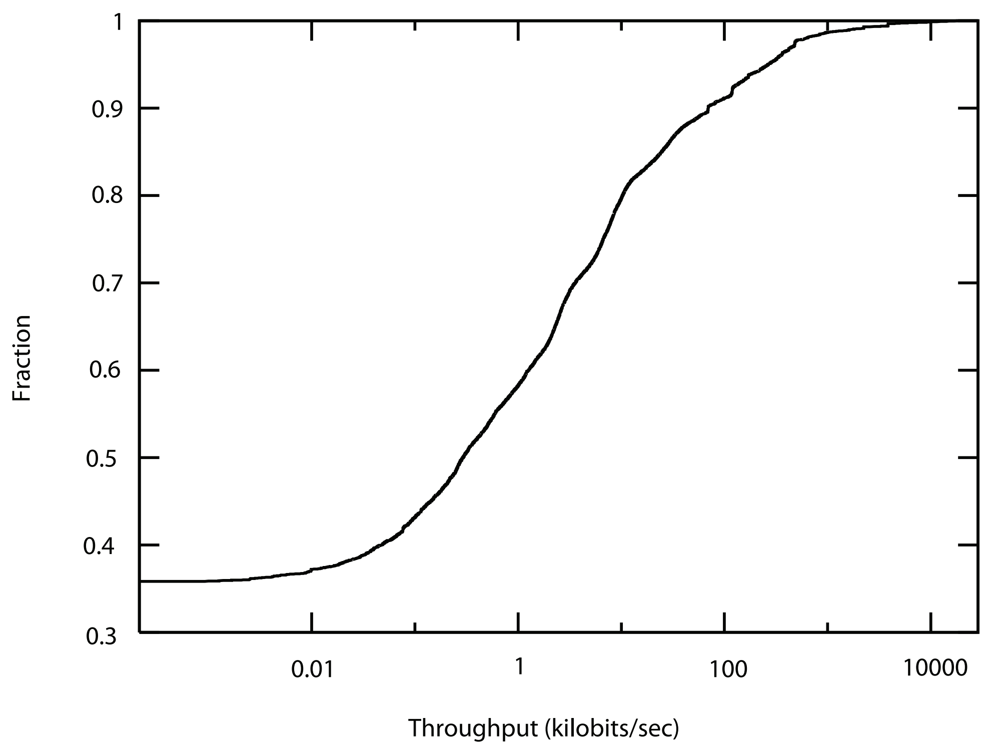


Figure 5.6: CDF of average throughput per Data MDT flow.

(and are defaulted to a value of 0 kilobits/sec). Since the Data MDT should primarily have high throughput flows, future research on multicast might investigate different mechanisms for switching flows to the Data MDT, as well as identifying situations when such transitions are appropriate.

Figure 5.7 shows the distribution of peak rates for the Data MDT. Peak rates are calculated by taking the maximum throughput seen across polling intervals. Peak rates are, on average, approximately 1.6 times greater than the average throughput. It is also interesting to note that for both peak rate and throughput, there are a small percentage of high bandwidth flows, indicating that multicast Data MDT trees may significantly reduce the amount of traffic sent over a network (relative to the Default MDT).

Figure 5.8 tracks the dynamics of receivers per session, measuring the maximum number of receivers seen, where a receiver is an egress PE in the backbone network.

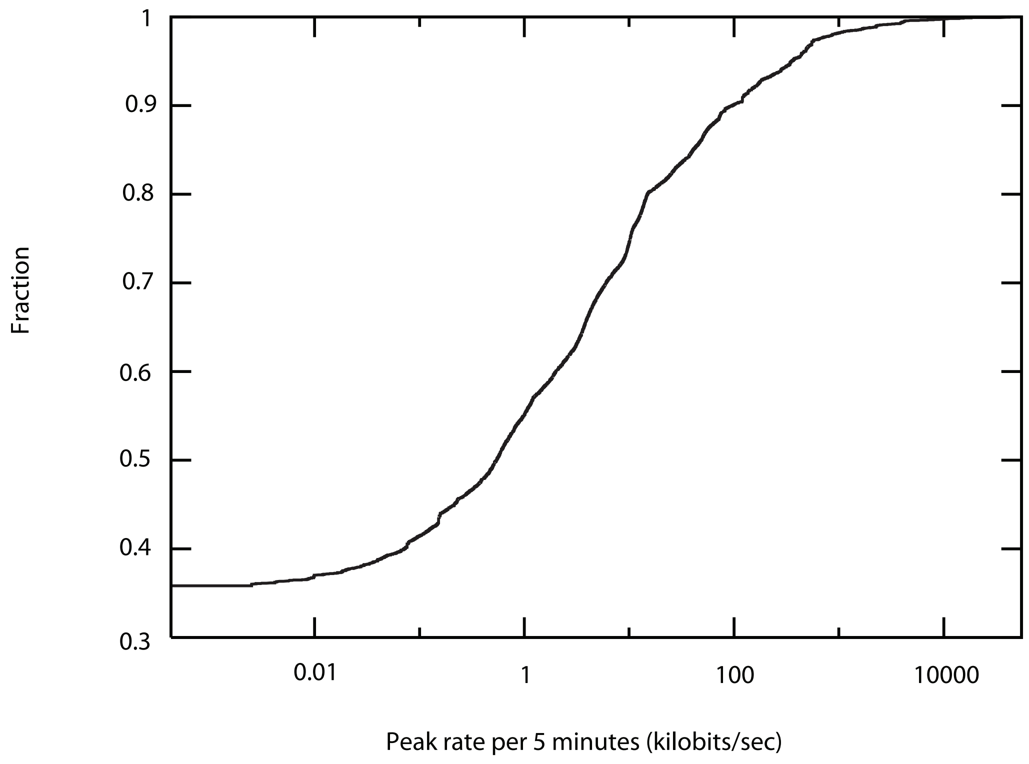


Figure 5.7: CDF of peak rates of Data MDT flows.

Surprisingly, a very large fraction of the flows (almost 50%) only have a single egress PE. In other words, using unicast to transport these flows across the backbone would use bandwidth just as efficiently (and incur less routing state overhead). These results imply that there may be an opportunity to reduce multicast state overhead; if an appropriate mechanism can be used to identify these flows in advance, a significant amount of multicast overhead can be removed by using unicast encapsulation across the backbone. As a separate note, there is a significant fraction (approximately 20%) of flows that reach at least 10 different egress PEs during their lifetime.

Finally, we perform clustering analysis on these characteristics (as well as on the average number of receivers seen per flow). In order to determine the number of clusters, we plot the unexplained variance for various numbers of clusters. Figure 5.9 shows the amount of variance that can be explained with different clusters, where variance is defined as the sum of the L_2 norms from each flow to its closest cluster.

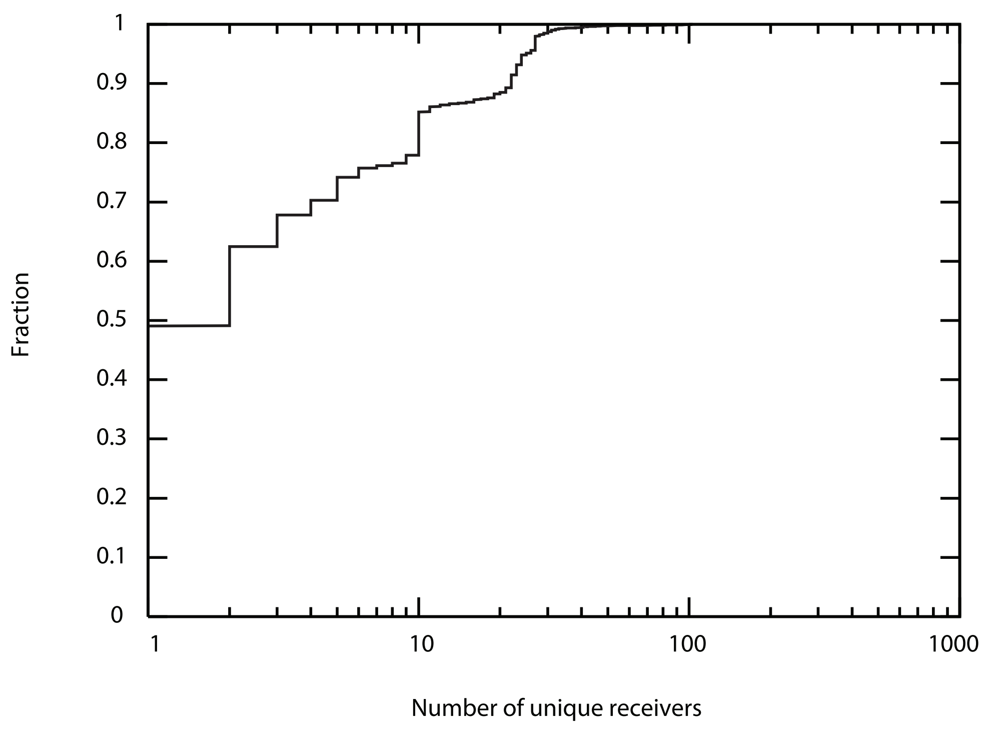


Figure 5.8: CDF of maximum number of receivers per Data MDT session.

We choose to label our flows with 4 clusters, as $k = 4$ is at the knee of the curve; it explains a significant fraction of variance while allowing us to label flows in a manageable manner. The cluster points are described in Table 5.1. A short name is given to each cluster point, highlighting its dominant characteristic. It is important to keep in mind that each cluster represents an average of many flows, and there is variance within a given cluster. From these points, we see many interesting aspects.

In the first cluster, called *unicast*, we see flows that are long-lived, very high throughput, with few egress PEs (usually one). As such, these backbone flows are not truly benefiting from multicast, but are adding routing state to the network. Although there are only a few flows that fall in this cluster, there may be value in investigating mechanisms to more efficiently support these flows, particularly if the applications they represent become more common.

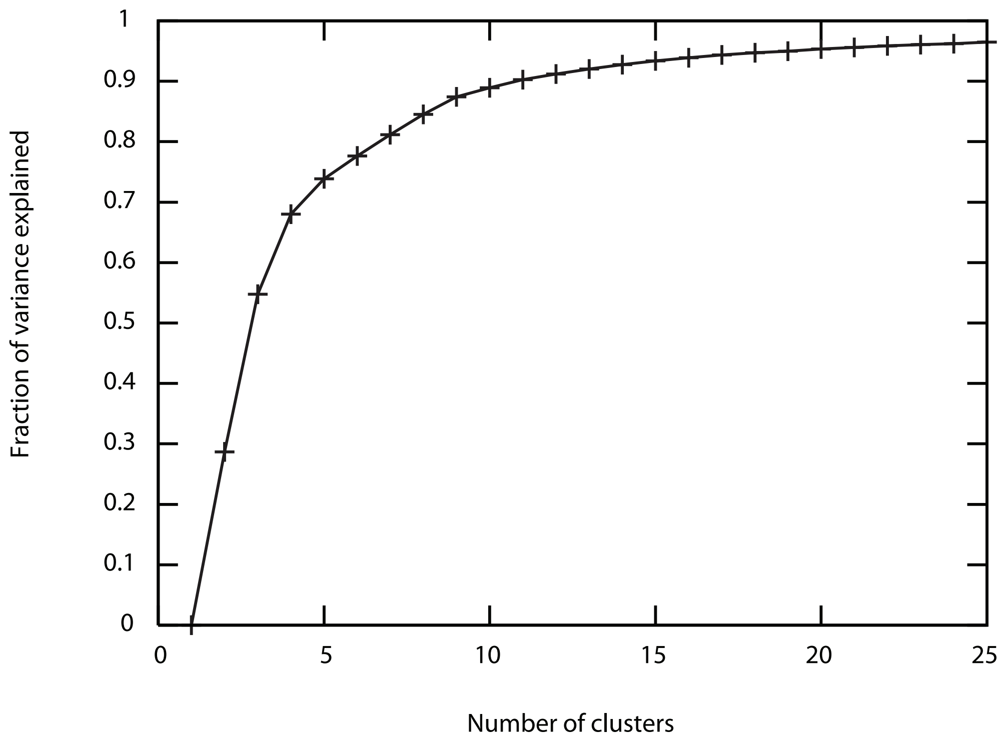


Figure 5.9: Total variance explained for a given number of clusters for Data MDT flows. L_2 norm is used.

The second cluster, *limited receivers*, contains most flows. It represents flows that are approximately 1 hour long, moderate bandwidth, with very few receivers. In fact, this cluster typically has a maximum of 3 receivers and an average of 2. Any improvements or optimizations that apply to these flows could be particularly beneficial to a network, given the size of the cluster.

The third cluster, *long lived*, represents flows that lasted approximately 1 month and consumed moderately high bandwidth. Moreover, the number of receivers, although small, is large enough to imply that multicast technologies are a good mechanism for handling these flows. However, the total number of flows in this category is very small. As such, the impact of these flows over the entire network is minimal.

Finally, we have a cluster, *well-fitted*, that represents moderate length, moderate bandwidth flows with a large number of receivers. The flows in this cluster benefit

Short Name	“Unicast”	“Limited rcv.”	“Long lived”	Well-fitted”
Duration	29 hours, 6 min	1 hour, 12 min	28 days, 2 hours, 10 min	59.3 min
Throughput	11.8 mbits/sec	39.7 kbits/sec	604.9 kbits/sec	20.5 kbits/sec
Peak rate	22.5 mbits/sec	56.3 kbits/sec	983.9 kbits/sec	30.7 kbits/sec
Max. Rcv.	1.4	2.7	9.5	25.4
Avg. Rcv.	1.2	2.1	3.1	19.6
Flows in cluster	0.1%	86.5%	0.05%	13.3%

Table 5.1: The four cluster centers.

greatly from using multicast. They are called *well-fitted* because these types of flows are the ones that derive much benefit from multicast (given the number of receivers). Reducing the router state imposed by these flows is tricky, as they represent “typical” multicast traffic that the protocol was designed for. Although they are the second largest cluster group, they are small in number, and thus are probably not a large contribution to total resource consumption. As such, optimizing them may not be of primary importance.

5.4.3 VPN Analysis

Finally, we analyze how individual customers (more precisely, the VPNs assigned to them) use multicast.

Figure 5.10 depicts the amount of time each VPN spends with at least one flow in the Data MDT, for a time period of one week. While there are a few VPNs that hardly use the Data MDT (less than 30% of their total time), the majority spend at least half of their time in the Data MDT.

Figure 5.11 plots the number of customers with active Data MDT entries over time. Very strong diurnal and weekday/weekend patterns can be seen. This corresponds roughly with what we would expect from enterprises, as they are more likely to be active during business hours. There is significant traffic at night and on weekends as well.

Figure 5.12 plots the CDF of the number of sessions per VPN. In this, we see that most VPNs (50%) only engage in a small number of flows over the course of one

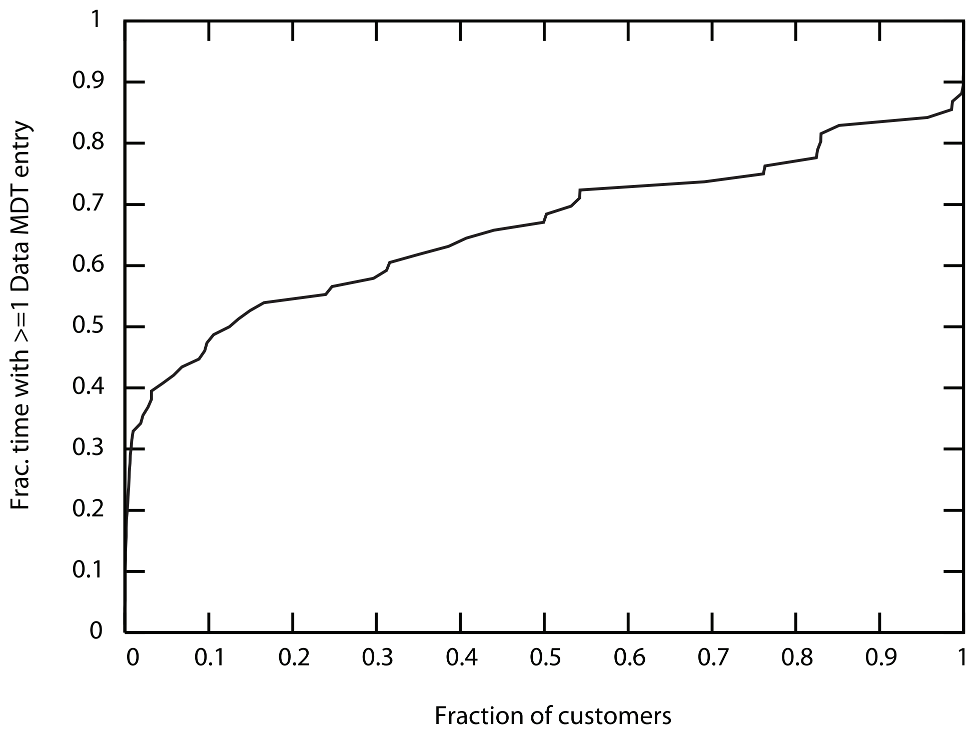


Figure 5.10: CDF of time spent with activity in the Data MDT, on a customer basis.

month. However, there are several heavy hitters that engage in hundreds (and up to a thousand or more) flows during this time period. Thus, although many VPNs use the Data MDT, they often use it in varying amounts.

Lastly, we investigated whether types of companies (*e.g.*, retailers, financial, *etc.*) use the Data MDT in different ways. We label the VPNs with categories and look at the clusters that they fall into. We summarize them in Table 5.2. Categories consist of health related industries, manufacturers, retailers, finance, tech, information services (including consulting and analysis firms), natural resources (either extraction or conservation), and other.

Some interesting trends can be seen. First, the *unicast* style flows are almost entirely confined to the manufacturers and financial companies. The *limited receivers* category was prominent across all categories. For the *long lived* flows, we see a clear

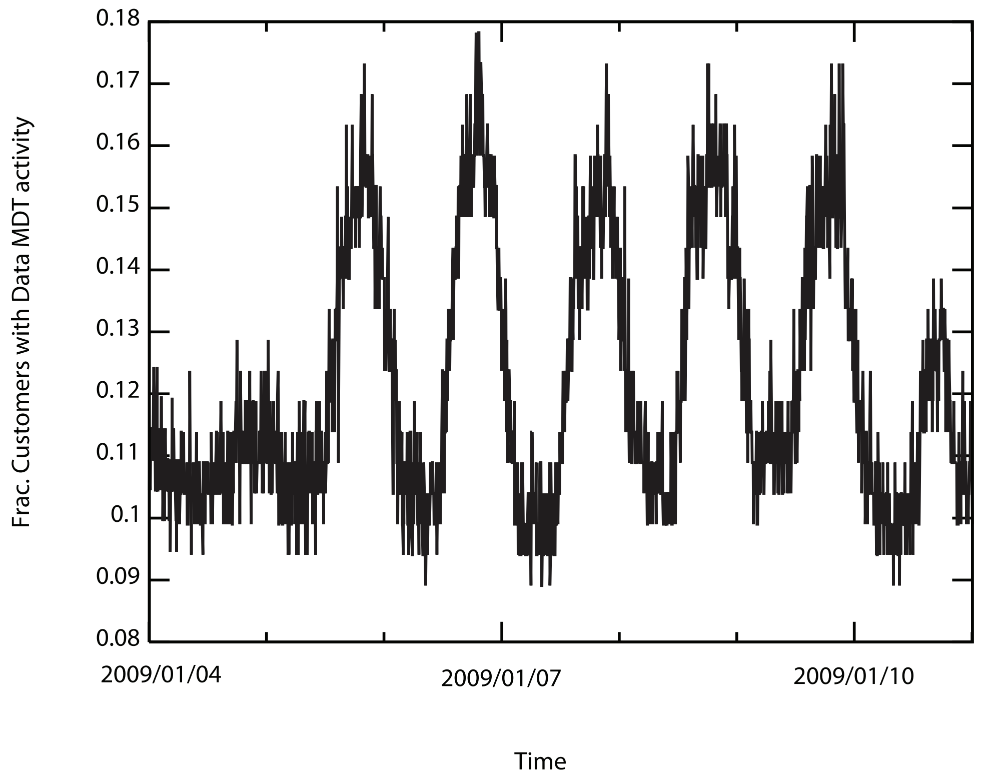


Figure 5.11: Time versus customers with Data MDT activity.

dominance in the health related industries. Finally, *well-fitted* flows were varying across industries.

These results shed some light on our understanding of multicast. The fact that many industries had many flows in the *limited receivers* category suggests that many different kinds of applications have this behavior. In contrast, *unicast* and *long lived* had clear dominators, possibly indicating this behavior is specialized to certain applications. Because of the variance in the *well-fitted* category, it is difficult to draw hard conclusions about the types of traffic that fit this classification. Overall, these results suggest that optimizations to general multicast networks should focus on the *limited receivers* case, and that there is potential for optimizations to the *unicast* and *long lived* flows for specialized cases.

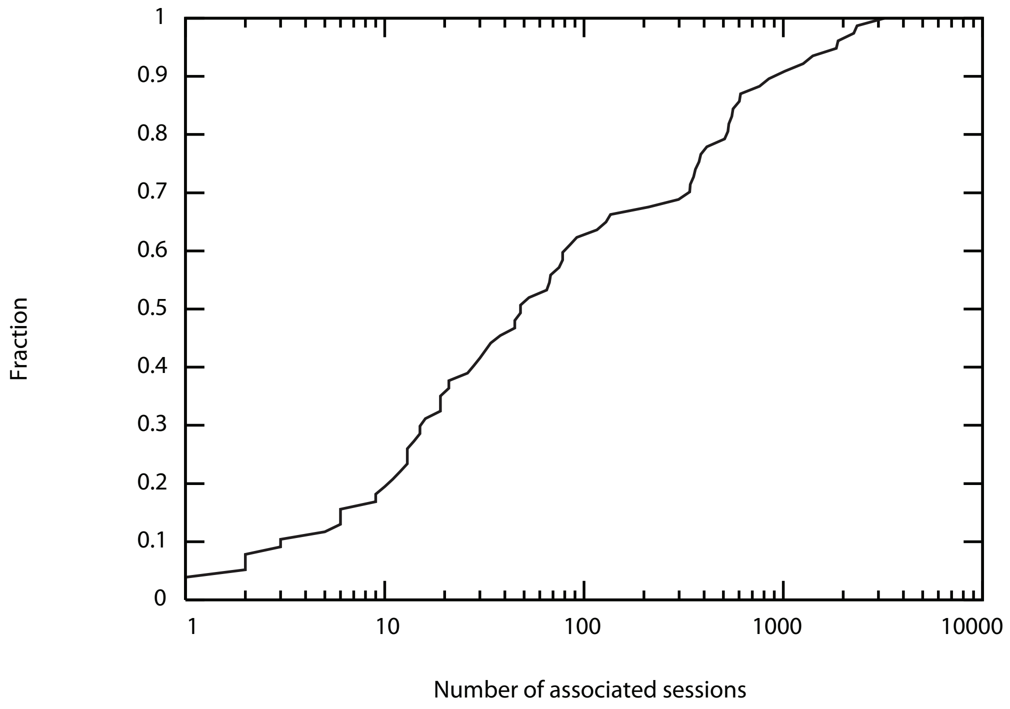


Figure 5.12: CDF of Data MDT sessions per customer.

5.5 Related Work

Our study is unique because, to our knowledge, there have been no other published results analyzing real VPN multicast traffic. However, there have been several areas of research related to multicast in general.

As previously mentioned, the MBone was a multicast backbone network that was free to use. Because not all routers that interconnect networks were multicast enabled, the MBone used tunnels between multicast islands. Unfortunately, this limitation presented scaling challenges as the bandwidth efficiency of multicast can be reduced when tunneling, rather than native multicast is used. While there have been studies done on the MBone [32, 31], as well as general IP multicast [102], our work differs from these since we evaluate VPN customers within a single ISP, as opposed to inter-ISP multicast traffic.

Category	% of *unicast*	% of *limited receivers*	% of *long lived*	% of *well-fitted*
Health Services	0	94.3	2.9	2.9
Manufacturers	0.3	78.5	0.03	21.1
Retailers	0	94	0	6
Finance	0.4	86.1	0.02	13.5
Tech	0	99.6	0	0.4
Information Services	0	75.1	0.04	24.8
Natural Resources	0	100	0	0
Other	0.3	98.5	0	1.2
Average	0.1	86.5	0.05	13.3

Table 5.2: A breakdown of how flows from each cluster center fall into different enterprise categories. Average for all flows across all groups is given at bottom.

To better aid operators, a tool known as VMScope [40] was created to help with network management. It can remotely monitor multicast VPNs and determine characteristics such as packet loss and latency. Deployed at a single location in a network, it provides a congenial interface for operators seeking high-level information about multicast sessions. We consider this work tangential to ours, as we are primarily concerned with longer-term characteristics such as flow duration and throughput.

Some research has continued on multicast protocol improvement. For example, Chainsaw [112] is a peer-to-peer overlay multicast system that does not rely on trees for message propagation. There has been information theoretic work on multicast in coded packet networks, where outgoing packets are generated from incoming packets [101]. Additional theoretical work has been done to shown that re-encoding packets in the middle of a network can result in a large increase to the maximum sending rate [91]. However, this research focuses on theoretical improvements to multicast. Our work supplements this research by providing real usage information to guide future work.

Finally, there has been a large amount of research concerning multicast support for IPTV. For example, the channel surfing problem has been given considerable study, where user behavior is expected to change during commercial breaks [51, 124]. Moreover, measurement studies have been done for IPTV multicast [89]. Because we study a different domain and are not focused on a single application, we consider this work to be complementary to ours.

5.6 Conclusion

Due to the growth of VPN multicast traffic, it is extremely important to understand how organizations are using multicast. Without this information, it is impossible to know how this service will continue to grow and change over the years. Additionally, such information can help us optimize multicast protocols to consume fewer resources.

Our results from a tier 1 ISP show several interesting aspects of multicast traffic. We see a wide distribution of flow duration, although most flows tend to be short-lived. Likewise, many flows use low or moderate amounts of bandwidth, with a small number of flows with very high throughput. Moreover, we see potential opportunities to make the Multicast VPN service more efficient. The large number of single egress PE flows in the provider domain indicates that unicast could be used as a replacement, resulting in no impact to efficiency but considerably less routing state. Likewise, a significant number of flows only communicate with a handful of receivers; converting these to unicast streams would decrease bandwidth efficiency, but greatly cutback on memory requirements.

There is future work to consider. First, it would be interesting to do a longer, longitudinal study on multicast traffic to understand the evolution of enterprise customer behavior. Second, this research does not extensively evaluate memory and bandwidth trade-offs present in today's networks. Further analysis to identify mechanisms for optimizing multicast memory usage multicast should be considered. Finally, a lower level application layer diagnosis of multicast traffic could provide insights into how particular applications are leveraging multicast technologies.

Chapter 6

Conclusion

Previous chapters have covered two kinds of router memory associated with BGP, the RIB and the FIB, and discussed mechanisms to reduce their resource requirements. Moreover, these chapters also provided insights into the emerging IPv6 and multicast protocols and they are used in practice today. This chapter concludes the thesis by discussing how Forgetful Routing and the MMS can be combined to simultaneously protect the RIB and FIB; in addition, the chapter looks at possible future directions, such as how the MMS could be used to enable other solutions, and how IPv6 and multicast memory could be reduced in the future.

6.1 Simultaneously Reducing the RIB and FIB

Although Forgetful Routing and the Memory Management System were described separately, it is possible to combine them within a single network. This hybrid can reduce both RIB and FIB memory consumption. However, the mode of deployment for the MMS directly affects its interactions with Forgetful Routing.

In the centralized mode, the MMS assumes responsibility of route propagation. Routers in the network no longer make routing decisions, and their RIBs contain stripped, coalesced entries. As such, these RIBs are already “compressed” and Forgetful Routing cannot be applied to these routers. However, the MMS itself can still benefit. By using border routers as proxies, the MMS can issue refresh messages to neighboring domains (as well as receive responses). Since the MMS stores an even larger RIB than what a router maintains, Forgetful Routing may be able to provide extra savings. Latency concerns and convergence delays become more problematic, since the refreshed information will be multiple hops away. Future research could determine if this extra delay is tolerable and/or reasonable.

In the distributed mode, Forgetful Routing may need some modification to work in unison with the MMS. If the full BGP decision process is always used, Forgetful Routing can be applied without any changes. If the MMS skips some BGP decision steps, Forgetful Routing must not evict "equally good" routes; otherwise, the router will be overly constrained during the MMS' coalescing process. However, other "worse" RIB entries can be discarded. Moreover, while the selected route should have its attributes preserved, the other "equally good" routes can be stripped of all attributes except $next - hop$, saving some space.

With these precautions in mind, Forgetful Routing and the MMS can be combined to reduce FIB and RIB memory consumption.

6.2 Enabling Clean Slate Design

The solutions proposed in this thesis attempt to push the limits of current protocols, with respect to efficiency. Given the constraint of backwards compatibility, is it possible to substantially reduce memory any further? If Forgetful Routing and the MMS are nearly optimal solutions, then the only way to significantly shrink memory is through backwards *in*compatible solutions. However, such clean slate designs have trouble with deployment because of a bootstrapping problem: deploying the solution incurs expense, and an AS cannot reap any reward unless others deploy. Without a critical mass of adopters, organizations will not to switch to these designs.

Although the MMS is backwards compatible, it can also act as a clean slate. If multiple ASes deploy centralized MMSes, these MMSes can be configured to communicate with each other directly. The MMS would still use standard protocols (like BGP) when communicating with legacy networks. However, the inter-MMS protocol has no constraints, and could support a clean slate design. Since the MMS provides incentives for deployment, it could circumvent the aforementioned bootstrapping problem.

6.3 Future Protocols and Memory Reduction

Forgetful Routing and MMS techniques can be equally applied to the IPv6 protocol. The Border Gateway Protocol operates on top of an Internet Protocol. Although this thesis evaluated BGP with IPv4, BGP operates in the same way with IPv6. The amount of memory that can be saved, however, may differ, and will depend on several questions. How fast does the number of prefixes grow in IPv6? Can IPv6 prefixes be coalesced easily? How many routing paths will exist per prefix? If IPv6 grows to replace IPv4, these questions can be answered and memory savings can be quantified. Even if the savings are not substantial, as long as *some savings* are possible, these techniques can be applied to reap the benefits.

However, Forgetful Routing and the Memory Management System do not appear applicable to multicast. Forgetful Routing and the MMS are designed for BGP, a fundamentally different protocol. A multicast router has no memory component that is similar to a RIB, and a multicast router FIB uses (S, G) entries that may not coalesce easily. Instead, the amount of memory needed for multicast can be reduced by using different techniques, such as converting some multicast flows into unicast flows). However, because of the nascent nature of multicast, it is unclear if other optimizations exist and how they should be applied. As such, the growth of multicast should be monitored.

6.4 Summary

Through Forgetful Routing and the Memory Management System, router memory today can be greatly reduced. Forgetful Routing introduces a space-time trade-off to reduce RIB memory at the cost of additional time to apply the BGP decision process. The Memory Management System provides a trade-off between FIB memory reduction and path quality. Both are backwards compatible with BGP, the *de facto* interdomain routing protocol of today.

The future, however, is uncertain. Upcoming protocols such as IPv6 and multicast have potential to substantially change the routing landscape. Because these protocols are nascent, it is difficult to evaluate their eventual impact. Instead of uncertain extrapolations, this thesis quantifies their usage and helps lay the groundwork for future research. In addition, preliminary analysis of multicast shows that there is potential to reduce the routing state associated with it.

Forgetful Routing and the Memory Management System are techniques for contemporary routers. As of today, ASes can protect their networks from memory overflow with these optimizations. The IPv6 and multicast studies are observations about the growth and usage of upcoming protocols; by studying how they evolve, operators and researchers can prepare themselves for the challenges of tomorrow.

Bibliography

[1] ARIN public stats. `ftp://ftp.arin.net/pub/stats/`.

[2] BGP reports. `http://bgp.potaroo.net`.

[3] CISCO IOS NetFlow. `http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html`.

[4] Enhanced interior gateway routing protocol. `http://www.cisco.com/warp/public/103/eigrp-toc.html`.

[5] Guidelines for 64-bit Global Identifier (EUI-64) Registration Authorit y. `http://standards.ieee.org/regauth/oui/tutorials/EUI64.html`.

[6] How much does planet Earth weigh? `http://science.howstuffworks.com/question30.htm`.

[7] I Root Server. `http://www.netnod.se/dns_root_nameserver.shtml`.

[8] IANA internet assigned numbers authority. `http://www.iana.org/`.

[9] Index. `ftp://ftp.arin.net/pub/stats/`.

[10] Internet Resource Management Policies in Latin America and the Caribbean. `http://lacnic.net/en/politicas/ipv6.html`.

[11] IPv4 Address Report. `http://www.potaroo.net/tools/ipv4/index.html`.

[12] K Root Server. `http://k.root-servers.org`.

[13] Query the RIPE Database. `http://www.db.ripe.net/whois?`

[14] Routing research group (RRG). `http://tools.ietf.org/group/irtf/trac/wiki/RoutingResearchGroup`. IETF-60.

[15] IPv6 Assignment and Allocation Policy Document. `ftp://ftp.ripe.net/ripe/docs/ripe-196.txt`, July 1999.

[16] Policy - AfriNIC IPv6 Address Allocation and Assignment Policy. `http://www.afrinic.net/docs/policies/afpol-v6200407-000.htm`, March 2004.

[17] AfriNIC Global policy proposal for IPv6 allocation form IANA to RIRs. `http://www.afrinic.net/docs/policies/drafts/afpol-glbipv6200508.htm`, August 2005.

[18] Domain Name System Client Behavior in Windows Vista. `http://technet.microsoft.com/en-us/library/bb727035.aspx`, September 2006.

[19] IPv6 Address Allocation and Assignment Policy. `http://www.ripe.net/ripe/docs/ipv6policy.html`, November 2007.

[20] Policy to Change the IPv6 HD ratio from 0.8 to 0.94. `http://www.afrinic.net/docs/policies/afpol-ipv6hdr20070330.htm`, March 2007.

[21] Teredo Overview. `http://technet.microsoft.com/en-us/library/bb457011.aspx`, January 2007.

[22] Address Space Managed by the RIPE NCC. `http://www.ripe.net/docs/ripe-434.html`, July 2008.

[23] ARIN Number Resource Policy Manual. `http://www.arin.net/policy/nrpm.html`, August 2008.

[24] The BGP instability report. http://bgpupdates.potaroo.net/instability/bgpupd.html, May-June 2008.

[25] forum.utorrent.com / uTorrent 1.8 released. http://forum.utorrent.com/viewtopic.php?id=44003, August 2008.

[26] Internet Protocol Version 6 Address Space. http://www.iana.org/assignments/ipv6-address-space, May 2008.

[27] IPv6 Address Allocation and Assignment Policy. http://www.apnic.net/policy/ipv6-address-policy.html, August 2008.

[28] IPv6 Global Unicast Address Assignments. http://www.iana.org/assignments/ipv6-unicast-address-assignments, May 2008.

[29] SixXS - IPv6 Deployment & Tunnel Broker :: IPv6 BitTorrent Clients. http://www.sixxs.net/tools/tracker/clients/, September 2008.

[30] Z. Albanna, K. Almeroth, D. Meyer, and M. Schipper. IANA Guidelines for IPv4 Multicast Address Assignments. http://tools.ietf.org/html/rfc3171, August 2001. Request for Comments: 3171.

[31] K. Almeroth. A long-term analysis of growth and usage patterns in the Multicast Backbone (MBone). *IEEE Infocom*, 2000.

[32] K. Almeroth and M. Ammar. Multicast group behavior in the Internet's multicast backbone (MBone). *IEEE Communications Magazine*, 1997.

[33] K. Asanovic, R. Bodic, B. Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams, and K. Yelick. The landscape of parallel computing research: A view from Berkeley. Technical report, University of California, Berkeley, December 2006. No. UCB/EECS-2006-183.

[34] H. Ballani, P. Francis, T. Cao, and J. Wang. Making Routers Last Longer with ViAggre. In *Proc. NSDI*, 2009.

[35] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (CBT). *SIGCOMM Comput. Commun. Rev.*, 23(4), 1993.

[36] A. Basu, C. Ong, F. Rasala, B. Shepherd, and G. Wilfong. Route oscillations in IBGP with route reflection. In *Proc. ACM SIGCOMM*, August 2002.

[37] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong. Route oscillations in I-BGP with route reflection. In *Proc. ACM SIGCOMM*, pages 235–247, 2002.

[38] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. `http://tools.ietf.org/html/rfc2796`, April 2000. Request for Comments: 2796.

[39] V. Bono. 7007 explanation and apology. `http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html`.

[40] L. Breslau, C. Chase, N. Duffield, B. Fenner, Y. Mao, and S. Sen. VMScope: a virtual multicast VPN performance monitor. In *Proc. SIGCOMM workshop on Internet network management*, 2006.

[41] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM Conference on Mobile Computing and Networking*, October 1998.

[42] A. Broido and kc claffy. Analysis of RouteViews BGP data: policy atoms. In *Proc. Network-Related Data Management (NRDM) Workshop*, Santa Barbara, May 2001.

[43] T. Bu, L. Gao, and D. Towsley. On characterizing BGP routing table growth. *Computer Networks*, 45(1), 2004.

[44] R. Bush. Email conversation, April 2006.

[45] R. Bush. report on prop-057: Proposal to change IPv6 initial allocation criteria. `http://mailman.apnic.net/mailing-lists/sig-policy/archive/2008/02/msg00015.html`, February 2008.

[46] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *Proc. NSDI*, April 2005.

[47] B. Carpenter and K. Moore. Connection of IPv6 Domains via IPv4 Clouds. `http://tools.ietf.org/html/rfc3056`, February 2001. Request for Comments: 3056.

[48] D.-F. Chang, R. Govindan, and J. Heidemann. An empirical study of router response to large BGP routing table load. In *Proc. Internet Measurement Workshop*, 2002.

[49] E. Chen. Cooperative route filtering capability for BGP-4. `http://www.ietf.org/internet-drafts/draft-ietf-idr-route-filter-13.txt`. IETF Internet Draft, expires September 2006.

[50] E. Chen. Route Refresh Capability for BGP-4. `http://tools.ietf.org/html/rfc2918`, September 2000. Request for Comments: 2918.

[51] C. Cho, I. Han, Y. Jun, and H. Lee. Improvement of channel zapping time in IPTV services using the adjacent groups join-leave method. *The 6th International Conference on Advanced Communication Technology*, 2004.

[52] Cisco. BGP best path selection algorithm. http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094431.shtml, August 2005.

[53] Cisco. Troubleshooting memory problems. http://www.cisco.com/warp/public/63/mallocfail.shtml, October 2005.

[54] How BGP Routers Use the Multi-Exit Discriminator for Best Path Selection. http://www.cisco.com/warp/public/459/37.html.

[55] B. Davie and Y. Rekhter. *MPLS: Technology and applications*. 2000.

[56] S. Deering and D. R. Cheriton. Host Groups: A Multicast Extension to the Internet Protocol. http://tools.ietf.org/html/rfc966, December 1985. Request for Comments: 966.

[57] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. An architecture for wide-area multicast routing. *SIGCOMM Comput. Commun. Rev.*, 24(4), 1994.

[58] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2), 1990.

[59] J. Doyle. IPv6 Hour at NANOG: A Follow-Up. http://www.networkworld.com/community/node/25276, February 2008.

[60] R. Draves, C. King, S. Venkatachary, and B. Zill. Constructing optimal IP routing tables. In *Proc. IEEE INFOCOM*, 1999.

[61] R. Dube. A comparison of scaling techniques for BGP. In *ACM Computer Communication Review*, volume 29, July 1999.

[62] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID Separation Protocol (LISP). http://tools.ietf.org/html/draft-farinacci-lisp-12, 2009. Internet-Draft Version 12 (work in progress).

[63] D. Farinacci, T. Li, S. Hanks, D. Meyere, and P. Traina. Generic Routing Encapsulation (GRE). http://tools.ietf.org/html/rfc2784, March 2000. Request for Comments: 2784.

[64] N. Feamster. Interdomain routing correctness and stability. http://nms.csail.mit.edu/6.829-f05/lectures/L5-rtg_correctness_slides.ppt.

[65] Federal CIO Council Architecture and Infrastructure Committee . IPv6 Transition Guide. http://www.cio.gov/documents/IPv6_Transition_Guidance.doc, February 2006.

[66] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). http://tools.ietf.org/html/rfc4601, August 2006. Request for Comments: 4601.

[67] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6), 1997.

[68] R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point in wireless sensornets. In *Proc. NSDI*, May 2004.

[69] P. Francois, O. Bonaventure, M. Shand, S. Previdi, and S. Bryant. Ordered FIB updates. In *Proc. of the 65th Internet Engineering Task Force*. http://www.ietf.org/proceedings/06mar/slides/rtgwg-1.pdf, March 2006.

[70] V. Fuller and T. Li. Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. http://tools.ietf.org/html/rfc4632, August 2006. Request for Comments: 3411.

[71] J. Gailly and M. Adler. The gzip home page. http://www.gzip.org.

[72] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Trans. Networking*, 9:733–745, 2001.

[73] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, December 2001.

[74] P. Gupta. Address Lookup and Classification. In *Course lecture*, May 2006. www.stanford.edu/class/ee384y/Handouts/lookup-and-classification-lec2.ppt.

[75] S. Halabi and D. McPherson. *Internet Routing Architectures.* Second edition, 2001.

[76] D. Harrington, R. Preshun, and B. Wijnen. An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks. http://tools.ietf.org/html/rfc3411, December 2002. Request for Comments: 3411.

[77] Y. Hei and K. Yamazaki. Traffic analysis and worldwide operation of open 6to4 relays for IPv6 deployment. In *Proc. 2004 Symposium on Applications and the Internet.*

[78] B. Hinden. 6bone Phaseout Planning. http://go6.net/ipv6%2D6bone/ngtrans/IETF-56-SanFrancisco/6bone-phaseout.pdf, March 2003.

[79] R. Hinden, M. O'Dell, and S. Deering. An IPv6 Aggregatable Global Unicast Address Format. http://tools.ietf.org/html/rfc2374, July 1998. Request for Comments: 2374.

[80] M. Hoerdt and D. Magoni. Distribution of multicast tree states over the IPv6 network topology. In *2004 IEEE Conference on Communications*.

[81] Y. Hu and V. O. K. Li. Satellite-based Internet: A tutorial. *IEEE Communication Magazine*, 39:154–162, March 2001.

[82] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel. INTERNET REGISTRY IP ALLOCATION GUIDELINES. http://tools.ietf.org/html/rfc2050, November 1996. Request for Comments: 2050.

[83] B. Huffaker and K. Claffy. Caida : research : topology : as_core_network. http://www.caida.org/research/topology/as_core_network/ipv6.xml.

[84] G. Huston. CIDR report. http://www.cidr-report.org/.

[85] G. Huston. IPv6 Reports. http://bgp.potaroo.net/index-v6.html.

[86] G. Huston. Re: [ipv6-wg] 2001:7f9::/32 still being announced even though the 'experiment' has finished 2 days ago? www.ripe.net/ripe/maillists/archives/ipv6-wg/2006/msg00036.html. IPv6 WG mailing list.

[87] G. Huston. Analyzing the Internet BGP routing table. *Internet Protocol Journal*, 4(1), March 2001.

[88] S. Iekel-Johnson, C. Labovitz, D. McPherson, and H. Ringberg. Tracking the IPv6 Migration. http://www.arbornetworks.com/IPv6research, 2008.

[89] K. Imran, M. Mellia, and M. Meo. Measurements of Multicast Television over IP. In *IEEE Workshop on Local and Metropolitan Area Networks*, 2007.

[90] Information Sciences Institute. INTERNET PROTOCOL - DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION. `http://tools.ietf.org/html/rfc791`, September 1981. Request for Comments: 791.

[91] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Transactions on Information Theory*, 51(6), June 2005.

[92] Juniper. Errors and error status messages. `http://www.juniper.net/techpubs/software/junos/junos72/swconfig72-fips/html/FIPS-mode5.html`.

[93] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 2002.

[94] E. Karpilovsky and J. Rexford. Using forgetful routing to control BGP table size. In *Proc. ACM International Conference on emerging Networking EXperiments and Technologies (ACM CoNEXT)*, 2006.

[95] C. Kim, February 2006. email conversation.

[96] C. Kim, A. Gerber, C. Lund, D. Pei, and S. Sen. Scalable VPN Routing via Relaying. In *Proc. ACM SIGMETRICS*, 2008.

[97] D. Krioukov, k c Claffy, K. Fall, and A. Brady. On compact routing for the Internet. In *ACM Computer Communication Review*, July 2007.

[98] H.-T. Kung and B. Karp. Greedy perimeter stateless routing for wireless networks. In *Proc. ACM Conference on Mobile Computing and Networking*, August 2000.

[99] B. N. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ack trees. In *Proc. ACM International Conference on Multimedia*, 1996.

[100] T. Li. Router scalability and Moore's law. In *Internet Architecture Board meeting (presentation)*, October 2006. `http://www.iab.org/about/workshops/routingandaddressing/Router_Scalability.pdf`.

[101] D. Lun, N. Ratnakar, M. Medard, R. Koetter, D. Karger, T. Ho, E. Ahmed, and F. Zhao. Minimum-cost multicast over coded packet networks. *IEEE Transactions on Information Theory*, 52(6), June 2006.

[102] B. Mah. Measurements and Observations from IP Multicast Traffic. Technical report, UC Berkeley, 1994. CSD-94-858.

[103] D. Malone. Observations of IPv6 Addresses. In *Proc. Passive and Active Measurement*. Springer Berlin / Heidelberg, 2008.

[104] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz. Towards an accurate AS-level traceroute tool. In *Proc. ACM SIGCOMM*, 2003.

[105] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 address allocation and the BGP routing table evolution. *ACM Computer Communication Review*, 2005.

[106] D. Meyer. University of Oregon Route Views Archive Project. `http://archive.routeviews.org/`.

[107] J. Moy. *OSPF: Anatomy of an Internet Routing Protocol*. 1998.

[108] NANOG. Looking glass sites. `http://www.nanog.org/lookingglass.html`.

[109] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. http://tools.ietf.org/html/rfc4941, September 2007. Request for Comments: 4941.

[110] D. Oran. OSI IS-IS Intra-domain Routing Protocol. http://tools.ietf.org/html/rfc1142, February 1990. Request for Comments: 1142.

[111] B. O'Sullivan. The Internet Multicast Backbone. http://ntrg.cs.tcd.ie/undergrad/4ba2/multicast/bryan/index.html. Accessed March 2009.

[112] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and E. E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Peer-to-Peer Systems IV, Volume 3640*, 2005.

[113] A. Percy. Understanding latency in ip telephony. Brooktrout Technology, February 1999.

[114] R. Perlman. *Interconnections: Bridges, routers, switches, and internetworking protocols.* Second edition, 1999.

[115] A. Popescu, B. Premore, and T. Underwood. The anatomy of a leak: AS9121, May 2005. NANOG presentation, http://www.nanog.org/mtg-0505/pdf/underwood.pdf.

[116] M. Quinn and N. Deo. Parallel graph algorithms. *ACM Computing Surveys*, September 1984.

[117] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure. Evaluating the benefits of the locator/identifier separation. In *Proc. ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*. ACM, 2007.

[118] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP routing stability of popular destinations. In *Proc. Internet Measurement Workshop*, November 2002.

[119] B. Roseman. ASO Statement Regarding IPv6 Policy Adoption. http://www.icann.org/en/aso/ipv6-statement-11jul02.htm, July 2002.

[120] E. Rosen and R. Aggarwal. Multicast in MPLS/BGP IP VPNs. http://www.ietf.org/internet-drafts/draft-ietf-l3vpn-2547bis-mcast-08.txt, March 2009. Network Working Group Internet Draft.

[121] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. http://tools.ietf.org/html/rfc3031, January 2001. Request for Comments: 3031.

[122] P. Savola. Observations of IPv6 Traffic on a 6to4 Relay. *ACM Computer Communication Review*, January 2005.

[123] W. F. Slater III. The Internet Outage and Attacks of October 2002. Chicago Chapter of the Internet Society, November 2002. www.isoc-chicago.org/internetoutage.pdf.

[124] D. Smith. IP TV Bandwidth Demand: Multicast and Channel Surfing. In *IEEE International Conference on Computer Comunications*, 2007.

[125] P. Smith. IPv6 Hour at NANOG42. http://www.nanog.org/mtg-0802/ipv6hour.html, January 2008.

[126] P. Smith, R. Evans, and M. Hughes. RIPE routing working group recommendations on route aggregation. http://www.ripe.net/ripe/docs/ripe-399.html, December 2006.

[127] J. W. Stewart. *BGP4 Inter-Domain Routing in the Internet.* 1998.

[128] S. Still, W. Bialek, and L. Bottou. Geometric clustering using the information bottleneck method. *Advances in Neural Information Processing Systems*, 16, 2004. MIT Press.

[129] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, August 2001.

[130] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proc. IEEE INFOCOM*, June 2002.

[131] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, 2001.

[132] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. `http://tools.ietf.org/html/rfc3065`, February 2001. Request for Comments: 3065.

[133] P. Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large networks. In *Proc. ACM SIGCOMM*, March 2006.

[134] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. Snoeren, and J. V. der Merwe. Wresting control from BGP: Scalable fine-grained route control. In *Proc. USENIX Annual Technical Conference*, June 2007.

[135] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. `http://tools.ietf.org/html/rfc1075`, November 1988. Request for Comments: 1075.

[136] Y. Wang, I. Avramopoulos, and J. Rexford. Design for configurability: Rethinking interdomain routing policies from the ground up. *IEEE Journal on Selected Areas in Communications*, 2009.

[137] X. Zhang, P. Francis, J. Wang, and K. Yoshida. Scaling IP routing with the core router-integrated overlay. In *Proc. International Conference on Network Protocols*, November 2006.