COMPACT ROUTING FOR THE FUTURE INTERNET

STEPHEN D. STROWES

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy

SCHOOL OF COMPUTING SCIENCE

COLLEGE OF SCIENCE AND ENGINEERING
UNIVERSITY OF GLASGOW

FEBRUARY 2012

Abstract

The Internet relies on its inter-domain routing system to allow data transfer between any two endpoints regardless of where they are located. This routing system currently uses a shortest path routing algorithm (modified by local policy constraints) called the Border Gateway Protocol. The massive growth of the Internet has led to large routing tables that will continue to grow. This will present a serious engineering challenge for router designers in the long-term, rendering state (routing table) growth at this pace unsustainable.

There are various short-term engineering solutions that may slow the growth of the inter-domain routing tables, at the expense of increasing the complexity of the network. In addition, some of these require manual configuration, or introduce additional points of failure within the network. These solutions may give an incremental, constant factor, improvement. However, we know from previous work that all shortest path routing algorithms require forwarding state that grows linearly with the size of the network in the worst case.

Rather than attempt to sustain inter-domain routing through a shortest path routing algorithm, *compact* routing algorithms exist that guarantee worst-case sub-linear state requirements at all nodes by allowing an upper-bound on path length relative to the theoretical shortest path, known as *path stretch*. Previous work has shown the promise of these algorithms when applied to synthetic graphs with similar properties to the known Internet graph, but they haven't been studied in-depth on Internet topologies derived from real data.

In this dissertation, I demonstrate the consistently strong performance of these compact routing algorithms for inter-domain routing by performing a longitudinal study of two compact routing algorithms on the Internet Autonomous System (AS) graph over time. I then show, using the k-cores graph decomposition algorithm, that the structurally important nodes in the AS graph are highly stable over time. This property makes these nodes suitable for use as the "landmark" nodes used by the most stable of the compact routing algorithms evaluated, and the use of these nodes shows similar strong routing performance. Finally, I present a decentralised compact routing algorithm for dynamic graphs, and present state requirements and message overheads on AS graphs using realistic simulation inputs.

To allow the continued long-term growth of Internet routing state, an alternative routing architecture may be required. The use of the compact routing algorithms presented in this dissertation offer promise for a scalable future Internet routing system.

Acknowledgements

I have been lucky through my PhD to have had an advisory team that has taken an active interest in my work. My thanks go to Colin Perkins, my supervisor, for all the time and the red ink. My thanks also go to Joe Sventek and Peter Dickman who were, respectively, latterly and formerly my secondary supervisors.

During my PhD I had the opportunity to take on an internship position at the Nokia Research Center in Espoo, Finland. For this, my gratitude to Lars Eggert and Pasi Sarolahti.

Finally, for the beer and for the laughs, my warmest thanks must go to all those who I shared office space with, some of whom had the pleasure of proof-reading sections of this dissertation. In alphabetical order: Martin Ellis, Paul Jakma, Paul Harvey, Michio Honda, Vamsi Kambhampati, Alexandros Koliousis, Ross McIlroy, Rachel Lo, Oliver Sharma.

To the inhabitants of the islands of arno, carney, cochrane, contreras, puffin, sibu, and soroya, without whom this dissertation would not have been possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet has seen massive growth since its inception, and in particular after interconnection of the research networks with commercial network owners was permitted in 1988. One of the keys to its success has been the network architecture. The Internet is designed such that most of the complex functionality resides on the computers that connect to the network, while the network itself need only concern itself with *routing* data between those computers. This has permitted the development of varied, and often complex, functionality such as web servers that users can access or peer-to-peer conference calling, without requiring modification to the network itself.

Ultimately, however, the growth of the network, *i.e.* the continued expansion of the Internet, will become a problem. The Internet is organised as a collection of interconnected networks. Its routing system operates by holding information for all destination networks to allow messages to be routed from any source to any destination. Since it stores references for all destinations, when data is sent from one computer to another, the network knows the shortest paths to use to transmit that data (subject also to policy and traffic engineering constraints). Thus, while the edge of the network bears the brunt of application complexity, the network bears the brunt of the massive growth of the Internet, since routing hardware must keep state for every destination. The current Internet architecture and the way it has grown is enumerated in Chapter 2.

The problem with the current network design is that the pace of growth is ultimately unsustainable. The fabrication of new routing hardware that is capable of handling this workload is already a difficult engineering task with core routers being expensive and power-hungry, and will be more so in the future. The architecture must change if the network growth is to be sustained in the long term. Alternative architectures that have been proposed are discussed in Chapter 3.

Rather than attempt to retain information representing all endpoints active in the network and continue engineering more complex, more costly, routers, an alternative approach could

be to instead reduce the volume of information held in the network. Previous work in the field of *compact routing* has explored precisely how to construct routing architectures that do not require holding information representing all endpoints. In particular, a smarter architecture can make use of *landmarks* as an aid when routing toward a destination, and limit the visibility of the destination to only a small subset of the full network. Conceptually, utilising landmarks in a network is similar to how road signs are designed to be used: distant destinations are aggregated into cities, counties, and even countries, but nearby destinations are described in considerably more detail.

Compact routing algorithms exist that use landmarks as a mechanism to aid scaling to larger networks. This dissertation explores the application of these algorithms to the Internet, and attempts to address the problems inherent in the algorithms to move toward a realistic solution to the Internet's routing scalability problem. I present this primarily in three stages. First, the suitability of compact routing algorithms to the Internet must be examined. In Chapter 4, I study the application of two such algorithms to Internet topologies dating back to 1997 and show that the algorithms handle routing between any two destinations with only small amounts of routing state, the trade-off being slightly longer paths in a minority of cases.

Next, for landmarks to be useful they must be more stable than the destinations themselves, or they do not provide any benefit. In Chapter 5, I study the stability of the Internet topology as a basis for landmark-based routing on the Internet, and show that the network is extremely stable.

Finally, the algorithm must be decentralised across the full Internet, or it is susceptible to failure. In Chapter 6, I describe in detail a fully decentralised routing protocol based on compact routing that utilises the stability of the network to enable sustainable routing in the long term. This protocol finds the shortest possible paths in the vast majority of cases.

1.1 Thesis Statement

I assert that compact routing algorithms provide a potential solution on which a future routing architecture can be built, a solution which offers near-shortest-path routing performance with routing tables that grow sublinearly in the vast majority of routers. I will demonstrate this by:

- First, performing a longitudinal study of two compact routing algorithms (the Brady-Cowen compact routing algorithm, and the Thorup-Zwick compact routing algorithm) on static Internet topologies derived from real-world data, assessing them for path

lengths and routing table sizes to show that routing tables are small and path stretch is bounded.

- Second, by evaluating the stability at the heart of the inter-domain Internet by applying the k-cores graph decomposition algorithm, and showing that this stable set is beneficial for a deployment of a routing protocol based on compact routing by also considering the geographical and topological suitability of the stable set.
- Third, by defining and evaluating a decentralised network protocol that uses compact routing principles, and is suitable for use on dynamic networks.

These algorithms aim to reduce forwarding state at all nodes, and in order to do so they cannot guarantee shortest path routing. Instead, they provide an upper-bound on the increase in path length relative to the shortest possible path. Thus, evaluation of these algorithms is primarily in terms of table sizes and path lengths.

1.2 Contributions

The contributions of this work are as follows:

1. an in-depth longitudinal study of the performance of the Thorup-Zwick compact routing algorithm and the Brady-Cowen compact routing algorithm on numerous snapshots of the Internet topology spanning 15 years. Previous studies used synthetic graphs, or solitary snapshots of the Internet topology.
2. an in-depth longitudinal study of the k-cores graph decomposition on Internet topologies spanning 15 years. Previous studies used synthetic graphs, or solitary snapshots of the Internet topology. These data are used to form the basis of a new algorithm to select landmarks on Internet topologies.
3. results for the stretch performance and table sizes for the Thorup-Zwick compact routing algorithm when applying the landmarks selected by the algorithm from point 2, above, to Internet topologies.
4. a description of a decentralised routing protocol based on the Thorup Zwick compact routing algorithm, with discussion of the trade-offs involved. Performance results are presented that demonstrate large reductions in path lengths when using compact routing protocols simply by exposing more information about the landmark set.

1.3 Publications

This work has produced two publications so far:

- *"Compact Routing on the Internet AS-Graph"*,
  S. D. Strowes, G. Mooney, and C. S. Perkins,
  in Proceedings of the 14th Global Internet Symposium, April 2011.
- *"Harnessing Internet Topological Stability in Thorup-Zwick Compact Routing"*,
  S. D. Strowes and C. S. Perkins,
  in Proceedings of IEEE INFOCOM 2012 Mini Conference, March 2012.

1.4 Dissertation Outline

This dissertation is structured as follows:

Chapter 2: looks at the deployed Internet architecture and its actual growth over time, starting from November 1997.

Chapter 3: covers related works in the areas of Internet scalability, and relevant background on routing theory, including compact routing.

Chapter 4: covers a longitudinal study of two compact routing algorithms applied to Internet graphs. This chapter highlights the stretch performance and routing table sizes generated by both these algorithms.

Chapter 5: studies the stability of the Internet graph during the same time period studied in Chapter 4. In doing so, I present a new landmark selection algorithm tuned to Internet graphs specifically, but which should be suited for many power law graphs in general.

Chapter 6: discusses the decentralisation of the routing protocol, based on an assumption that there exists a mechanism to derive landmark nodes as discussed in Chapter 5. This chapter presents routing protocol convergence times and an analysis of the control traffic overhead of the protocol from simulation. It also presents path stretch results and routing table sizes as in previous chapters. Finally, this chapter covers some details of the application of a compact routing protocol to realistic Internet operations.

Chapter 7: outlines directions for future work, and concludes the dissertation.

Additionally, there are two appendices:

Appendix A: this dissertation uses lots of real-world data in various different ways. This appendix covers the data handling and cleansing process that took place.

Appendix B: describes the simulator software that was used to handle the simulation that generated many of the results presented in the preceding chapters.

Chapter 2

Background on Internet Growth

Shortest path routing protocols are typically used in networks to minimise the lengths of paths traversed between nodes. The Border Gateway Protocol (BGP) is the protocol that is currently used on the Internet to exchange and propagate routing advertisements between distinct networks, Autonomous Systems, thus facilitating inter-domain packet forwarding. An Autonomous System (AS) is defined as "a connected group of one or more IP prefixes run by one or more network operators which has a single and clearly defined routing policy" [1], and often represents a network privately owned by one organisation. (Multiple ASes may be owned by the same organisation [2], often by the process of buyouts or mergers.) BGP allows local policy enforcement, but it is otherwise a path-vector protocol used to compute the shortest policy-compliant AS paths to all destinations. A path vector protocol is a variant of a distance vector routing protocol, but rather than maintain distance counters, a path vector protocol maintains full paths to the origin of a routing advertisement. Retaining paths solves the count-to-infinity problem and, more generally, allows simple detection of loops.

Routing state collected at various locations has been archived by the Route Views project [3], a data repository containing BGP table dumps dating back to 8 November 1997. The number of collectors associated with the Route Views project has grown over time from just one to the current set of ten; see Figure A.1 (page 127). This archive is the primary data source that I use throughout this dissertation. The BGP table dumps reveal information on the address space that was active on the Internet over time, and also the interconnections between ASes. All AS topologies and IPv4 prefixes I use in this dissertation are derived from these BGP routing tables and aggregated according to the date they were collected. More detail on the data, and how I manage it, can be found in Appendix A.

This chapter covers the growth of the inter-domain Internet using this archived data. In order to provide some background on the growth of the Internet as is relevant to this dissertation, I will outline some of the Internet's growth characteristics: in particular, the number of participating Autonomous Systems, the inter-connections between those networks, and the growth

of the advertised address space (both in terms of the number of prefixes, and the number of addresses these represent). I will also describe the AS graph model used throughout the remainder of this dissertation, as well as additions to the model not used here (in particular inferred types of relationships between ASes and the use of traceroute-derived topology maps), and the difficulties inherent in modelling the inter-domain Internet.

The chapter is structured as follows:

Section 2.1: describes the data sources and the modelling assumptions that are used throughout this dissertation.

Section 2.2: summarises the current Internet architecture.

Section 2.3: covers the growth of the graph of Autonomous Systems since 1997, in terms of the number of participating Autonomous Systems, the number of links, and measures of the size of this graph.

Section 2.4: covers the expansion of the advertised IPv4 address space since 1997, and presents an analysis of the prefixes that form the full BGP table.

Section 2.5: covers in more detail the difficulties inherent in modelling the AS topology.

Section 2.6: discusses some of the constraints that the routing state places on forwarding hardware.

Section 2.7: summarises the chapter.

2.1 Data Sources & Modelling Assumptions

In this chapter and all subsequent chapters, I use archived BGP routing data to model Internet topologies. This section describes the data used, and the network model constructed. This data and this model are used throughout the dissertation, unless explicitly stated otherwise.

2.1.1 Data Sources

Public routing data archived by the Route Views project [3] is the primary data source used in this dissertation. The Route Views project archives the BGP routing tables from dedicated collectors located in different parts of the world. Full routing tables are captured by the Route Views project every two hours at each collector. The earliest data in the Route Views archive was collected on 8 November 1997.

The dataset used for this dissertation consists of one BGP table per collector per day. The table taken for a given collector on a particular date is the table with the earliest timestamp. Most commonly, the earliest table available for a collector on a given date was collected at midnight.[1] Where multiple collectors were active on a given date, one routing table from each collector is stored. These multiple vantage points are used to create an aggregated graph that represents the AS topology. The methodology used to process the BGP tables into AS topologies is described in Appendix A.

Some of the results presented in this dissertation use all of the daily AS topologies, from 8 November 1997 to the end of the dataset. Some of the results sample from this data by using topologies built from routing tables collected 6 months apart (starting 8 November 1997, and using every 8 May and 8 November topology to the end of the dataset).

2.1.2 Modelling Assumptions

Each AS in the Internet is composed of multiple routers, and an AS may have multiple ingress and egress points. In this dissertation, ASes are modelled as nodes, or vertices, connecting the graph of links. This is a common simplifying assumption when modelling Internet graphs, based on the information available in BGP data. The process used to build the AS graphs is covered in Appendix A.

This model provides unweighted, undirected links. In reality, individual ASes will have differing agreements with the ASes they do business with, and this is reflected in the links chosen for propagating advertised address space, which in turn affects link visibility. Paths taken on the Internet may deviate from the paths visible in the data because of these differing agreements. Information on policy is not publicly available and cannot be derived. The effect of this is covered in Sections 2.5 and 2.5.1. Without additional information, the graph model used in this dissertation is necessarily a policy-free model, *i.e.*, the policies used at each node are unknown and therefore cannot be simulated.

The data does, however, allow for a rough distinction to be made between ASes that offer transit services, and ASes that act only as sources/sinks for routing updates (*i.e.*, stubs, that do not offer transit services). This distinction is highlighted in Section 2.3, and used later in Chapter 6.

[1] All collectors are configured to the same timezone, and so a midnight timestamp on each represents approximately the same moment in time.

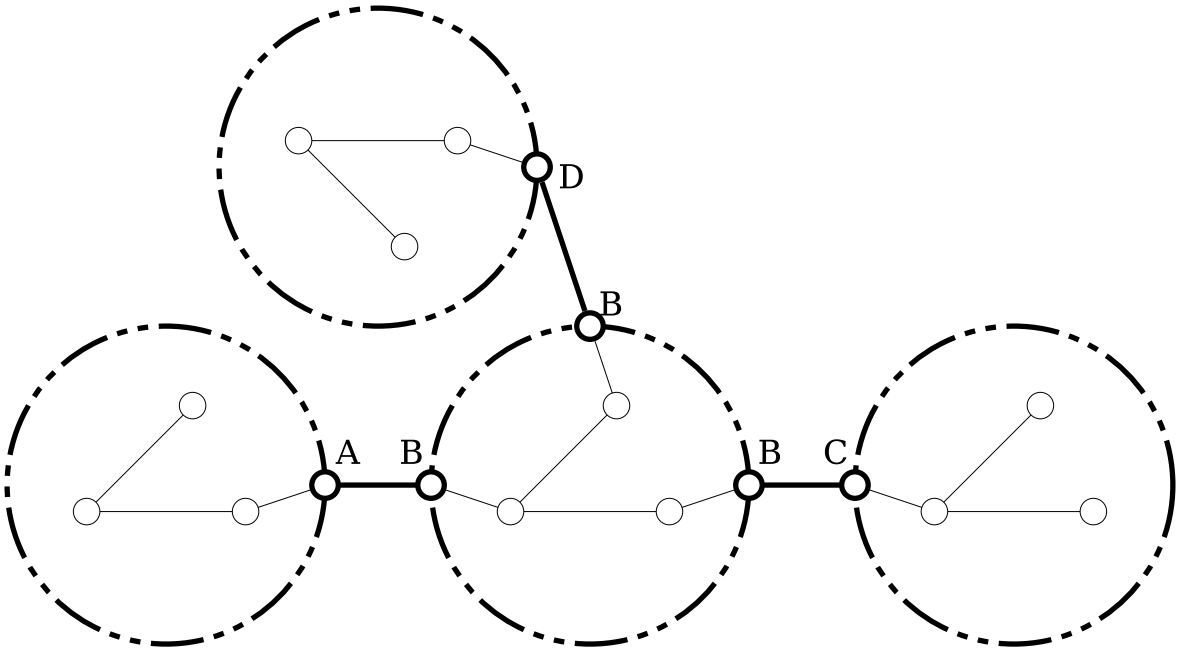


Figure 2.1: High-level overview of four trivial Autonomous Systems interconnected with BGP.

2.2 Current Internet Architecture

The current inter-domain protocol is the Border Gateway Protocol (BGP) [4], and has been in operation since its original specification in 1994 [5]. Previously, the Exterior Gateway Protocol (EGP) was the common inter-domain protocol [6]; the EGP required loop-free (tree) structures, and thus was not suitable for non-trivial networks and became a problem as the Internet grew. The BGP iterated through four versions of culminating in BGP-4, the most recent specification of which is documented in [4].

Operation of the Internet routing system is based on two primary components: network prefixes (also known as address blocks, or network layer reachability information (NLRI)), and Autonomous System numbers. Network prefixes represent portions of the routable IPv4 or IPv6 address space. These prefixes can be assigned at various lengths to networks. They can be disaggregated, and it is possible for two prefixes of different lengths to be advertised, one which is a proper subset of the other. Current BGP deployments support Classless Inter-Domain Routing (CIDR) [7] which allows these variable-length prefixes, and the possibility of aggregating network prefixes prior to advertising them to the inter-domain network.

Autonomous System numbers are used to uniquely identify networks. Portions of the address space – the network prefixes – are advertised by ASes with the AS number of the origin attached in the BGP advertisement. The advertisements are propagated by other ASes that are willing to offer a transit service.

BGP is used *between* ASes, with other routing protocols used internally by each AS (com-

monly OSPF [8] or IS-IS [9]). Figure 2.1 presents a simplistic model to demonstrate where BGP operates; each of the smaller circles represents a router linked by edges to other routers, and each large dotted circle represents an AS. The border routers in the diagram, *i.e.* the set of routers that are BGP speakers that interact with routers in neighbouring ASes, are those on the perimeter of each AS, and labelled by the AS number assigned to each AS[2]. Internally to the AS, these border routers share state between each other either by forming a full mesh with the other border routers or through the use of route reflectors that connect to all border routers and share state indirectly. Externally to the AS, routing advertisements are forwarded between BGP speakers.

BGP propagates advertised address blocks between ASes, with border routers prepending the local AS number to the *AS path* when an advertisement is propagated. This path information allows the detection of routing loops. The number of AS numbers in the AS path gives the *AS hop* count, a metric for the length of the path traversed. ASes can sometimes engage in "path prepending", by which they prepend the local AS number *multiple* times such that the resulting AS hop count for that AS path will be longer than it actually is, thus making the path less desirable to subsequent recipients and therefore affecting inbound traffic for this route.

BGP is a *hard-state* protocol, with an explicit removal required for each advertisement, and no refresh timers. Connections between BGP border routers are TCP connections, and so BGP update messages between two neighbouring border routers arrive reliably and in-order, as long as the connection exists. On forming a new connection, two border routers exchange full tables, upon which updates thereafter act. Thus, the data at any BGP router is a view of the current state of the inter-domain Internet, excepting any updates to that state that are still propagating from their origin. Further, not all AS links may be visible to other ASes: some connections between ASes are formed between those ASes to avoid the cost of routing via a provider, allowing the ASes to install "shortcut" routes to the address space advertised by the other AS, but in such a way that this link does not "leak" out to other ASes for public use. This type of arrangement accommodates peering ("settlement-free") agreements, where traffic is shared between ASes thus saving on volume of transit that must be purchased from an upstream provider [10]. Therefore, there are extant links not visible in the BGP data (discussed further in Section 2.4), and so the state that is revealed in the BGP data is correct, albeit partial, within some small time delta from when the snapshot was taken.

[2]Such numbers are assigned by registries split across geographic regions.

2.3 Network Growth

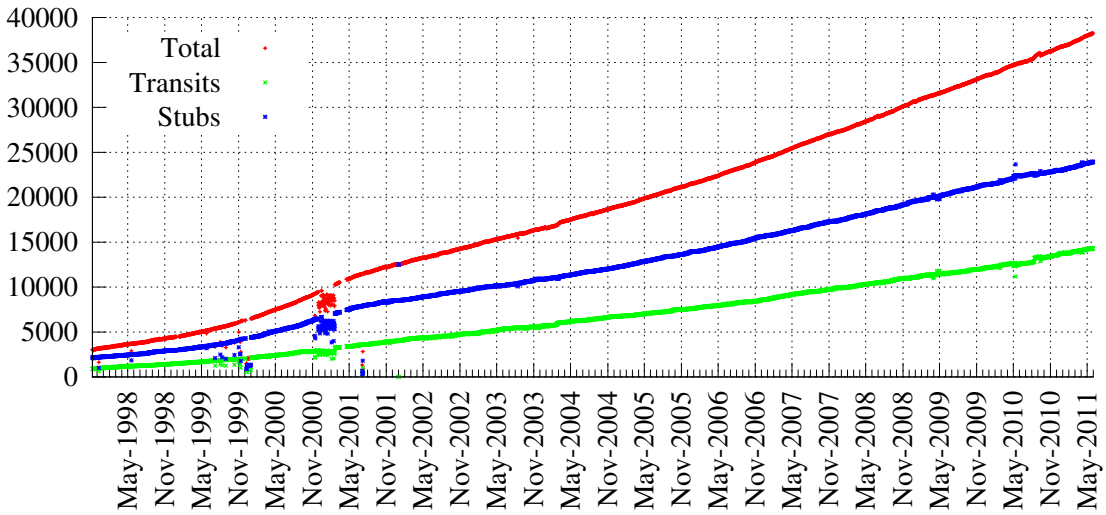
Between 8 November 1997 and 31 May 2011, the number of Autonomous Systems visible in the BGP data has grown from 3,030 ASes to 38,102. The growth of the total number of ASes is shown in Figure 2.2a. This is an absolute measure of the number of ASes participating in BGP. Many AS numbers are no longer visible in BGP, if their parent company has ceased trading or they have merged with another company; of the 3030 ASes in 8 November 1997, 1746 remain in the BGP data as of 31 May 2011. The growth of the number of ASes was super-linear until mid/late 2001, followed by a linear phase for a short time [11]. The growth of the curve appears to currently be super-linear.

BGP path data does not reveal the types of business relationships between ASes, but it is easy to identify two broad groups of ASes from the data by drawing a distinction between nodes that are clearly stubs (that do not offer transit services to other ASes) and those that are not stubs (that do offer transit services to other ASes). This is achieved by examining the AS paths stored at the Route Views collectors for a given date; a stub network never forwards advertisements to other ASes and so will only ever be observed in the path data as an origin for advertisements and will never appear in the middle of an AS path. Figure 2.2a shows that approximately two thirds of the ASes visible in the BGP table are clearly stubs, but a decreasing proportion, with 71.0% on 8 November 1997 and 62.6% on 31 May 2011. More detail on inferring AS relationships is covered in Section 2.5.1.

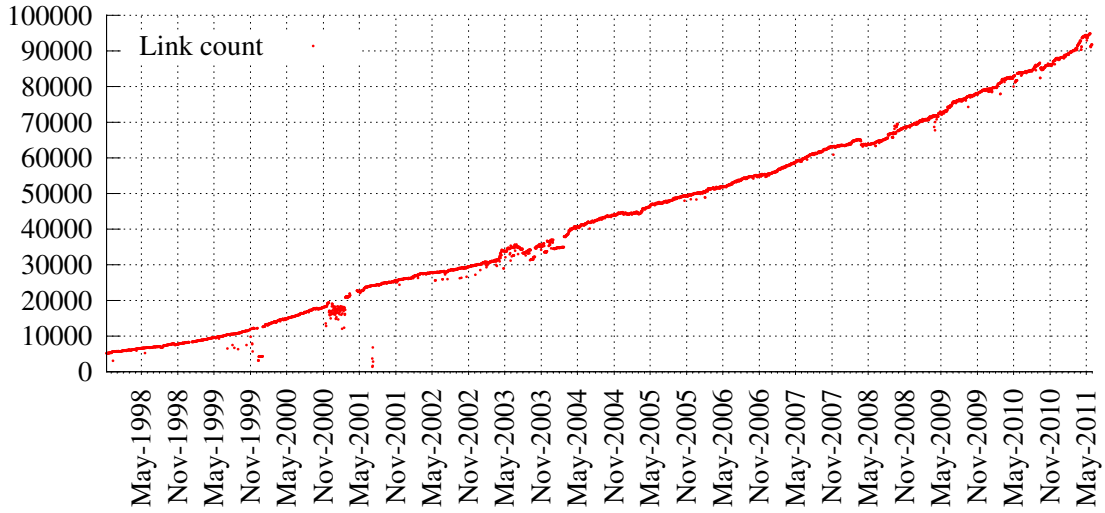
Also observable from the data is the number of links that the BGP advertisements have traversed. As the number of ASes grows, so too must the number of links. Figure 2.2b shows a superlinear growth curve of the number of links in the graph. The growth pattern is similar to the AS growth pattern.

The combination of these two measures, the number of ASes and the number of links, allows a simple measure of the density of the graph, and how this has changed over time. The ratio of links to ASes is shown in Figure 2.2c. In this instance, a lower number indicates a higher average node degree, and possibly a more densely connected graph. However, this ratio has been effectively constant at around 0.42 since 2004. (That is, 2.38 neighbours per AS, on average.) This is surprisingly stable given the rate of growth of the network.

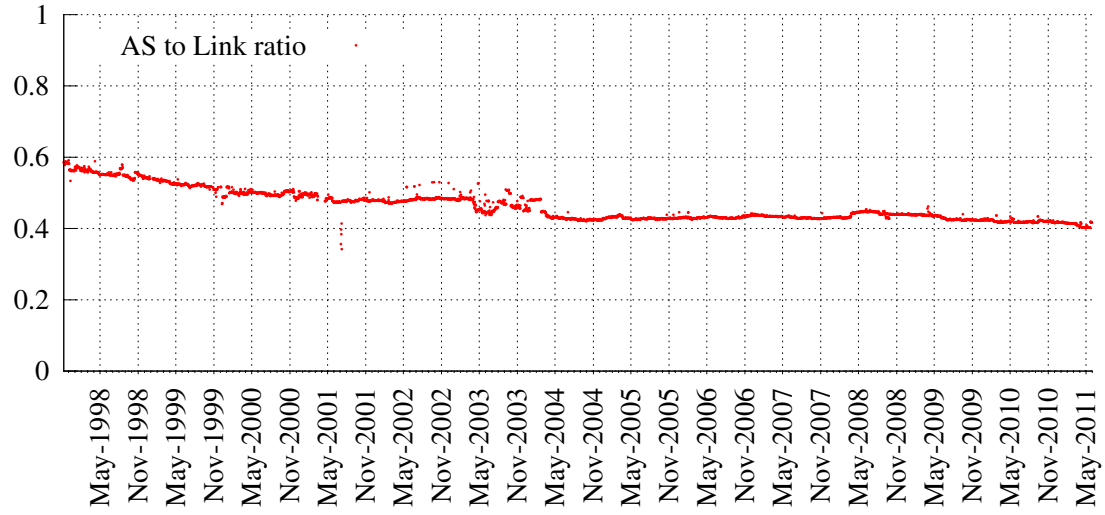
The size of the network can additionally be measured in terms of distance between any two ASes. The shortest path between two nodes is the path between those two nodes with the fewest hops. The mean network path length is the mean of all the shortest path lengths in the network, and the diameter of a network is defined as the longest path in the collection of shortest paths between all pairs of nodes in the network. Figure 2.3a shows mean path lengths over the dataset used throughout this dissertation, as well as the 75^{th}, 95^{th}, and 99^{th} percentiles on path lengths, and the network diameter. Given the full set of shortest



(a) Number of ASes, indicating a simple split between "stub" networks and "transit" networks.

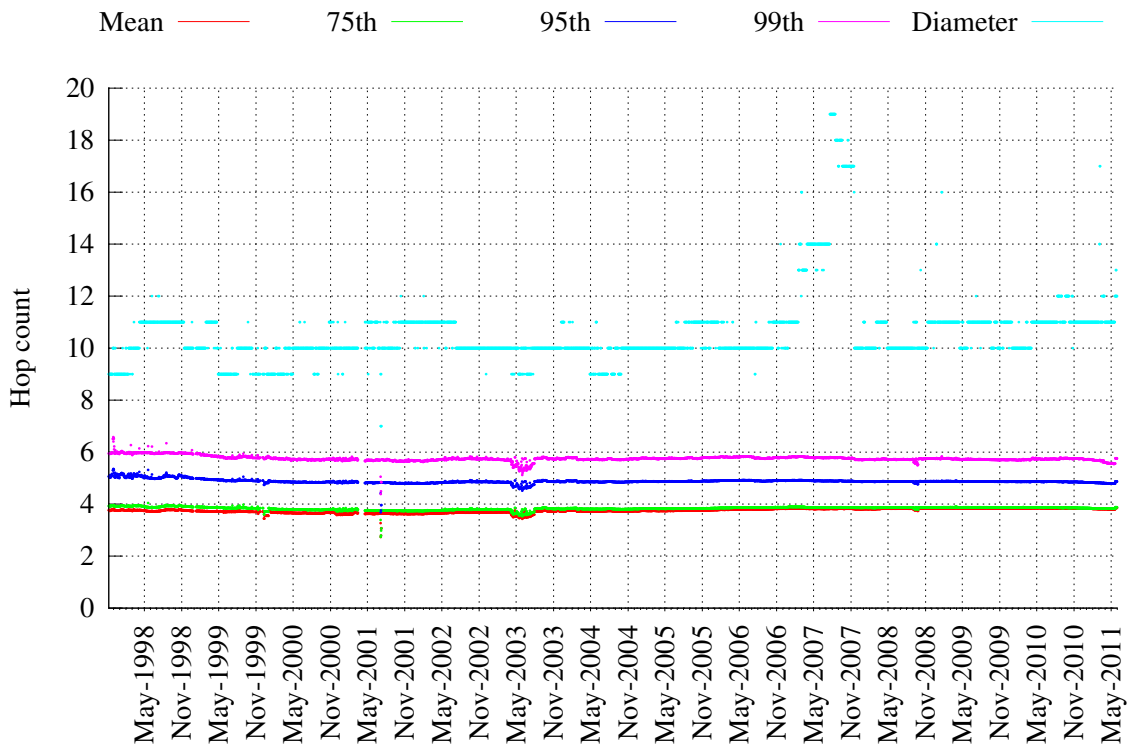


(b) Number of inter-AS links visible.

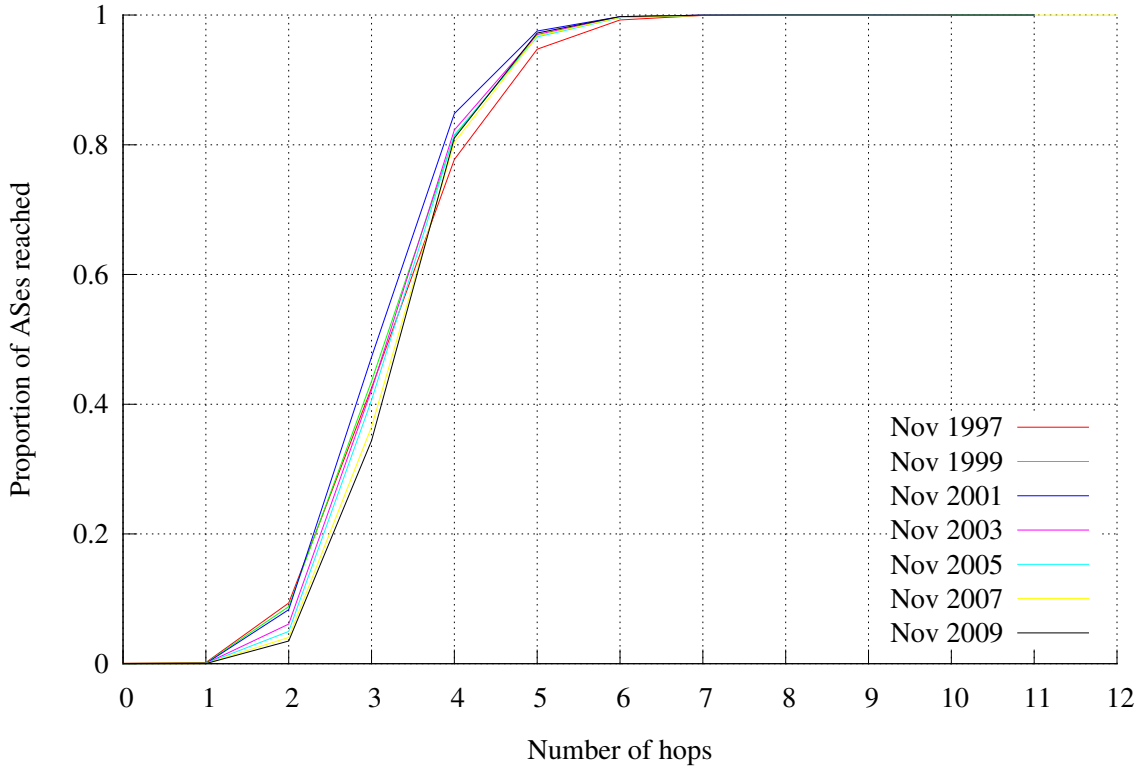


(c) The ratio of links to ASes.

Figure 2.2: Growth of the AS graph.



(a) Path lengths, showing the mean, the 75th, 95th, & 99th percentiles, and the maximum path lengths (network diameter).



(b) Cumulative mean distance to rest of network.

Figure 2.3: Network path lengths.

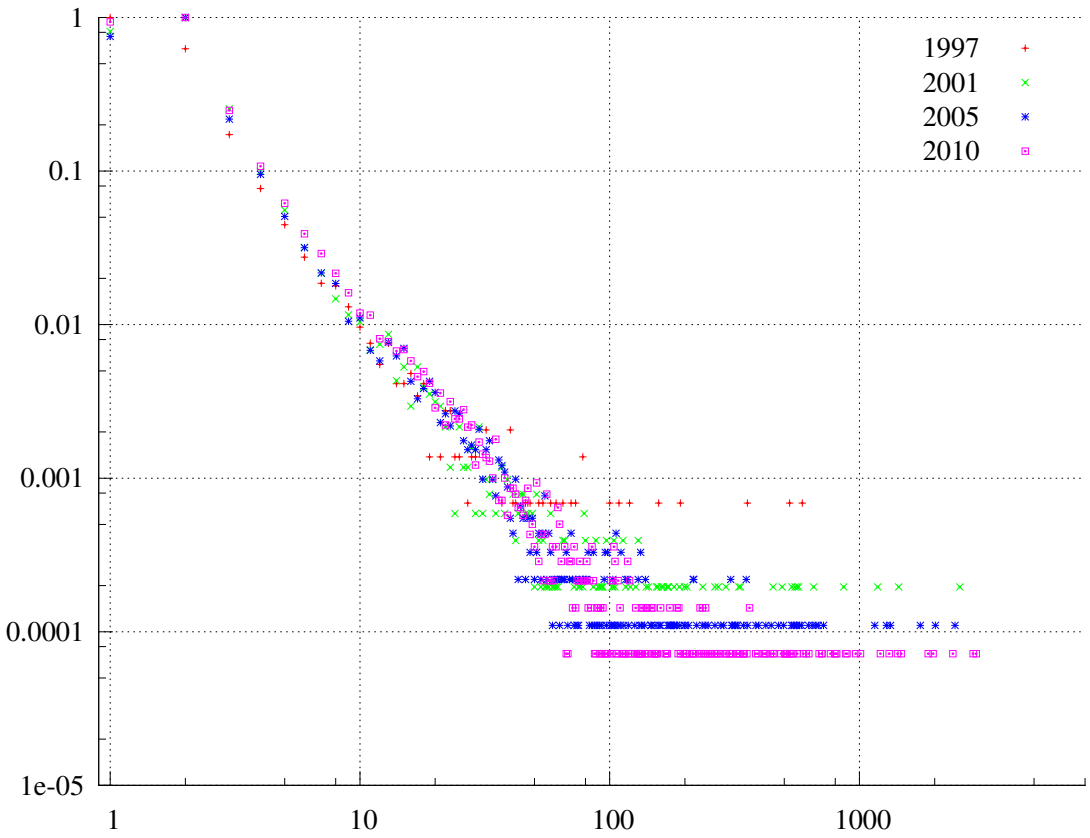


Figure 2.4: Normalised node degree distributions.

path distances between all pairs of ASes, Figure 2.3b shows cumulative distributions of the proportion of path lengths, indicating the remarkable similarity between graphs of different sizes.

Despite the order-of-magnitude growth of the AS graph, mean path lengths have been effectively constant, and the diameter of the network over time has increased only slightly. The average AS path length has essentially remained constant between 3.5 and 4 AS hops. Even the 99^{th} percentile is consistently fewer than 6 AS hops. This has been achieved while the AS graph has maintained a consistent link density. This confirms previous observations on a much smaller date range [12].

The low AS-to-link ratio already shown in Figure 2.2c, if applied to a uniform random graph, would not produce the same static average path lengths shown in Figure 2.3, but instead would exhibit gradually increasing path lengths. AS graphs demonstrate more structure than random graphs. They exhibit an apparent power law distribution on the node degrees, first characterised in [13], but also in other work [14]. A power law function is a function $f(x) = \alpha x^{\beta}$ where α and β are non-zero constants. For network degree distributions typically with $-3 < \beta < -2$, the network is a *scale-free* network, where a small proportion of the network is well-connected to the rest, and a large proportion of the network is not so well

connected. Empirical data often maps onto a power law in the extremities, as $x \to \infty$, and such distributions have been noted in many empirical data sources for some time [15].

The degree distributions for a selection of AS topologies are shown in Figure 2.4, each normalised according to the number of nodes in the network. The heterogeneous degree distribution goes some way to describe the relatively constant length of paths despite the size of the graph.

Power law degree distribution may follow naturally from a form of preferential attachment (the concept of "the rich get richer") and incremental growth: new networks choose to connect to networks that already have large customer bases relative to those networks that do not [16]. Three years of structural network growth seems to confirm demonstrates preferential attachment as the source of the power law, and the growth of links between existing networks [17]. A subtly different model is that, rather than preferential attachment that implies foreknowledge of the degrees of other ASes, that ASes form links with each other as they grow their userbase. The model relies on growth patterns of the number of ASes and the number of hosts on the Internet, and results in a similar highly variable degree distribution [18].

Many ISPs are well-connected to the rest of the Internet and offer transit services for other ISPs. Offering a transit service implies good service which in turn implies minimal latencies between pairs of endpoints. Thus, ISPs offering this level of service often maintain full routing tables to make best use of their connectivity. (Also, unless they have a limited hierarchy of routers, they have to retain the full table.).

2.3.1 Summary of Network Growth

The IPv4 AS graph has grown by an order of magnitude in the duration for which archived data is available, and currently exhibits a greater link density than at the start of the archive, though the link density now appears constant. The continued expansion has not increased path lengths, which are effectively constant, and the network diameter has grown slowly.

Supporting this consistent level of connectivity is the structure of the graph, exposed by the node degree distribution. The highly heterogenous node degree distribution exhibited appears to be that of a power law degree distribution, or a *scale-free* degree distribution, where there exists a small set of ASes of high degree (connected to a high proportion of the network) and a large set of ASes with low degree (connected to a small proportion of the network). In this sense, the network has something of a central "hub", forming a hub-and-spoke arrangement where traffic passes through a small handful of highly connected transit networks. This type of topology is not trivial, however, with many links between the low-degree nodes themselves.

2.4 Advertised IPv4 Address Space

Internet routing relies on a shared, universal address space which endpoints use to address each other. The prevalent address space in use today is IPv4. To facilitate routing between endpoints, independent networks advertise to each other the portions of the address space they are entitled to manage. These portions of address space, advertised in blocks, are passed between networks, and installed into network routers. Thus, packetised data can be sent across the network with the destination endpoint indicated in each packet header, and the routers match the destination endpoint to one of the address blocks stored locally to determine the correct link on which to *forward* the packet such that it ultimately arrives at its destination.

These address blocks can vary in size, and they can overlap; the convention being that if a router contains multiple advertised address blocks that match a packet's destination, then the smaller block (*i.e.*, the longest prefix, and the most specific block) takes precedence in choosing the next hop on the packet's path.

CIDR has helped slow the growth of the number of IPv4 prefixes because it allows greater flexibility in address block assignments. While subnetting beneath the assignment encourages hierarchies in local networks, the BGP system is effectively a flat address space with no hierarchy. BGP does allow the advertisement of *AS sets*, wherein one IPv4 block can be advertised by a network upsteam to represent a *set* of ASes, but this requires manual management and cooperation between networks if one of the ASes in the set chooses to multihome. Kleinrock and Kamoun's 1977 paper on hierarchical routing explored the relationship between forwarding table sizes in a hierarchical network and the distance between nodes in that network, suggesting that depths of hierarchies must grow quickly with respect to the number of nodes present to maintain reasonable sizes for forwarding tables [19]. However, multihoming implies no hierarchical address aggregation can be applied, and thus contributes to the rapid growth of the size of the routing table.

The size of the full BGP forwarding table is shown in Figure 2.5. The tables have seen massive growth since the start of the Route Views collection, holding 56,698 unique prefixes on 8 November 1997, and 386,771 unique prefixes on 31 May 2011. This growth largely comes from the edge of the network. Much of the expansion in the forwarding tables in the Internet's *default-free zone* (DFZ), *i.e.*, much of the BGP infrastructure where there is no "default" upstream provider, can be attributed to end-sites opting to multihome; a study from 2004 showed that 20-30% of the prefixes in routing tables originated from multihomed ASes [20]. It is often the case that networks will participate in inter-domain routing to multihome as a means to either load-balance multiple traffic flows, or to use one provider as a backup link. However, many others participate to enable multiple geographic points of presence, and there are many networks in the business of providing transit who operate with

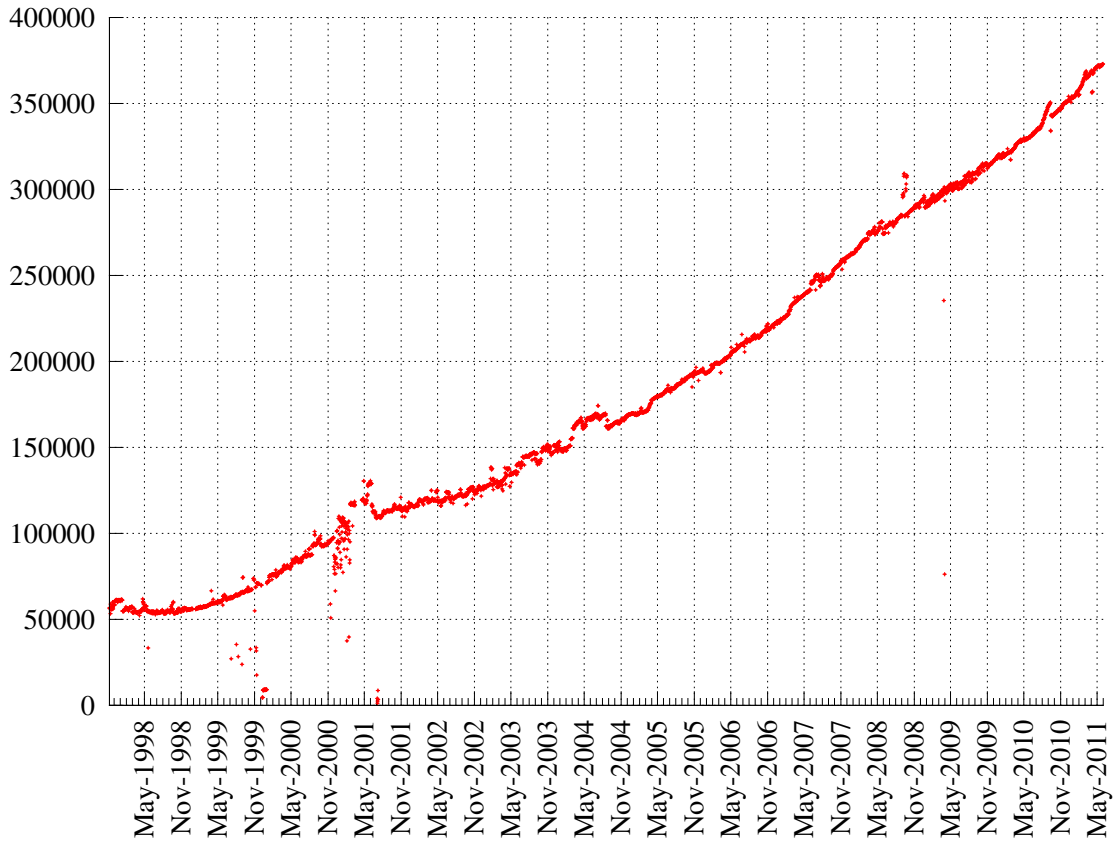


Figure 2.5: Growth of total number of prefixes observed in the BGP data.

full, *default free*, routing tables. Such an arrangement provides the benefit to the end site that they are no longer reliant on one ISP to make strong service guarantees; instead, they opt for redundancy with more than one route in and out of their network. To achieve this redundancy, the network's address space must be advertised into the BGP routing system via all provider networks, suggesting that the advertised address block cannot be aggregated by at least one service provider.

Multihomed end-sites are commonly addressed is via provider-independent (PI) address blocks, as opposed to aggregatable provider-assigned (PA) address blocks. Provider independent addresses make up a substantial number of address allocations by regional Internet registries (RIRs). As none of these allocations can be aggregated, these relatively small blocks must be advertised independently of all other addresses in the DFZ.

CIDR allows prefixes of variable lengths to be advertised. Current best practice states that, in general, the longest prefix propagated across the inter-domain network is a $/24$. No prefixes shorter than $/8$ are assigned, and so there is variability in prefix lengths of 16 bits. Figure 2.6a shows the distribution of IPv4 prefixes advertised on selected dates across the dataset. The number of prefixes present has grown significantly as already shown in Figure 2.5, and so these distributions are normalised to show the proportion of the table occupied per prefix

length.

Given that two adverts may represent different sizes of blocks of the address space. Figure 2.6b shows the relative size of the address space advertised by block size, displayed proportionally by prefix length. Viewed in this manner, the /24 prefixes advertised on 8 November 2010 represent 52.4% of forwarding table entries, but cover less than 1.5% of the total space advertised.

The average span of these prefix lengths has been fairly constant throughout the dataset: the average prefix length on 8 November 1997 was 22.68, and on 31 May 2011 was 22.43. The mean throughout the set is 22.44, with a standard deviation of 0.28. There has been an expansion of prefix lengths between /16 and /24 with a corresponding reduction in the *proportion* of /24s being advertised and the number of /16s being advertised, with the overall effect being that the average span is effectively constant.

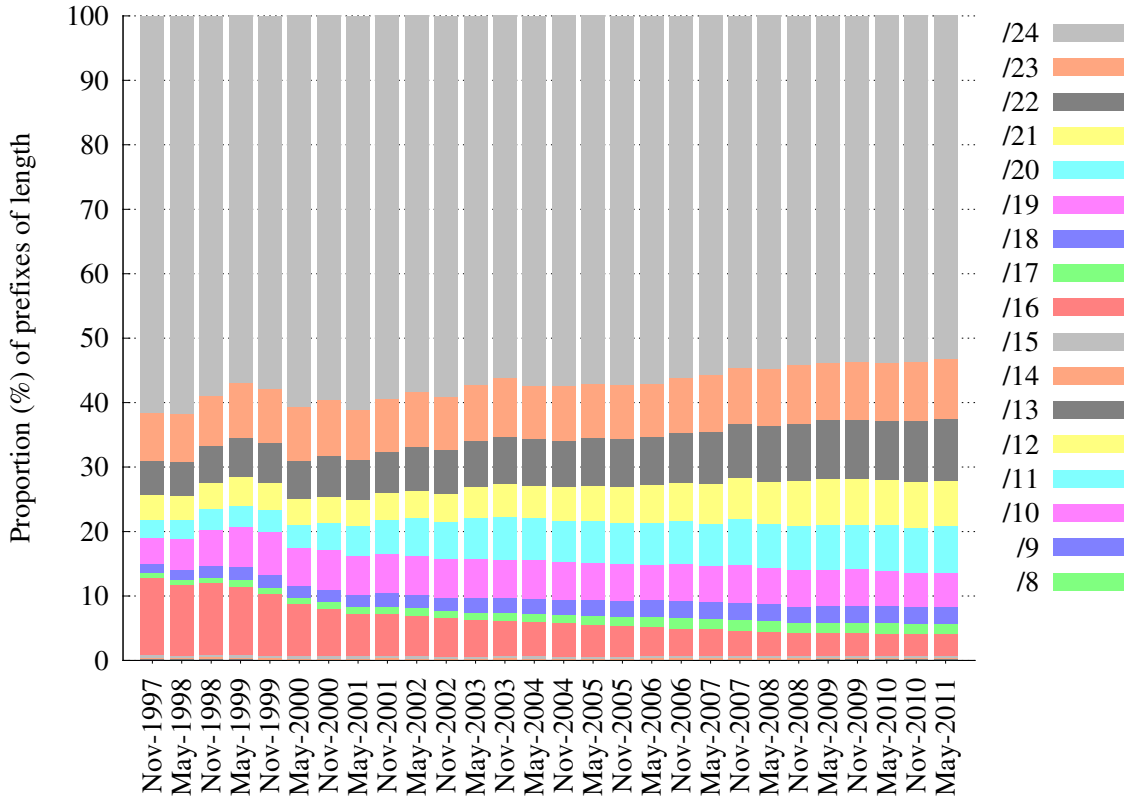
The prefixes advertised between 2001 and 2008 show no particular trend toward aggressive prefix disaggregation or traffic engineering over time [21]. Also, the same research indicates that disaggregated prefixes do not generate an increased level of update traffic. That is, there has been no significant proportional change as the network has grown, though naturally the network grew significantly.

The continued growth in the size of the forwarding tables suggest diminishing returns in terms of the address space advertised, but a growing burden for the routing hardware to maintain. Additionally, networks charge each other for bandwidth used, but not for the prefixes they advertise; there is no link between cost passed on to customers, and the size of the routing table [22].

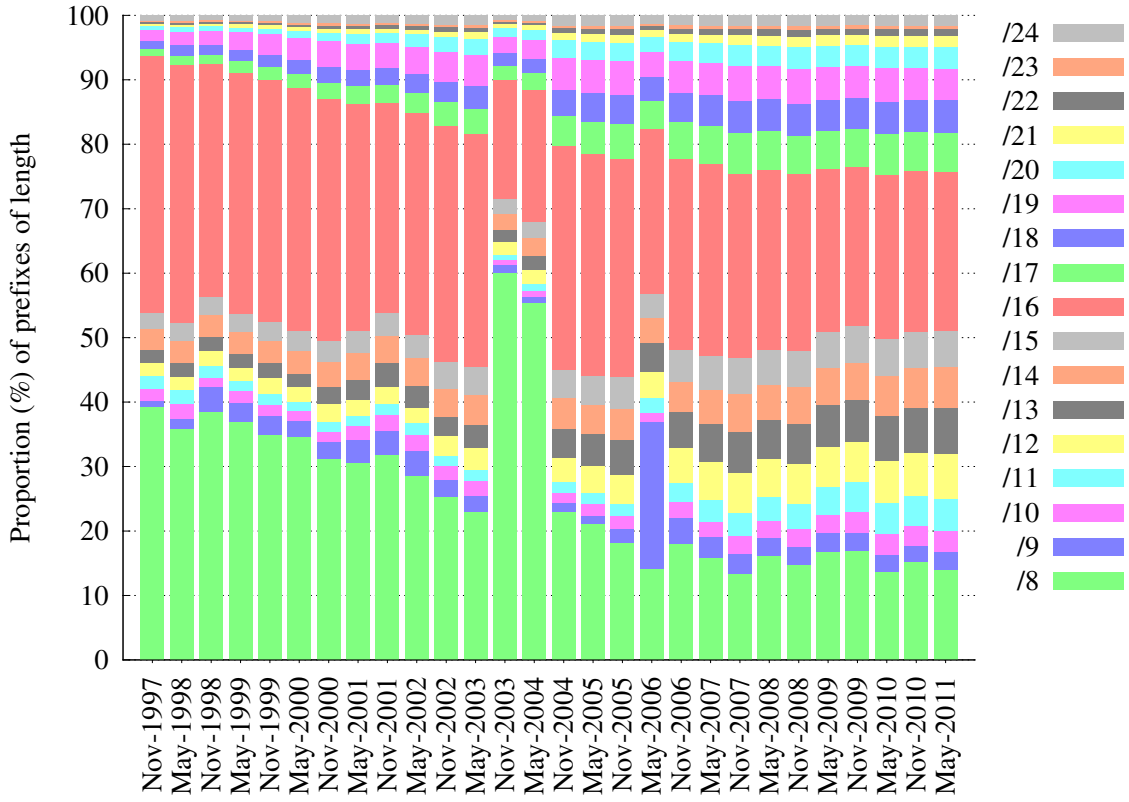
The IPv4 address space has now been exhausted from the top-level free pool managed by the Internet Assigned Numbers Authority (IANA). The last of the available address blocks were assigned to the five RIRs in February 2011. One of those, APNIC, has already consumed the last of its IPv4 addresses.

IPv6 is also not yet ready for global deployment; there is the possibility it may never achieve ubiquity. The exhaustion of the IPv4 space as the only ubiquitous Internet address space implies a continuous subdivision of the address space, leading to a continuation of the trend toward advertising smaller blocks (and possibly ultimately an *increase* in the proportion of /24s advertised). Creating markets to permit the trade of IPv4 address space between networks is a possibility for extending the life of IPv4, assuming that networks will only pay for what they need. The economics of this outcome are discussed further in [23].

In summary, the number of IPv4 address blocks advertised continues to grow. The address space has, over time, tended toward prefix lengths longer than /16s, *i.e.*, there has been a tendency toward advertising smaller address blocks. The exhaustion of IPv4, and lack of readiness of IPv6, suggests that this trend will continue.



(a) Variation in prefix distribution over time.



(b) Variation in contribution to the address space

Figure 2.6: IPv4 prefix distributions over time.

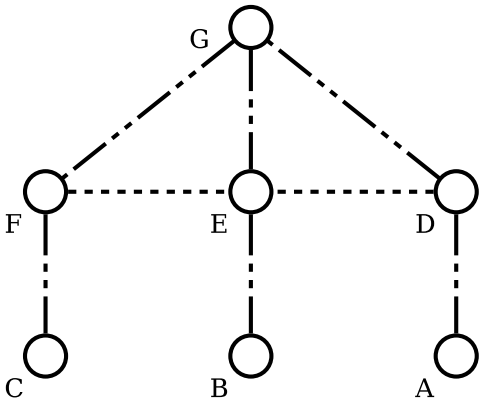


Table for A:

A	-
B	D E B
C	D G F C
D	D
E	D E
F	D G F
G	D G

Table for B:

A	E D A
B	-
C	E F C
D	E D
E	E
F	E F
G	E G

Table for C:

A	F G D A
B	F E B
C	-
D	F G D
E	F E
F	F
G	F G

Figure 2.7: A high-level overview indicating customer/provider links. Peering links exist between D & E, and E & F, all other links are provider/customer transit links. The peering links are visible only to these nodes and their children: the table at node C, for example, does not contain the link E – F.

2.5 Modelling of the Internet Graph

Deriving unweighted, undirected graphs of AS connectivity is simple when using the AS paths stored in archived BGP data. The set of passive BGP collectors utilised by the Route Views project [3] provide an important source of BGP data. However, this data may not represent all inter-AS links [24, 25, 26, 27, 28, 29, 30]. Although I already aggregate data from the multiple Route Views collectors, there is scope for improving the sampling by aggregating more paths from other repositories, such as the BGP data archives held by RIPE. For an overview of the processing of the data used in this dissertation, see Appendix A.1. The aggregation of the data builds a more complete view of the network.

Synthetic network models have been constructed, but these generally model the network as it appears from a solitary vantage point and fail to capture the complexity of the network when multiple vantage points are aggregated [31]; a better view of the network may affect the observed degree distribution [32].

We know that the relationships between the ASes are not homogeneous. The links missing from the data primarily represent *settlement-free* links between ASes (*i.e.*, no-cost links, where no currency is exchanged and there exists some level of equality between the networks). These agreements are in place to provide a direct path between the two ASes and avoid the costs of using a shared transit provider. An illustrative example is shown in Figure 2.7. ASes that form a peering relationship tend to share only their routes and the routes of their customers, such that a "shortcut" is provided in the network. These links are visible between peers and their children, but by their nature do not propagate beyond the peering (the settlement-free nature of the arrangement makes public use of the peering links undesirable).

Revealing more of the underlying AS-level structure of the Internet is an active research

area [33]. Modelling the evolution of the network by inferring the difference between real topological changes and mere path changes over time would allow a more complete picture of the network, for some confidence level [34]. There is also work on performing active measurement, using traceroute to reveal active paths not visible in the BGP data [35].

Each of these network models is static. Building dynamic models from archived BGP data should be possible, and would aid our understanding of how the Internet evolves. How to model real-world Internet dynamics is discussed in [36]. Models have also been built that aim to capture the growth of the network, featuring rules for attachment between transit and stub ASes [37]. These, however, are modelling the visible portion of the network and cannot capture the behaviour of any links missing from the data.

2.5.1 Relationships between Autonomous Systems

Augmenting the AS topology with AS relationships would allow a better representation of the flow of routing state across the network, and therefore provide a better indication of routing overheads, as the business relationships between networks affect the flow of advertisements, and therefore path selection, across the Internet. However, ASes tend to consider their business relationships to be proprietary information, and generally do not divulge it.

Inferring the relationships between ASes is known as the type-of-relationship (ToR) problem. Specifically, it involves assigning types to links in the unweighted, undirected graph revealed by the BGP data. The initial work in the area defines a hueristic algorithm based on analysis of the AS paths visible in BGP data to categorise links into one of four types: customer-to-provider, provider-to-customer, peer-to-peer, and sibling-to-sibling [38]. Similar techniques were formalised in [39], and modified heuristics have been presented in other research [40, 41]. Sibling relationships indicate ASes owned by the same organisation; other work has attempted to use the WHOIS database to infer such ASes, with limited success [2].

The network model usually assumed is that of a valley-free network topology model, where on an AS path a provider-to-customer link is only followed by additional provider-to-customer or sibling-to-sibling links, and a peer-to-peer link is followed by exclusively provider-to-customer or sibling-to-sibling links. This inference is based on node degree, though, which may not be descriptive enough to indicate the type of relationship.

It has been shown that these algorithms tend to be accurate when inferring provider-customer links, but are less accurate when inferring peer-to-peer links [42]. Further, it is known that the ToR problem is NP-complete [43, 44, 45].

2.6 Forwarding Hardware

The problem with the size of the forwarding tables indicated in Section 2.4 is that they must be contained within the forwarding hardware, which must be able to handle packet forwarding at line-rate. Routers must also be able to update forwarding tables in a timely manner without interrupting packet forwarding. As forwarding tables increase in size, so too does the time to analyse the table *per packet*, or the time taken to modify the data structure *per update*.

The rate of BGP updates is increasing, but the average rate is not necessarily a problem; a related problem is that the BGP system is not predictable, and bursts of updates are not uncommon [46]. Future routers must be able to handle increased workloads, including extremely noisy periods of BGP updates. The primary bottleneck within routers will be the forwarding hardware [47].

Broad approaches to address lookups in forwarding hardware are algorithm-based and hardware-based. Algorithmic approaches aim to construct a suitably compressed data structure stored in RAM which requires $O(\log n)$ lookups to find a match. Patricia trees [48] and the Luleå algorithm [49] are common examples of data structures that are used for longest-prefix matching. Hardware-based lookup uses Ternary Content Addressable Memory (TCAM), which performs longest prefix matching with a constant lookup times regardless of the number of prefixes in the table or how many prefixes match a given address.

Algorithmic approaches rely on the construction of efficient trie structures to minimise memory lookups. Although the emphasis is on minimising lookup times, they must also be able to handle updates quickly. Performing incremental updates on the forwarding state without disrupting packet forwarding is preferable [50].

Hardware approaches use TCAMs to store the forwarding table. TCAMs are less dense than conventional RAM, are considerably more expensive, require more transistors to store the same amount of data, and consume much more power. They are incorporated into some router designs, but do not seem to be used in large-scale routers today. Indeed, the trend away from TCAM usage in high performance routers is supported by related work that defines an allocation and growth model for IPv4 addresses and asserts that tries scale better than TCAMs with increasing forwarding table sizes [51]. A concern related to the growth of the routing tables is that there has been some discussion on the formation of a market to allow the exchange of IPv4 address blocks between networks [23].

Given $/24$s, there is the possibility of treating the IPv4 address space as a flat address space with no prefixes of different lengths. By doing this, forwarding algorithms are considerably simpler, and then routing on $/24$s is perfectly feasible [52]. This specific design may work for a flat $/24$ address space with 2^{24} address blocks, but if network growth and evolv-

ing best practice mandates common usage of /25 or longer, memory requirements render this approach infeasible. In particular, the size of IPv6 addresses will render this scheme unworkable.

A related concept is the "memory wall" [53], which describes the inherent problem in the increasing difference in growth rates between processor speeds and DRAM access rates. The key problem stated is that, on a single-core processor architecture, regardless of the existence of cache layers, the number of processor cycles consumed per memory access becomes overwhelming, leading to a heavily underused processor. Naturally, process scheduling and multi-core architectures help mitigate this problem, and different processing workloads make different demands on memory [54]. Although the original memory wall paper considers a standard processing architecture with a processor, some cache, and main memory, the same argument can be made of DRAM access rates on line routers which are not amenable to a layer of cache (implying direct memory access for each lookup) and yet are servicing larger tables on faster links.

Ultimately, forwarding hardware is highly specialised hardware that must be able to handle full forwarding tables and the stream of modifications to that state. As the number of prefixes increases, and as the number of updates increases, achieving this becomes much harder. Large scale routers are already non-trivial equipment, but engineering them to support future growth without any reduction in routing state or routing state growth will be difficult.

2.7 Summary

In this chapter, I have demonstrated the growth that the Internet has exhibited over the last 14 years for which BGP data is publicly available, with some additional datasets derived from Traceroute data. This growth is currently manageable, but long-term sustainability is questionable.

I have shown the order-of-magnitude growth in the number of ASes advertising address space, and also in the address space advertised, but also that the address space advertised is continuing to disaggregate over time. This growth is likely to continue: new users in growth regions require address space; new services must be addressable.

It is likely that the IPv4 space will continue to disaggregate for as long as IPv6 is not ubiquitous. Currently, IPv6 growth has been marginal, and the most optimistic projection is that IPv4 and IPv6 network growth will happen in tandem. The state burden on the legacy IPv4 space as organisations begin trading smaller IPv4 blocks, in the absence of any available addresses from the RIRs, mean that despite IPv4 being "exhausted", there is still much room for growth. As discussed in Section 2.6, this growth will eventually become problematic for

router vendors. The network cannot sustain this growth on the long-term with the existing architecture.

Chapter 3

Routing Scalability and Related Work

To achieve long-term sustainable inter-domain Internet routing, the size of the forwarding tables or the rate of table growth must be reduced. Achieving long-term scalable packet forwarding introduces unique challenges: routers must be capable of forwarding packets at line-rate, implying that data structures are stored in close proximity to the processing hardware to minimise lookup latency. Forwarding infrastructure for inter-domain routing is high-performance, specialised hardware. As discussed in Section 2.6, forwarding hardware is a bottleneck because of physical constraints on the size of the memory buffers that can be accessed in the timeframes available and also the rate at which they can be modified. Thus, a solution for state scalability requires a reduction in the state held, or at least a reduction in the rate of growth on the state that must be held.

Alternative network architectures have been proposed that aim to reduce forwarding state in the inter-domain Internet. Two broad categories for new architectures are "dirty-slate" architectures and "clean-slate" architectures: dirty-slate architectures are sympathetic toward the deployed hardware and build their design to operate with as much existing technology as possible; clean-slate architectures imply deployment of new infrastructure to operate. Orthogonal to this distinction is whether the proposal is incrementally deployable: a clean-slate proposal that is incrementally deployable is more useful than a dirty-slate proposal that is not.

Many of these proposals aim to reduce routing state or slow its growth, but are not able to make any guarantees. By revisiting the fundamental algorithmic underpinnings of the routing problem, I also cover algorithms that achieve routing while retaining very little state. Of particular note is compact routing research, which has produced routing algorithms that may be feasibly decentralised and which have shown promise on power law graphs.

In this chapter, I cover some of the various proposals and algorithms that offer some scope for forwarding state reduction on the Internet.

This chapter is structured as follows:

Section 3.1: covers some of the dirty-slate solutions proposed that may help slow routing state growth. These primarily aim to instigate some form of identifier/locator separation between the network layer and the transport layer.

Section 3.2: outlines some clean-slate network architectures that have been proposed, including architectures based on the principles driving distributed hash tables, and evaluates them in terms of their applicability to solving the routing scalability problem.

Section 3.3: covers some fundamental routing theory, and the application of compact routing algorithms and hyperbolic mappings to the problem of scaling Internet routing.

Section 3.4: summarises the chapter.

3.1 Scaling the Internet

In Chapter 2, I outlined the growth of the AS graph, and the growth in the number of IPv4 prefixes advertised across it. In this section, I cover "dirty-slate" solutions to the routing state problem, so called as they aim to retain the most of the existing architecture, while modifying part of it in an attempt to reduce or slow the growth of the routing state.

One factor influencing the growth of the BGP routing table is the practice of AS network multihoming. A network multihomes by advertising its own address space (acquired from its regional Internet registry (RIR)) via multiple transit providers. The address space acquired is usually a small provider-independent (PI) address block, and is not hierarchically assigned. The benefits of multihoming for the network include increased reliability and the possibility of load balancing across multiple links. Additionally, network renumbering when changing provider is difficult [55], so networks often purchase address space specifically to avoid it. The routing system accepts the additional routing state, but the network advertising the address space is unlikely to provide transit services to other networks. The end hosts on the local network are unaware of the multiple provider links, and all routing decisions are handled by the network.

One of the contributing factors to the difficulty in network renumbering is the use of the IP address as both a network locator at layer 3 (the network layer) and as an endpoint identifier at layer 4 (the transport layer) in the network stack. For end hosts, this makes hierarchical assignments from multiple providers difficult to manage. Local network routing decisions are simplified if the network has its own address space which is advertised into the inter-domain network.

Many of the architectures presented here attempt to split network locators from endpoint identifiers to remove the conflation of the two functions, endpoint identification and network

location. In doing so, they aim to encourage adoption of hierarchical address assignments to networks and thus reduce the need for networks to advertise directly into BGP. Critically, without an identifier/locator split, the task of transitioning to a new network layer is difficult. For example, an identifier/locator split could have allowed an easier transition from IPv4 to IPv6.

There are two broad categories of identifier/locator separation techniques: host-based and network-based. Host-based solutions aim to provide a identifier/locator split within hosts, without affecting the network infrastructure, and network-based solutions that implement the split within the network but aim to avoid affecting the hosts.

3.1.1 Host-Based Identifier/Locator Separation

In this section, I discuss two examples of host-based identifier/locator separations: the Host Identity Protocol, followed by Shim6.

The Host Identity Protocol (HIP) introduces a new transport layer namespace, and a new protocol layer between the network and the transport layers [56]. In doing so, it provides a clearer distinction between the layers of the network stack. A key tenet of HIP's design is host-level authentication built into the transport layer. A host identity may be the public part of a public-private key. The host-identity is the input to a hash function to generate fixed-size identities of either a 128-bit "Host Identity Tag" (HIT) or 32-bit "Local Scope Identifier" (LSI, to allow the possibility of migration for older applications using 32-bit wide addresses) to provide fixed-size fields for use in HIP packets. Thus, HIP operates over IPv4 or IPv6, and may even ease communication between end hosts using the different network layers. With the transport layer using host identities, hosts should be able to roam freely without terminating existing data flows, provided there exists a good enough mapping service between identities and IP addresses, or some other rendezvous service. For HIP identities less likely to move frequently, a new DNS entry type to map identities to IP addresses may be viable.

Given that HIP does not require network modification, it could be incrementally deployed onto hosts. However, all hosts on a private network would have to be HIP-enabled to displace the desire for a network to multihome. Ideally, a host in a multihomed network would receive one hierarchically derived IP address from each provider network, and the HIP layer could decide which one to use.

Shim6 is another approach to offering multihoming without the use of PI addressing [57]. When using Shim6, end hosts within a network are aware of the various IP addresses they have been assigned, corresponding to the prefixes assigned by the provider networks. When initiating a communication, the host chooses one of its network locators, after which the

"shim" layer allows the network stack to present that locator to the transport layer as an identifier; the network layer can switch between the available prefixes to reflect service availability or policy change, without having to reset transport-layer associations. Shim6 only operates with IPv6, and otherwise relies on all hosts being shim6-aware to displace the desire for PI addressing.

In effect, each of these host-based solutions implies the removal of hosts with legacy transport layers if inter-domain routing state is affected. This may be possible, over time, on corporate networks with strict usage policies, but modification of *all* end hosts is often difficult [55]. Newer transport protocols can architect a better split between the transport layer and the network layer for better multihoming support. One such example is SCTP, which uses connection IDs at the transport layer rather than IP addresses [58].

3.1.2 Network-Based Core/Edge Separation

Network-based identifier/locator split solutions require some network modification, but aim to avoid modifying the end hosts. Like host-based solutions, there are various network-based solutions which have received attention. Network-based identifier/locator split schemes effectively mandate a core/edge separation architecture. Re-homed networks would be able to renumber with fewer concerns for breaking configurations across end hosts.

Here, I discuss two examples of network-based core/edge separation architectures, GSE and LISP.

GSE (so named after its Global, Site, and End-system address elements) proposes an alternative way to structure IPv6 addresses, in three parts: the end system designator (ESD); the site topology partition (STP); and the "routing goop" (RG) [59]. The ESD is the "identifier" part, uniquely identifying a host; the RG is the locator part, drawn from the hierarchically assigned portion of the address space in the public Internet; and the STP part is used to describe a subnet within the AS that the RG resolves to. The End System Designator (ESD) is an 8 byte, fixed size, globally unique identifier for the node to which it belongs. The ESD represents an end point (though each end system may have multiple ESDs); a suitable ESD may be a MAC address, but dynamically allocated ESDs for links which do not have a suitable MAC address to use are also required. The RG portion of an address is re-written on packets entering and leaving a network: on a packet leaving the origin network, the RG locator of the network is written into the source address field of the packet header; on entering the destination network the RG locator part of the destination address is set to a "site-local" value, not routable across the public Internet, after which the STP and the ESD are used to route the packet to the endpoint.

GSE is not incrementally deployable, and interoperation with the existing infrastructure

would require complex translation mechanisms. However, its hierarchical address assignment would help the inter-domain state scale.

LISP, the Locator/ID Separation Protocol, defines a "map and encap" (map and encapsulate) scheme for IPv4, where end hosts are identified by endpoint identifiers (EIDs) and routing locators (RLOCs) are distributed hierarchically from network to network [60]. Only the parent RLOC address enters the inter-domain routing system, similar to how IP addressing would currently work if PI addressing were not allowed. On routing a packet from one EID to another, if the EID does not exist within the same AS then a lookup takes place to discover the current RLOC of the AS in which it resides ("map"), and packets are then essentially tunnelled ("encapped") across the inter-network routing infrastructure. By enforcing hierarchical RLOC assignment, LISP would be able to solve the multihoming issue; the mapping function can simply return two or more valid RLOCs representing the network the destination EID is located within, the choice of which to use being left to the originator network.

LISP requires no changes to existing hosts, and little additional hardware in networks to facilitate the mapping from an EID to a RLOC, and subsequent encapsulation of packets sent to that RLOC. LISP also doesn't require modification of the IPv4 core of the network.

Different LISP variants are proposed which differ only in their implementation of the mapping service from an EID to a current RLOC. Massey *et al.* proposed a very similar scheme to LISP, which uses similar ideas albeit under different terminology [61].

However, LISP does not guarantee any reduction in state growth in the Internet's DFZ. Indeed, it does not guarantee reduction in state growth, but by introducing new dedicated ingress and egress points for networks, does introduce an additional point of failure, or an additional bottleneck for network performance.

The technologies covered here, GSE and LISP, have been considered as solutions to slow the growth of inter-domain routing state. However, they are designs that offer no guarantees, introduce additional complexity for network administrators and often imply manual management. As such, they may offer compelling solutions for network mobility and multihoming, but do not necessarily solve the routing state scaling problem.

3.1.3 Discussion

Host-based schemes are useful primarily as host mobility solutions, but do little to reduce routing state except from networks that can exert tight control over the devices in use. Network-based schemes, in particular LISP if it is transparent to the end hosts, could, in principle, reduce the rate of growth of routing state by encouraging new networks to connect to multiple providers via LISP rather than using PI addressing.

Some evaluation has been done on the effectiveness of ID/locator separation [62], suggesting that a suitably deployed locator/identifier separation scheme could vastly reduce FIB sizes in the Internet's DFZ. The paper makes the assumption, however, that such a scheme could be applied from the top down, and evaluates the routing gains based on this assumption; it does not present an analysis of future routing state growth assuming an evolution from the current system.

The common unanswered question for identifier/locator separation architectures is how best to implement the mapping function between the identifiers and locators, though it is quite reasonable to expect that DNS could be used for many nodes, perhaps alongside a similar mechanism dedicated to more mobile nodes.

As noted in Sections 2.3 and 2.4, much of the growth in the network, and therefore the origin of many prefixes, are the edge networks. Thus, one of the core/edge split schemes would seem a viable technology to reduce the state in the default-free zone. The primary attraction of a core/edge split is that it would allow greater flexibility for networks to choose their providers. Such a split, however, does not necessarily guarantee reductions in state scaling; indeed, while they may slow scaling, routing state would likely still grow linearly. Ultimately, the use of shortest path routing between indirection points in a hierarchical network structure inevitably leads to elongated paths when compared to the potential shortest physical path between any two points. The use of such indirection presents essentially unbounded path lengths.

ViAggre [63] suggests reducing routing overhead within an AS by dividing the IPv4 space into a known set of aggregates to be held at aggregation points in the AS. For example, the address space could be represented within an AS as 128 /7s, and distributed across multiple routers. The aggregation points hold forwarding references only for the address space advertised under the aggregates they hold. The aggregation point determines the next hop for the packet, and forwards it to the correct egress router over a tunnel (the tunneling mechanism is not prescribed, but MPLS Label Switched Paths are described). Thus, by introducing some additional router management, ASes may be able to scale their routing infrastructure. The use of the aggregation points may increase the path length through an AS, and the setup relies on additional router management, but ViAggre can potentially help forwarding tables scale. This is an entirely internal solution that does not seek to reduce the total amount of routing state, and so assumes that BGP continues to be a flat space and, by extension, that the number of advertisements and the rate of routing update will continue to increase.

The solutions presented, therefore, do not provide a guaranteed solution to the routing scalability problem, yet they necessarily elongate paths. They make no guarantees on maximum path elongation, and yet require additional manual management.

3.2 “Clean-Slate” Architectures

Many problems inherent in the current Internet architecture have stemmed from homogeneity in protocols and difficulty in being able to evolve [64, 65, 66]. Clean-slate architectures assume we can throw away the existing network and start over. The emphasis in many of these works is that there may be more to be gained from clean slate designs rather than by making minor tweaks to the existing system.

NIRA (the New Internet Routing Architecture) aims to encourage competition in the ISP market by allowing users choice in the routes their packets take, and which networks a packet will traverse [67]. Protocols would be in place for a node to discover its “up-graph” toward the core of the network, and also to query the destination’s “up-graph”, allowing choice in which networks a packet will traverse while in transit. The chosen route is then encoded into packet headers. This architecure describes a form of source routing. It aims to reduce the amount of routing state exchanged outside of a dense network core, but no analysis has been performed on the architecture.

HLP aims to ease the volume of routing updates observed in the current BGP system by exploiting the hierarchy of provider-customer relationships across most of the Internet graph and employing explicit information hiding, such that minor updates within one hierarchy will not be visible within other hierarchies [68]. BGP, as a path vector protocol, exposes path information to the rest of the Internet. HLP utilises a link-state protocol within a hierarchy to reduce levels of churn and improve convergence times, and a distance-vector protocol to communicate between hierarchies. HLP constrains routing updates, but does not affect table sizes.

‘Routing on Flat Labels’, ROFL, draws on distributed hash table (DHT) research to present an Internet architecture which does not require a hierarchical addressing structure [69]. The two projects ROFL draws on are Virtual Ring Routing (VRR) for intra-domain routing [70], and Canon for inter-domain routing [71].

VRR is a ring-based DHT system, intended for ad-hoc wireless networks, but employed by ROFL as an intra-domain protocol. In ROFL, routing takes place over identities alone, abandoning any sense of hierarchical network locators. The routers maintain successor and predecessor routing entries for all the nodes that it “owns” within the AS (*i.e.*, end hosts which consider that router to be its default router), which creates at the very minimum a ring-structure; routers will cache other routes that hosts are using. Forwarding is greedy, in that a node will route toward the nearest ID, but not beyond, when forwarding a packet.

Canon allows construction of hierarchical DHTs, thus allowing for a hierarchical graph of ASes somewhat similar to the current Internet graph. The Canon routing protocol guarantees that a packet will not leave an AS if the destination is contained within the same AS.

ROFL as an Internet architecture does not attempt to use network locators (in the hierarchical sense, at least), and does not attempt to always construct shortest paths between endpoints. Thus, it is an extreme example in the trade-off between routing state and path length: it retains minimal state, but lacks knowledge of actual topological information, and can create long paths between endpoints. Connecting new nodes into the existing infrastructure is also an expensive operation.

These architectures effectively provide unbounded path lengths. Building a routing infrastructure to use principles from distributed hash tables will provide consistently small table sizes, but to do so they rely on elongating path lengths. In the presence of a dynamic network, such an architecture may degrade to forming one large ring along which routing takes place.

3.3 Routing Fundamentals

The proposals covered in the preceding sections do not sufficiently solve the routing scalability problem. Fundamental principles of routing state that shortest path routing will always have at least worst-case linear state growth at each node [72], and this bound also applies to BGP, as the current inter-domain routing algorithm.

Shortest path routing algorithms are well-known, most commonly Dijkstra's algorithm [73] and Bellman-Ford [74]. These are commonly used in distributed routing protocols; the former is run locally on each node in a link-state routing protocol (in which all links are shared with the whole network), and the latter is run locally over the known distances to all other nodes as in a distance-vector routing protocol (in which nodes share distances, but not necessarily link state changes if the distance is not affected).

Intuitively, shortest path routing can move a packet through, and out of, the network in as few hops as possible. This reduces the number of packets in transit, and the bandwidth requirements on the links in the network. Many of the solutions proposed in the preceding sections, however, imply an additional level of indirection as a mechanism to reduce routing state. In doing so, they introduce additional hardware to manage, and simultaneously make no guarantees about the length of paths traversed.

Routing algorithms exist that construct highly compressed routing tables on special types of graphs: grid, tree or ring topologies, for example. Constraining the upper bounds on routing state while simultaneously adhering to bounds on path lengths on general graphs, however, is not so trivial.

The investigation of the trade-off between routing table sizes and the elongation of path lengths is not new, and it is reasonably well understood in graphs where hierarchical ad-

dressing can be applied [19]. As Section 2.4 discussed, the inter-domain routing infrastructure forms a flat space and, thus, in the absense of hierarchical addressing, alternative techniques are required.

3.3.1 Compact Routing Algorithms

We know that shortest path routing in general graphs with arbitrarily named nodes has worst-case linear state growth as the network expands. BGP forwarding table sizes in the DFZ are dictated by the number of prefixes publicly advertised.

The solution space for the routing problem is richer than simply shortest path routing. There is a set of algorithms known as compact routing algorithms (alternatively, compact routing *schemes*), that do not guarantee shortest path routing between all sources and destinations. Instead, the algorithms guarantee delivery between all sources and destinations with worst-case sublinear state retention at all nodes, with a guaranteed upper bound on the *path stretch*. Given a path length cr_{uv} when using a compact routing algorithm between two nodes u and v, and a theoretical shortest path length sd_{uv} between the same nodes, the path stretch is usually defined as:

$$\alpha_{uv} = cr_{uv}/sd_{uv} \tag{3.1}$$

In this form, $\alpha_{uv} = 1$ represents a shortest path. Alternatively, path stretch can be referred to as an additive measure of deviation from the shortest path, *i.e.*:

$$\beta_{uv} = cr_{uv} - sd_{uv} \tag{3.2}$$

In this form, $\beta_{uv} = 0$ represents a shortest path. The additive metric is less common than the multiplicative metric in compact routing literature, but can be useful. Whenever I refer to stretch in this dissertation, I am referring to multiplicative stretch unless I state otherwise.

The research that improves on the known bounds when trading path stretch against table sizes forms a clear progression:

Peleg initially introduced the area with discussion of unweighted graphs and bounds that applied to the state requirement of the sum of all nodes [75]. He observes the trade-off between all nodes in a network retaining state for all destinations, requiring tables of size $\Omega(n)$, or a network where nodes perform source routing, such that headers contain the full path, requiring packet headers of size $\Omega(n)$.

Awerbuch *et al.* improved upon this work by defining a compact routing algorithm for weighted graphs [76]. They prove it is possible to achieve an upper bound on the

space requirement at each node of $\tilde{O}(n^{\frac{1}{k}})$, and a maximum multiplicative path stretch of $\tilde{O}(k^2 9^k)$, for any $k \geq 1$. They improved on this bound again with space requirement at each node of $\tilde{O}(n^{\frac{1}{k}})$, and worst-case multiplicative stretch of $\tilde{O}(k^2)$ [77].

Eilam *et al.* improved on the bound again to achieve a space requirement at each node of $\tilde{O}(\sqrt{n})$, for a maximum multiplicative stretch of 5 [78], then Cowen asserted the same space requirement at each node but with a maximum multiplicative stretch of 3 [79].

Thorup and Zwick presented a compact routing algorithm that improves on the state bound with a space requirement of $\tilde{O}(\sqrt{n})$ at each node and and the same maximum multiplicative stretch of 3 [80].

Additionally, it is known that no routing algorithm can exist that guarantees sublinear state growth at all nodes while simultaneously guaranteeing a worst-case multiplicative path stretch better than 3 [72]. The compact routing algorithm presented by Thorup and Zwick is effectively optimal, in terms of minimising path length while achieving sublinear state growth.

Some work that has applied compact routing algorithms to single snapshots of the Internet AS graph have shown, however, that despite the worst-case bound of stretch-3, the mean multiplicative stretch between all nodes is around 1.1 [81, 82, 83]. This suggests that near-shortest-path routing is attainable with the benefit that state growth is sublinear. Put another way, this suggests that long-term scalable routing may be achievable with a slight weakening of the shortest path requirement, and not a total abandonment. Thus, there appears to be scope for a scalable future Internet that uses a different routing inter-domain protocol. The work in this area has, so far, lacked depth, and there been a lack of experimental evaluation of these protocols.

Improved compact routing based on the TZ compact routing algorithm has been described by Agarwal *et al.* that complements TZ compact routing through use of an additional distance querying infrastructure [84]. This scheme generates favourable path stretch results, offering shortest path routing in almost all cases.

An implementation of the TZ compact routing algorithm as a network protocol called S4 is defined in [85]. This was defined for large-scale wireless networks with static nodes as a mechanism to reduce routing state overhead. It relies on a fixed set of landmark nodes, and implies control of the network being configured, rather than decentralised networks operating under different owners as in the Internet.

Additionally, Brady and Cowen defined a compact routing algorithm intended for graphs with power law degree distributions. Their compact routing algorithm, referred to here as the BC compact routing algorithm, unlike most compact routing research, places an *additive* bound on path stretch but no bound on multiplicative stretch. Their compact routing algorithm constructs a spanning tree rooted at the highest-degree node, then constructs additional

smaller trees on connected regions on the periphery of the network, some number of hops away from the root of the primary tree; nodes then use a distance labelling to decide which spanning tree to use.

Enachescu *et al.* [86] study the trade-off between landmark set size and path stretch on specific graphs. They use a greedy heuristic that iteratively grows the landmark set, each iteration choosing landmarks that minimise the forwarding table entries required across the network. This achieves sublinear state growth at all nodes for many graphs, but cannot guarantee success for worst-case stretch less than 3 in general.

These routing algorithms are known as *labelled* schemes. They require additional information to a node ID to route packets toward a destination. *Unlabelled* algorithms that must use arbitrary IDs have been defined with multiplicative worst case stretch 3, and routing state requirements of $\tilde{O}(\sqrt{n})$ [87]. Analysis of this algorithm on two different Internet topologies leads to mean multiplicative path stretch of approximately 1.4 [83].

3.3.2 Complex Networks and Hyperbolic Routing

Scale-free networks are networks whose degree distribution follows a power law; they were discussed earlier in Section 2.3. They feature a heterogeneous degree distribution, with a large set of low-degree nodes connected to a smaller set of higher-degree nodes.

A recent development is the study of the geometry of these complex networks [88, 89, 90]. Understanding the geometry of complex networks allows a better understanding of their scale-free nature, and also a mechanism to harness the structure in routing.

Research on using complex network geometry has found that the Internet's topology is well suited to hyperbolic mapping which allows near-shortest-path routing [91, 92]. The study of hyperbolic mapping is promising early research, but an area which has not yet produced algorithms that may be feasibly decentralised.

3.3.3 Summary

Routing algorithms with compact state requirements exist that appear to be highly applicable to Internet topologies. Both the compact routing algorithms and the hyperbolic mapping techniques covered in this section are active research areas that may offer viable future routing architectures for the Internet.

Compact routing investigates the intuitive trade-off between routing state storage and path length, *i.e.*, the theoretical bounds on simultaneously minimising routing state and worst-case path length. One key result is that $O(n)$ bits of storage are still required at each node for any routing algorithm that guarantees paths less than 3 times the length of the shortest path

between a given source and destination [72]. However, it is possible to guarantee sublinear state growth at all nodes by guaranteeing that paths are maximally 3 times the length of the shortest path between two nodes, and so the TZ compact routing algorithm presented above is essentially optimal. Knowledge of a worst-case bound on path stretch is better than having potentially unbounded path stretch, and the work on this algorithm suggests that path stretch may usually be considerably lower than the upper bound. Routing within this upper bound with sublinear state growth is highly desirable for a future Internet routing architecture.

3.4 Summary

In this chapter, I have outlined various mechanisms and algorithms related to the topic of reducing routing state in networks, with a particular focus on solutions for the Internet.

First, I covered some of the alternative routing architectures that have been proposed for the Internet. Many of these are architectural modifications that aim to split endpoint identification from network location. Some modifications are host-based ID/loc schemes that genuinely enforce a split in the network stack between the network and transport layers. Others are network-based schemes more akin to core/edge separation, that do not alter hosts.

Next, I covered routing algorithms that seek to reduce routing state algorithmically. Some DHT-based architectures were discussed, but much of the interesting work involves the field of compact routing, which seeks to minimise the conflicting bounds of path stretch and routing state.

The algorithms that appear most promising as mechanisms for reducing inter-domain forwarding state on the Internet are the compact routing algorithms covered in Section 3.3.1, and the hyperbolic mappings covered in Section 3.3.2. The best known compact routing algorithms offer routing state at each node that scales as $\tilde{O}(\sqrt{n})$ with respect to the number of nodes in the network n, and have maximal multiplicative path streth 3. Of these, the compact routing algorithms appear more directly applicable to real graphs, and do not present too much of a departure from current routing nomenclature. Compact routing algorithms have shown much promise on Internet graphs, promising extremely small routing state requirements with near-shortest-path routing, but investigation of them on Internet topologies has been limited, and they are not defined as algorithms suitable for dynamic graphs.

The remainder of this dissertation will focus on compact routing.

Chapter 4

Static Compact Routing on Internet Topologies

In Chapter 3, I outlined various works related to minimising Internet routing state. One of the most promising directions covered is compact routing. In this chapter, I perform a systematic analysis of two of those compact routing algorithms, the Thorup-Zwick [80] (TZ) compact routing algorithm and the Brady-Cowen [93] (BC) compact routing algorithm on Internet AS topologies spanning 15 years.

The fundamental basis for investigating compact routing algorithms is that no shortest path routing algorithm for general graphs can guarantee better than linear state growth at all nodes. Compact routing research asserts that there is no known routing algorithm that can guarantee sublinear table growth unless we accept that paths may be *stretched* by a multiplicative factor of three in some cases [72]. Path stretch was introduced in Section 3.3.1 as a measure of path length relative to the theoretical shortest path between two nodes. Compact routing algorithms specifically aim to simultaneously minimise the bounds on path lengths and per-node routing state in a network, while satisfying the routing problem of forwarding data between any two nodes. Compact routing algorithms are known that guarantee all paths selected between any two nodes are not more than three times longer than the shortest possible path, while simultaneously guaranteeing sublinear state growth.

The TZ compact routing algorithm defines an upper bound of multiplicative stretch-3 on path lengths; the BC compact routing algorithm instead defines an upper bound of additive stretch d, where d is a configurable parameter (discussed in Section 4.2.2). Previous work on applying compact routing algorithms to synthetic power law graphs has shown that the performance of the TZ compact routing algorithm and the BC compact routing algorithm offer mean path stretches of around 1.1, considerably better than the worst-case upper-bounds [81, 82, 83], while retaining sublinear routing tables. However, the synthetic power law graph models used in these evaluations do not fully describe the AS topology, and

fail to describe where the scale-free nature of the node degree distribution originates [94]. However, the strong performance on synthetic graphs suggests that these algorithms will also perform well on the Internet graph, with its power law degree distribution [13], and may also perform considerably better than the theoretical upper bounds.

This previous work also tested on solitary AS topologies, which indicated similar performance. However, evaluation of compact routing algorithms on solitary Internet AS topologies does not provide enough information to assert that these algorithms perform well on *all* AS topologies. As I have shown in Section 2.3, the number of ASes and the number of IPv4 prefixes have grown by an order of magnitude between 1997 and 2011. Further, the Thorup-Zwick algorithm features a pseudo-random component, and thus running the algorithm on the same graph will provide different results. As an algorithmic solution to the long-term routing scalability problem in the default-free zone of the Internet [95], it is important to properly study these compact routing algorithms on topologies derived from real-world data. In the context of long-term inter-domain Internet routing, on the order of decades, it is reasonable to expect the continued fragmentation of the remaining IPv4 address space and the uptake of IPv6 to lead to massive routing and forwarding state requirements.

A more in-depth analysis is required to assert good or bad stretch performance with these compact routing algorithms. In this chapter, I offer such an analysis, incorporating AS graph data between 1997 and 2011 to evaluate the performance of the algorithms during continuous growth of the graph, and multiple evaluations on the same graph to assert consistent behaviour in the presence of the pseudo-random node selection.

In this chapter, I provide such results that show the TZ compact routing algorithm to be consistently stable over long periods of time, and also reasonably stable in the presence of the pseudo-random landmark selection. I show that TZ stretch performance means that, in all cases, the vast majority of paths are shortest paths, and the vast majority of the remainder require only one additional AS hop. I also show that the Brady-Cowen compact routing algorithm exhibits similar performance, but that the resultant path stretches are less stable during the timespan evaluated.

The contributions of this chapter are as follows:

1. I present the first systematic study of the performance of the TZ and BC compact routing algorithms on the AS graph, in terms of forwarding table sizes and the distribution of path stretch values, to demonstrate that these algorithms can perform as well on the AS graph as suggested by previous studies on power law random graphs. As part of my analysis, I show both multiplicative and additive stretch over the 14 year period using one snapshot of the Internet topology every six months starting from 8 November 1997 and running to 8 November 2010. This offers two snapshots per year, totalling

27 snapshots of the AS graph, to provide a representative coverage of the Internet as it has evolved.

2. I present an analysis of the TZ landmark selection algorithm, to better understand *why* this pseudo-random landmark selection algorithm performs so well on the AS graph. I discuss why a pseudo-random selection is not appropriate for practical Internet routing.

3. I present an analysis of the contribution of various components of the BC compact routing algorithm to its performance. I show that the addition of a heuristic built into the algorithm [93] is one of the key contributors to lowering its mean path stretches.

Both of these compact routing algorithms require total knowledge of the graph to label nodes appropriately before routing can take place. This is not useful in a distributed and dynamic network. I address these problems of heavy centralisation, in Chapters 5 and 6.

This chapter is structured as follows:

Section 4.1: covers some of the related work on applying compact routing to the Internet graph.

Section 4.2: describes in detail the TZ and BC compact routing algorithms.

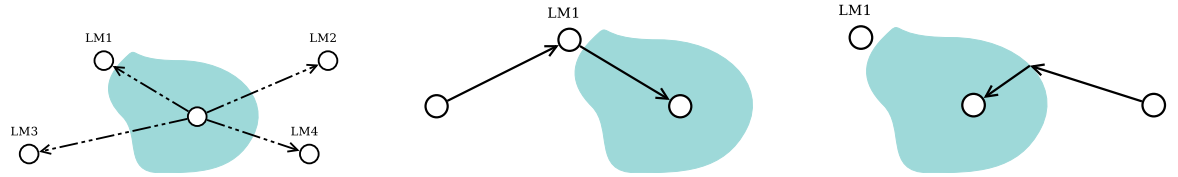
Section 4.3: presents an analysis of forwarding table sizes, path stretches, and the behaviour of the TZ landmark set algorithm.

Section 4.4: summarises the chapter.

4.1 Previous Work

Previous work has evaluated compact routing algorithms, including the TZ and BC compact routing algorithms [82], on power law random graphs with node degree distributions obeying a power law, such as the classical Barabási-Albert model of preferential attachment [96]. The previous work has indicated that these compact routing algorithms offer mean multiplicative path stretch of around 1.1 on these graphs, considerably better than the worst case of multiplicative stretch 3. However, degree distribution alone does not accurately model the Internet AS graph and ignores topological features such as the clustering evident in AS topologies. To enable assertions about the Internet graph it is desirable to analyse these algorithms on Internet topologies derived from real data rather than on synthetic topologies.

The algorithms have been tested on two Internet topologies derived from different sources in [83], which indicated results comparable with the results from the synthetic topologies.



(a) A node, with a cluster of nodes (not shown) that it retains state for, and the landmark set.

(b) Packet forwarding via the nearest landmark, LM1.

(c) Packet forwarding, shortcut prior to LM1.

Figure 4.1: Illustrative example of TZ landmark-based routing.

However, observations on so few snapshots do not capture the growth or variation in complexity of the AS graph over the years. It is necessary to confirm these previous results by running the algorithms on additional AS topologies, and it is also important to perform an in-depth analysis of the behaviour of these algorithms on Internet topologies spanning the available historical data. Using this real data will allow a better understanding of how the algorithms behave on realistic network topologies that have undergone massive growth, and provide indications of the variability of the results on different topologies.

4.2 Compact Routing Algorithms

In this section, I describe the two compact routing algorithms that I evaluate later in this chapter, with the TZ compact routing algorithm in Section 4.2.1, and the BC compact routing algorithm in Section 4.2.2.

4.2.1 Routing using the Thorup-Zwick Algorithm

Given a graph $G = (V, E)$, the TZ compact routing algorithm [80] selects a set of nodes $A \subseteq V$ to act as *landmarks*, and restricts forwarding tables at all nodes to contain only the set A and a partial view of the rest of the network. A node u is retained in the forwarding table of node v where $u, v \in V$ if u is in the *cluster* C_v as specified in Equation 4.1; $d(u, v)$ is the shortest path distance between u and v, and $D(u, A)$ is the shortest path distance between u and the closest node in the landmark set A.

$$C_v = \{u \in V \big| d(v, u) < D(u, A)\} \tag{4.1}$$

That is, u is retained in the forwarding table of node v if the shortest distance between u and any of the nodes in A is less than the distance between u and v.

The TZ landmark selection algorithm operates as follows. Initially, A is populated with a set of nodes $W \subseteq V$ by selecting nodes independently from V with uniform probability of

$2\sqrt{(n \log n)}/n$, where $n = |V|$. The algorithm uses Equation 4.1 to construct a cluster for each node, and then derives a set of nodes $W \in V$ whose clusters are larger than $4n/s$.

$$W_i \leftarrow \{v \in V \big| |C_v| > \frac{4n}{s}\} \tag{4.2}$$

In the evaluation of the TZ compact routing algorithm in [80], s is set to $\sqrt{n/\log n}$. I use this value of s throughout this dissertation. With $s = \sqrt{n/\log n}$, all W_i must contain fewer than $4\sqrt{n \log n}$ nodes.

The algorithm expands A by incorporating the nodes in W into A, and repeats this process until the cluster for each node is smaller than the limit (*i.e.*, $W_i = \emptyset$). The forwarding table of each node v holds entries for the union of C_v and A. The algorithm terminates on the first iteration that W is empty, *i.e.* no nodes has a cluster that breaches the bounds.

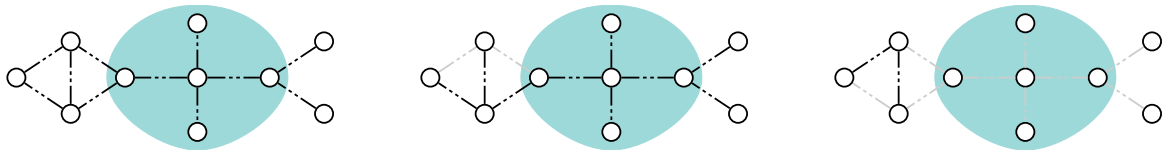
With $s = \sqrt{n/\log n}$, the following has been shown [80]:

$$|A| \leq 2\sqrt{(n \log n)} \tag{4.3}$$

With these components combined, each node v is aware of the full landmark set A and its cluster C_v. This is illustrated for a small network in Figure 4.1a. Note that Equation 4.1 means that C_v is empty for all $v \in A$, and so landmark nodes are only aware of other landmarks.

Packet forwarding under TZ routing requires three pieces of information: the identifier of the destination node, the identifier of the destination's landmark node, and the next hop from the landmark to the destination. The forwarding algorithm at each node attempts to match on the destination, and forwards the packet toward the destination's landmark if the destination is not found. If the packet arrives at the landmark, then the landmark node uses the next hop information in the packet header to forward the packet toward the destination. The landmark is used by the forwarding algorithm at intermediate nodes only when the destination is not known.

It is important to understand that paths traversed by packets do not necessarily include the landmark. To illustrate, the same landmark is used in the small example network in Figures 4.1b and 4.1c. Paths that use the landmark imply that the *longest* possible TZ path has been taken. It is possible, and expected, that packets arrive at an intermediate node which has, through the clustering process, retained a reference to the packet's destination prior to reaching the landmark. Shortest paths are always used from the source toward the landmark until a route to the destination is found, at which point the shortest path is used from that node to the destination itself. Thus, stretch is only incurred if the path toward a destination's landmark takes the packet off the shortest path from source to destination.



(a) Core with d=2 around highest degree node.

(b) Spanning tree originating from highest degree node.

(c) Additional spanning tree on the connected component to the left of the core.

Figure 4.2: Illustrative example of BC compact routing.

Building clusters according to Equation 4.1 means that for each node v, any node u adjacent to v will not be included in v's forwarding table if u is itself adjacent to a landmark. Previous analysis modified the basic algorithm to insert entries to all adjacent nodes into forwarding tables after the landmark selection and clustering phase [81]. In this chapter I look primarily at performance with all direct neighbour links in use, but I also evaluate the performance gains of this modification to the algorithm in Section 4.3.1.

4.2.2 Routing using the Brady-Cowen Algorithm

The BC compact routing algorithm [93] aims to improve on the worst-case performance of the TZ algorithm, specifically in power law graphs. It provides an upper bound on *additive* path stretch of d hops, *i.e.*, maximally d hops longer than the shortest path. Using the highest-degree node as the root, d specifies the diameter of a *core* of nodes at most $d/2$ hops from this root; nodes outside the core constitute the *fringe*.

The algorithm then operates as follows. First, a primary spanning tree is constructed on the full graph from the root. One spanning tree is then constructed on each connected region contained within the fringe. I refer to BC using the union of these two sets of trees in Section 4.3.1 as the simplest BC variant, *bcn*. The core, the primary spanning tree, and the additional spanning tree built within a connected component are illustrated in Figure 4.2 for a small graph.

Next, the algorithm identifies the minimal set of edges within the fringe that must be removed to make the fringe acyclic, and constructs for each edge e a full spanning tree that includes e itself. I refer to the union of the resulting set of trees and *bcn* as a second variant, *bce*. In the evaluation of the BC algorithm in [93], if fewer than five edges are identified, then edges in the fringe are chosen randomly from which additional spanning trees are constructed to create five trees. The selection of the number five is not explained in [93]. I refer to this third variant of the algorithm as *bch*.

Once all trees have been constructed, each node in each tree is given a Peleg distance label [97] and a TZ tree label (described in [80], and different to that described in Section 4.2.1). Peleg distance labels are constructed over a full graph, and allow a node to

calculate its distance to any other node from only the information held in the label. Running the Peleg distance labelling algorithm on each of the trees will give different, tree-specific, distance labellings for each node. Forwarding from a node u to a node v requires the following steps: u compares its own Peleg distance labels against v's distance labels to determine the tree with the fewest hops to v, then forwards packets using v's TZ tree label for that tree. Once the shortest tree has been selected, subsequent nodes need not use the distance labels, only the tree labels.

Increasing the d parameter reduces the size of the set of edges identified for *bce*, but increases stretch. In this chapter, I set $d = 6$ as the largest value that identifies additional edges for the *bce* variant on our snapshots. Graham Mooney explored the properties of d on AS graphs further [98]. This choice of d is natural for Internet topologies. Since the value d indicates a diameter around a root node, this diameter will extend $d/2$ hops from that root. Recall from Figure 2.3 (page 13) that the mean shortest path lengths on observable Internet topologies is consistently between 3.5 and 4 AS hops. Thus, if d were 8, the core will usually encompass the full graph, and if d were 4, would encompass fewer than 10% of the nodes. $d = 6$ is the only natural parameter for Internet topologies.

This concludes the description of the two compact routing algorithms evaluated in this chapter. The remainder of this chapter is devoted to the experimental analysis and performance results.

4.3 Experimental Analysis

To understand the behaviour of these compact routing on the Internet graph, I evaluate the algorithms against the dataset described in Section 2.1. That is, a series of AS topologies derived from full BGP routing tables, with one AS topology every six months starting 8 November 1997 (the first available date in the Route Views archive) through 8 November 2010, inclusive. The AS topologies are derived from archived BGP table snapshots, using the AS paths stored within. The data processing pipeline, and the routing tables themselves, are discussed in detail in Appendix A.

The compact routing algorithms presented here both pre-process a graph according to the outlines described in Sections 4.2.1 and 4.2.2 to generate the appropriate node labelling to permit compact routing. Both compact routing algorithms were run on each graph to generate the appropriate labels.

Once the pre-processing is complete, all forwarding table sizes are known. To compute path stretches, the shortest paths between all nodes must be computed for each of these graphs, to compare with the path lengths when using these compact routing algorithms.

The algorithms differ so significantly that the manner in which the path stretch is evaluated is different. The BC compact routing algorithm's labelling scheme features a compact distance labelling that allows all nodes to compute the distances between each other (such that the tree with the fewest hops between source and destination can be chosen), and thus all BC path lengths are easily computable immediately after the initial pre-processing phase. The TZ compact routing algorithm's labelling contains no distance information. The TZ forwarding algorithm will attempt to forward along a shortest path toward a landmark node, and from there to the destination, but it is possible that paths are "shortcut" to the destination prior to reaching the landmark. Thus, the path stretch incurred by TZ compact routing is dependent on the forwarding state held at intermediate nodes between source and landmark, and computation of distances when using the TZ compact routing algorithm is not so trivial as to compute for two nodes u and v the sum of $d(u, l_v)$ and $d(l_v, v)$, where l_v indicates the landmark selected during pre-processing for v. Accordingly, the forwarding algorithm must be simulated to determine the number of hops between source and destination pairs. Further, paths under the TZ compact routing algorithm are not necessarily symmetrical. That is, $d(u, l_v)$ then $d(l_v, v)$ may be different in length to $d(v, l_u)$ followed by $d(l_u, u)$, and so the distances in both forward and reverse directions may differ.

The complexity of comparing routes between all possible pairs of nodes grows as $O(n^2)$, where n is the number of nodes. Thus, it becomes computationally infeasible to simulate the TZ forwarding algorithm on the larger graphs. Rather than exhaustively test routing between all pairs of nodes, representative subsets of the paths in each snapshot were selected on which to evaluate the stretch performance. The sample sets are generated by selecting, uniformly randomly, 1% of all nodes in each snapshot, and simulating the forwarding algorithm from each of those nodes to all other nodes in the graph, and then the reverse path. By selecting uniformly randomly, there is no bias toward high-degree nodes or low-degree nodes, and landmark nodes can also be selected. The sample sizes are large enough that the range of observed stretches is representative of the full graph, and also large enough to expose a representative number of maximum-stretch paths. To help confirm that the results are consistent, the smallest of the graphs, 8 November 1997, is tested using *all pairs* of nodes.

As the TZ landmark selection algorithm builds landmark sets from an initial random selection of nodes, the algorithm is *not* deterministic. Stretch results may vary given different landmark sets. To accommodate this, I generate a representative sample of 50 landmark sets for each snapshot, and simulate forwarding on 5 of these. The 5 sets are chosen according to the size of the 50 landmark sets, so that I evaluate with varying landmark set sizes from smallest to largest.

Routing under the BC algorithm is deterministic, so path stretch values can be directly determined without simulation. This information is used to compute all-pairs of BC distances.

In the following subsections, I primarily focus on the path stretches observed on AS topologies using these compact routing algorithms (Section 4.3.1), and also the forwarding table sizes required to support packet forwarding (Section 4.3.2). Additionally, in Section 4.3.3, I discuss the behaviour of the TZ landmark selection algorithm when applied repeatedly to AS topologies.

4.3.1 Path Stretch

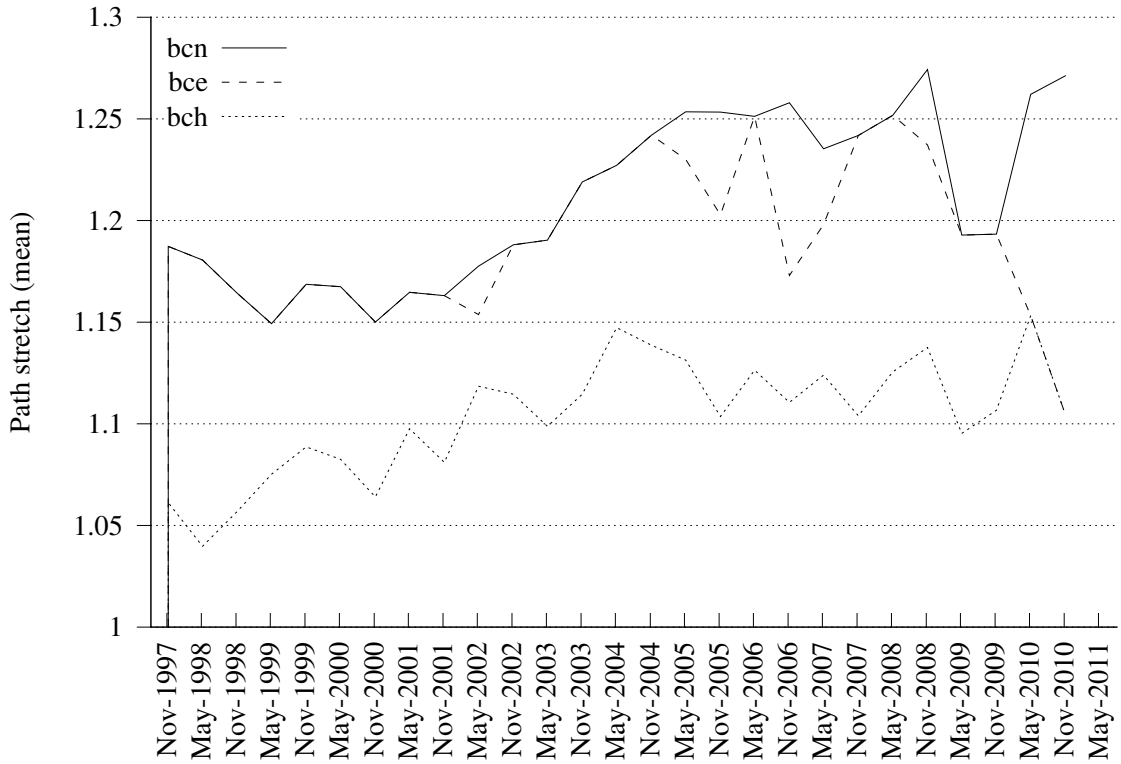
In this section, I discuss the stretch behaviour of the different algorithms, to confirm that they perform well on the AS graphs. In particular, I show the behaviour of the minor variants of the two algorithms already identified in Section 4.2.

Figure 4.3a shows the mean multiplicative path stretches for the three variants of the BC compact routing algorithm defined in Section 4.2. The algorithm is deterministic, so there is no variation in the result when applied to the same graph multiple times. There is a clear reduction in stretch when extra trees are added in *bch*, in addition to the primary tree set *bcn*, but with $d = 6$ *bce* often does not offer any improvement. The performance gain observed here is quite simple to understand: additional spanning trees use more of the underlying links in the core of the network, offering multiple paths to all destinations. It is clear that BC performance declines slightly over time, and is more variable than TZ.

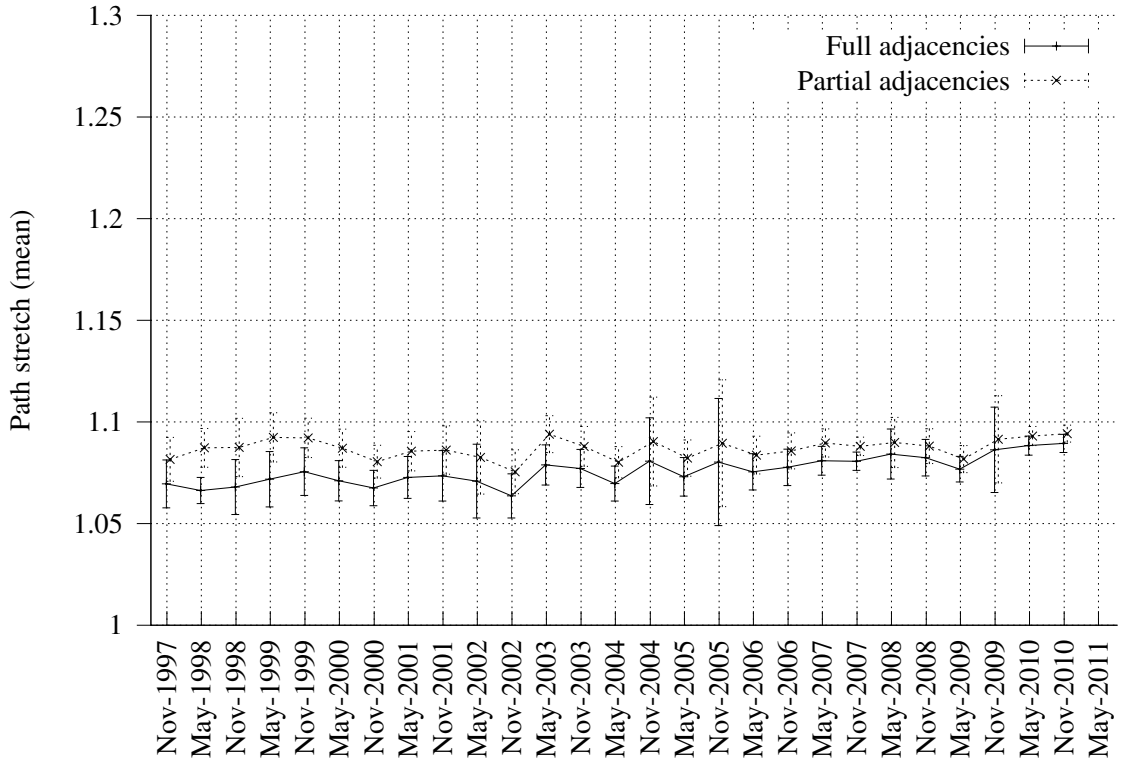
Figure 4.3b shows mean multiplicative path stretches (and their standard deviations) derived from applying the TZ compact routing algorithm to AS graphs, using the five landmark sets described earlier in Section 4.3, both with and without full adjacencies. Adding all adjacencies that are missing after the algorithm's clustering process into forwarding tables leads to a minor improvement in performance, though this gain has narrowed over time. This decrease in performance is due to the sublinear growth of the landmark set relative to the full network, and the corresponding reduction in nodes directly adjacent to a landmark. Given that full adjacencies always offer best performance, for the remainder of this chapter I use this variant of the algorithm. The average multiplicative stretch in November 2010 with full adjacencies is around 1.09 (with a standard deviation of 0.004).

Figure 4.4a shows the multiplicative stretch for the BC experiments. The results here are variable, as may be expected from Figure 4.3a, but in all tests the majority of paths are multiplicative stretch-1 (63.21% for November 2010), and a very small proportion are above multiplictive stretch-2.0 (0.37% for November 2010). Correspondingly, Figure 4.5a shows that most stretch is incurred by one additional AS hop.

By setting the parameter $d = 6$, BC guarantees a maximum additive stretch of 6 hops. Additive stretch of 6 occurs between only 381 pairs of nodes spread across all snapshots tested (out of over 650 million pairs in total). This additive bound means 6 additional hops

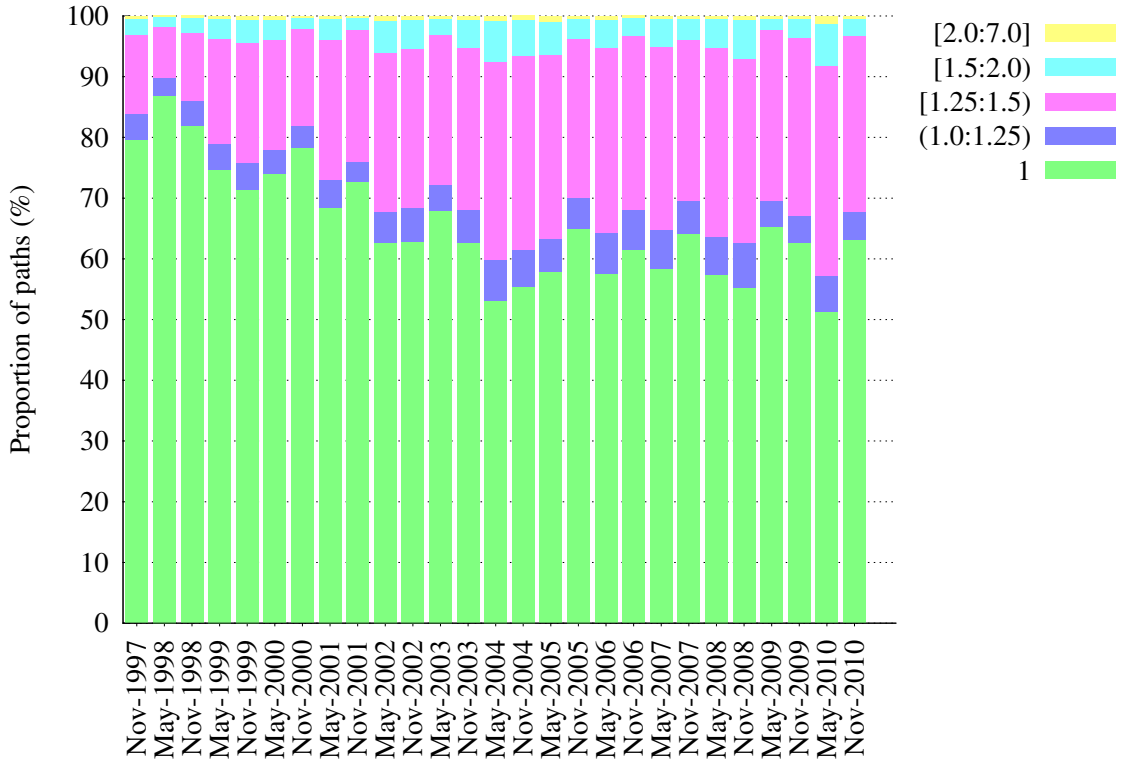


(a) BC means of means: Path stretches are consistently low, on average, but are more variable than TZ, and indicate an upward trend.

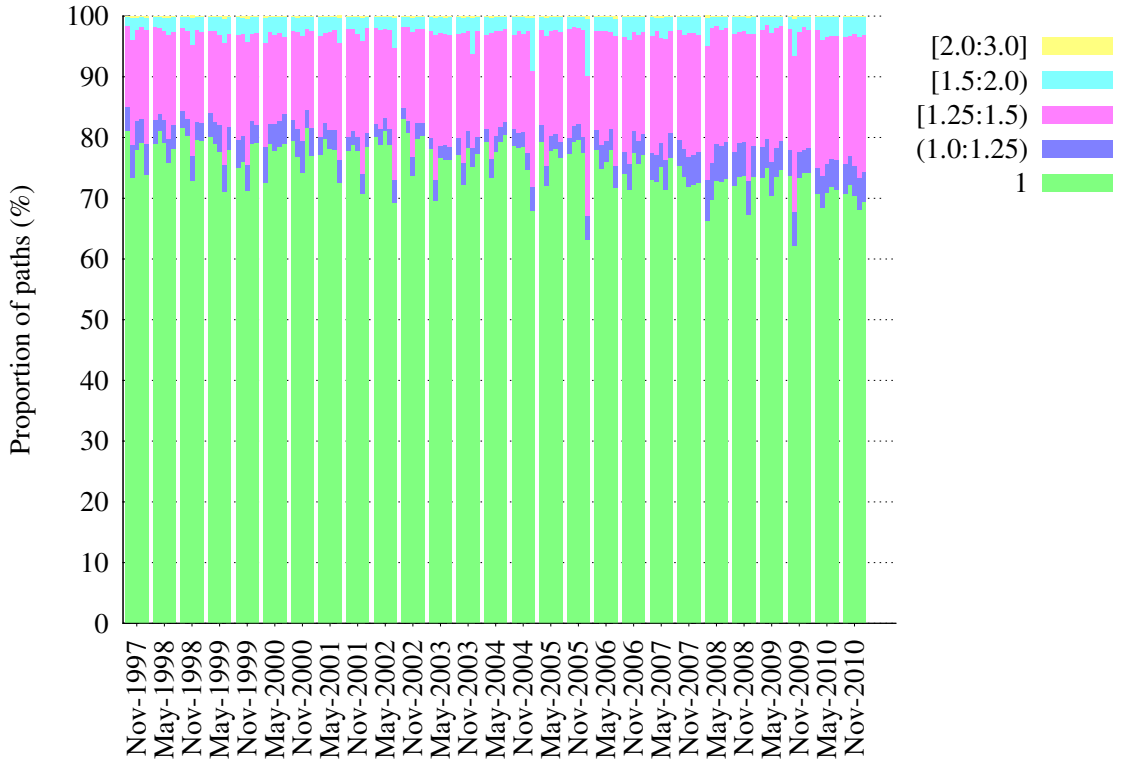


(b) TZ means of means: Path stretches are consistently low, on average, and show little variation across landmark sets.

Figure 4.3: Effect of variations in each algorithm on multiplicative path stretch.

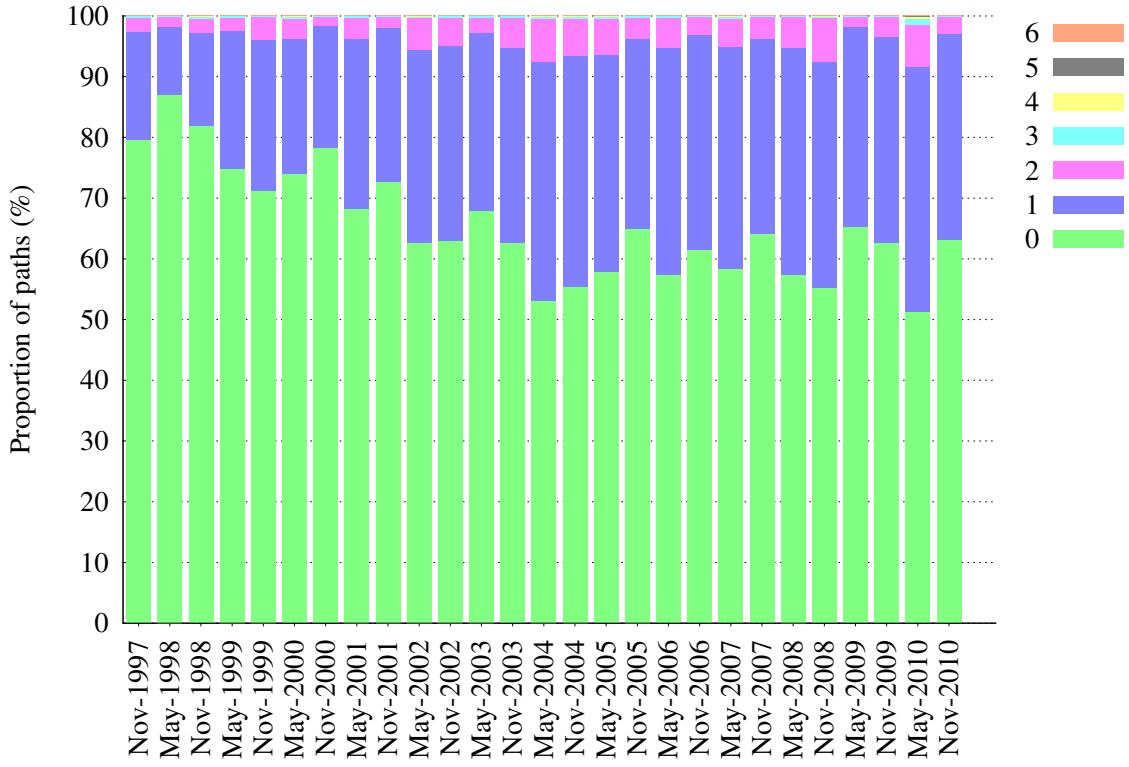


(a) BC routing multiplicative stretches.



(b) TZ routing multiplicative stretches.

Figure 4.4: Multiplicative stretch behaviour for the TZ and BC compact routing algorithms across all snapshots.



(a) BC routing additive stretches.

(b) TZ routing additive stretches.

Figure 4.5: The additive stretch behaviour of the BC and TZ compact routing algorithms across all snapshots.

which, for adjacent nodes, means the maximum *multiplicative* stretch is 7. In November 2010, 5,260 paths with multiplicative stretch higher than three were observed. The majority of these would be eliminated if a routing algorithm were used that deviated from BC's tree-based labelling algorithm to use a different form of labelling for direct neighbours.

I show actual multiplicative stretch distributions for each of the TZ experiments in Figure 4.4b. The results from multiple tests on each snapshot are shown. The resulting stretch is clearly more stable. The vast majority of paths do not exhibit any stretch at all (*i.e.*, are shortest path); across all tests on the November 2010 snapshot, 70.2% of paths are shortest paths. 4.95% exhibit multiplicative stretch of less than 1.25; 26.54% exhibit multplicative stretch of less than 1.5, and a further 3.17% exhibit multplicative stretch between this and 2.0. Fewer than 0.01% of paths exhibit stretch higher than 2.0. It is important to realise that, given the upper bound of multiplicative stretch-3, this performance is exceptional. Figure 4.5b shows that the path stretch is generally only one additional AS hop on the shortest path: 26.6% of paths in November 2010 have one additional AS hop; 3.14% have two additional AS hops.

It is claimed in [82] that the average stretch with TZ compact routing decreases as the network size increases, for power law random graphs. These results suggest the opposite is true on Internet graphs, that average stretch has increased slightly. Most of this increase appears to come from single additional AS hops. This result emphasises the importance of testing these algorithms against real-world data.

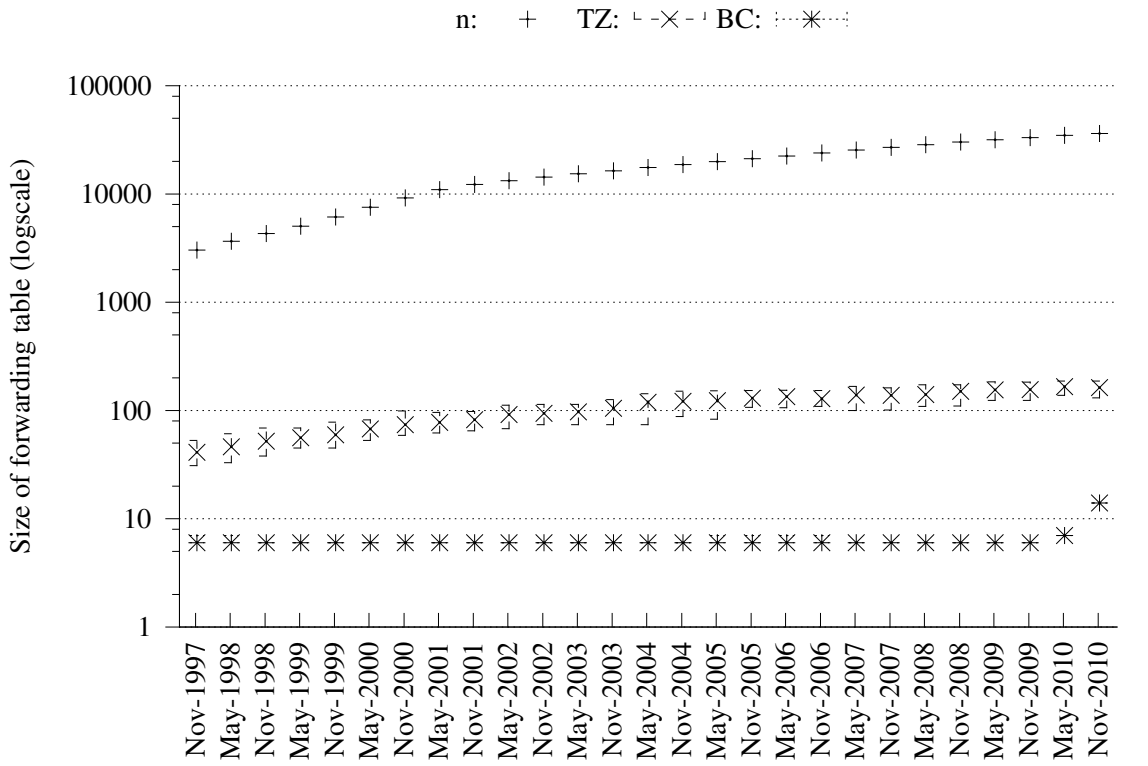
On all of the simulations with the TZ routing algorithm, only two tests featured paths with five additional AS hops. In all other cases, four additional AS hops was the maximum additive strech.

The performance of these two algorithms when applied to a range of AS topologies confirms the previous observations that the application of compact routing algorithms to AS topologies can lead to strong performance, *i.e.*, approaching shortest path routing with small amounts of forwarding state. Running these algorithms on a range of topologies also indicates that the TZ compact routing algorithm is considerably more consistent in its path stretch distribution when presented with an evolving network than the BC compact routing algorithm.

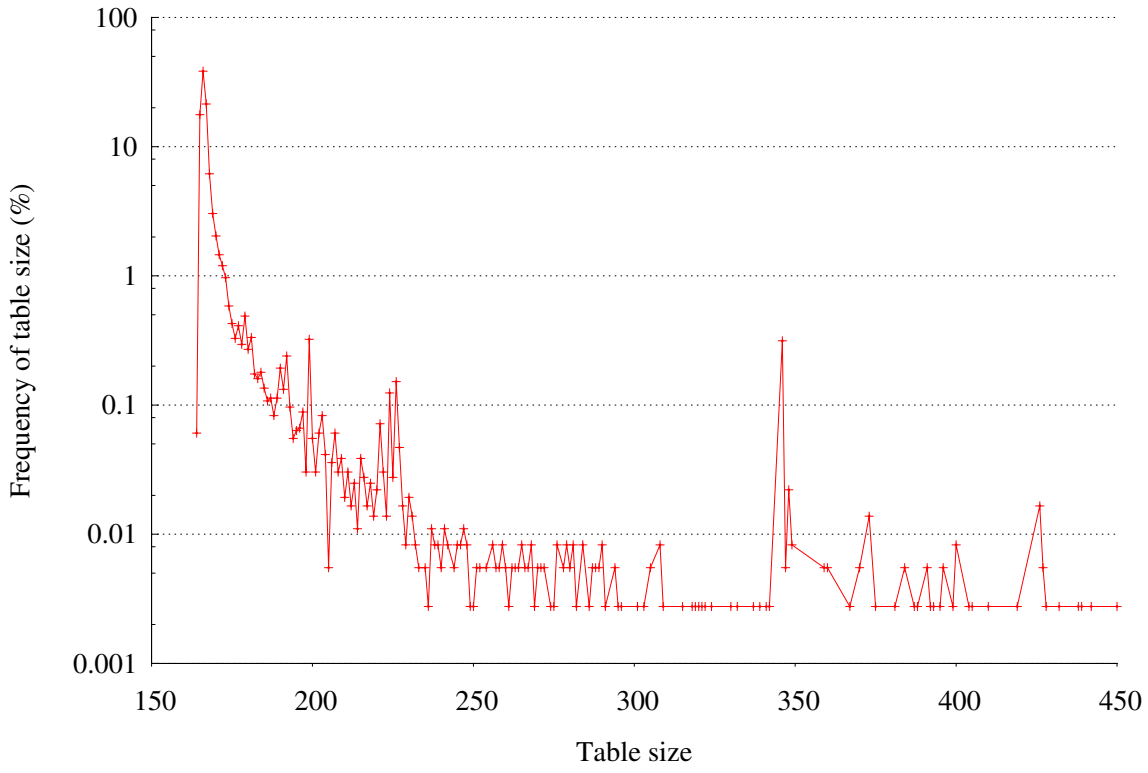
4.3.2 Forwarding Table Sizes

As the path stretch introduced by these compact routing algorithms is extremely low, approaching shortest path routing, next I look at the size of the forwarding tables that are required at nodes to support this level of performance.

Figure 4.6a shows average forwarding table sizes at all nodes for each of the TZ (with full adjacencies) and BC (*bch*) experiments. The upper line on this graph is n, the number of



(a) Mean routing table growth. TZ and BC values indicate means, with 1^{st} and 99^{th} percentiles. Note that all nodes here retain *full* forwarding tables according to their routing algorithms, and no default routes have been used.



(b) Distribution of TZ routing table sizes.

Figure 4.6: Forwarding table growth.

participating ASes in the Internet for each snapshot. This has grown from 3030 in November 1997, to 36255 in November 2010.

In this time, the average size of TZ forwarding tables on these snapshots has risen from 41.13 to 162.59 entries, representing 0.83% to 0.45% visibility of the full graph respectively. This, given the proportion of shortest paths discovered by the TZ algorithm, is extremely small. I show a representative distribution of forwarding table sizes for one of the landmark sets generated for the November 2010 snapshot in Figure 4.6b. As these forwarding tables retain all adjacencies, the largest of the forwarding tables are dictated by node degree, and the upper bound on the forwarding table size of a node $w \in V$ becomes, maximally, $d_w + |A| + |C_w|$, where d_w is the degree of w. Given the highly heterogenous degree distribution of the network [13], it is not surprising that over all of the tests run using the November 2010 sample, only 0.0332% of nodes had forwarding tables containing more than 1,000 entries, and 91.19% have forwarding tables containing 180 entries or fewer.

The BC forwarding table sizes shown in Figure 4.6a are even smaller, as they are based on the number of trees each node is contained within. Given that each node is a member of the primary spanning tree, any other spanning trees generated across the full graph, and possibly one smaller tree connecting nodes in the fringe, the forwarding tables generated by BC show little variance (error bars are drawn, but are not visible). These forwarding tables are constant in size until 2010 owing to the use of $d = 6$ and the algorithm's reliance on a minimum of five spanning trees in addition to the primary tree. The 2010 snapshots make use of additional trees, indicating network growth outwith the core region that the algorithm has made use of. These forwarding tables for November 2010 contain 14 entries in over 99% of nodes, and 15 entries in the remainder.

The forwarding tables stored by these algorithms are tiny; the tables computed by the TZ compact routing algorithm grow sublinearly with the size of the network, but the tables computed by the BC compact routing algorithm are effectively constant throughout. The distinction appears to be that the tables presented by TZ compact routing are larger because the algorithm will use more links than the BC compact routing algorithm's trees, and is thus more stable in the presence of massive network growth. The conclusion is the same in both cases, that these tiny forwarding tables are capable of supporting near-shortest path routing on Internet AS topologies derived from real data.

4.3.3 Remarks on TZ Landmark Selection

The TZ landmark selection algorithm constructs its landmark sets from an initial random set of nodes, as described in Section 4.2.1. Intuitively, multiple landmark sets generated independently by the TZ landmark selection algorithm on a large graph seem likely to contain no overlapping nodes.

However, this seems to not be the case. For each of the AS graph snapshots used to investigate path stretch in this chapter, I generated a representative sample of fifty independent TZ landmark sets, and counted the recurrence rate of landmarks. Figure 4.7a shows a histogram of the recurrence rates for ASes from the fifty landmark sets generated for the 8 November 2010 Internet AS graph snapshot, and is representative of the pattern of return rates across all snapshots. Most ASes do appear only once, a few nodes (2.89% of all ASes observed) appear more than twice, and one node was selected in all fifty landmark sets. This is an unintuitive result, as the first iteration of this algorithm randomly selects nodes from the graph irrespective of node degree.

There is a clear correlation between the likelihood of a node being selected as a landmark and its node degree. In Figure 4.7b, I aggregate the frequency of selection against node degree for each node observed in the landmark sets for each snapshot. It is clear that nodes with degree $\gtrapprox 100$ are very likely to feature in the landmark set. This is unexpected, but can be explained as follows. The degree distribution of the AS graph obeys a power law [13], *i.e.*, it is dominated by low-degree nodes. As the first iteration in the landmark set consists of nodes drawn from the full graph with uniform probability, it will contain primarily low-degree nodes. This will, with high probability, lead to large clusters forming around high-degree nodes based on their proximity to this initial landmark set. The landmark selection algorithm attempts to subdivide large clusters by adding nodes with large clusters to the landmark set on the next iteration, following Equation 4.2. Since high-degree nodes are intrinsically 'near' to much of the network, they are disproportionately likely to have clusters large enough to break the constraints specified in the equation, so they are therefore often selected by the algorithm in subsequent iterations W_i. Note that high degree nodes on these graphs are rare compared to nodes with a low degree, and hence Figure 4.7b is still dominated by nodes that only appear once across all landmark sets.

It is reasonable to believe that this landmark choice aids the stretch performance of the algorithm. As stated in Section 4.2.1, if the path toward a destination's landmark does not deviate from the shortest path to that destination, no stretch will be incurred. As high degree nodes are connected to a large proportion of the network, many paths already naturally use these nodes for transit as the shortest path toward a destination. This is beneficial: this level of connectedness increases the likelihood that this landmark is already on a shortest path to many destinations.

Note that I am discussing a policy-free network, and that node degree is an arbitrary measure of the *importance* of a node in a real network [99]. Algorithms that rely on node degree as a measure of importance [100] are unlikely to perform well on the Internet, since high node degree may indicate that a node has many customers and few providers, or it may indicate a node with many providers and a high level of peering with other transit nodes. However, as discussed in Section 2.5, the Internet is not an undirected graph and ASes implement

various policies over certain links implying that some ASes may not form part of a valid path between certain sources and destinations. That is, node degree alone is not a useful mechanism to select landmarks for the Internet, and the particular landmarks selected here may not be willing to operate as such in the real-world. What is more important is that landmarks be well-connected to the wider network, and be long-lived (and therefore less likely to depart the network in the immediate future); this is covered in Chapter 5.

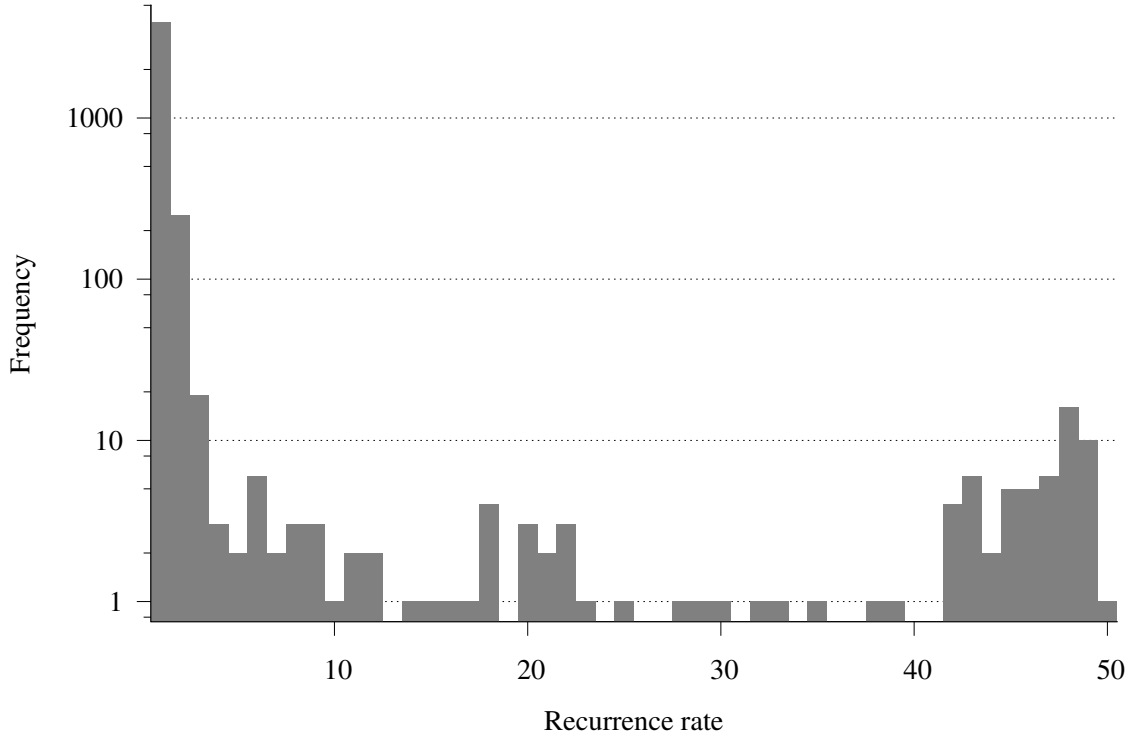
This section has covered the results from the experimental analysis of two compact routing algorithms on AS topologies, the TZ compact routing algorithm and the BC compact routing algorithm. I have shown results indicating path stretches when applying both of these algorithms to AS topologies, and discovered them both to exhibit mean path stretches close to 1. TZ compact routing is considerably more stable than BC compact routing over time. I also indicated the table sizes observed when using these algorithms, and showed that they produce tiny tables when compared to the size of the full network.

4.4 Summary

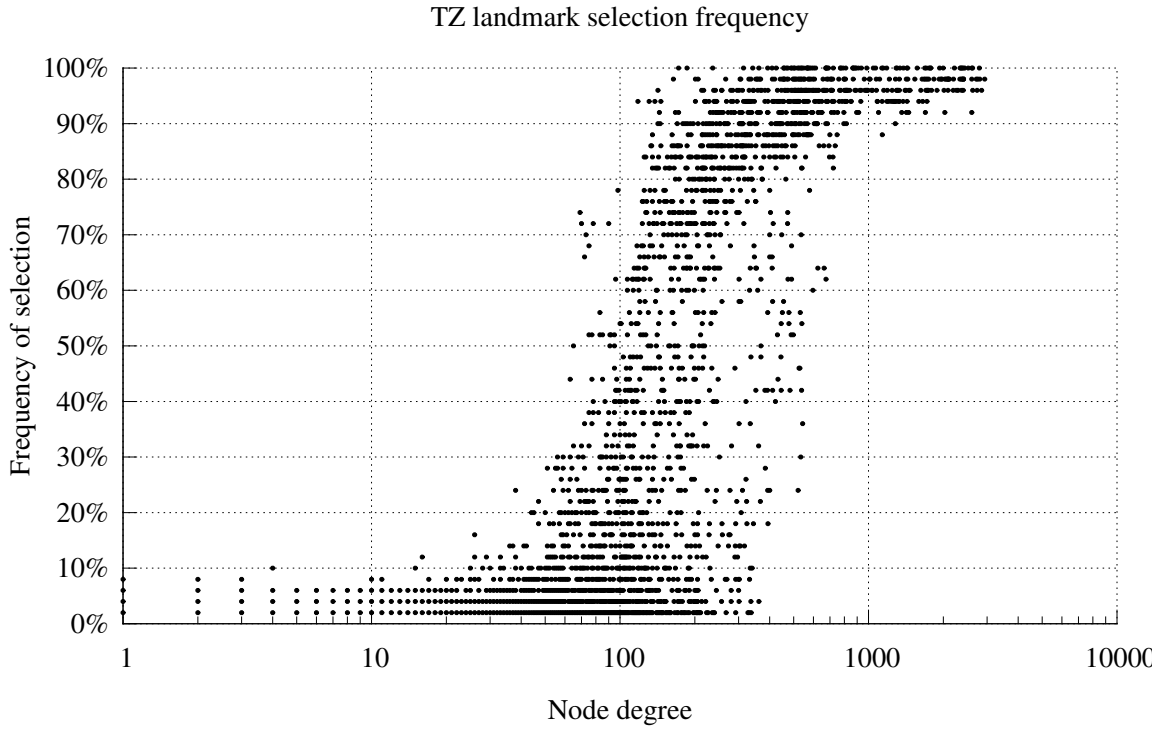
This chapter has presented the first systematic analysis of the TZ and BC compact routing algorithms on the Internet AS graph as it evolved. In summary, these compact routing algorithms perform extremely well on Internet graphs, and have continued to perform well through the massive growth of the network. As highlighted in Section 2.5, there is further scope for evaluating the algorithms with more accurate models of the AS graph, should they become available.

If either algorithm can be decentralised, the potential reduction in forwarding state is massive. Consider that a full BGP table for May 2011 contains 386,771 IP prefixes and that the TZ compact routing algorithm requires that, on average, nodes retain forwarding entries to 0.45% of the rest of the network on the later graphs, then a TZ-based protocol would provide forwarding tables of 1,521 entries on average. This is a simplistic mapping, but the potential savings are profound and warrant investigation.

Neither of these algorithms is immediately deployable as a network protocol. Both algorithms, as stated, rely on a global view of the network. Of the two algorithms analysed in this chapter, the TZ compact routing algorithm seems most easily decentralisable. The TZ landmark selection algorithm grows its landmark set from an initial random selection of nodes, but on the Internet a stable set of centrally located transit networks acting as landmarks would be more suitable. In Chapter 5, I focus on the stability of a subset of the core of the network and its suitability as the landmark set. The definition of a distributed routing protocol that uses this landmark set, is presented and analysed in Chapter 6.

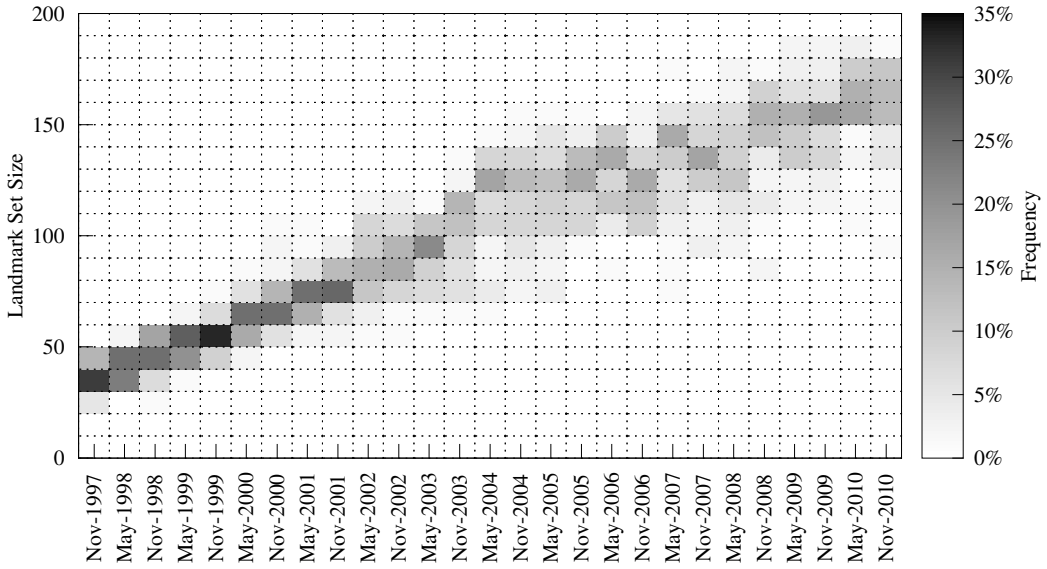


(a) Node selection histogram, showing dominance of ASes selected only once across the fifty landmark sets generated for November 2011.



(b) The selection frequency for each snapshot tested (*i.e.*, how often a landmark appears in each of the fifty landmark sets, overlaid onto the same plot).

Figure 4.7: TZ landmark selection frequency.



(c) Distribution of landmark set sizes.

Figure 4.7: TZ landmark selection frequency.

Policy-based routing, and the ability to make local decisions on link utilisation, or to use preferred links (for cost purposes, etc), is considered an important aspect of network management. The algorithms evaluated in this chapter do not make allowances for policy, though there is still scope for affecting policy on routes within clusters, and routes toward landmarks; issues with policy routing are discussed as future work in Chapter 7. It is not easy to evaluate against the path inflation introduced by policy-based routing in BGP, as such information is generally considered proprietary. It may be possible, however, to begin evaluating this using publicly available network maps annotated with inferred AS relationships [101, 41].

Chapter 5

Stable Landmark Selection

In Chapter 4, I demonstrated that the Thorup-Zwick (TZ) compact routing algorithm has offered consistently low path stretch on Internet AS topologies spanning 15 years. To achieve scalable routing, the TZ compact routing algorithm defines a globally visible set of landmark nodes A, and for each node u a set of forwarding table entries defined by the clustering equation, Equation 4.1 (page 40). The landmarks are used to forward packets if the destination for the packet is unknown. The TZ landmark selection algorithm assumes a static graph for its input, and the pseudo-random landmark selection leads to variation in the path stretch results. In the context of a dynamic network without central control, its flaws are as follows:

- Landmark sets based on a pseudo-random selection will be inherently unstable between executions of the algorithm. Using such an algorithm in a dynamic network implies highly disruptive routing state changes upon the selection of a new landmark set. The landmark selection algorithm offers no graceful evolution of the landmark set over time.
- Decentralised pseudo-random selection is difficult to achieve while retaining the routing bounds defined in Section 4.2.1, and though it would be possible for nodes to independently select themselves as landmarks according to a chosen uniform probability, treating all ASes on the Internet as equals ignores the possibility that some are longer-lived and more reliable than others.

Thus, the pseudo-random landmark selection algorithm for the TZ compact routing algorithm is a problem when considering deployability in dynamic networks. For TZ compact routing to be useful in a dynamic network, it must be capable of maintaining a stable set of landmark nodes in the presence of change and must be computable in a distributed manner, neither of which are goals of the pseudo-random landmark selection algorithm.

Therefore, a better landmark set would be one that is central to the network and is stable. In this chapter, I briefly discuss measures of node centrality before focussing on k-cores

graph decompositions of AS topologies, on the intuitive basis that the nodes most "central" to the AS graph are providers who are also likely the most *stable*, suggesting the feasibility of distributed and incremental landmark set calculation. The k-cores graph decomposition offers a promising alternative mechanism for selecting landmarks. I focus on using the k-cores decomposition of the AS graphs as input to a new landmark selection algorithm for the Thorup-Zwick (TZ) compact routing algorithm.

Stability and centrality are essential properties of a landmark set for a dynamic network, and are necessary precursors for the development of a deployable compact routing protocol. Note that I do not directly address policy concerns for landmark selection, but I outline future directions to address this and other issues in Section 5.5, and later in Chapter 7.

The contributions of this chapter are as follows:

1. I show that the k-cores algorithm on the Internet graph reveals a well-connected *nucleus* corresponding to the k_{max}-shell (see [99] and Section 5.1.1). I present quantitative data on the rate of arrivals to, and departures from, the nucleus over time to show that this set of nodes is remarkably stable. In doing so, I present a comprehensive longitudinal study on the stability of this set, confirming previous results that made similar assertions. I also show that the nucleus is topologically well-placed for use as a landmark set for TZ compact routing.

2. I define the TZ_k compact routing algorithm, a modified version of the TZ compact routing algorithm that uses the k-cores graph decomposition to nominate landmark nodes. Previous work has performed k-core graph decomposition on the AS graph for analysis [99, 102], but no compact routing algorithms have used this form of graph decomposition to intelligently place landmarks.

3. I run the TZ_k landmark selection algorithm on snapshots of Internet AS-level connectivity collected between November 1997 and November 2010 inclusive, and show that using the k_{max}-core of each snapshot as the landmark set satisfies the TZ compact routing constraints for sublinear forwarding table sizes in the overwhelming majority of cases. In the remaining cases the addition of the k_{max-1}-core is sufficient to satisfy the constraints.

4. Using the same AS graph snapshots as in Chapter 4, I demonstrate the performance of the TZ_k routing algorithm, in terms of stretch factors and forwarding table sizes, and show that it offers comparable performance to regular TZ compact routing.

This chapter is structured as follows:

Section 5.1: discusses measures of graph centrality, and introduces the k-cores graph decomposition algorithm.

Section 5.2: evaluates the k-cores graph decomposition applied to the AS graph, in particular the stability of the maximum core, its size relative to the full network, and its connectivity.

Section 5.3: describes the application of k-cores graph decomposition as the TZ landmark selection algorithm.

Section 5.4: evaluates the routing stretch when using these landmarks in comparison with the TZ landmark section evaluated in the previous chapter, and also covers forwarding table sizes.

Section 5.5: discusses the potential for distributing the k-cores graph decomposition.

Section 5.6: summarises the chapter.

5.1 Measures of Centrality

The TZ compact routing algorithm [80] is designed for general graphs, and therefore makes no assumptions about graph structure. It selects its landmarks uniformly randomly from the full graph. In the context of the Internet, however, there is structure in the graph that could be exploited by a compact routing algorithm to make landmark selections that are more suitable for the Internet. Intuitively, a central collection of nodes in a graph will offer a more suitable landmark selection in a dynamic graph. The principle of preferential attachment, discussed briefly in Section 2.3, suggests that long-lived nodes are also likely to be well-connected, and so are located very few AS hops from the rest of the graph.

There are various measures of node centrality for graphs that may offer a reasonable mechanism to compute the most central of all the nodes. Measures of node centrality are discussed in [103], some of which are discussed briefly in this section.

Node degree is commonly used as an important attribute when discussing graph structure. Some compact routing algorithms explicitly use node degree when processing graphs: the BC compact routing algorithm and the compact routing algorithm by Chen *et al.* [100] aim to use this property to improve performance. The Chen *et al.* compact routing algorithm uses node degrees to select landmarks, reducing forwarding state compared to the theoretical bound in general graphs. However, although the node degree distribution of the AS graph is known to follow a power law [13], node degree is an unsuitable measure of node importance: it gives a highly-localised view around each node, and disregards the importance of nodes with relatively low degrees through which many paths pass.

The graph density, discussed earlier in Section 2.3, is related to the node degree. It is trivial to compute when given a full view of the graph. The density of AS topologies is already shown

in Figure 2.2c (page 12). Density is not always a useful measure of how well connected a graph is: it is possible for a highly decentralised graph (with low-degree nodes) to have the same density as a graph which is sparse but for a few dense regions. Further, it is a broad metric that does not differentiate particular nodes.

An alternative measure of centrality is the closeness centrality, which is the mean shortest path length for each node to every other node. However, to perform all-pairs shortest path computations to determine the closeness centrality measure of every node is computationally expensive for large graphs. Further, as shown in Section 2.3, Internet graphs generally have mean path lengths of around 3.8 AS hops and little variation, suggesting the centrality measure for most nodes will be very similar, thus limiting the metric's capability to differentiate nodes.

Another possibility is searching the graph for the largest possible clique, thereby locating the most strongly connected region of transit nodes in a graph. However, locating the largest possible clique requires a full view of the graph, and is a NP-complete problem.

The other more fundamental problem with searching for cliques, or similar structural artifacts, is that analysing networks for cohesive regions such as these does nothing to find the weak links connecting those cohesive regions. A more coherent measure of network structure is one that takes into account both cohesiveness and density. One such algorithm is the k-cores graph decomposition algorithm [104], which is not computationally expensive when given a full view of a graph (linear runtime algorithms exist [105]), and aims to reveals cohesive regions that are themselves not tightly connected.

5.1.1 k-cores Graph Decomposition

The k-cores decomposition of a graph is a mechanism that produces a sequence of subgraphs of gradually increasing cohesion. The number k in this context is an indicator of a node's centrality in the network: the removal of a node with a higher k value affects more of the network than a node with a lower k value.

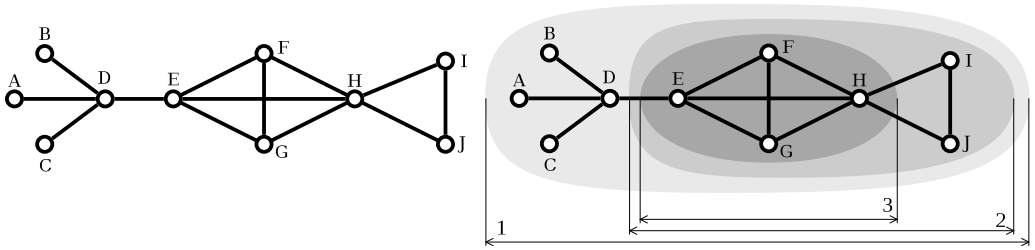
Formally, a subgraph $G' = (V', E|V')$ induced by the set $V' \subseteq V$ is a k-core if and only if $\forall v \in V' : \deg_{G'}(v) \geq k$ and G' is the maximum subgraph with this property [106].

A node v has *shell index* of k if it belongs to the k-shell, but not the $(k+1)$-shell [102]. k-cores graph decomposition defines the following sets:

A k-shell is composed of all nodes whose shell index is k.

The k-core of a graph is the union of all shells with index $\geq k$.

The k-crust of a graph is the union of all shells with index $\leq k$.



(a) A fully connected graph; node H has the highest degree, then D and E.

(b) The k-cores decomposition, with shells marked. Node D has been isolated in the 1-shell as part of a tree-like structure, and the cluster E, F, G, H is identified as the 3-shell.

Figure 5.1: Example of k-cores decomposition on a small unweighted, undirected network.

All k-shells of a graph can be obtained easily by recursively removing all the nodes of degree less than k until all nodes in the remaining graph have degree at least k, incrementing k when no additional nodes qualify for that k-shell. The largest k which generates a non-empty k-shell is k_{max}, and I refer to the k_{max}-shell as the *nucleus*. The k_{max}-shell and the k_{max}-core are equivalent.

An intuitive example is shown in Figure 5.1. An example graph is shown unannotated in Figure 5.1a, and an annotated version showing the k-core decomposition of the same graph is shown in Figure 5.1b. The decomposition clearly reveals something of the structure of the graph. Node D, for example, is a high-degree node with 4 connections, but it belongs in the 1-core, while node E, directly adjacent, exists in the 3-core, despite both nodes having the same degree. The algorithm makes a distinction between those regions of the graph that are poorly connected (for example, the $\{ABCD\}$ cluster on the left hand side), and those regions that are more strongly connected. In this context, E is more useful to the graph in the event of the failure of any link in the $\{EFGH\}$ cluster, while there is no redundancy in the $\{ABCD\}$ cluster.

This k-cores decomposition exposes structure in the network not obvious from node degree alone, and identifies the k_{max}-shell, the *nucleus*, as a highly-connected component at the heart of the network. That is, using the k-core graph decomposition, the chosen nodes are structurally important. In the following section, I apply the k-cores graph decomposition algorithm to Internet AS topologies and analyse the output.

5.2 k-Cores Decomposition on the AS Graph

Using the k-cores decomposition as a centrality measure on the AS graph allows the evaluation of the k_{max}-core selection against additional metrics for real-world suitability. Previous work has performed k-cores graph decomposition on a single snapshot of the Internet AS topology, and indicated that the nucleus is a hub containing well-connected ASes [99].

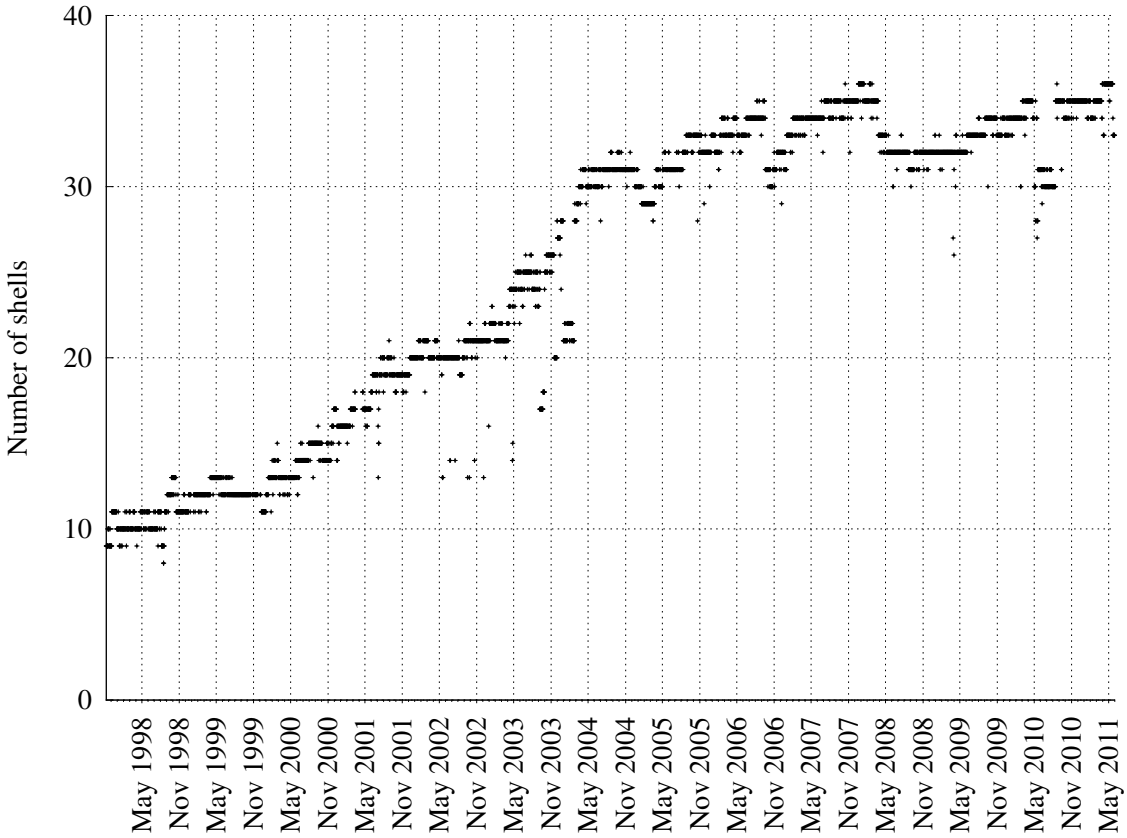


Figure 5.2: Total number of k-shells in each AS snapshot.

In this section, I use the daily snapshots of the AS topology in the dataset described in Section 2.1. This covers 8 November 1997 to 31 May 2011 inclusive. For each of these topologies I generated its k-cores decomposition. I analyse the growth of the nucleus in Section 5.2.1, and the stability between sets in Section 5.2.2. I discuss centrality in terms of distance to the rest of the network in Section 5.2.3, and also the geographical distribution of the set in Section 5.2.4.

5.2.1 Nucleus Growth

The Internet AS graph has grown from 3,030 ASes on 8 November 1997 to 38,250 ASes on 31 May 2011. As the Internet has grown, the number of k-shells generated by the k-cores graph decomposition algorithm has also grown from around 10 shells in 1997 to around 35 in 2011, as shown in Figure 5.2. The maximum observed across all snapshots is 36. The growth in the number of shells is small compared to the growth in the size of the network, but suggests that the nucleus will also have changed in this time.

In this section, I look at the growth of the nucleus over the same period. Frequent changes in the size of the nucleus would suggest the network is not stable enough as a measure of

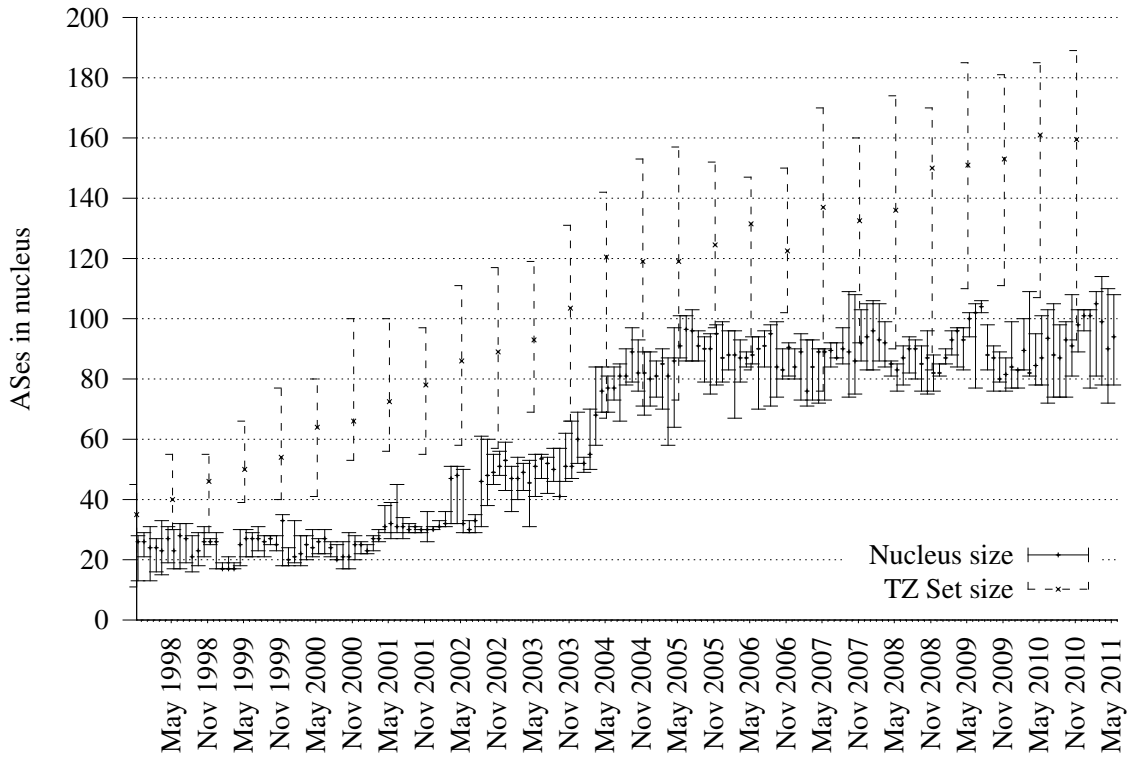


Figure 5.3: TZ landmark set growth and nucleus growth for snapshots of the AS graph, showing the 5^{th} percentile, median, and 95^{th} percentile for each.

centrality to form a useful landmark selection for compact routing, while a nucleus that varies in size slowly suggests there may be stability within the network structure which can be exploited.

Figure 5.3 compares representative landmark set sizes generated by the pseudo-random TZ landmark selection algorithm with the size of the nuclei for all snapshots. According to Equation 4.3, the maximum landmark set size that the TZ landmark selection algorithm will generate ranges from 205 on 8 November 1997 to 838 on 31 May 2011. The clustering constraint, Equation 4.2, combined with the parameter s, generally gives a landmark set smaller than the maximum. Figure 5.3 shows the range of landmark set sizes for multiple TZ landmark sets generated for a variety of dates. The figure shows the median value (with 5^{th} and 95^{th} percentiles). For each date shown, the range represents the fifty landmark sets generated in Section 4.3.3. It is clear that valid landmark sets generated by the TZ landmark selection algorithm vary considerably in size.

In Figure 5.3, I also show the growth of the nuclei generated by the k-cores decomposition. The figure aggregates the range of nucleus set sizes for each month and shows the median value for each month (with 5^{th} and 95^{th} percentiles).

The growth rate of the nucleus in Figure 5.3 is not linear as one may expect from the growth of the network [107]. The flattening of the growth curve is likely an artifact of sampling:

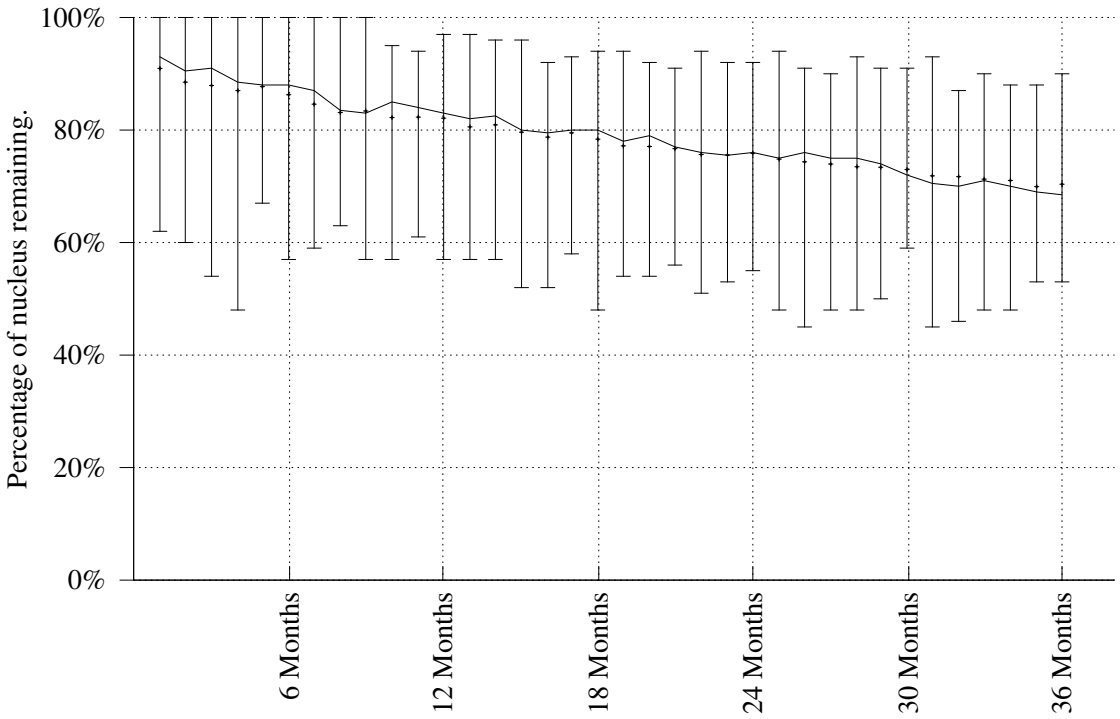


Figure 5.4: Proportion of nucleus remaining after n months. For a selection of evenly spaced dates d_i this shows the proportion of the nucleus set A_{d_i} which remains in the TZ_k nucleus set in each of the snapshots in subsequent months. Error bars are minima and maxima; points are medians; the line tracks the mean.

Route Views collectors were added after the initial collector in 1997 until 2005, with few additions since; see Appendix A.1. Graphs derived from Traceroute data [35] claim to represent a fuller picture of the topology, in particular edge-peering. While AS topologies generated from traceroute data can suffer from inaccuracies [108], the nucleus set generated from the data released with [35] contains 153 nodes, suggesting that with a full view of the network, the nuclei will be larger. I show in Section 5.4.1 that despite the flattening of the curve, these sets are sufficient as landmark sets in the vast majority of cases and therefore any larger nuclei will also be sufficient.

This data shows that these nuclei are smaller than the landmark sets generated by the standard TZ landmark selection algorithm. The size of these nuclei are reasonably constant. What is also important is how consistent the sets are, and how often nodes are returned to the set of nuclei nodes on repeated executions of the algorithm. I show in Section 5.2.2, that these nuclei are very stable in terms of their membership, and I show in Section 5.4.2 that routing stretch performance is not affected by using these smaller sets. As the algorithm uses the structure of the network and eliminates the random element of TZ, these landmark sets are deterministic. Thus, in an evolving network, these landmark set sizes are more predictable, and vary with respect to the network itself.

5.2.2 Nucleus Stability

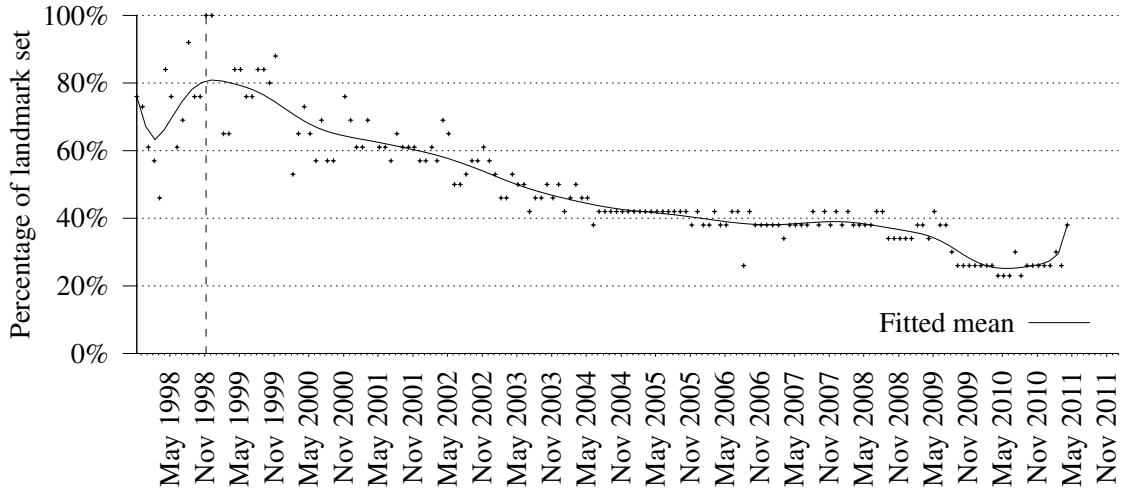
Given the evolving nature of the network, the membership of the k_{max}-core will also evolve over time. A landmark set that is stable would help reduce the computational overhead involved in recomputing and readvertising forwarding tables. I measure stability in terms of additions to, and departures from, the landmark set over time.

First, I measure departures from the nucleus by calculating the fraction of the set A_{t_1} from time t_1 that is still present in the landmark set A_{t_2} at time t_2. Figure 5.4 plots this at monthly intervals up to 3 years from a range of initial dates (one snapshot every three months, commencing 8 November 1997 through 8 November 2010). The error bars show the maxima and minima on each data point. I choose 3 years from each initial date as a long-term measure in terms of Internet evolution. The departure rate is clearly linear, with around 70% (mean *and* median) of a nucleus remaining after the 3 years. Given the rate of growth of the network, one might reasonably expect greater instability; to observe such stability is a key result. The lower bound on the error bars in Figure 5.4 is introduced primarily by the networks earlier in the dataset with smaller nucleus sets, and the upper bounds on the error bars are introduced by more recent networks.

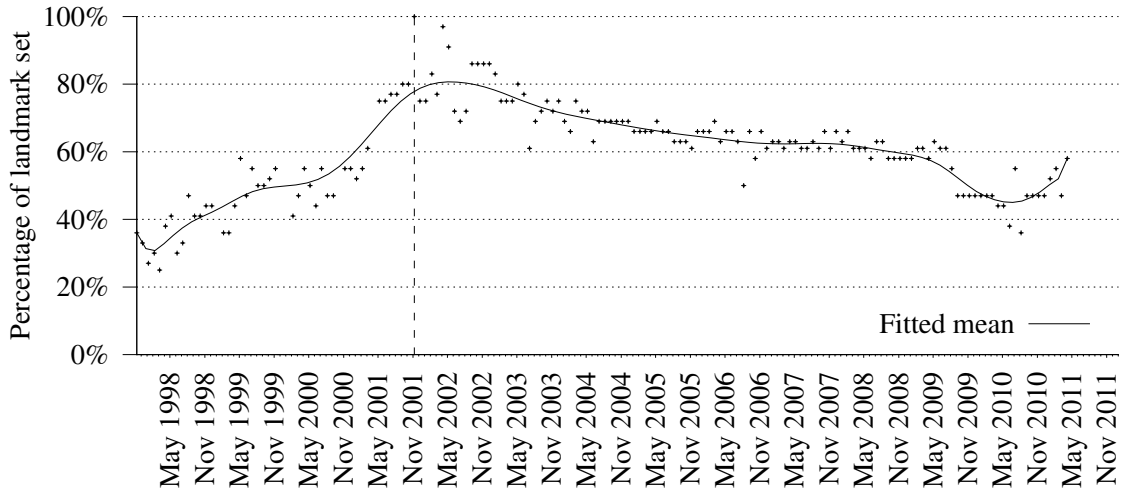
Figures 5.5a, 5.5b and 5.5c show a similar pattern. Here, I take the nucleus for three snapshots spaced three years apart (representing snapshots taken for 8 November 1998, 2001, and 2004), and plot the proportion of that nucleus which existed in the prior and subsequent months. Not only is there a slow degradation of the membership of the nucleus from each date selected, but also the nodes in the nucleus in Figure 5.5a depart faster than those shown in the later nuclei shown in Figures 5.5b and 5.5c. As the nucleus set grows in size, so too does the proportion of the set that is highly stable: a smaller proportion of that set will cease to exist in future months.

In Figure 5.6a, I measure the similarity between nuclei sets taken across different time spans. I take the nuclei sets generated on snapshots during each time span, and plot the Jaccard index of neighbouring aggregate sets. The Jaccard index provides a measure of similarity between two sets, in the range 0 to 1 inclusive where 0 indicates disjoint sets and 1 indicates identical sets. To demonstrate stability, the Jaccard index between sets should be as close to 1 as possible. Figure 5.6a demonstrates that given the union of all nuclei generated over the course of a year, there is a strong overlap with the union of all nuclei generated in the following year. Since 2004, the similarity between adjacent year-long sets has been effectively constant around 0.8. The month-by-month similarity measures appear slightly more variable, but are consistently similar to each other with an index around 0.9 indicating little change. The data suggests a very stable set of nodes that always feature in the nucleus, and a smaller subset of nodes that move between the k_{max} and the k_{max-1} shells.

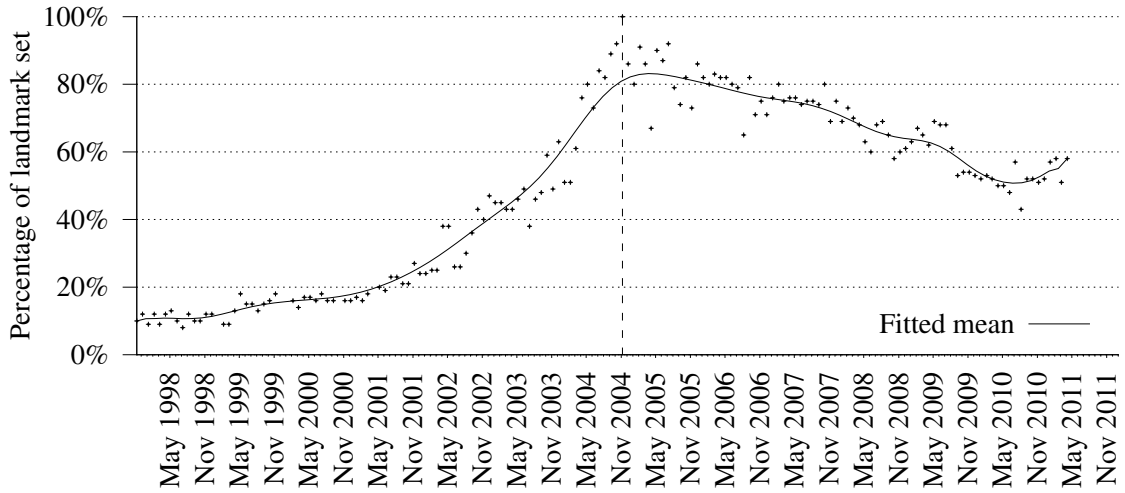
Note, however, that the Jaccard index is heavily influenced by the larger of the two sets being



(a) Growth/decay for 8 November 1998.

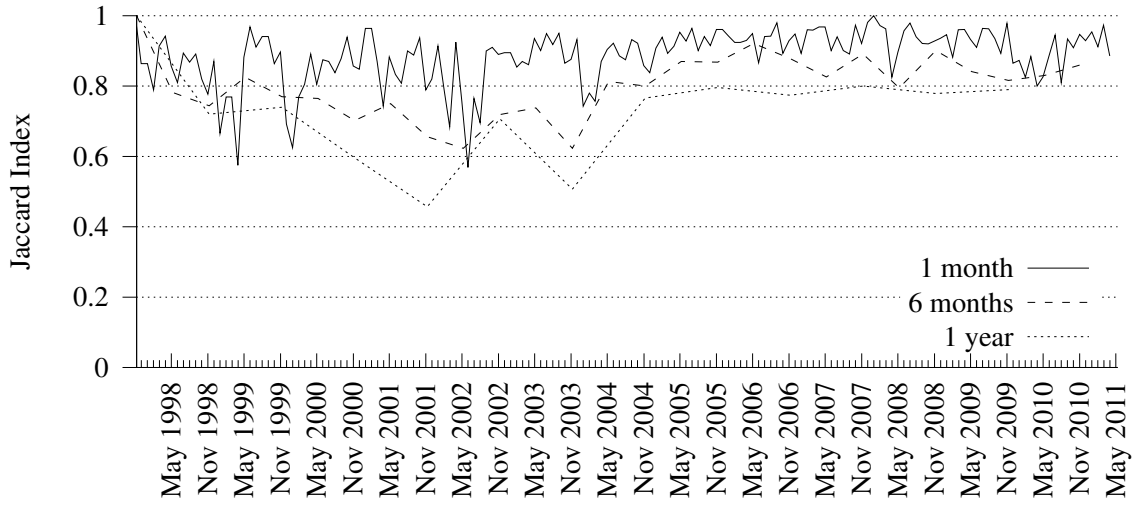


(b) Growth/decay for 8 November 2001.

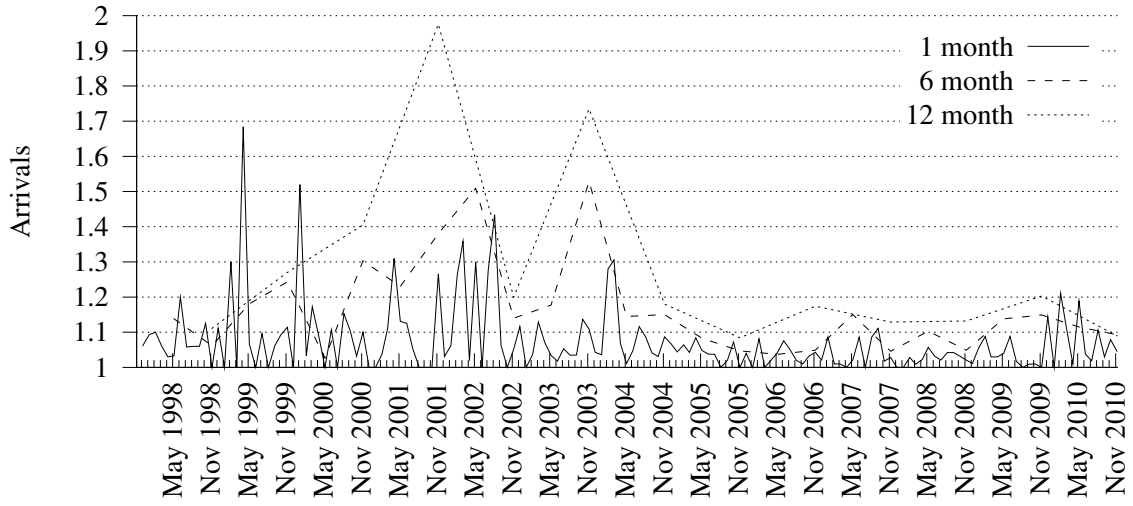


(c) Growth/decay for 8 November 2004.

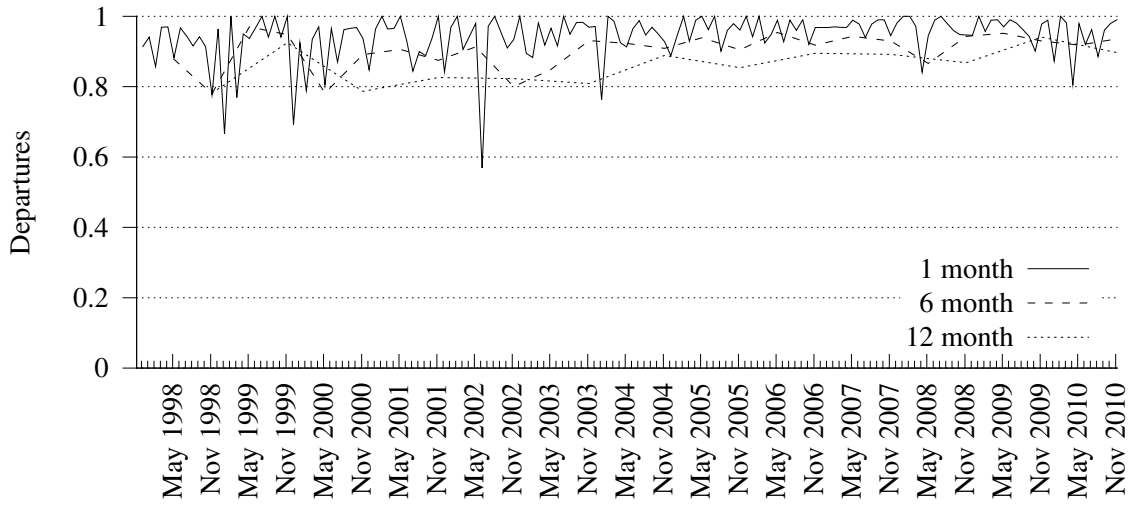
Figure 5.5: Nuclei generated on three different dates demonstrate a rapid arrival rate into the nucleus followed by a slower departure rate.



(a) Jaccard index on adjacent 1, 6, and 12 month spans. Each span is the union of all nucleus sets generated by the k-cores graph decomposition algorithm on intermediate dates.



(b) Landmark set arrival rates, not including departures from the set.



(c) Landmark set departure rates, not including new arrivals to the set.

Figure 5.6: Jaccard Indices

compared, and that an index less than 1 may indicate growth without removal of any nodes. It is a similarity measure between two sets with no concept of causal ordering. To measure arrivals and departures, I define two new measures. The purpose of these measures is to encapsulate how much change can be attributed to node arrivals in the nuclei, and how much can be attributed to departures. Given two sets A_{t_1} and A_{t_2}, I define arrivals as the number of nodes present in A_{t_2} over the number of nodes present in the intersection of the two sets (thus, I do not count nodes that have left A_{t_1}). This gives a value greater than or equal to 1, relative to the size of A_{t_1}. Formally:

$$arrivals = \frac{|A_{t_2}|}{|A_{t_1} \cap A_{t_2}|} \tag{5.1}$$

I define *departures* similarly, as a measure of the intersection of the two sets divided by the size of A_{t_1}:

$$departures = \frac{|A_{t_1} \cap A_{t_2}|}{|A_{t_1}|} \tag{5.2}$$

Figures 5.6b and 5.6c show arrival and departure rates respectively according to these definitions. These demonstrate that much of the change in the nucleus set comes from the arrival of new nodes, and not the replacement of old nodes. Note that the lowest similarity values shown in Figure 5.6a on the "1 year" trace correspond with the peaks on the same trace in Figure 5.6b, indicating that the dissimilarities comes from growth in the set, and not departures. The departure rate observed in Figure 5.6c is low and appears essentially constant.

These results show a long-term stability at the centre of the network. Over the course of the entire dataset more than 45,000 unique AS numbers have been observed, but only 261 nodes have appeared in the nucleus. Figure 5.7 shows when each of the 261 ASes that have appeared in the nucleus over this dataset were present. Of these, 224 were still in the BGP data, even if not in the nucleus, on 31 May 2011. This figure demonstrates at a high-level the stability of the set, with a tendency toward older ASes (*i.e.*, those with lower AS numbers). Note that this set shows a considerably higher rate of returning nodes than the pseudo-random TZ landmark selection algorithm, demonstrated in Figure 4.7, where there is little continuity between landmark sets generated even on the same graph.

5.2.3 Nucleus Membership & Connectivity

Textual descriptions of ASes are publicly available in the WHOIS system, including the name of the registrant, the geographic region (specifically, the two-character ISO 3166 country codes), contact details, etc. Grouping these manually according to the textual description of the AS, the nucleus consists of a variety of different networks, ranging from international

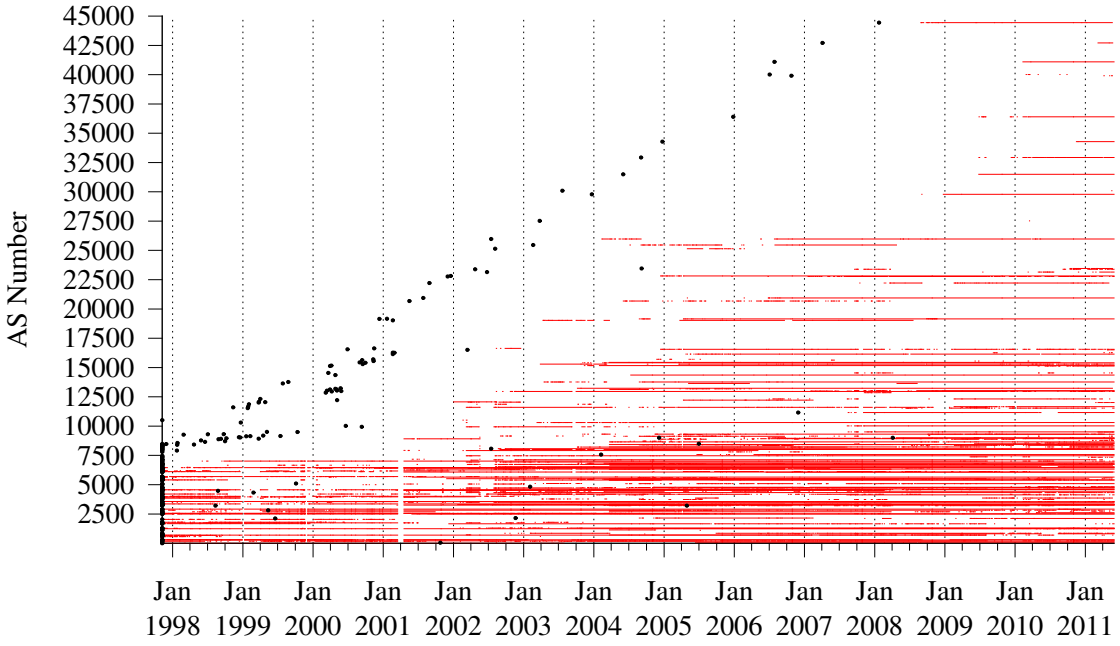


Figure 5.7: Dates on which ASes appear in k_{max}-core. Circular points indicate when an AS was first introduced to the network, and horizontal bars indicate dates the AS appears in the nucleus set.

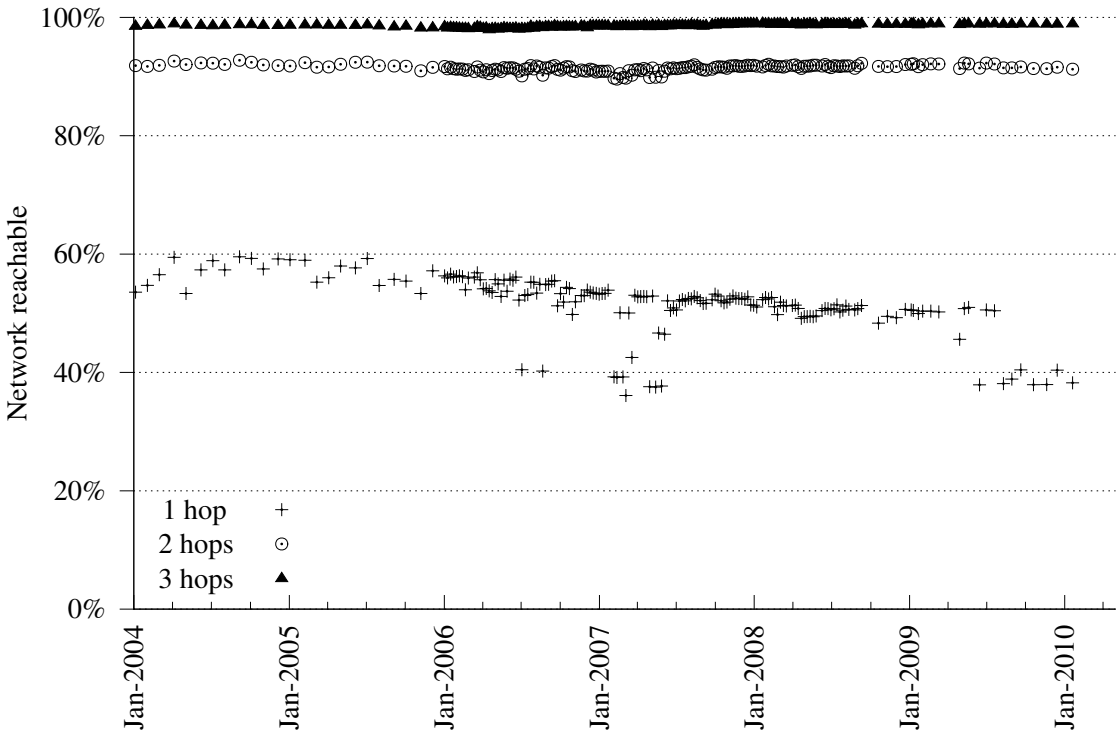


Figure 5.8: Proportion of network reachable within n AS hops from the nucleus.

transit networks spanning continents (e.g., Level3, Cogent), national networks (e.g., ChinaNet, Deutsche Telekom, Swisscom), regional networks (e.g., NORDUnet), and data/cloud providers (e.g., Akamai, Google, Microsoft).

Inferred AS relationship data [101] is available between January 2004 and January 2010. This data gives an indication of the link types between ASes. How these types are calculated was covered in Section 2.5.1, page 21. Using this data to determine the transit paths from the nucleus, Figure 5.8 shows the proportion of the network reachable within 1, 2 and 3 AS hops from the nucleus. Figure 5.8 shows that although there has been a slight decrease in the proportion of the network reachable within one AS hop, most of the network is reachable in 2 – 3 AS hops. The reduction in the proportion of the network reachable within one AS hop is small, given the rate of growth of the full graph (as shown in Figure 2.2a on page 12). When compared to the size of the nucleus shown in Figure 5.3, approximately 50% of the network is directly connected to this relatively small set of nodes. This demonstrates that the nucleus is very well placed to cover a large proportion of the network.

The proximity of the nucleus to the rest of the network is also important. Calculating the shortest AS-path distance from the nucleus set to all other nodes, the average path length lies in the range 2.5 – 2.7, with the 99^{th} percentile distance being 4 or 5 in all cases. These values place the nucleus at the heart of the network: the mean distance between *any* two pairs of nodes in the AS graph has been constant over time, lying in the range 3.5 – 4.0; the maximal diameter has risen from 9 AS hops to 11. On this basis, the nucleus appears to be ideally located as the network's natural "core".

5.2.4 Geographical Distribution of the Nuclei Sets

In a network so widely distributed as the Internet, it is important that the landmark set be geographically widely distributed such that latency introduced by any additional AS hops is small. It is possible to infer some information on the geographical placement of TZ_k landmark nodes on the AS graph through the WHOIS system. Although more accurate indicators of the geographic location of each AS are difficult to obtain (and not meaningful for large inter-continental networks), it is possible to determine the regional Internet registry that issued the AS number, and the country in which the AS is registered.

Using this as a basis to infer the geographic regions served by these ASes, Table 5.1 shows the geographical distribution of the nucleus for the snapshot on 8 November 2010. This nucleus contains 100 ASes in 27 different countries: a geographically diverse set covering Europe, the US, the Asia-Pacific region, and Africa. Notably, one RIR is not accounted for: LACNIC is not represented in this set, though this would change as demand improves network deployment. Deriving the nucleus of the AS graph reveals a topologically well-placed set of landmark nodes. This data supports that assertion, on the reasonable assumption

RIR	Country	No.	RIR	Country	No.
afrinic	ZA	1	ripencc	DE	3
apnic	HK	3	ripencc	CH	2
apnic	JP	2	ripencc	SE	2
apnic	KR	2	ripencc	AE	1
apnic	AU	1	ripencc	AT	1
apnic	CN	1	ripencc	ES	1
apnic	MY	1	ripencc	FI	1
apnic	SG	1	ripencc	FR	1
apnic	TW	1	ripencc	IT	1
arin	US	38	ripencc	NL	1
arin	CA	7	ripencc	NO	1
ripencc	EU	12	ripencc	PT	1
ripencc	GB	9	ripencc	UA	1
ripencc	RU	4			

Table 5.1: Geographical distribution of nucleus on 8 November 2010.

that network deployment correlates with demand. Thus, the best-connected regions of the network exist in areas with the highest Internet usage. The nucleus set defined in this chapter naturally uses this property, and will evolve as the network evolves.

5.2.5 Summary

In summary, the k-cores graph decomposition of AS topologies, and in particular the k_{max}-core, has several desirable properties that can be harnessed for use as a landmark set for TZ compact routing. The set of nodes are highly stable, and relatively small compared to the rest of the network. Very few nodes have entered this set throughout the 15 years of data analysed, but the set is geographically diverse, and the vast majority of the network can be reached within two AS hops suggesting the set is highly centrally located.

These properties are highly beneficial for a landmark set for a dynamic network. In the following sections, I cover the use of the k-cores graph decomposition algorithm as a landmark selection algorithm, followed by analysis of path stretches and table sizes when using these new landmark sets.

5.3 TZ Compact Routing with k-cores Landmarks

In this section, I define a modified TZ compact routing algorithm that uses the k-cores graph decomposition algorithm as a replacement landmark selection algorithm. I refer to the result as TZ$_k$.

Algorithm 1 : **landmark**(G, s)

Generate k-shells $k = 1, 2, ..., max - 1, max$
$A \leftarrow \emptyset$
$W = \emptyset$
$i \leftarrow max$
do
  $A \leftarrow A \cup i$-shell
  $i \leftarrow i - 1$
  $C_v = \{w \in V \big| d(v, w) < D(w, A)$, for every $v \in V\}$
  $W = \{v \in V \big| |C_v| > 4n/s\}$
while $W \neq \emptyset$
return A

The motivation for using k-cores decomposition of the Internet graph as the basis for its landmark set is that this decomposition exposes a highly connected, highly visible region of the network, yet is cheap to compute (linear runtime [105]) and is stable over time.

The nucleus region is structurally important: on the Internet topologies analysed in [99], 30% of nodes are reachable *only* via the nucleus, and distances between the other 70% are shortened considerably by routing through the nucleus. Given that so many paths already require the nucleus, using this set as a landmark set will incur no stretch at all for many paths, and is expected to have low stretch for the others.

The TZ$_k$ landmark selection algorithm is listed in Algorithm 1. Intuitively, given a static graph and setting the parameter s as defined in Section 4.2.1, the TZ$_k$ compact routing algorithm determines the k-cores for the graph. Starting from the k_{max}-core (the nucleus), the algorithm tests the clustering constraints as per Equation 4.2 on page 41. If the test fails, the algorithm iterates and expands the landmark set to include the $(k_{max} - 1)$-shell, and so on. Note that on the graphs studied in this dissertation, adhering to Equations 4.1 and 4.2 ensures that the TZ$_k$ algorithm will not violate the upper bounds defined in Equation 4.1. Although the algorithm cannot guarantee Equation 4.3 and will fail on graphs with uniform or near-uniform node degree (grids, rings, etc), it does not violate this constraint on any of the Internet AS graph snapshots across the entire collection. This algorithm generates forwarding table sizes of $O(\sqrt{n \log n})$.

A landmark selection algorithm such as this which takes into account the structure of the graph can select highly important and topologically well placed ASes to be landmarks. Section 5.4 shows that TZ$_k$ provides *at least* the same performance as TZ, but with the benefit that it will provide a highly-stable, geographically distributed set of ASes that are likely to be already performing transit functions to act as landmarks.

5.4 Performance Evaluation of TZ$_k$ on the AS Graph

Having determined in Section 5.2 that the k-cores decomposition offers a stable set of nodes on AS topologies that is central to the network, I defined a landmark selection algorithm, TZ$_k$, in Section 5.3 that uses the output of the k-cores decomposition. In this section, I evaluate the performance of TZ$_k$ compact routing using these landmark sets.

Using the dataset described in Section 2.1 (in particular, one AS topology snapshot every six months across the full dataset), I use Algorithm 1 to generate the landmark sets for each snapshot. Using these landmark sets, forwarding tables are constructed as defined in Section 4.2.1, with the addition of forwarding entries toward direct neighbour nodes as introduced in [81] and also used in Section 4.3.

With this information, I present results on the constraint-checking behaviour of the TZ$_k$ landmark selection algorithm in Section 5.4.1. Then, in Section 5.4.2, I evaluate the stretch performance of TZ$_k$ against the performance of TZ on the same graphs, and cover forwarding table sizes in Section 5.4.3.

5.4.1 Constraint Checking

To check the suitability of the k-cores decomposition and the TZ$_k$ algorithm for landmark selection, the TZ$_k$ landmark selection algorithm was applied to all AS snapshots in the dataset for this dissertation.

The algorithm determines the graph's nucleus to be a sufficient and valid landmark set in 98.18% of the 4,662 snapshots in the dataset (*i.e.*, on the first iteration, the algorithm does not produce any clusters that violate the clustering constraint in Equation 4.2). Only 85 (1.82%) snapshots have nuclei that lead to one or more clusters large enough for the algorithm to enter its second iteration. Generally in these cases, only one node had a cluster that broke the size constraint. In no case is a third iteration required. Snapshots requiring a second iteration were often sequential days (primarily in 2004, 2006, and 2007), indicative of the network's natural evolution rather than fundamental change in the network structure.

It is important to understand that breaking the clustering constraint does not negatively impact on path stretch; it only implies that some nodes will store routing tables that are larger than the upper bounds defined by the TZ compact routing algorithm. In the case of nuclei that lead to clusters that are too large, routing will still function correctly, the upper bound of multiplicative path stretch-3 will be maintained, and average case path lengths may be reduced. This dissertation does not evaluate path stretches in these outliers.

It is clear that the structure of the network, despite massive growth, remains extremely well-suited to the landmark sets generated by the TZ$_k$ landmark selection algorithm. As discussed

in Section 5.2, the k_{max}-core revealed by the k-cores graph decomposition is highly stable, centrally located and well-connected. By also satisfying the TZ clustering constraints in almost all cases tested, it is ideally suited to function as a landmark set for TZ compact routing.

5.4.2 Path Stretches

To analyse the stretch performance, the TZ packet forwarding algorithm must be run across the graph from sources to destinations. To sufficiently test the stretch behaviour of the algorithm, packets were generated from each node using the same test sets defined in Section 4.3.

Figure 5.9 shows the multiplicative path stretch results for both the TZ algorithm and the TZ$_k$ algorithm, and Figure 5.10 shows the additive path stretch results. Note that the TZ results plots in these figures were already shown in Figures 4.4b (on page 47) and 4.5b (on page 48); recall that a subset of five landmark sets is shown from the fifty generated for each snapshot for the TZ compact routing algorithm, such that those five span the range of landmark set sizes.

Figures 5.9a and 5.9b illustrate the multiplicative stretches observed when using the TZ pseudo-random landmark selection and the TZ$_k$ landmark selection respectively. The results are binned such that stretch values higher than 1 are visible. Calculating the mean multiplicative stretches for TZ$_k$ across these snapshots, 77.4% of all TZ$_k$ paths tested are stretch 1, while 90.0% of all paths tested are less than stretch 1.3; only 0.10% of paths tested are stretch 2.0 or more. The maximum observed value for range $[2.0 : 3.0]$ on all of the graphs tested is 0.27% of all paths, in November 2009. Only a small proportion of paths are stretch-3. For example, on the 8 November 1997 graph where I test the forwarding algorithm for all pairs (resulting in 9,180,900 pairs), 20 paths are stretch-3.

The TZ$_k$ stretch results are comparable to the TZ stretch results, which in turn are in line with previous work [82, 109]. Growth of the graph appears to have introduced a gradual reduction in the number of paths experiencing any stretch, but the results indicate that TZ$_k$ continues to perform extremely well on these networks, with the majority of paths experiencing no stretch at all.

While most compact routing research focusses on multiplicative stretch, shortest paths are not of uniform length and so multiplicative results can be misleading. Additive stretch tells us precisely how many additional AS hops were used. I present additive stretch results in Figures 5.10a and 5.10b for the TZ pseudo-random landmark selection and the TZ$_k$ landmark selection respectively. As above, 77.4% of TZ$_k$ paths are shortest paths, but 21.3% of paths are stretched by only 1 additional AS hop. Only 1.3% of paths experience more than one additional AS hop, with the proportion diminishing rapidly as the number of additional AS

hops increases. No paths on the graphs tested are additive stretch 5. Once again, these TZ_k results are comparable to the TZ results.

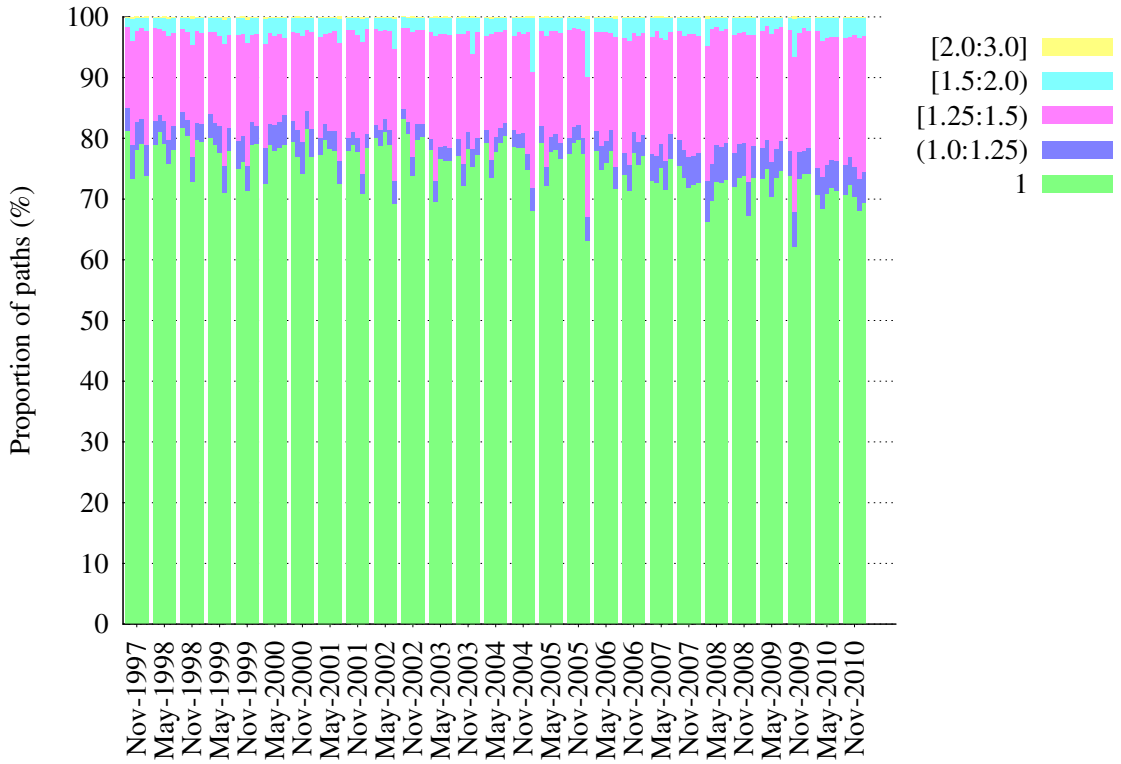
As clusters can overlap and are not symmetric (given two nodes a and b, it is possible for C_a to contain b, while C_b does not contain a), it is also possible for path lengths to differ and thus for path stretch to be non-symmetric. When using the TZ_k landmark sets, 64.66% of paths are symmetric, with 86% of those being stretch 1 in both directions. Of the 35.34% asymmetric paths, 99.79% were stretch 1 in one direction. Out of all the paths observed, 82.46% showed stretch less than 1.3 in both directions, and no paths were stretch 3 in both directions

It is clear that stretch performance with these landmark sets is consistently good. As shown in this section, long stretches are outliers, with paths predominantly shortest paths or near-shortest-path. Although AS graphs derived from BGP data can be incomplete [26, 27], it is reasonable to assume that these stretch results are worst-case, in that additional usable paths will lead to improved performance. A real operational network might additionally insert routes for popular destinations to ensure low-stretch paths, although this would increase forwarding table sizes.

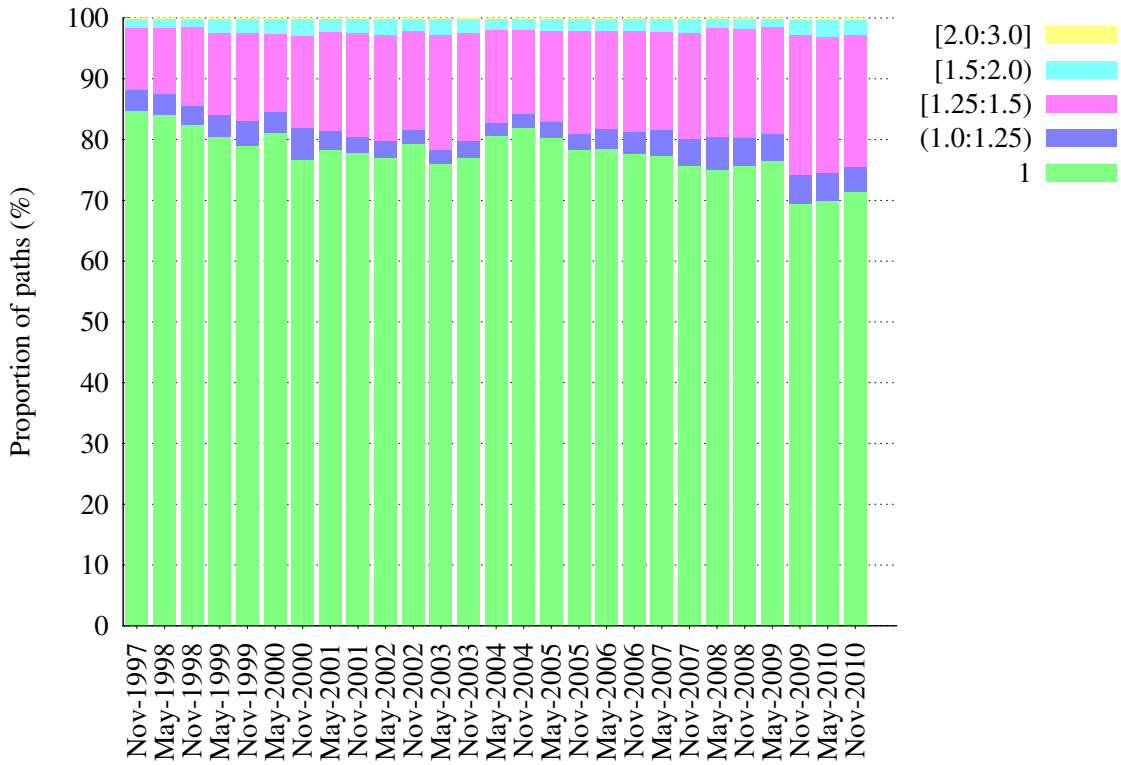
5.4.3 Forwarding Table Sizes

Figure 5.11a shows the distribution of forwarding table sizes observed across all snapshots of the AS graph tested with the TZ_k compact routing algorithm. Even though direct neighbours are inserted into forwarding tables after the landmark selection algorithm has completed (as was discussed in Section 4.3.1), the forwarding tables are still extremely small, generally around a few tens of entries. By adding all direct neighbours, the routing table size for a node directly adjacent to a landmark is maximally the size of the cluster defined by the TZ clustering equation, or the node degree. Thus, the largest of the routing tables are about two orders of magnitude larger than when using TZ compact routing without explicitly adding direct neighbours. Even tables of this size are small when compared with storing references for all ASes, as in shortest path algorithms.

Figure 5.11b shows the distribution of TZ_k table sizes on one AS graph snapshot. Only 5 nodes maintain forwarding entries for more than 5% of the network, and no node holds a forwarding table referencing more than 9% of the network. Other snapshots of the AS graph show very similar forwarding table size distributions. 99.8% of nodes retain forwarding entries for less than 1% of the full network. Thus, forwarding tables for TZ_k have the desirable property that they are consistently small.

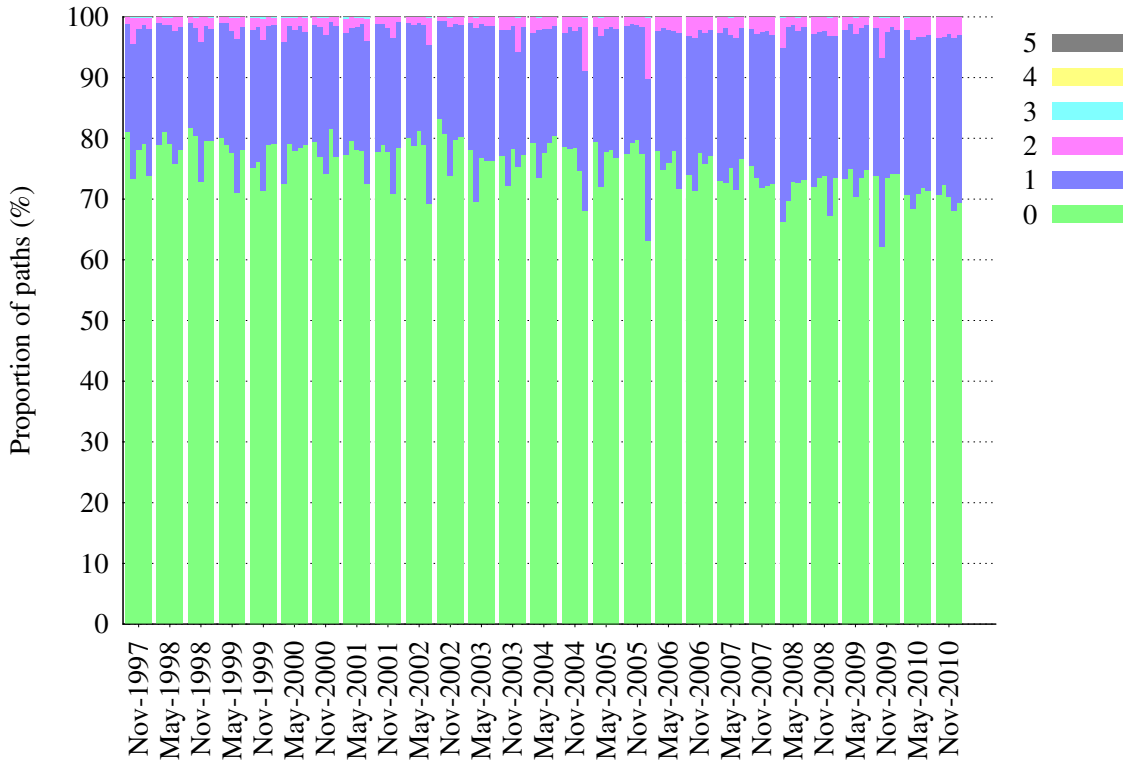


(a) Multiplicative stretches for TZ.

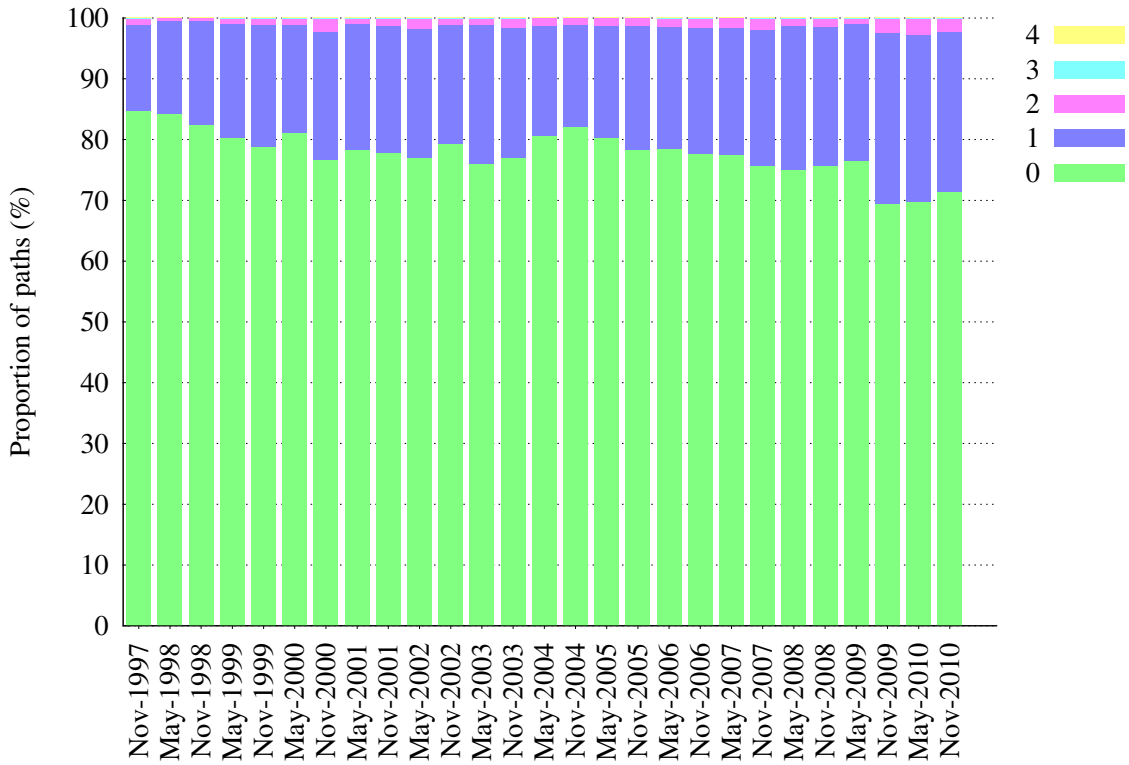


(b) Multiplicative stretches for TZ_k.

Figure 5.9: Multiplicative path stretch for the TZ and TZ_k algorithms.

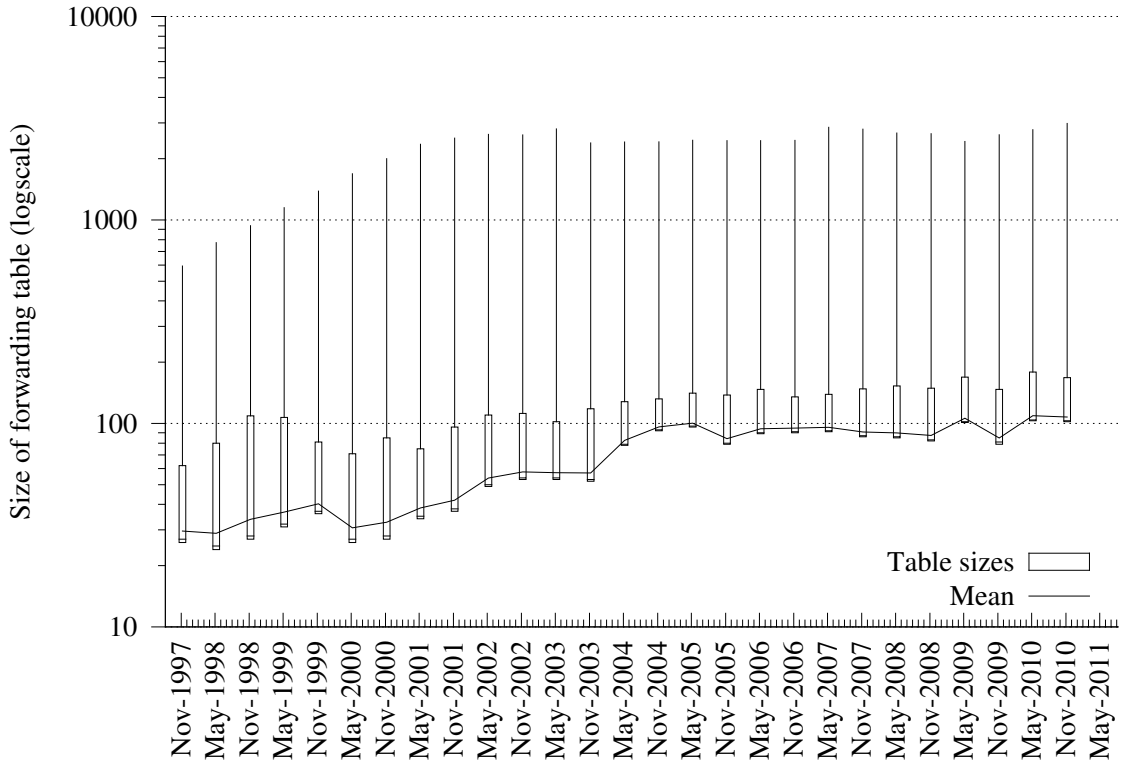


(a) Additive stretches for TZ.

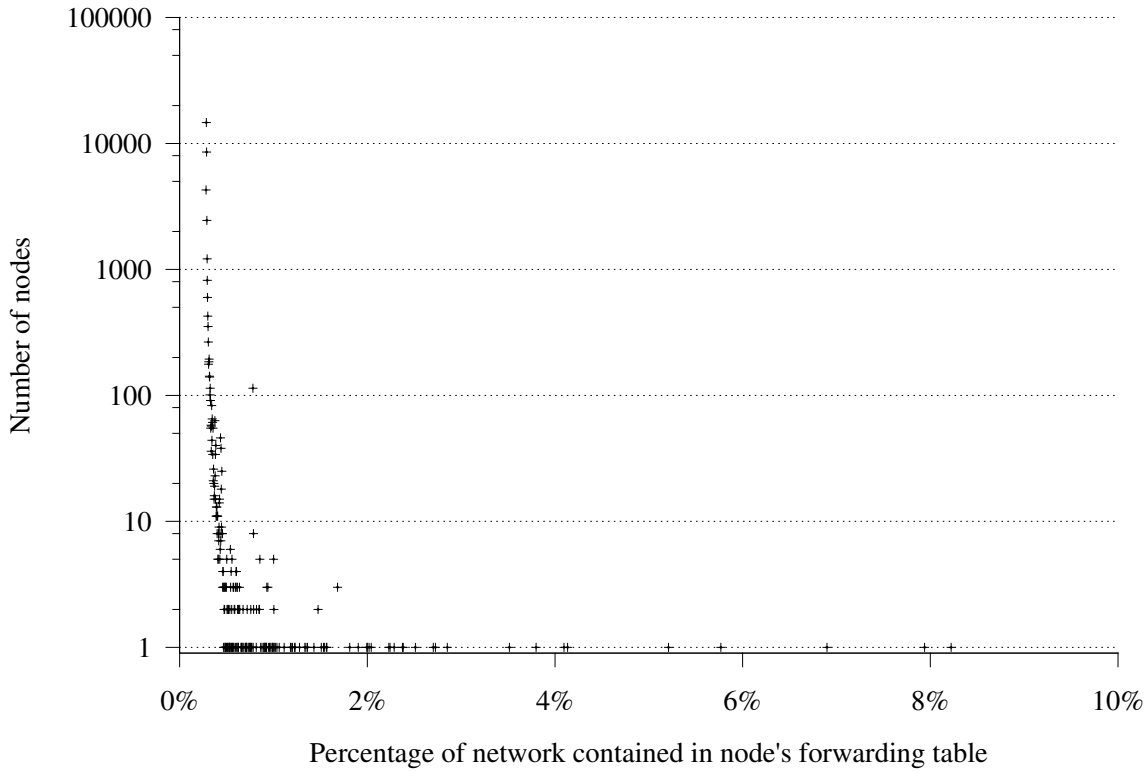


(b) Additive stretches for TZ_k.

Figure 5.10: Additive path stretch for the TZ and TZ_k algorithms.



(a) TZ_k forwarding table sizes across many snapshots. These tables are generally very small. Boxes show 1^{st} percentile, median, 99^{th} percentile. The upper extremity indicates the maxima. The minima overlap with the 1^{st} percentile in all cases.



(b) Forwarding table sizes relative to the size of the full network, November 2010.

Figure 5.11: TZ_k forwarding table sizes.

5.4.4 Summary

In this section, I presented constraint checking conformance, path stretch distributions, and forwarding table sizes and distributions for TZ_k on snapshots of the AS graph from 1997 to 2010. The results are consistently tightly bounded, with low path stretch and small amounts of state in forwarding tables. The average path stretch is consistent with the standard TZ algorithm applied to AS graphs. Finally, and importantly, the revised landmark set selection algorithm presented in this chapter, with the desirable properties enumerated in Section 5.2, does not impact the routing performance of TZ compact routing.

5.5 Discussion

This chapter has presented data on the stability of the nucleus derived from the k-cores decomposition of Internet AS topologies over time, followed by the definition of the TZ_k compact routing algorithm and performance results comparing it to standard TZ compact routing. These are two additional issues that must be resolved for this landmark selection to be useful in a dynamic network: first, the decentralisation of the landmark calculation, and second, being able to reason about policy constraints and how this affects landmark selection.

Regarding decentralisation of the algorithm, the TZ_k algorithm presented in Section 5.3 is an offline algorithm, designed to run on a snapshot of the AS graph. It is not ideal in a dynamic network to entrust the calculation of such a set to one entity, and so a fully distributed computation of the k-cores decomposition is required. A distributed algorithm for computing a k-shell is outlined in [110]. This requires k as an input parameter, but the goal of such an algorithm for TZ_k compact routing is to find the k_{max}-shell. This distributed algorithm could be readily modified to operate without prior knowledge of k_{max}, either by operating in rounds (increasing k until no nodes identify themselves as belonging to the k-shell), or by calculating each k simultaneously (each node maintaining a list of k-shells which it may or may not belong to). The algorithm operates iteratively until an equilibrium is reached. A rigorous analysis of the number of messages required to compute the k_{max}-shell in a distributed manner on a graph of this size is desirable, although absolute performance is not expected to be critical since the landmark set is highly stable over a period of months. In the context of a future Internet protocol, any decentralised k-cores algorithm requires networks to exchange information about how many other networks they have agreements with. Much of this information is exposed implicitly via BGP in the current network.

Also important is determining *when* recomputation of the landmark set is required. I have shown landmark sets to be remarkably stable over time (Section 5.2.2), but there are two eventualities that will require that the landmark set to be recomputed if TZ constraints are

to be met: when an AS acting as a landmark wishes to withdraw from the network, or if the clustering constraint for any AS is violated.

Once recomputed, readvertisement of the set is required to announce the ASes that have accepted to become landmarks. Recall that the landmark set represents the only globally visible set of nodes, and its reachability information is required to facilitate packet forwarding between any pair of nodes across the whole network. Given that the new landmark set is likely to contain primarily the same nodes as in the previous landmark set, this should result in only a partial recomputation of routing state at the other ASes. Landmark set advertisements can be similar to BGP advertisements, propagating outward from their origin and maintaining path vectors. The primary requirement is that the whole network has reachability information for the landmark set so that each AS can determine the nearest landmarks in that set, and also forward data packets towards any landmark required to reach a given destination.

The enforcement of policy requirements, currently managed in BGP through path inflation and modification of local preferences, is not taken into account by any compact routing algorithm. Clearly, any architecture which routes on AS number alone conflicts with the desire for fine-grained control over prefixes. However, there is scope for weighting links according to policy, or considering links to be unidirectional, when determining the k_{max}-shell of a network. This would allow ASes to control which links are visible to the k-cores algorithm and apply some form of policy. In particular, if it were possible to use only customer/provider links to calculate the nucleus, then only those willing to carry data would be nominated as landmarks, and highly connected peers (e.g., CDNs) would not be nominated.

Further, utilising additional AS hops in the network may incur additional monetary costs, or incur excessive traffic load across links, and so this trade-off must be considered. More investigation of the effects on traffic flows is required.

Thus, this chapter has indicated that stable landmark sets in the dynamic Internet AS graph are feasible, and that the selection of such landmarks is reasonably simple to achieve when provided a full graph. The two additional problems, of decentralised computation for that set, and operating within a policy-constrained environment, are common to *all* compact routing protocols.

5.6 Summary

In this chapter, I presented an analysis of stability of the k_{max}-shell derived from applying the k-cores graph decomposition algorithm to Internet AS graph snapshots spanning 15 years.

Then, I demonstrated that the nucleus revealed by the k-cores graph decomposition of Internet

graphs has remained stable despite the network's growth. This nucleus offers a highly-visible, well-connected, stable, and long-lived set of nodes.

Next, I presented the TZ_k compact routing algorithm, a modification to the TZ compact routing algorithm that uses the k-cores graph decomposition to provide a landmark set that is both stable and topologically well-placed. Such a stable set of landmarks is beneficial for real-life deployment of a decentralised protocol designed to operate according to the constraints defined by the TZ compact routing algorithm.

The TZ_k routing algorithm offers the same consistently excellent routing performance on AS graphs as the TZ routing algorithm, usually achieving shortest paths and, for non-shortest paths, usually adding only one additional AS hop. Using the landmark sets generated by the TZ_k algorithm, forwarding tables are extremely small. These properties constitute an important step toward the development of a deployable compact routing protocol, and the application of compact routing algorithms to dynamic networks.

Chapter 6

Distributed TZ Compact Routing

Compact routing research has defined provable bounds on the routing problem in terms of the trade-off between simultaneously minimising routing table sizes and path stretch. It presents a promising direction for a future long-term scalable Internet routing architecture.

The earlier chapters in this dissertation focussed on evaluating compact routing algorithms on static graphs. In Chapter 4, I confirmed observations from previous work by others that the TZ compact routing algorithm and the BC compact routing algorithm demonstrate strong performance, with tiny routing tables and low mean path stretches of around 1.1 on the Internet AS graph. I presented a longitudinal study of the path stretch performance and routing table sizes generated by these compact routing algorithms when applied to Internet AS topologies derived from archived BGP routing data spanning 15 years. This demonstrated that the TZ compact routing algorithm exhibits considerably more consistent path stretch than the BC compact routing algorithm, and is more appropriate as a candidate for a future Internet routing architecture.

The TZ compact routing algorithm is not, as defined in [80], easily deployable in a network as a routing protocol. Its landmark selection algorithm selects landmarks pseudo-randomly which is not useful as the network changes, and the labelling algorithm is centralised. In Chapter 5, I demonstrated that there exists a stable set of nodes central to the Internet AS graph that offer the same consistently low mean path stretch demonstrated in Chapter 4. This set forms a stronger basis for a landmark set that is more useful in a deployable TZ compact routing protocol than a pseudo-random selection, and is potentially decentralisable.

In this chapter, I present the design of a decentralised path vector protocol which uses landmark nodes and distance measures to perform TZ-style compact routing on dynamic graphs. For the purposes of this discussion, in this chapter I will use the landmark sets from Chapter 5, and assume that there is an orthogonal protocol in use whose purpose is to determine such a valid landmark set; Section 5.5 discusses how this might operate.

The contributions of this chapter are as follows:

1. I present a distributed network protocol based on the TZ compact routing algorithm which is suitable for dynamic networks. I define the protocol in terms of routing advertisements dispatched by ordinary nodes, routing advertisements dispatched by landmark nodes, and the conditions in which these advertisements are propagated by other nodes. Also covered are link addition and removal behaviour.

2. From simulation, I present results showing the routing overhead when using this protocol on static Internet AS graphs, in terms of messages exchanged in order to distribute all state starting from only local state at each node. This indicates that the scoping of the routing state according to the constraints defined in the TZ compact routing algorithm leads to a considerable proportion of routing overhead being attributed to the (globally visible) landmark set. I also present measures of path stretch and table size, indicating that this protocol predominantly reveals shortest paths through the network.

3. Additionally, I present results specific to the compact routing protocol. This includes numbers indicating the proportion of the network from which nodes *do not* receive routing updates, and the length of paths traversed by the routing updates, as an indication of the proportion of the network that routing updates generally affect (*i.e.*, as a measure of how well landmark-based routing keeps state updates local to the cluster and the landmarks). I also show the size of the valid landmark sets that source nodes can select from for any given destination.

This chapter is structured as follows:

Section 6.1: discusses various requirements and trade-offs for a deployable compact routing protocol, with a particular focus on the decentralisation of the TZ compact routing algorithm. This section also briefly discusses supporting infrastructure to facilitate packet forwarding with a compact routing protocol.

Section 6.2: presents a decentralised compact routing protocol suitable for dynamic graphs.

Section 6.3: presents the forwarding algorithm used by this distributed routing protocol.

Section 6.4: outlines the model used for experimentation, including the latency model applied to the same AS topologies used earlier in this dissertation.

Section 6.5: presents an evaluation of the routing protocol in terms of the protocol's messaging overheads. This section also covers the path stretch results and the forwarding table sizes as presented in previous chapters.

Section 6.6: describes how such a compact routing protocol should operate in the presence of overlapping advertisements, such as IPv4 prefixes, and indicates the routing overhead required.

Section 6.7: summarises the chapter.

6.1 Design Rationale and State Trade-offs

The TZ compact routing algorithm is sufficiently different from shortest path routing algorithms that it is useful to first consider the primary components that allow TZ compact routing to operate with sublinear state at all nodes. These include landmarks, distance calculations, and the supporting infrastructure, which are discussed in the following subsections.

6.1.1 Landmarks

The TZ compact routing algorithm utilises a small set of nodes as a globally visible set of landmarks. It is important to consider, for the design of a fully distributed routing protocol, that the introduction of landmark nodes should be complentary to its operation, and not compulsory. If this is the case, then the following is true:

1. Given a network with no landmark nodes, shortest path routing is achieved between any two points. Routing advertisements are not scoped. State requirements are linear with respect to the number of nodes in the network. This model includes any distance-vector or path-vector routing protocol (such as RIP or BGP).

2. Given a sufficient number of well-placed landmark nodes, a network can achieve sublinear state requirements with stretch-3 routing, as proven in [80] and demonstrated on AS topologies in Chapter 4.

3. Given an increasingly large number of landmark nodes, and the fact that landmark nodes are globally visible, the routing system will tend toward linear state requirements and shortest path routing as the landmark set tends toward $|V|$. If all nodes are landmarks, then routing is shortest path, and state requirements are linear with respect to the number of nodes in the network.

Thus, given a network with no landmarks, systematically introducting landmarks will reduce the routing state held in the network while simultaneously tending multiplicative path stretches toward a mean multiplicative path stretch of approximately 1.1, as shown in Chapter 4. The bounds presented in Section 4.2.1 allow point 2, above, to guarantee sublinear

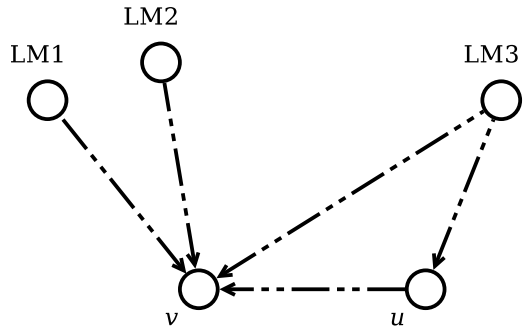


Figure 6.1: A high level indication of the distance measures known to a node in a distance vector protocol. The diagram indicates the messages received at v, and thus v knows the full landmark set $A = \{LM1, LM2, LM3\}$ and distances to those nodes, and the distance between u and v, but not the distances between u and A.

state at all nodes. To achieve this, the decentralised compact routing protocol defined in this chapter assumes a reasonable landmark selection mechanism exists. The details of such a mechanism are left as future work, but Chapter 5 shows that the k-cores graph decomposition offers a promising direction.

6.1.2 Distance Calculations

The TZ compact routing algorithm uses the clustering equation (equation 4.1, page 40) to determine what state is retained in routing tables at all nodes. The cluster for a node v contains every node u if u is nearer to v than v is to any of the landmark set A. This equation determines precisely what routing state should be retained at each node, and thus forms the basis of a decision algorithm for storing and propagating advertisements in a decentralised protocol. This clustering equation generates a "horizon" beyond which knowledge of a node should not propagate, therefore restricting visibility of the origin. Note that the distance that advertisements will propagate relies on the distance between the advertisement's origin and its nearest landmark(s) (as per equation 4.1 on page 40), and so v will retain forwarding references to nodes that are varying distances from the landmark set.

To allow nodes to make local decisions on what state should be retained, each node requires enough information to perform the distance calculation as defined for the clustering equation. Given a node v which receives a routing advertisement from another node u, to make a decision according to the clustering algorithm, v requires the following two distance measures to evaluate the equation: the distance between the two nodes $d(u, v)$, and the distance between u and its nearest landmarks $D(u, A)$ (*i.e.*, the same functions as defined in Section 4.2.1). Figure 6.1 indicates the information known to node v in a standard path vector protocol. For both of these measures in turn: $d(u, v)$ is carried in the routing advertisement, as it will be indicated in the path retained in the *update* message, and although $D(v, A)$ is known by v,

$D(u, A)$ is unknown. $D(u, A)$ is required at recipients of a routing advertisement to make appropriate decisions on which state to retain in a compact routing protocol, and thus this information must be carried by the routing protocol.

Landmark nodes present an interesting case for the distance calculation. The cluster for each landmark $l \in A$ is defined as the empty set, since $D(l, A)$ is always zero for a landmark node. Therefore, according to the TZ compact routing algorithm, landmark nodes hold only routing state to reach the rest of the landmarks in the set A, since it is globally visible. The TZ compact routing algorithm is designed this way to assert the sublinear bound on the routing state stored at every node. The lack of routing state at landmarks is compensated for by providing a third field in the packet headers for packet forwarding: a *port* or *next hop* from the landmark towards the destination. This is a reasonable choice for the TZ compact routing algorithm because it is heavily centralised and designed for static graphs. However, this *next hop* information is derived from path information local to the landmark node, and may change more frequently than the landmark itself, and so introduces an additional point of path failure. This is undesirable in a decentralised protocol.

For a distributed protocol, it seems more reasonable to deviate from the TZ compact routing algorithm's state requirements for the small set of landmark nodes, and allow them to retain the routing updates that they receive, thus allowing them to make local forwarding decisions. A landmark l should retain in its local forwarding table references to every node u that considers l to be its local landmark. Given a network with n nodes, the size of the landmark set is $O(\log n)$ (as per equation 4.3, page 41), and each of those nodes may have multiple valid landmarks. If a node u is equidistant from all landmarks, then all landmarks will retain forwarding state for u. Thus, the state requirement at the landmarks (indeed, *only* the landmarks) is $O(n)$, in that no lower bound can be guaranteed, although state retained will usually be considerably less than n.

In summary, state retention at landmark nodes must be treated differently from state retention at ordinary nodes.

6.1.3 Supporting Infrastructure

To facilitate packet forwarding, a source node v with a destination u must know at least one of the landmarks that are $D(u, A)$ hops from u. The landmark is used by the forwarding algorithm as an intermediate destination for data packets at nodes where u has not been held in local routing state by the clustering equation.

The description in Section 6.1.2 states that a node u indicates its minimum distance $D(u, A)$, from the landmark set, but not directly the set of landmarks that are this minimum distance

from u. The discovery at the source node of the set of landmarks for a destination is left to supporting infrastructure.

The existence of the set of landmark nodes, and the partial routing state retained in the rest of the network, requires that either a supporting naming infrastructure exists to map from destinations to valid landmarks, or that the landmark nodes form a default-free zone to permit initial packet forwarding if a source node does not have a valid landmark to forward packets to a destination. This chapter describes only the routing protocol, and does not specifically define the naming infrastructure, but asserts that some supporting infrastructure is required. Two possibilities are as follows:

Naming Infrastructure: In the current Internet architecture, the Domain Name System (DNS) is an out-of-band mechanism that an end-host can use to determine the locator of another end-host if only a name is known. Conceptually, this provides a direct mapping between names and IP addresses. Often, multiple names may map onto one IP address, and vice versa.

For a decentralised TZ compact routing protocol, there is an additional requirement for a naming service to facilitate packet forwarding: packet headers require the ID of a valid landmark to use, as well as the ID of the destination node. These must be discoverable to the source. Thus, an out-of-band mechanism is required to provide enough information about the destination. Utilising the DNS is a reasonable approach, as are in-network alternatives such as LISP-ALT [111]. The key requirement is that it is possible to map the endpoint ID onto a valid landmark to use during packet forwarding.

A Default-Free Zone: An alternative solution may be for landmark nodes to retain *all* advertised state, leaving the rest of the network free to hold considerably less state. As the landmark set is globally visible, there is no barrier to landmarks connecting to each other and sharing full tables. In this case, the landmark nodes retain full state, much as BGP does currently. The benefit of defining the set of landmark nodes as a default-free zone for the network is that it obviates the need for any additional lookup infrastructure, with the downside that the routing scalability problem is not solved at this small subset of landmark nodes. When emitting packets, if a node is able to forward initial packets via the landmark region, in the absence of a direct path to the destination and with no known landmark for the destination, then destination nodes can tag return packets with a list of valid landmark nodes to use in future. Thus, all state is retained locally and there are no additional infrastructure requirements.

An additional benefit of having a supporting infrastructure is that it provides scope for exposing to a source node the full set of *candidate* landmarks for a destination. The TZ compact

routing algorithm as defined in [80] labels each node v with only one landmark from the subset of A that is the minimum distance $D(v, A)$ from v.

However, recall that the longest path is discovered when forwarding packets between two nodes u and v if the landmark for the destination, l_v, is reached. That is, packets follow the full path $u \rightarrow l_v \rightarrow v$. Note that path from $l_v \rightarrow v$ will always be shortest path, but the length of the $u \rightarrow l_v$ segment is dependent on which landmark is chosen for v. Thus, exposing the set of landmarks to u allows u to choose the nearest of the valid landmarks for v, and thus minimise the worst-case path length and therefore also path stretch.

The remainder of this chapter focusses on the design and evaluation of the routing protocol, and leaves any supporting infrastructure as future work.

6.2 A Distributed Compact Routing Protocol

In this section, I discuss the design of a network routing protocol based on TZ compact routing. The routing protocol described is a hard-state path vector protocol, similar to BGP. That is, advertisements and withdrawals are explicitly announced to the network. Routing state is not periodically refreshed as it would be in a soft-state protocol. The advantages of a path vector protocol are that it allows an easy mechanism for loop detection and therefore also prevents the count-to-infinity problem inherent in distance vector protocols.

The protocol is designed according to the rationale described in Section 6.1. The protocol carries additional information in advertisements generated by a source node that allows recipients to perform the clustering calculation and thus place a limit on the distance an advertisement will propagate. An ancillary lookup infrastructure is not covered, but the logic that dictates when it should be updated is described.

In this section, I present the operation of a network routing protocol based on the TZ compact routing algorithm. Section 6.2.1 covers the basic conditions in which routing state should propagate. The four primary message types that are covered in Sections 6.2.2 through 6.2.5 define how ordinary nodes and landmark nodes advertise and withdraw from the network. Respectively, these sections define the following messages:

update(src, path, landmark_distance)
withdraw(src, path)
landmark_update(src, path)
landmark_withdraw(src, path)

Additionally, Sections 6.2.6 and 6.2.7 describe how link addition and removal affects the routing state. Finally, I discuss how this state is retained in local routing tables in Section 6.2.8.

6.2.1 Routing State

Routing state should propagate across every node u where the clustering equation C_u (equation 4.1, page 40), holds. Given the distance information required to perform this calculation, each node is able to determine whether an advertisement should be retained in local routing state, and possibly propagated to neighbours.

In a dynamic routing environment there is scope for horizons changing as the landmark set changes or as links are added and removed. By defining this routing protocol as a hard-state routing protocol, advertisements propagate as far as the clustering equation holds, and not further. Routing state need not be refreshed. In the event that changing network conditions causes the horizon of an existing advertisement to recede, an updated advertisement must propagate beyond its new horizon so that paths retained in the local routing tables according to the previous horizon can be removed. If an advertisement arrives at a node where it should not be retained according to Equation 4.1, but the path is in the local RIB, then the path associated with this advertisement *must* be removed and propagated if it has previously been propagated, to allow neighbours that have received the same advertisement to also remove it.

Propagation has a natural stopping condition. If an advertisement arrives at a node, but the advertisement should not be retained according to Equation 4.1 and the path is *not* stored in the local routing table, then it has never been seen before. The advertisement can be safely ignored and not propagated to neighbours, and thus the distance travelled by the advertisement is scoped. Details are discussed in Sections 6.2.2 and 6.2.3.

6.2.2 Node Advertisements

To describe how node advertisements propagate across the network, I will first describe the behaviour of the source (origin) of an advertisement, and subsequently I will describe the decision process when other nodes receive advertisements. Landmark announcements are discussed in Section 6.2.4.

When a node v joins the network by connecting to another node w, w forwards its current routing state to v, updating the path vectors appropriately by pre-pending its own ID. This gives v its local view of the routing system including both nodes and landmarks, upon which all future routing messages act. Note that after the table exchange, v may be able to trim the routing table it has received by evaluating all advertisements against the clustering equation (for all advertisements received, $d(u, v)$ will be one AS hop longer than at the previous AS hop, thus affecting the outcome of the clustering equation).

Having received a full table, v is aware of all known landmark nodes. This provides v enough information to advertise itself to the network with its known value for $D(v, A)$.

Algorithm 2 : **At u, handling update(v, path, landmark_distance)**

```
case update(v, path, landmark_distance) ⇒ {
  if ( path contains u ){  // Message has looped.
    return
  }
  if ( u is a landmark ){
    if ( path_len ≤ landmark_distance ){
      rib = rib + (v → path)
      fib = fib + (v → rib.bestPathTo(v).path.head)
    }
    else if (  path_len > landmark_distance ){
      rib = rib −− (v → path)
      if ( rib contains no paths to v ){
        rib = rib −v
        fib = fib −v
      }
    }
  }
  else if (  path_len == 1) ∨ (path_len < landmark_distance ){
    rib = rib + (v →, path)
    fib = fib + (v → rib.bestPathTo(v).path.head)
    Propagate update(v, u :: path, landmark_distance) to neighbours
  }
  else if (  path_len ≥ landmark_distance ){
    if ( rib contains (v → path) ){
      rib = rib −− (v → path)
      if ( rib contains no paths to v ){
        rib = rib −v
        fib = fib −v
      }
      Propagate update(v, u :: path, landmark_distance) to neighbours
    }
  }
}
```

An ordinary node v announces an advertisement to the network by sending an *update* message to its direct neighbours. Routing advertisements with the minimal state take the form:

update(src, path, landmark_distance)

That is, after joining the network, v will send an *update(v, $\{v\}$, $D(v, A)$)* message to its neighbours. If A is empty, then v will send *update(v, $\{v\}$, ∞)*, permitting the advertisement to propagate throughout the network, fulfilling the requirement that the protocol operate in the absence of landmarks, as discussed in Section 6.1.

A node v will send a new *update* message for a previously dispatched advertisement if its nearest landmark distance $D(v, A)$ changes. The new update is dispatched with a modified *landmark_distance* field. This occurs if a landmark announcement is received that lowers $D(v, A)$ or, similarly, v's nearest landmark departs and therefore $D(v, A)$ increases. Similarly, the establishment of a new link in the network can reduce $D(v, A)$, and the removal of a link can increase $D(v, A)$. Link addition and removal are discussed in Sections 6.2.6 and 6.2.7 respectively.

When $D(v, A)$ increases at the origin v, the new update must propagate across the network for as many AS hops as Equation 4.1 holds. When $D(v, A)$ decreases at v, the routing advertisement should be updated, with the new information propagating across the network for as many AS hops as Equation 4.1 holds, or as far as the advertisement travelled previously to remove invalid state. The *landmark_distance* measure can only be updated by v itself, and must not be modified by any other node. It is state local to v, and no other node can infer this value (unless they are v's sole neighbour).

This section has so far described the conditions under which a node will emit an *update* message to issue an advertisement. Next, I discuss how the recipients of that *update* message react to it.

An advertisement sent by v will be received by neighbouring nodes who may choose to store the advertisement, and may also choose to propagate it to their neighbours. A node u, on receipt of an *update* message, executes the full decision process described in Algorithm 2. Bearing in mind the different semantics for state retention between landmark nodes and ordinary nodes described in Section 6.1.2, I describe the logic at a high-level here in two parts. First, the logic for ordinary nodes operates as follows:

- The advertisement is dropped if it has looped.
- The advertisement is retained in the local RIB if the advertisement arrived from a direct neighbour, *or* the the length of the path traversed is less than the *landmark_distance*.

- The advertisement and path pair is removed from the local routing state if the path length is not less than *landmark_distance*. If the removal of this path is successful, then the advertisement must be propagated, and if the removal of this path leaves the local routing state with no suitable next hop back to the advertisement, then the node must remove all references to the advertisement.

The logic for landmark nodes is then similar, the primary difference is that paths with length equal to *landmark_distance* are retained, compared to the non-landmark case where they are not:

- The advertisement is dropped if it has looped.
- The advertisement is retained in the local RIB if the length of the path traversed is not greater than the *landmark_distance*.
- The advertisement and path pair is removed from the local routing state if the path length is greater than *landmark_distance*. If the removal of this path leaves the local routing state with no suitable next hop back to the advertisement, then remove all references to the advertisement.

Note that this does not describe state sharing between landmarks in the case that they are forming a default-free zone, as discussed in Section 6.1.3.

It is possible for a local node to retain multiple paths to a destination that are valid according to the clustering equation (all paths satisfy the distance constraints), but for some paths to be longer than others. On installation of a new path to a destination in the routing table, the next-hop on one of the shortest available paths is chosen to be installed in the local forwarding table.

6.2.3 Node Removal

When a node wishes to leave the network, it must explicitly withdraw its advertisements from the network. Node removal hence requires a distinct message type:

withdraw(src, path)

When a node wishes to leave the network, it transmits a *withdraw* message to its neighbours prior to disconnecting.

When other nodes receive a *withdraw* message, they must remove their local state for this advertisement and, if the advertisement had been held locally, propagate the *withdraw* message to neighbours. This only affects state relating to this advertisement, and does not affect anything else. As before, all packets that have looped are dropped.

Algorithm 3 : **At u, handling withdraw(v, path)**

```
case withdraw(v, path) ⇒ {
  if ( path contains u ){  // Message has looped.
    return
  }
  if ( rib contains (v → path) ){
    rib = rib −− (v → path)
    if ( rib contains no paths to v ){
      rib = rib −v
      fib = fib −v
    }
    Propagate withdraw(v, u :: path) to neighbours
  }
}
```

If an implementation of the protocol can authenticate that messages received have arrived from the origin rather than an impostor, perhaps by using public-key infrastructure as per Secure-BGP, then a withdraw can be acted upon immediately and all paths to the destination can be removed immediately: if the local RIB and FIB contain reference to this advertisement, then all paths to a destination can be removed from both the RIB and FIB, and the withdrawal message propagated.

If the routing system is not secure, then a more sensible option is to operate as described in Algorithm 3, and handle withdrawal messages as if they were standard *update* messages with the *landmark_distance* field set to 0.

On the event of a node failure prior to dispatching a *withdraw* message, neighbouring nodes must dispatch a withdrawal on the origin's behalf. To do so, reasonable attempts to assert liveness of the node must be attempted. For example, if a TCP connection has failed and cannot be re-established.

6.2.4 Landmark Announcements

Landmark announcements are defined as distinct from ordinary announcements, because all other nodes must be aware of the full landmark set. In terms of propagation rules they are equivalent to an ordinary advertisement with the *landmark_distance* field set to ∞; they must propagate across the whole network. However, a distinction is required that echoes the distinction in the state logic presented in Section 6.2.2, and to allow other nodes to easily identify the global landmark set.

As before, I first describe behaviour at the origin of a landmark announcement. As shown in Chapter 5, there exists a stable set of nodes at the heart of the Internet that provide a good candidate landmark set. This set changes very slowly over time, and so the introduction of a

Algorithm 4 : **At u, handling landmark_update(l, path)**

```
case landmark_update(l, path) ⇒ {
  if ( path contains u ){  // Message has looped.
    return
  }
  Propagate landmark_update(l, u :: path) to neighbours
  old_landmark_distance = D(u, A)
  A = A ∪ l
  if ( d(u, l) < old_landmark_distance) ){
    Send update(l, u :: path, D(u, A)) to neighbours
    u must update supporting naming system
  }
  else if (  d(u, l) == old_landmark_distance ){
    u should update supporting naming system
  }
}
```

new landmark to a network is an infrequent occurrence. When the landmark selection algorithm nominates a new landmark, it must advertise itself as such with a landmark message, which is simply:

landmark_update(src, path)

Thus, a node l will send *landmark(l, $\{l\}$)* to all of its neighbours if it becomes a landmark.

Having described how a landmark announcement is emitted, I now describe the behaviour at all other nodes that receive the landmark announcement. Landmark advertisement handling is simpler than for update messages from normal nodes. In Algorithm 4, I specify pseudocode for landmark advertisement handling. In effect, all landmark routing advertisements are propagated at all nodes, and all advertisements that have looped back to a previous hop on the path are dropped.

The additional logic concerns the behaviour if receipt of a landmark advertisement reduces, for a node u, the $D(u, A)$ measure; this implies that u has to update its advertisement with the new state. The lower distance affects the cluster calculation at other nodes, and reduces the number of nodes that will retain references to u. There are three possible outcomes when a landmark advertisement is received. Given a landmark l that is $d(u, l)$ hops from u:

- If l is nearer to u than any of the paths to any of the landmark set previously observed, then l becomes the nominated landmark for u. In order for this to be the case, the landmark must be installed in the lookup system (if one is running) for this destination, and second, the routing state for u must be updated to reflect the new distance between u and the landmark. If the new distance is not advertised to update a previous

advertisement, packet forwarding will still operate correctly, but without the advantage of minimised state.

- If the current nearest landmark and l are equidistant from u, the routing state for u need not change. The landmark can be installed in the lookup system to be used by sources.
- If l is farther from u than the current-nearest, no routing state changes, and the landmark announcement must be forwarded to neighbours to maintain global visibility, and the landmark stored in local routing/forwarding tables.

On the arrival of a new landmark, existing nodes need only update their routing advertisements if the landmark is *nearer*, implying that routing advertisements for these nodes should persist fewer AS hops into the network than they did previously. The updates propagate through the network as described in Section 6.2.2.

There is one additional caveat regarding the receipt of a *landmark_update*, and the conditions under which a node will issue an *update* for its existing advertisements. A node will only readvertise its advertisements in the event of a new landmark announcement if that new landmark is nearer than any of the previous landmarks it was aware of. Thus, for the new landmark node to attain a full list of nodes that may consider it their landmark, direct neighbours for the new landmark will reissue any advertisements they previously propagated to the landmark that it may have dropped. This allows the landmark node to recalculate its routing tables appropriately, and resolves the issue.

6.2.5 Landmark Removal

If the landmark selection algorithm decides to remove a node from the landmark set, the node must advertise this state change to the network. Landmark status is removed by announcing a *landmark_withdraw* message:

landmark_withdraw(src, path)

Recipients of the message behave as described in Algorithm 5. It operates as follows:

- If l is farther from u than the current best. In this case, no routing state changes. The withdrawal must be forwarded to neighbours.
- If l is equidistant from u to other nearest landmark nodes. No routing state changes, but u node must update the lookup mechanism to remove l as a candidate for sources to use.

Algorithm 5 : **At u, handling landmark_withdraw(l, path)**

```
case landmark_withdraw(l, path) ⇒ {
  if ( path contains u ){  // Message has looped.
    return
  }
  Propagate landmark_update(l, u :: path) to neighbours
  old_landmark_distance = D(u, A)
  A = A − l
  if ( d(u, l) < old_landmark_distance ){
    Send update(l, u :: path, D(u, A)) to neighbours
    u must update supporting naming system
  }
  else if ( d(u, l) == old_landmark_distance ){
    u should update supporting naming system
  }
}
```

- If l is nearer to u than the next-nearest landmark. In this case, advertised routes *must* be updated with the *landmark_distance* field set to the distance to the nearest landmark once l is removed completely from local routing state. The lookup system must be updated immediately.

Having advertised its change in status, the routing state stored at the landmark should be trimmed. The landmark can perform the distance calculation on all local state by following the non-landmark branch in Algorithm 2.

Alternatively, a node can remove its landmark status by sending an ordinary *update* with the *landmark_distance* set to the number of AS hops to the nearest of the other landmarks. Nodes must forward the update advertisement according to Algorithm 2.

Sections 6.2.2 through 6.2.5 so far describe the two different forms of node insertion, and the two different forms of node withdrawal, and how the network reacts to each of those. Two additional cases involve the addition or removal of additional links between nodes, described in the following two sections.

6.2.6 Link Addition

The addition of a new link between two nodes brings parts of the network closer together. When two nodes connect to each other, they must share full routing state with each other, with path vectors updated appropriately. This routing state contains both landmark advertisements and ordinary advertisements, and each is handled according to the algorithms 2 and 4 respectively, implying that advertisements received by either node may be propagated.

The propagation of state across this new link may reduce the distance between nodes and the landmark set. Since the origin of an advertisement u is the only node that can update its *landmark_distance* announcement, the arrival of landmark advertisements nearer than previously observed will trigger an *update* that will propagate outward according to Algorithm 2 and ultimately remove excess routing state.

6.2.7 Link Removal

In the case of a link removal, the opposite is true. Nodes on either side of the link must withdraw all of the advertisements they have propagated that traversed the link. The removal algorithms (Algorithms 3 and 5) are defined on a per-path basis, and so will remove paths appropriately, but will not completely remove the origin of the advertisement provided a valid path still exists.

A link removal may *increase* the path lengths between nodes and landmarks, and thus may also increase the minimum distance between a node u and its nearest landmark(s), prompting an *update* be emitted by u to extend the distance that its advertisements will propagate.

In each of these cases, the algorithms specified already will handle the propagation of the required state. No complex recomputation is required at intermediate nodes when links are added or removed, but the distance that advertisements travel from their origin may change; this is a change that is driven by the origin itself once it learns of changes in the paths to the landmark set.

6.2.8 Building the Routing Information Base

As nodes receive advertisements from other nodes, they must retain them in local routing state. The protocol described in this chapter is a path vector protocol, and so paths must be retained in local routing state. However, for forwarding state, only the destination and an appropriate next hop for that destination need be stored. The routing state is held in the Routing Information Base (RIB), and forwarding state is held in the Forwarding Information Base (FIB).

The type description for the RIB is:

$$\text{Map}[u \rightarrow \text{Set}[\text{paths}]]$$

That is, it is a mapping from a destination to the set of paths that this advertisement traversed to reach the local node.

As in BGP, and especially if policy constraints are being applied to routing state, there may be an additional *RIB-out* that contains the routing advertisements that were propagated, according to the neighbour they were sent to. This is especially useful for re-sending updates to nodes if they become landmarks, for example.

If routing were able to employ policy to determine which routes are retained, then additionally a *RIB-in* would be useful to hold all routing advertisements and their associated paths from each neighbour they were received from. Policy reconfigurations can be applied over the contents of the RIB-in to recompute local routing state. A RIB-in may be additionally useful at nodes to alleviate overhead associated with table exchanges when nodes become landmarks, though this is a relatively rare event.

The FIB contains destinations and the next hop along the path toward that destination. The type description of the FIB is:

$$\text{FIB: Map}[u \rightarrow next\_hop]$$

That is, it is a direct mapping from a destination to the next hop.

It is important to maintain a distinction between the FIB and the RIB in implementations of the protocol. New node advertisements and node withdrawals will affect the contents of both the RIB and the FIB. During normal operations, the RIB will be updated more frequently with path changes for existing destinations that often do not affect the contents of the FIB. Additionally, retaining a more complete view of the network in the RIB allows for local decisions to be made in terms of best next-hop selection for the FIB (possibly involving associated link costs), while also keeping the FIB simple for packet processing.

6.3 Packet Forwarding

The packet forwarding algorithm is simpler than the packet forwarding algorithm defined in [80] for TZ compact routing, because of the removal of the third field in packet headers that the TZ compact routing algorithm requires. In the context of IPv4 or IPv6 packets, the landmark destination could be carried in an IP option in the packet header.

Quite simply, packet forwarding operates as in Algorithm 6. The algorithm is simple to understand. If the packet has arrived at its destination, then the forwarding algorithm is complete. If not, but the destination for the packet is held in the local forwarding table, then forward directly towards the destination. Otherwise, if the destination is not held locally, forward towards the nearest landmark node for the destination. The landmark node is only used while forwarding data packets when the destination is not held locally.

Algorithm 6 : **Packet Forwarding**

if ($u == v$){
  We are done.
}
else if (v is contained in FIB){
  Forward to v
}
else if (l_v is contained in FIB){
  Forward to l_v
else
  Forward to nearest landmark.
}

Finally, the algorithm as described here includes the possibility of routing toward the landmark set in the eventuality that the landmark is not known/not set. This is not general to the protocol, but rather, specific to having a well-connected landmark set.

6.4 Evaluation

To evaluate the messaging overhead for this the compact routing protocol, I simulate the exchange of all routing state on the Internet AS topologies defined in Section 2.1. Nodes are initiated with connections to their neighbours and no locally retained routing state. The simulation progresses until all routing state has propagated and is stable. For the purposes of adding non-uniform traversal latency to the simulation to affect the behaviour of routing state propagation, I augment the same AS snapshots as used previously in Chapters 4 and 5 to include latency measures. The latency model describes latency that occurs *between* and *within* ASes. Rather than choose random latency values, the latency values are derived from public CAIDA Ark traceroute data as described in Appendix A.2. The networks are simulated in a distributed time-based event simulator, the design of which is described in Appendix B.

To help scale the simulation of the routing protocol, I construct a more accurate model of the AS topologies. Previously, the graphs were undirected, unweighted graphs. For this chapter, I use the AS paths stored in the BGP data to determine the set of ASes that offer transit services to other ASes, and those that operate only as stubs. This is the distinction highlighted in Figure 2.2a on page 12. This change to the model modifies the mean shortest path lengths for the graphs only slightly[1], but reduces considerably the complexity of the event simulation for routing update propagation. This modification does not have a major effect on the path stretch with compact routing (the effect is enumerated in Section 6.5.2).

[1]The mean path length for 31 May 2011 if treating all nodes as transit nodes is 3.81 AS hops, and the mean path length for the same date if some nodes are stubs is 3.89 AS hops.

Having allowed all routing state to propagate, I test the path stretch by generating traffic to run the forwarding algorithm. I use the same test pairs as were used previously in Chapters 4 and 5.

In order to discuss the behaviour of a routing protocol, I make the following assumptions:

- As in BGP, links between nodes (here, modelling ASes) are reliable; messages sent *between neighbours* arrive complete and in-order. This is equivalent to border routers exchanging messages over TCP connections.
- I assume the existence of a lookup mechanism for a source node to derive a set containing at least one landmark to use to forward packets toward any given destination. The requirement of a lookup mechanism was highlighted in Section 6.1.3. Some infrastructure may be required to facilitate mapping from a destination to the set of valid landmarks, as discussed in Section 6.1.3, but for the purposes of this section where routing overhead is the focus, is not directly relevant. For the simulation, nodes register their landmark set with a central lookup node that is used when performing the stretch evaluation and packet forwarding after routing state has been exchanged.
- Given that these graphs are static, landmark nodes are known at the start of the simulation, and advertise themselves as such. The landmark sets are those derived by Algorithm 1 (page 71) that were generated for Chapter 5, which in all cases for these AS snapshots is the k_{max} core.
- For propagating routing state, nodes operate a per-origin update timer, similar to the minimum route advertisement interval (MRAI) operated in BGP. This rate-limits advertisements for a particular origin to reduce the flow of updates for that origin, and allow nodes to receive multiple copies of the same advertisement from neighbours before deciding which paths to propagate. For the purposes of simulation, the timer is set very low, at 2ms, based on inter-arrival times for messages on the 8 November 1997 snapshot.

As in Chapters 4 and 5, I evaluate the path stretches observed on these AS topologies by generating traffic and running the forwarding algorithm across the networks.

As with the rest of this dissertation, I do not address routing policy in this chapter. Network policy constraints are important, but the application of policy to a routing protocol based on the TZ compact routing algorithm is non-trivial, and outwith the scope of this thesis. Policy is left as future work, disscussed in Section 7.3.

6.5 Simulation Results

In this section, I present the results from simulation of the routing protocol described in Section 6.2. I do this in the following stages.

First, in Section 6.5.1, I present results on the message overheads generated by the algorithm on the AS topologies tested. All nodes log the number of messages they propagate, broken down into message types: landmark announcements, ordinary announcements, and also the number of times a naming infrastructure may need to be updated. I present detail on where the communication costs lie when using this protocol on these topologies, and how the overhead has grown over time.

Then, as in the previous chapters, I present an analysis of the stretch performance derived by simulating packet forwarding from each node to a reasonable subset of the rest of the graph, and their reverse paths (using the same test traces as used in Chapters 4 and 5; *i.e.*, 1% of all nodes, chosen with uniform probability, except from 8 November 1997 were all pairs are tested). For this part of the evaluation, presented in Section 6.5.2, I use the routing state retained by nodes in Section 6.5.1.

I also discuss the distribution of forwarding table sizes in Section 6.5.3. The distinction here between previous chapters is that landmark nodes as defined for this decentralised compact routing protocol retain state, and so I look at how much state the landmark nodes in particular retain.

Finally, I discuss the distribution of the number of landmarks that are viable for use by each AS in Section 6.5.4.

6.5.1 Communication Costs

This section evaluates the message overhead incurred by the routing protocol defined in Section 6.2. I break down message overhead into the different message types to better understand how much overhead is incurred by landmarks and how much by ordinary nodes.

When the simulation commences, all nodes have no knowledge of the network aside from their neighbours; landmark nodes additionally know that they are landmarks and advertise themselves as such. The receipt of a landmark message prompts an updated landmark distance measure from all other nodes, and the routing updates propagate according to the algorithms defined in Section 6.2.

It is difficult to perform a direct comparison between this routing protocol and real BGP update overhead on the Internet, because BGP updates are influenced heavily by the policies enforced by ASes, but here I assume undirected, unweighted links between all non-stub nodes. This implies that where BGP update propagation might be heavily scoped, the updates

in the simulation of the compact routing protocol will propagate freely across all non-stub paths. These message overheads therefore indicate worst-case overheads. Further, by using the topologies used throughout this dissertation, it is possible to indicate the relative growth of the routing protocol overhead as the network has grown.

Figure 6.2 shows the total number of messages sent, and a breakdown of the types of messages sent into landmark advertisements, and regular advertisements. This figure also shows an indication of how frequently the supporting naming infrastructure would have to be updated. The "naming traffic" line on these plots indicates the frequency with which nodes would have to update a naming infrastructure, and not the number of messages required *within* the naming infrastructure to achieve consistency. The other measures are absolute.

The volume of landmark updates has increased linearly with respect to the size of the landmark set as the network has grown (*i.e.*, sublinearly with respect to the size of the network). As the landmark set has grown slowly over time, and landmark advertisements are not scoped like advertisements from ordinary nodes, it is reasonable to expect this growth from landmark advertisements.

The volume of updates from ordinary nodes, however, is considerably more variable. The variability is a shortcoming of the network latency model used for these simulations rather than the protocol itself. The spikes in routing update overhead are caused by advertisements propagating from an origin, then being removed if the origin subsequently receives a landmark advertisement along a shorter path and updates its *landmark_distance* variable appropriately. As stated in Section 6.4, there is a very short minimum advertisement interval timer at each node of 2ms. Modifications to update timers ought to alleviate the variability in update overhead, as might the development of a graph latency model with finer granularity. Further, the undirected, policy-free graph model implies that advertisements that arrive at high-degree nodes will suffer an amplification effect, especially if the advertisement is later updated.

Despite the variability, recall that the landmark sets are tiny compared to the size of the full network (fewer than 1% of the nodes). Thus, routing updates are heavily scoped, despite the apparent variability, if the landmark updates contribute significant portions of the traffic. By aggregating the traffic from all of the simulations, the landmark traffic represents 39% of the messaging overhead, and routing updates represent the remaining 61%.

To enumerate the extent to which updates are scoped, many nodes do not receive updates from most of the network. On the 8 November 1997 snapshot, some of the landmarks receive updates from a large proportion of the network (around 50% of all nodes). Similarly, a large proportion of the network (again, around 50% of all nodes) receive updates from fewer than 100 nodes (that is, about 3% of the ASes.) All nodes receive updates from more nodes than they retain in their local routing state, but still receive updates for significantly fewer nodes

than the full network.

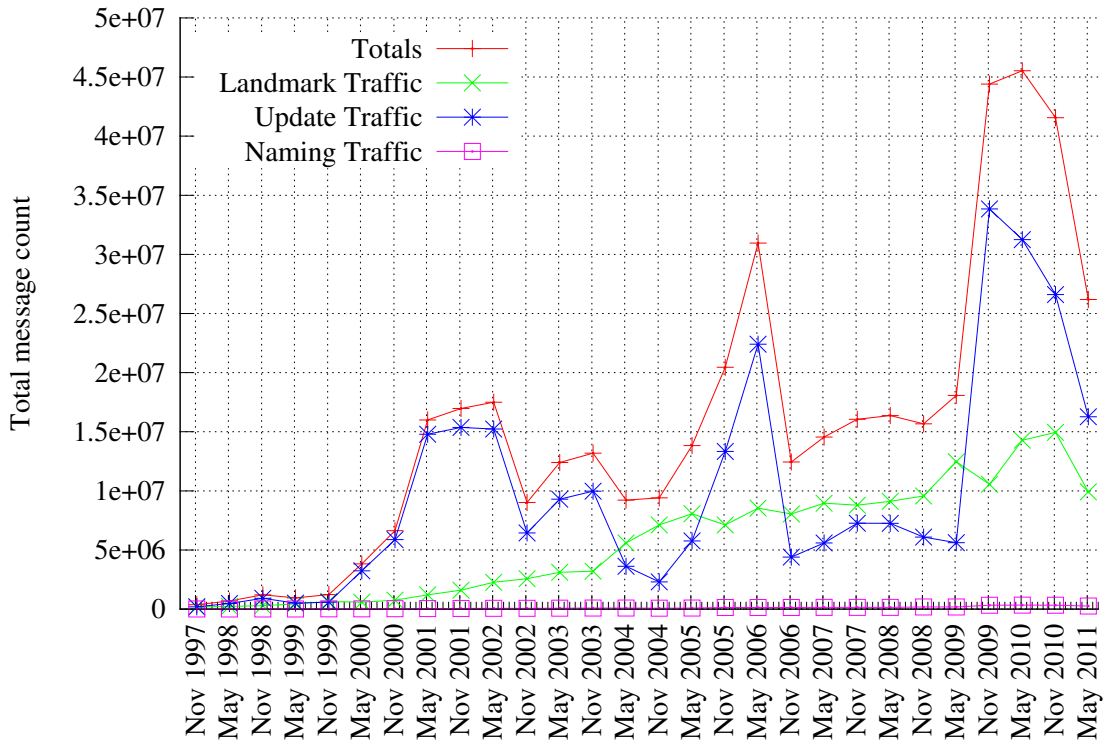
The number of times the nodes indicate that they required an update to a supporting naming infrastructure has grown slowly. These counters support the assertion that the variability in ordinary update traffic comes from variation in the arrivals of landmark announcements (*i.e.*, distant landmark advertisements arrive prior to nearby landmark advertisements); the 8 November 2010 AS topology required the most naming updates (337,570 across the full network, and thus a mean of between 9 and 10 updates per node).

Figure 6.7 (page 114) shows the cumulative message counts for each network as time series. These plots show the total number of messages sent across the network on the y-axis, with time advancing in milliseconds on the x-axis. The vast majority of all updates propagate in less than 15 minutes of simulation time on all graphs. There is a long tail of update propagation, with some graphs taking hours of simulation time to propagate all updates; this implies only that there are highly latent links in the graph model used for simulation. As noted in Section 6.4, all nodes maintain a minimum advertisement interval timer of 2ms, which keeps message propagation times low. With a low uniform minimum advertisement interval timer, advertisements propagate across the graph quickly (recall from Figure 2.3a on page 13 that the mean AS path length is fewer than 4 AS hops and that the 99th percentile AS path length is fewer than 6 AS hops in length). In all cases here, it is the landmark announcement that prompts other nodes to update its *landmark_distance* value by sending an *update* message, and so the count of *update* messages sent across the networks tends to trail the number of *landmark_update* messages sent.

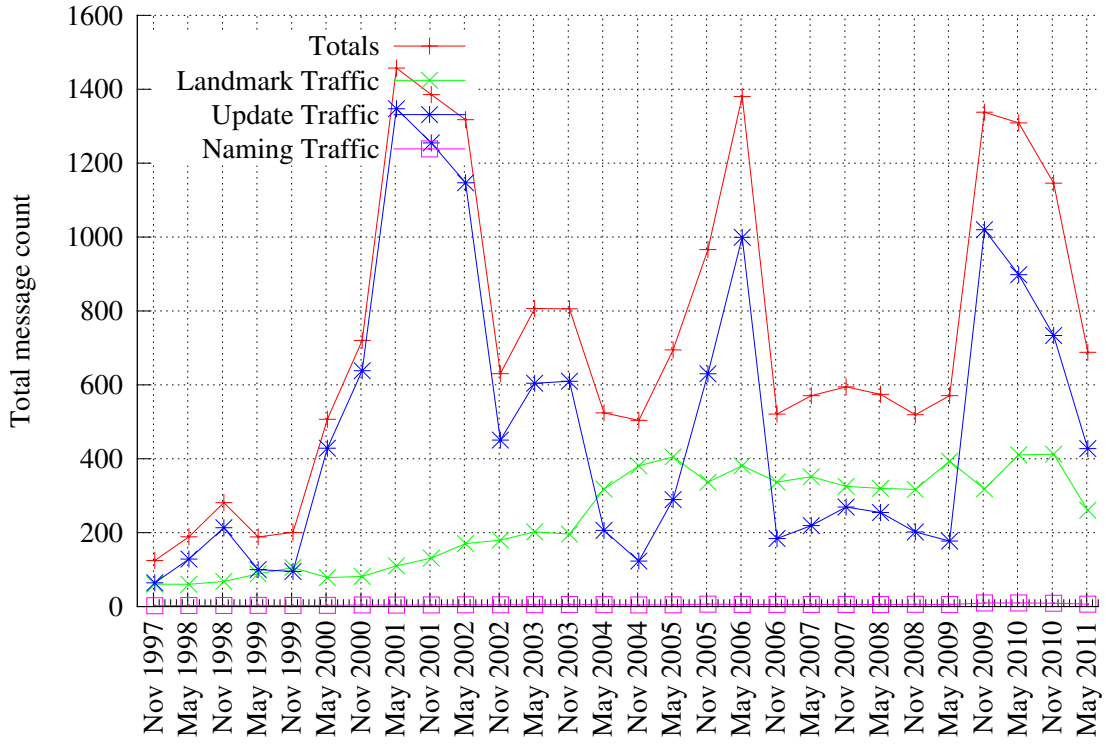
Figure 6.8 (page 116) shows the distribution of the number of messages sent by each node for each of the graphs tested. As may be expected, some nodes propagate many more messages than others, and many nodes propagate only a handful of messages. These exhibit an unusual distribution, but the long-tail is not unexpected: there are many stub nodes (approximately two thirds of all nodes) that will emit their own advertisements but not propagate advertisements from other origins, but only one third of nodes are transit nodes that will propagate routing traffic. Without additional information on how network policy affects routing updates through the network, the distributions shown here are highly influenced by node degrees.

6.5.2 Routing Stretch

As in Chapters 4 and 5, in this section I evaluate the routing stretch incurred by the routing protocol. In this instance, the primary difference is that every node will advertise via the naming system all of its nearest landmarks. Thus, the source node potentially has a set of landmark nodes to route toward, from which it selects the nearest to itself. Since the landmark set is globally visible, the source node can select the nearest of the landmarks, thus



(a) Total message counts, categorised by message type, across the whole network.



(b) Mean message overhead per-node, categorised by message type.

Figure 6.2: Indication of message overhead.

reducing the length of the path taken by the routing system even in the worst-case of routing via the landmark without shortcutting.

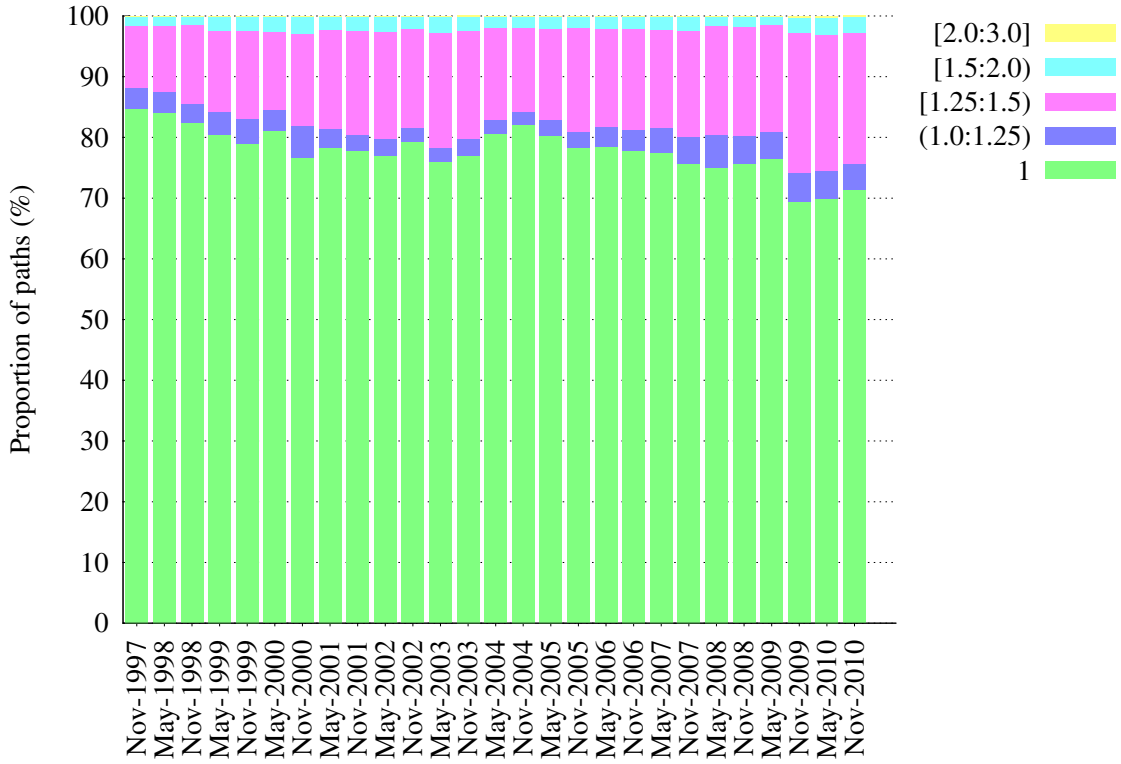
Multiplicative stretch results are shown in Figure 6.3. The resulting stacked histograms are organised into the same bins as in previous chapters, although some of the smaller bins are so small as to not be visible. Across all the snapshots tested, the mean proportion of paths that are shortest path is 97.70% (with a standard deviation of 1.06). This is a massive improvement over the 77.4% (standard deviation 3.75) of paths that were stretch 1 in the evaluation of the TZ_k landmark sets in Section 5.4.2, where full sets of valid landmarks are not exposed to source nodes. Of the remainder of the stretch values, taking the mean result for each bin across all of the snapshots evaluated, 0.44% of paths lie between multiplicative stretch 1 and 1.25; 1.62% of paths lie between multiplicative stretch 1.25 and 1.5; 0.21% of paths lie between multiplicative stretch 1.5 and 2.0, and only 0.033% of paths have multiplicative stretch greater than 2. A minuscule proportion of paths are stretch-3. For example, on the 8 November 1997 graph where I test the forwarding algorithm for all pairs (resulting in 9,180,900 pairs), only 6 paths are stretch-3.

Additive stretch results are shown in Figure 6.4. To achieve multiplicative stretch-1, 97.70% of paths obviously have no additional AS hops. Of the remainder, 2.16% have one additional AS hop (standard deviation 0.99), 0.14% have two additional AS hops (standard deviation 0.1), 0.006% have three additional AS hops (standard deviation 0.010), and 0.0008% have four additional AS hops (standard deviation 0.0001). For reference, the 8 November 1997 contained only 35 paths with this maximum of four additional AS hops.

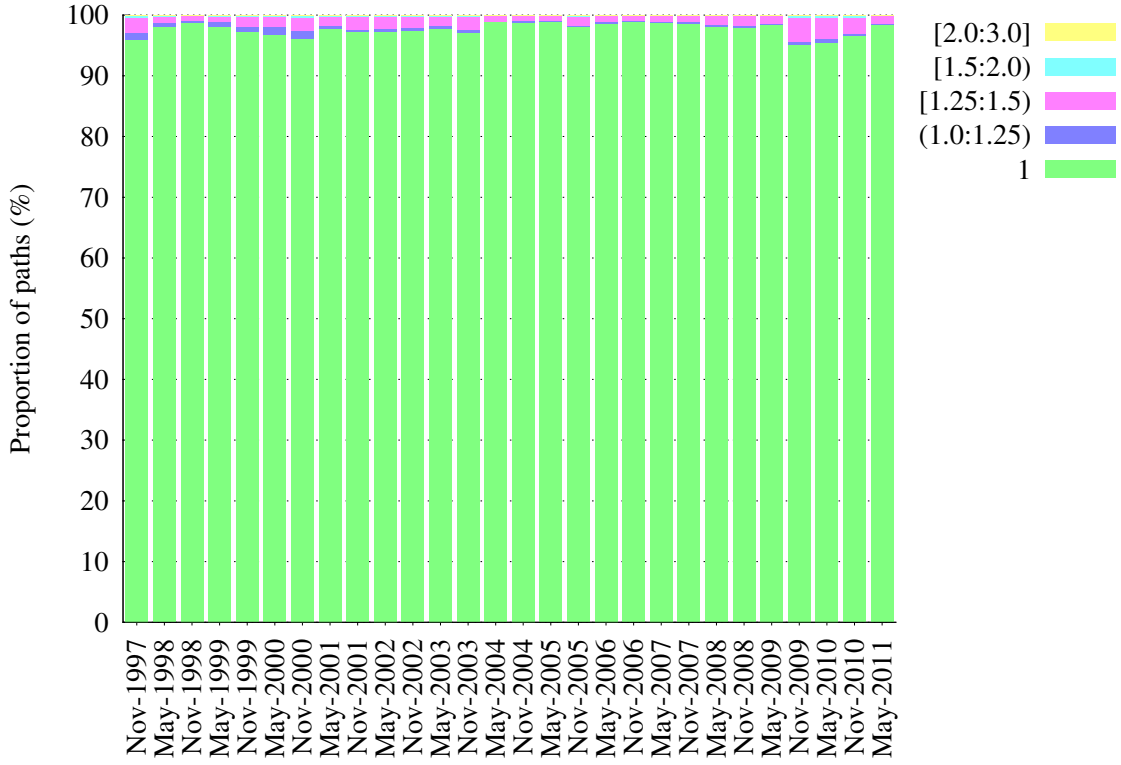
The stretch results shown in this section are noticeably better than those presented in previous chapters. There are two reasons for the improvement: first, the AS graphs used in this section are more accurate, with stub nodes appropriately identified as nodes that will not forward routing updates, thus constraining the path selection; second, all source nodes are able to select the landmark to use from a set of valid landmarks equidistant from the destination.

It was possible that the strong stretch performance is derived from the modification to the graph structure, with the introduction of stub nodes that do not offer transit to traffic from other nodes, rather than from exposing the full landmark set to source nodes. To compare the two effects, I ran the forwarding algorithm for all pairs on the 8 November 1997 AS snapshot.

The mean shortest path length on this snapshot with stub nodes in place is 3.768, and the mean path length with the compact routing protocol applied is 3.813 (a mean path stretch of 1.012). For comparison, the mean shortest path length on this snapshot if all nodes offer transit is 3.763, and the mean path length with the compact routing protocol applied is 3.810 (also a mean path stretch of 1.012). For reference, the mean path length on this graph for TZ_k compact routing, which does not expose the full set of valid landmarks to source nodes,

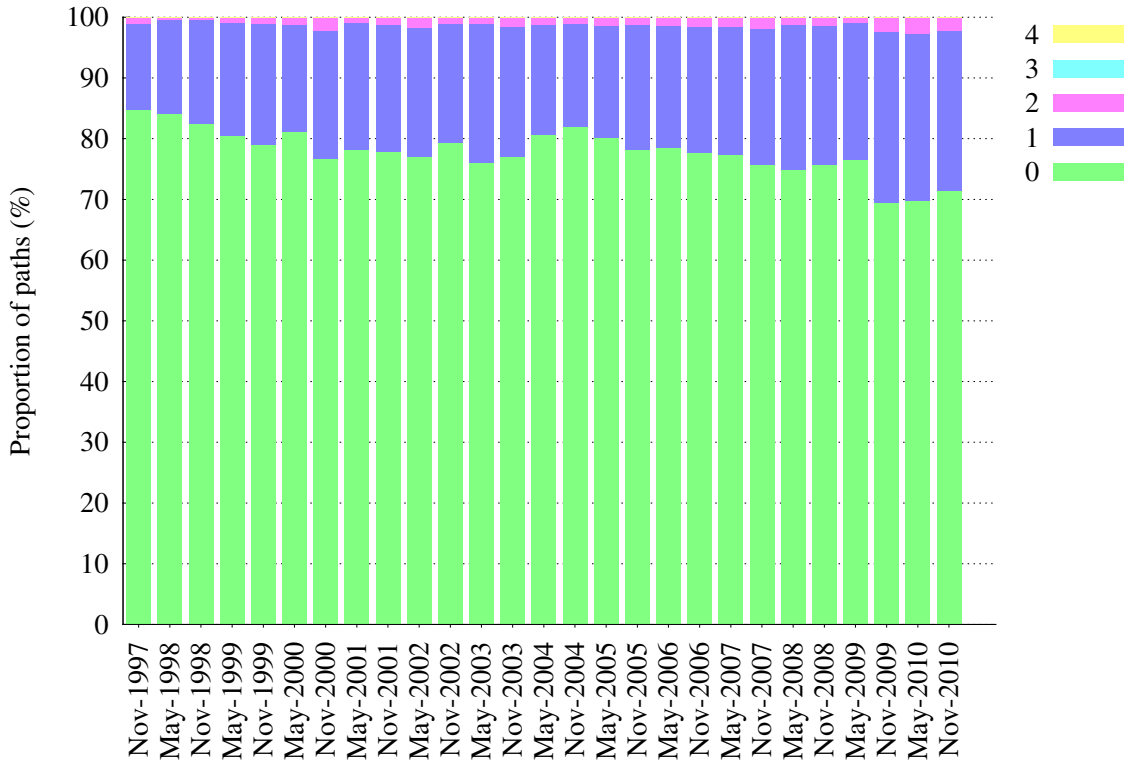


(a) Multiplicative stretches for TZ_k.



(b) Multiplicative stretches for the compact routing protocol.

Figure 6.3: Multiplicative path stretch for the TZ_k algorithm and the compact routing protocol.



(a) Additive stretches for TZ_k.

(b) Additive stretches for the compact routing protocol.

Figure 6.4: Additive path stretches for the TZ_k algorithm and the compact routing protocol.

is 3.928 (a mean path stretch of 1.044).

These results indicate that the effect of the modification to the graph model is minor as expected, and that the primary factor in improving path stretch comes from exposing the full set of valid landmark nodes to source nodes.

6.5.3 Routing Table Sizes

Figure 6.5a shows the distribution of means for routing table sizes, with 1^{st} and 99^{th} percentiles, maxima and minima. As expected, forwarding state is tiny in relation to the size of the network, and grows sublinearly with respect to the number of nodes in the network. These table sizes are equivalent to those shown in Figure 5.11a (page 77), apart from the landmark nodes, some of which have considerably larger routing tables.

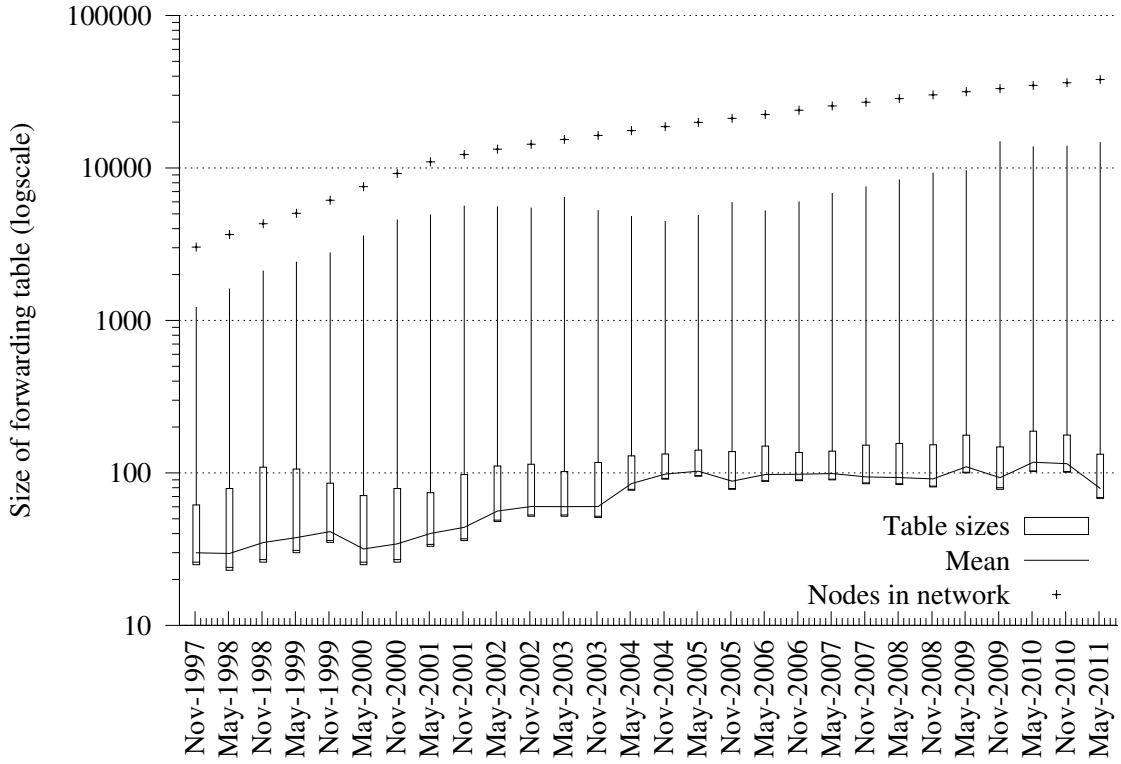
Figure 6.5b shows the distribution of means, with 1^{st} and 99^{th} percentiles, maxima and minima, for the landmark nodes on each of these graphs. Each landmark l in these experiments held state for all nodes that could use elect to use l as a landmark. The landmarks are not holding full state, and thus do not form a DFZ. There is wide variation in the size of tables, and some tables approach the full graph. Note that the k-cores algorithm does not necessarily select by node degree alone, and takes structure into account. Some of the landmark nodes will be structurally important, binding together highly connected regions, but do not themselves have many connections and so do not store so many advertisements as the highly connected landmarks.

6.5.4 Landmark Usage

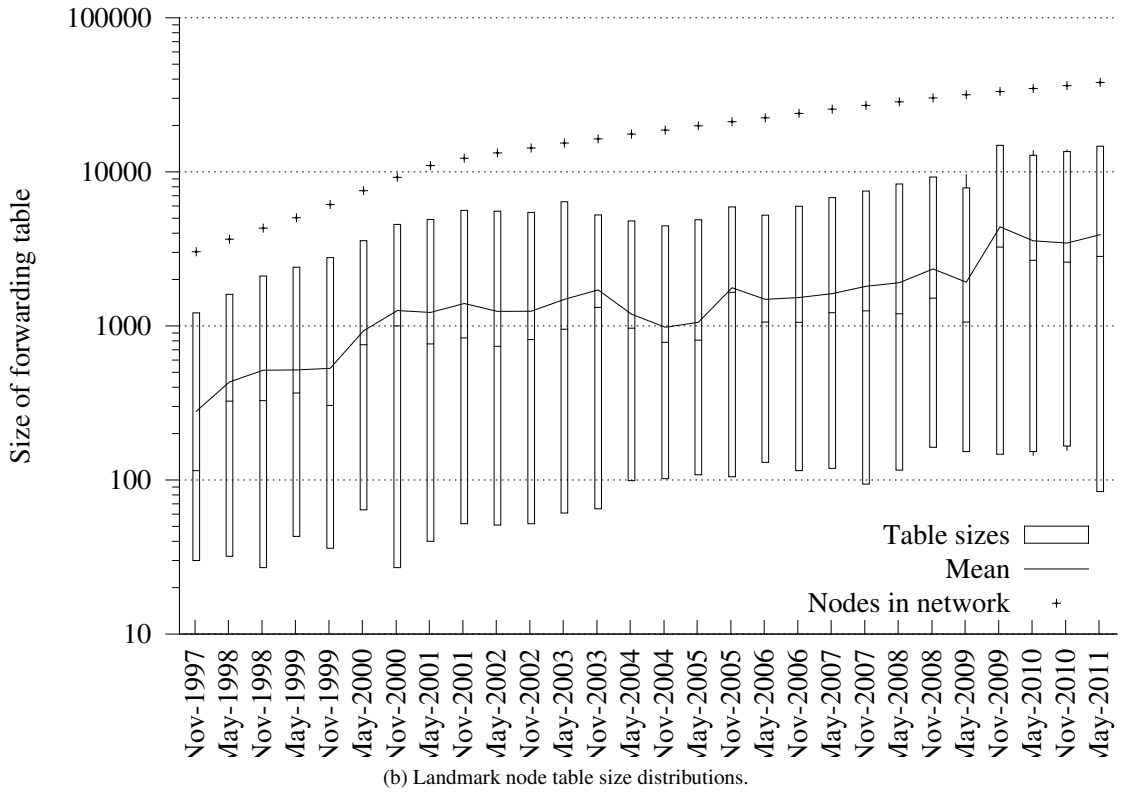
As described in Section 6.1.3, a node u may have many valid landmarks drawn from the landmark set A, each of distance $D(u, A)$.

Distributions of the size of the set of *valid* landmarks is shown in Figure 6.6, for a range of the AS topologies on which the routing protocol was tested. The data shows that although many ASes have only one valid landmark, many others have more. The ratio has changed over time: on 8 November 1997, 1,768 ASes (54.35%) had one landmark and 1,262 ASes (41.65%) had multiple; on 8 May 2011, 12,493 ASes (32.93%) had one landmark and 25,450 ASes (67.07%) had multiple.

The exposure of multiple landmarks via supporting infrastructure enables source nodes to choose the nearest landmark and minimise path lengths when the destination is not known at the source. As shown in Section 6.5.2, this brings the proportion of stretch 1 (shortest) paths to 97.70% of all paths examined. Additionally, exposing multiple landmarks potentially



(a) Table sizes generated by the routing protocol. Boxes show 1^{st} percentile, median, 99^{th} percentile. The upper extremity indicates the maximum table size. The minima overlap with the 1^{st} percentile in all cases.



(b) Landmark node table size distributions.

Figure 6.5: Growth of tables generated by the routing protocol.

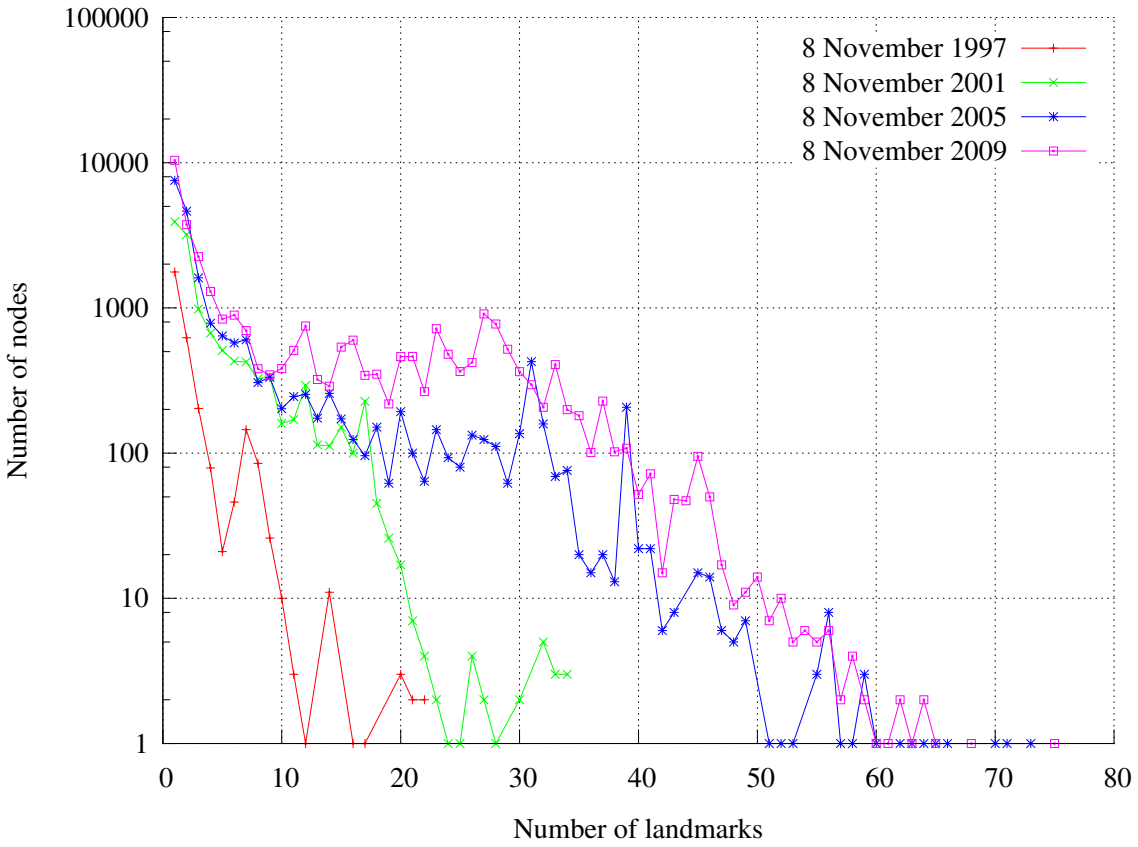


Figure 6.6: Distribution of the number of valid landmarks per-AS.

provides increased reliability in a dynamic network, but this is less of a concern having already shown that in Chapter 5 this landmark set is extremely stable.

6.5.5 Summary of Results

This section has presented results from simulating the distributed TZ protocol described in Section 6.2 on the same Internet AS topologies used in Chapters 4 and 5.

Section 6.5.1 showed communication overheads. Of note here is that the advertisements from landmark nodes often contribute about one third of all messaging overhead (39% across all snapshots tested), and landmark advertisement traffic has grown linearly with respect to the size of the landmark set. It is important to consider here that the size of the landmark set is tiny compared to the full graph, and to account for such a high proportion of network traffic implies that routing update scoping is highly effective at reducing routing state from the rest of the network.

Section 6.5.2 covered path stretches. The path stretches exhibited here are considerably better than indicated in Chapters 4 and 5, due to the exposure of multiplie landmarks per destination. By exposing the set of potential landmarks for a destination to the source nodes

for every path examined, shortest path routing is achieved in the vast majority of cases. This simple tweak provides a mean path stretch of 1.012 on tiny forwarding tables.

Section 6.5.3 discussed forwarding table growth. As shown in previous chapters, the forwarding table requirements for the TZ compact routing algorithm are tiny relative to the full graphs, and that is obviously also the case here. The median table size being around 100 entries.

Finally, Section 6.5.4 discussed the variation in the size of sets of potential landmarks per destination. This information is important: nodes who have multiple landmarks permit alternative paths from a source node, and offering multiple landmarks to a source node allows appropriate path selection to reduce path stretch across the network without any increase in the amount of state kept in the routing system.

These results are extremely positive, and improve on the (already promising) results from Chapters 4 and 5.

6.6 Overlapping Prefixes

The evaluation of compact routing algorithms so far has focussed on the stretch properties using unique advertisements from each node, *i.e.*, unique AS numbers. In BGP, however, networks advertise address blocks which can overlap; the current Internet addressing architecture, CIDR, allows variable length prefixes to be advertised. One network can advertise an address block which is a proper subset of an address block advertised by another network. Importantly, longest-prefix matching encourages smaller blocks to be advertised as distinct from a larger block to load balance flows across data centres, to advertise more specific destinations, or to multihome a network through different upstream providers. For an inter-domain routing protocol based on the TZ compact routing algorithm to be useful for the inter-domain Internet, it must also be able to correctly handle prefixes of varying lengths. The advertised IPv4 address space was discussed in Section 2.4.

Such variable-length prefixes are potentially incompatible with a compact routing protocol. The aim of using a compact routing protocol is to limit visibility of advertisements, but the use of variable length prefixes on the Internet relies on global visibility to allow all ASes to make routing decisions according to the longest prefix match.

To resolve this issue, there are two solutions: require that small, fixed size address blocks be advertised, and that networks only advertise the space they are using and not subsets they have sold; or, define a more complex routing state algorithm that allows the share of more specific (longer) prefixes.

The first solution simplifies the problem considerably. It would be possible to accept current routing advertisements and map those into the smaller fixed-size blocks for the compact routing protocol, though larger blocks for which a subset is advertised elsewhere are still a problem.

The second solution, modifying the routing algorithm, is feasible. The key issue with longest-prefix matching is that, in the presence of a short prefix for which a longer prefix exists elsewhere, the longer prefix must also be present such that the more-specific prefix is followed. Without being aware of variable length prefixes, the protocol outlined in this dissertation will constrain the distance that an advertisement will travel regardless of prefix length. Less specific prefixes are not a problem, but more specific prefixes must propagate across any space where a less specific prefix has been advertised. That is, given an advertisement with prefix length l, a node must be aware of all prefixes longer than l.

Variable-length prefixes are, in the first instance, considered useful by the operating community, and have real value. In the second instance, they can provide a mechanism for a direct comparison between the current BGP system and these TZ simulations by using real BGP UPDATE feeds hosted on Route Views.

Given a routing architecture that operates as described in Chapter 6, with the set of landmark nodes forming a default-free zone to enable correct routing even in the absence of landmark information, some additional logic is required to handle variable length, potentially overlapping prefixes. In the presence of of overlapping prefixes, the logic relies on the following:

- If an advertisement *is not* a subset of any other advertisement, then it need not be advertised beyond its natural horizon as defined in Section 6.1.2.
- If an advertisement *is* a subset of another advertisement, then it *must* be advertised beyond its natural horizon. In particular, it must be propagated to all nodes who hold the less specific advertisement.

That is, treating the landmark set as the DFZ is required, as it absorbs all advertisements from all nodes. The DFZ has enough information to decide when an advertisement should be propagated beyond its boundary. The propagation mechanism must be aware of variable-length prefixes to enable it to decide whether to propagate on the prefix length *prior to* the distance calculation.

This is a feasible addition to a routing protocol, but it immediately violates the constraints of the TZ compact routing algorithm at more nodes than just the landmark set. However, given that the routing tables are tiny when using the TZ compact routing algorithm, the state savings may still be significant when compared to full BGP tables. Investigation of the application of this algorithm is left as future work.

6.7 Summary

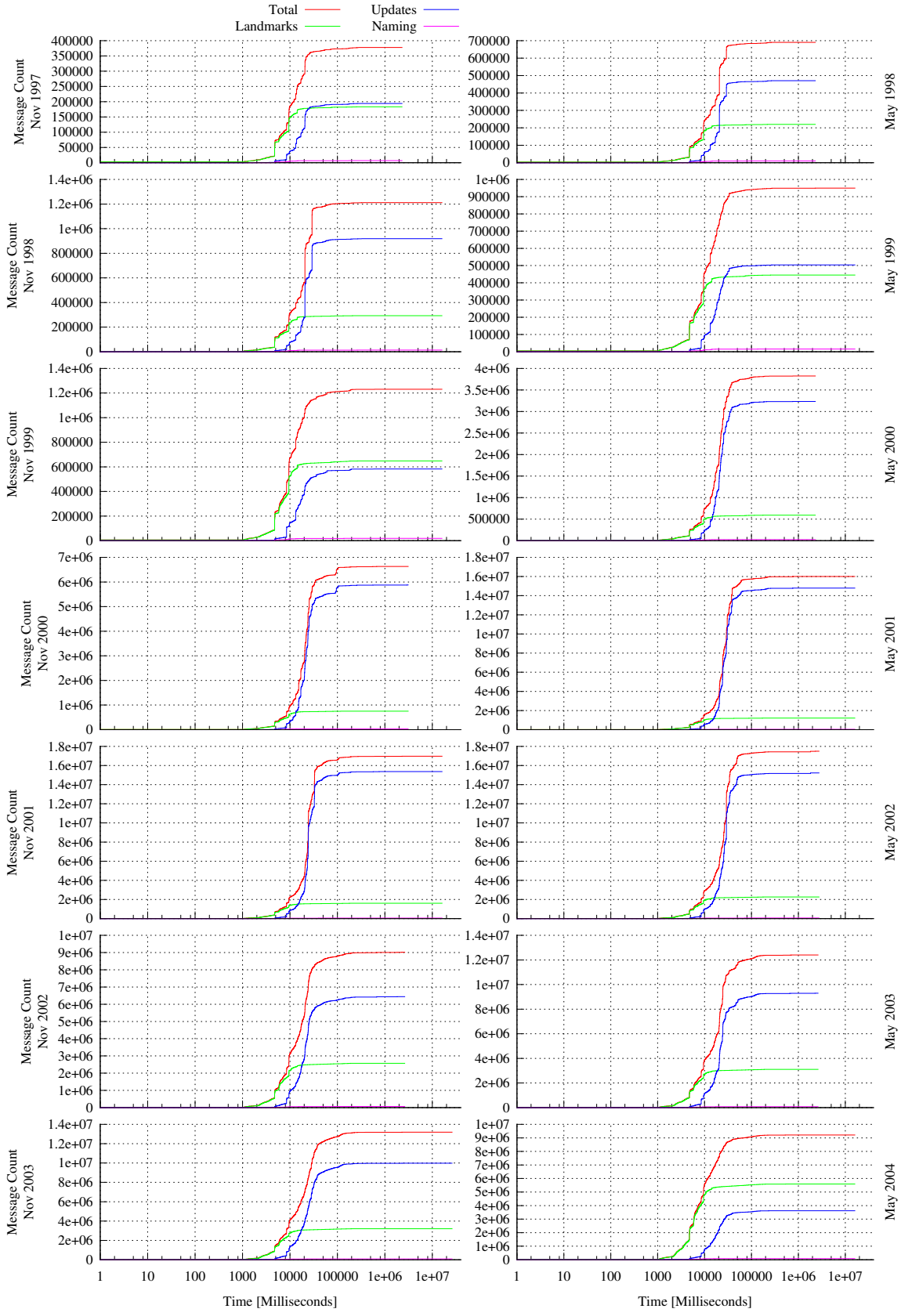
This chapter has presented a hard-state path-vector routing protocol based on the TZ compact routing algorithm. The protocol is intended for dynamic graphs, and was tested by simulation on the same Internet AS topologies as used in previous chapters. It uses the same clustering equations defined in [80] to define how the required routing state is propagated across the network.

In Section 6.1, I discussed the design rationale, including some of the state trade-offs inherent in performing landmark routing and indicating the additional state required in routing updates to facilitate the implementation of a compact routing protocol. Sections 6.2 and 6.3 described the protocol in considerably more detail, including important features such as how nodes react to the addition or removal or a node, a landmark, or a link.

In Section 6.4 I presented the experimental design, and the details of the simulation environment, and Section 6.5 demonstrates the results from simulation of the protocol on AS graphs.

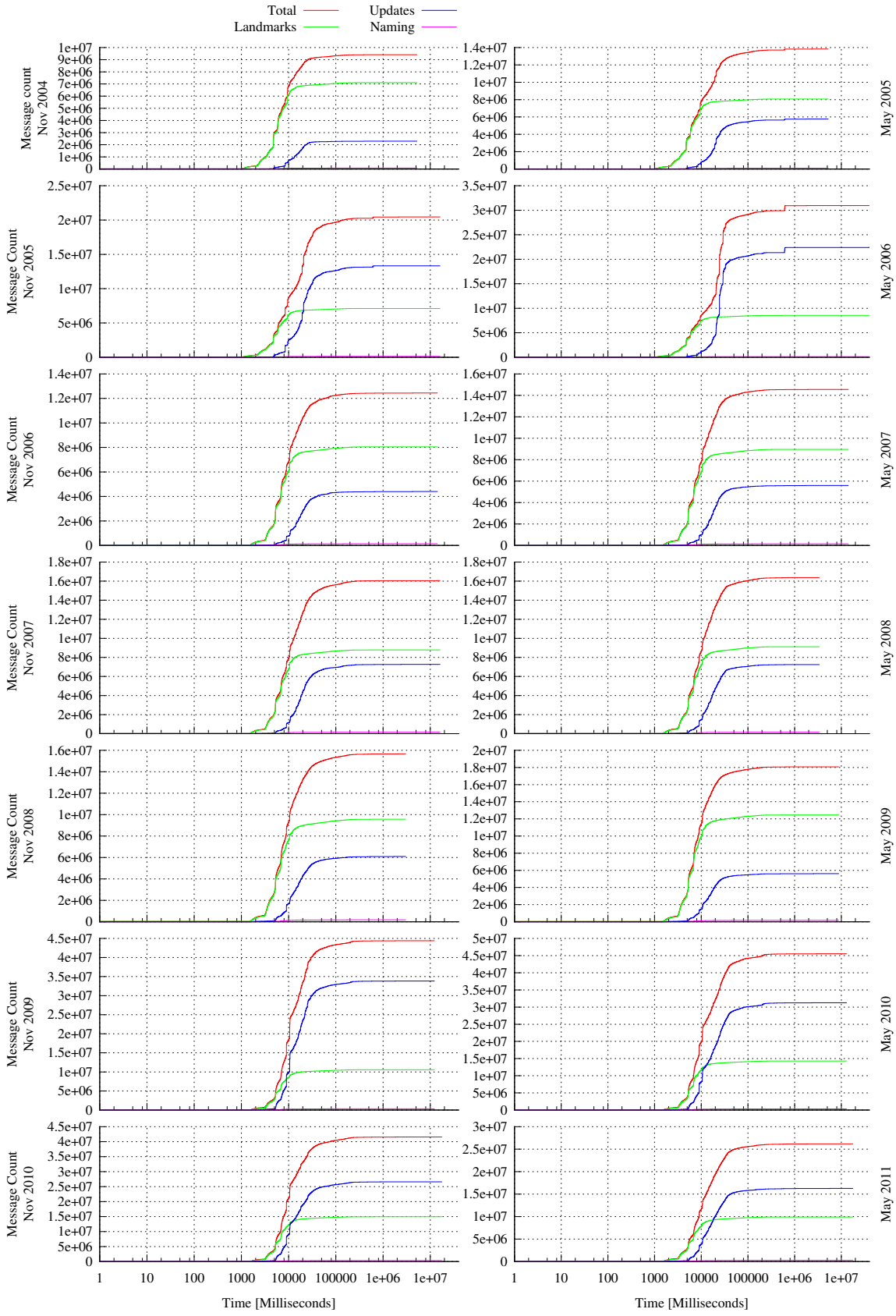
The simulation results indicate multiple useful properties: the presence of landmarks in the network helps constrain the flow of routing state, to the extent that 39% of network traffic is landmark advertisement traffic despite the landmark sets accounting for less than 1% of the nodes in the network. The routing tables generated by the protocol are tiny, as small as covered in Chapters 4 and 5. Additionally, by exposing the full set of potential landmarks for each destination to all source nodes, the mean path stretch is reduced considerably, with the vast majority of all paths tested being shortest paths (97.56% of all paths exhibit no stretch, representing an average multiplicative path stretch of 1.00766 across all paths tested).

These results indicate the suitability of a compact routing protocol as a long-term scalable solution for Internet routing. Achieving shortest path routing on 97.56% of all paths with tables that scale sublinearly with respect to the size of the network is extremely strong performance given the tiny amounts of routing state required at most nodes.



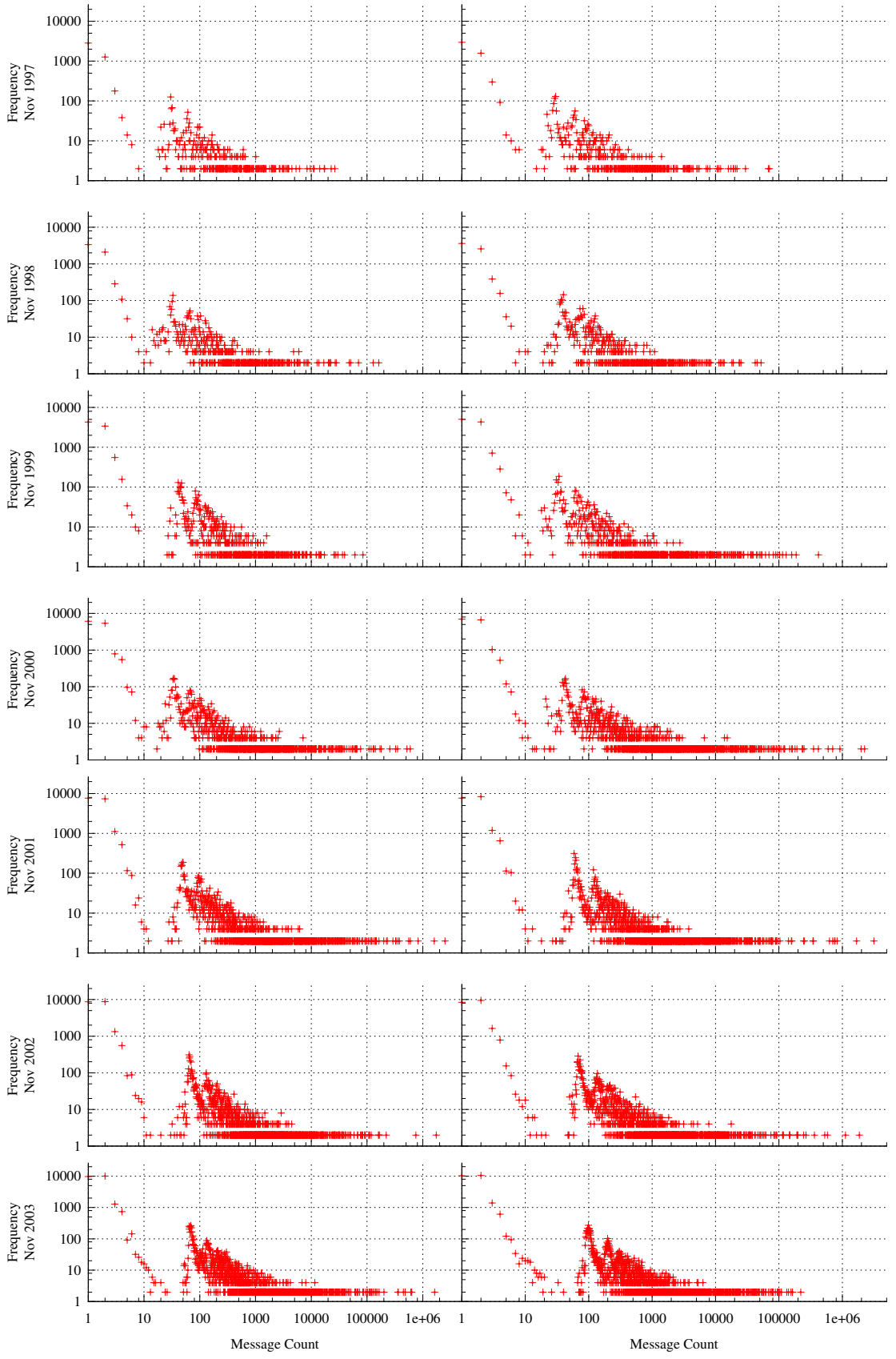
(a)

Figure 6.7: Protocol message overheads (November 1997 – May 2004).



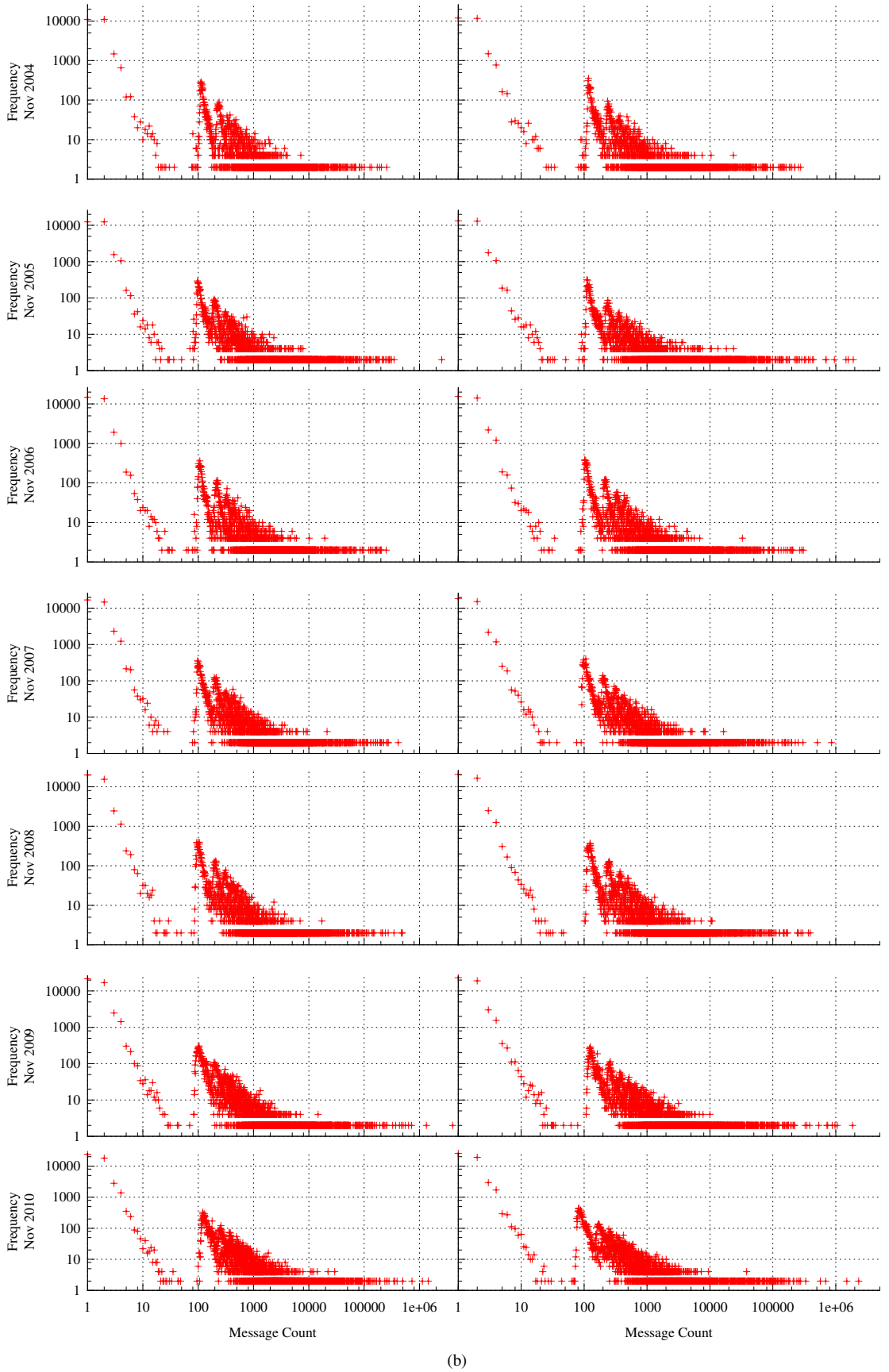
(b)

Figure 6.7: Protocol message overheads (November 2004 – May 2011).



(a)

Figure 6.8: Protocol message distributions. (November 1997 – May 2004)



(b)

Figure 6.8: Protocol message distributions. (November 2004 – May 2011)

Chapter 7

Conclusions & Future Work

The Internet architecture is designed so that complex functionality resides on the end hosts, and the network handles the forwarding of packetised data between those end hosts. That is, the desktop computers and servers attached to the network contain the complex software required to operate the wealth of Internet applications in use today, and the network retains reachability information on how to reach all endpoints. The inter-domain network achieves reachability by exchanging routing state for the address space advertised by all the networks that are connected together. This implies large forwarding tables at routers, each describing how to forward packets toward any given destination.

As discussed in Chapter 2, the forwarding hardware required to maintain large inter-domain forwarding tables ultimately cannot scale as the network continues to grow, and no satisfactory solution offering long-term scalability has been proposed. However, there is a rich field of routing research known as *compact routing* that investigates the bounds that can be placed on routing state against the maximum deviation on path length from the theoretical shortest path.

In this dissertation, I have focussed on compact routing as a promising algorithmic solution for the long-term scalablity of the Internet's inter-domain routing system. In short, the best-known bounds for compact routing on general graphs imply paths maximally three times longer than the shortest possible (multiplicative path stretch 3) with sublinear routing table growth.

This chapter is structured as follows:

Section 7.1: revisits the thesis statement, and discusses how this dissertation has addressed the assertion made in the statement.

Section 7.2: discusses the contributions presented in this dissertation.

Section 7.3: discusses directions for future work, including improved Internet graph models, and the effects of policy and traffic engineering on compact routing.

Section 7.4: summarises this chapter and concludes the dissertation.

7.1 Thesis Statement

The thesis statement was stated in Section 1.1. I repeat it here for reference:

> I assert that compact routing algorithms provide a potential solution on which a future routing architecture can be built, a solution which offers near-shortest-path routing performance with routing tables that grow sublinearly in the vast majority of routers. I will demonstrate this by:
>
> - First, performing a longitudinal study of two compact routing algorithms (the Brady-Cowen compact routing algorithm, and the Thorup-Zwick compact routing algorithm) on static Internet topologies derived from real-world data, assessing them for path lengths and routing table sizes to show that routing tables are small and path stretch is bounded.
> - Second, by evaluating the stability at the heart of the inter-domain Internet by applying the k-cores graph decomposition algorithm, and showing that this stable set is beneficial for a deployment of a routing protocol based on compact routing by also considering the geographical and topological suitability of the stable set.
> - Third, by defining and evaluating a decentralised network protocol that uses compact routing principles, and is suitable for use on dynamic networks.
>
> These algorithms aim to reduce forwarding state at all nodes, and in order to do so they cannot guarantee shortest path routing. Instead, they provide an upper-bound on the increase in path length relative to the shortest possible path. Thus, evaluation of these algorithms is primarily in terms of table sizes and path lengths.

To back up the assertion that compact routing algorithms are a valid candidate for long-term inter-domain routing scalability, I did the following.

Chapter 4 presented a longitudinal study of the Thorup-Zwick (TZ) compact routing algorithm and the Brady-Cowen (BC) compact routing algorithm on Internet AS topologies

between November 1997 and May 2011, to determine the effect of network growth on the path stretch performance achieved by the algorithms. This chapter also ran multiple experiments for the TZ compact routing algorithm for various AS snapshots spanning the full dataset to eliminate the pseudo-random element of its landmark selection from affecting the final results.

I presented results that show the stretch performance of the TZ compact routing algorithm has been remarkably stable throughout the growth of the Internet AS graph during this time period. The results support the assertion that near-shortest-path routing is attainable on these topologies using compact routing algorithms to generate tiny routing state. In Chapter 4, I also highlighted some of the algorithm's primary problems for application in real graphs: that its landmark selection algorithm chooses nodes pseudo-randomly, and that it is highly centralised and requires a full view of the topology.

In Chapter 5, I evaluated a central set of nodes in AS topologies by applying the k-cores graph decomposition algorithm. Given an input graph, the k-cores decomposition algorithm derives layers of nodes of increasing cohesion, ultimately revealing the k_{max}-core which is centrally located and highly connected. It is less computationally expensive to compute than locating the largest cliques in a network or finding the nodes most centrally located by distance. The algorithm identifies structurally cohesive regions, and so is more suitable than simply selecting the nodes of highest degree.

I presented results in this chapter that indicate the stability of the k_{max}-core over long periods of time. I also presented the growth of the set, its distance to the rest of the network, and the geographical locations of the ASes selected. Having asserted that the cores revealed by this algorithm meet suitable criteria for a landmark set for compact routing in a globally distributed, dynamic network, I defined a modified landmark selection algorithm for TZ which I refer to as TZ$_k$. TZ$_k$ uses the k-cores decomposition as its input. I showed that on the vast majority of graphs tested, the k_{max}-core is sufficient as a landmark set according to the constraints defined by the TZ algorithm in [80]. I applied these landmark sets to the same AS topologies used in Chapter 4. The results support the assertion that a routing architecture based on compact routing is feasible for long-term Internet routing scalability by showing that the path stretch performance of TZ compact routing with these landmark sets is as consistently strong as it was with the pseudo-random landmark selection, but with the distinct advantage that the landmark set is more stable, and thus more suitable for a dynamic network.

Next, in Chapter 6, I presented a routing protocol based on the TZ compact routing algorithm that operates as a scoped path vector protocol similar to BGP that shares the additional state required for the scoping rules defined by the TZ compact routing algorithm (namely, the distance between the origin of an advert and its nearest landmarks).

By defining this protocol, I presented results that indicate the routing overhead associated with the protocol on AS topologies. I showed that, although the graph model reveals variable behaviour in the routing update messages, that updates are heavily constrained to the extent that routing advertisements from landmark nodes are a primary contributor to messaging overhead. This chapter supports the assertion that a compact routing architecture is feasible by showing that the compact routing protocol heavily scopes routing updates, and additionally by showing that exposing the full set of valid landmarks to use for any destination massively improves path stretch while retaining tiny routing tables. Shortest path routing is achieved in 97.70% of all paths tested, while routing tables normally contain around 100 entries on networks containing over 35,000 nodes.

7.2 Contributions

The contributions of this thesis are as follows:

A longitudinal study of two compact routing algorithms on AS graphs. This work examined the routing stretch and forwarding table sizes of two compact routing algorithms on Internet AS topologies ranging from November 1997 to May 2011. The Thorup-Zwick (TZ) and the Brady-Cowen (BC) compact routing algorithms show different routing state requirements but also different levels of stability with respect to routing stretch; TZ compact routing provides considerably more consistent path stretch as the graph has grown.

A longitudinal study of the k-cores decomposition on AS graphs. This work examined the k-cores decomposition of the range of AS topologies studied in this dissertation. The k_{max}-core of all the graphs represent a highly stable region within the network that is structurally important, centrally located, and geographically diverse. These properties are useful if we are to consider the deployment of a compact routing protocol based on the TZ compact routing algorithm.

The definition of an alternative landmark selection algorithm, for the TZ compact routing algorithm called TZ_k. Harnessing the stability of the most cohesive k-shells, I define TZ_k. The benefits of TZ_k are such that the landmarks are placed within the most topologically important region of the network, which I have already shown is highly stable. The introduction of TZ_k's landmark selection also means that this selection process is deterministic, unlike the original TZ landmark selection algorithm.

The definition of a routing protocol based on TZ compact routing. Accepting that there is a stable set of nodes in the network that are suitable landmarks, I defined a decentralised protocol based on TZ compact routing. I tested the protocol on the same

AS topologies as in preceding chapters, and found that the algorithm achieves better stretch performance than the standard TZ compact routing algorithm by exposing multiple landmarks per destination. I provide detail on approximate convergence time, and the number of control messages required on AS topologies annotated with intra-AS and inter-AS latencies.

7.3 Future Work

This dissertation presents a promising direction for a routing architecture for the future Internet, but in itself leaves open various unsolved problems requiring further work. In this section, I describe some of these future directions.

7.3.1 Modelling on Dynamic Networks

The results presented in Chapter 6 indicate messaging overhead to apply the routing protocol to a static graph, but stops short of analysing behaviour on dynamic graphs. Dynamic networks introduce additional overheads for the routing protocol presented in Chapter 6. The gradual evolution of the landmark set, and the more fluid evolution of links between ASes being added and removed, and ASes themselves joining and leaving the network, introduces routing update overhead not accounted for in the analysis of static graphs presented in this dissertation.

Understanding the behaviour of the protocol in a dynamic environment is an important next step. Modelling of the dynamic Internet graph using BGP *UPDATE* feeds, as archived by Route Views alongside BGP routing tables, is a feasible approach. Throughout this dissertation, I have sought to apply compact routing algorithms to realistic topologies derived from real-world data, and modelling the dynamics of the network should be no different.

Understanding the BGP *UPDATE* feeds allows two approaches. One approach is to use the data to model an approximation of how often ASes join or leave the network, and where links are created or removed. This approach is particularly useful for modelling the network into the future, if we are willing to accept the assumption that future network growth will be similar to recent network growth. The other approach would be to use the data to directly "replay" the network between two states, and monitor the routing overhead as the network is modified. This is perhaps as close to approximating real network overhead as it is possible to get without a significantly more complex model of the AS graph, or a deployment of the protocol on a real, dynamic network. Some of the complexities of modelling real-world data are covered in [36], and the description of a model based on this data is presented in [37].

7.3.2 Improved Internet Topologies

Related to modelling dynamic Internet topologies is the development of more accurate Internet topologies, or the augmentation of existing topologies. Augmentation of BGP data with traceroute data exists as a mechanism to reveal AS paths that are not necessarily visible in solitary BGP snapshots, but traceroute data also allows the creation of a more fine-grained AS topology map. Since the traceroute data reveals router hops, and the IP addresses for many of those hops can be mapped to the ASes that originate them, it should be possible to map not only the ingress and egress points of an AS, but also the number of paths through that AS, the number of interconnections between that AS and its neighbours, and also the relative latencies of paths through each AS.

This improved model encourages a deeper understanding of the on-path latencies between any two endpoints on the Internet as packets move from network to network and, in particular, allow the modelling of diverse regions connected via transit providers (for example, paths through a large inter-continental AS may incur high latency if a trans-oceanic link is used, but equally there will be paths through the same AS within the same geographic region, implying a considerably lower traversal time.)

Modelling the diversity of path selection across the Internet from different vantage points would also be a useful addition to the topology description [112].

7.3.3 Policy Based Compact Routing

In this dissertation, I have not considered policy and how it affects routing in the network. This area is complex, in part because peering relationships and policy decisions are considered proprietary by many, and thus correctly modelling AS-level policy is difficult. The valley-free, inferred policy models discussed in Section 2.5.1 offer a simple model of the relationships between ASes, but they are difficult to evaluate against reality. Revealing settlement-free peering links extant between networks relies on having collectors at multiple vantage points. Also, the BGP data does not reveal any information about the business relationships between ASes.

The graphs evaluated in this dissertation are largely undirected graphs of links between ASes generated from the vantage points available through the Route Views project (a list of the repositories is given in Appendix A.1). In reality, many links are missing from the data and some of the visible links would only carry data between certain subsets of the AS graph, according to local policy constraints. Currently with BGP, policy is applied to the routing infrastructure to adhere to the business realities of Internet operations: some links will be preferred due to costs, some settlement-free links should only be visible to peers and cus-

tomer ASes, and some links should be advertised but retained primarily as backup links through the use of AS path pre-pending.

The contention between shortest path routing and policy-based routing is that policy prioritises factors other than path length, often emphasising other factors such as preferred providers or costs associated with link usage. Policy can be described as a local decision process that is then applied to the network, which then attempts to derive shortest paths in the presence of these constraints. However, in the presence of these constraints, convergence is not guaranteed [113].

Some recent work on handling policy with compact routing algorithms has been presented in [114], which uses the formalisms defined in [115].

7.3.4 Incremental Deployability

Related to allowing local policy decisions with a compact routing protocol is the prospect of incremental deployability. To encourage deployment, ideally a compact routing protocol must be deployable locally and offer state saving benefits without participation from other networks. However, the protocol relies on reducing visibility of the full graph. Incrementally deploying such a protocol is left as an open problem.

7.3.5 Managing IPv4/IPv6 Prefixes

One of the key hurdles is that prefixes in the current addressing architecture overlap. This was covered in Section 6.6, and requires a more complex routing algorithm. This additional complexity may manifest itself in overbearing routing updates when the advertised set of prefixes change (for example, the advertisement of a large address block could encourage a flood of smaller blocks drawn from the same address space that have been advertised by other ASes). To understand this overhead requires a detailed analysis of the use of the address space, and a model of how frequently prefixes or varying length are advertised and withdrawn.

7.4 Summary & Conclusions

This dissertation has presented an in-depth analysis of the application of the previously theoretical field of compact routing to Internet topologies. It has presented important steps forward in understanding the applicability of compact routing to the Internet. First, presenting a longitudinal study of compact routing behaviour on AS topologies validates previous claims that the algorithms may perform well on the Internet. Moving beyond this assertion,

the discovery of a set of ASes that are stable and well-placed, and are therefore a suitable candidate for a stable landmark set for the Internet is a fundamental step toward a deployable protocol: landmarks must be stable transit nodes to be useful and to minimise churn. Finally, the definition of a decentralised protocol that follows the principles of TZ compact routing moves the discussion of this work beyond discussion of a static algorithm for static graphs.

The results presented in this dissertation indicate that long-term routing scalability for the Internet is feasible by changing the *routing* architecture. The results presented in this dissertation show that compact routing is a promising candidate for a future Internet routing architecture that retains shortest path routing for the vast majority of paths while retaining only a fraction of the state.

Appendix A

Data Management

The work described in this dissertation makes use of publicly available data primarily from two repositories. This data is processed into useful graphs models for this dissertation. This appendix describes the data sources, and the processes by which it was obtained and processed.

The data sources are:

- Archived BGP tables from multiple BGP collectors, archived by the Route Views project.
- Traceroute data spanning the full advertised IPv4 address space originating from multiple locations, archived at the CAIDA Ark project.

I use these sources to derive various datasets:

Prefixes: Lists of prefixes advertised on a daily basis, derived from the aggregate of the prefixes visible to each Route Views collector on a particular date. There are generally small differences between the contents of these tables [116], so the aggregate presents a more complete snapshot of IPv4 address usage. The prefixes are used in Chapter 2, and referenced in Section 6.6.

AS paths: Each Route Views collector reveals a different set of AS paths, and so the daily aggregate of these presents a more complete view of the AS topology. The AS paths are used directly Chapter 2, and are used to derive the prefix-to-AS mapping required for Chapter 6.

AS links: Deriving the links between the ASes from the daily aggregate sets of AS paths allows the construction of daily AS graph snapshots. The AS graph snapshots are used directly Chapters 2, 4, 5, and 6.

Latency data: Round trip times to IP addresses selected from the full BGP table, and all responsive intermediate router hops, are held in the data collected by the Ark project. The traceroute probes are launched from multiple sites, and so these provide good coverage of router-level hops. It is possible to carefully map from IP addresses on to AS numbers. From there, I derive information on inter-AS and intra-AS latencies. The latency data is used in Chapter 6.

This appendix is structured as follows:

Section A.1: discusses the handling of BGP data from multiple Route Views collectors to derive the prefix lists, AS paths, and AS links data.

Section A.2: discusses the collection and handling of the Traceroute data to derive the latency data.

Section A.3: summarises this appendix.

A.1 BGP Data

This section describes the BGP data handling pipeline used to build the data used throughout this dissertation.

Full BGP tables and logged BGP updates are publicly available from Route Views repositories. Route Views repositories passively monitor the BGP routing system by peering with existing networks and collecting the full routing table as it appears from at that vantage point. The Route Views monitors do not themselves advertise any prefixes. There are currently 10 routeviews collectors holding IPv4 BGP data, plus one which is no longer operational[1], and one collector specifically for IPv6 data. This dissertation focusses on the IPv4 Internet, and so I do not use the IPv6 collector. The lifetimes for the various collectors, and their names, are shown in Figure A.1.

The data processing pipeline for the AS graph creation is shown in Figure A.2.

A.1.1 BGP Data Formats

The BGP data is available in two different formats: a human-readable format generated by Cisco routers, the output of running the "*sh ip bgp*" command on the router; and the Multi-threaded Routing Toolkit (MRT) format [117], a machine-readable format parsed by the

[1]The archive for route-views.1 (oix-route-views) has data available beyond 26-March-2008, but these contain data from route-views.2, modified to look like the Cisco human-readable data.

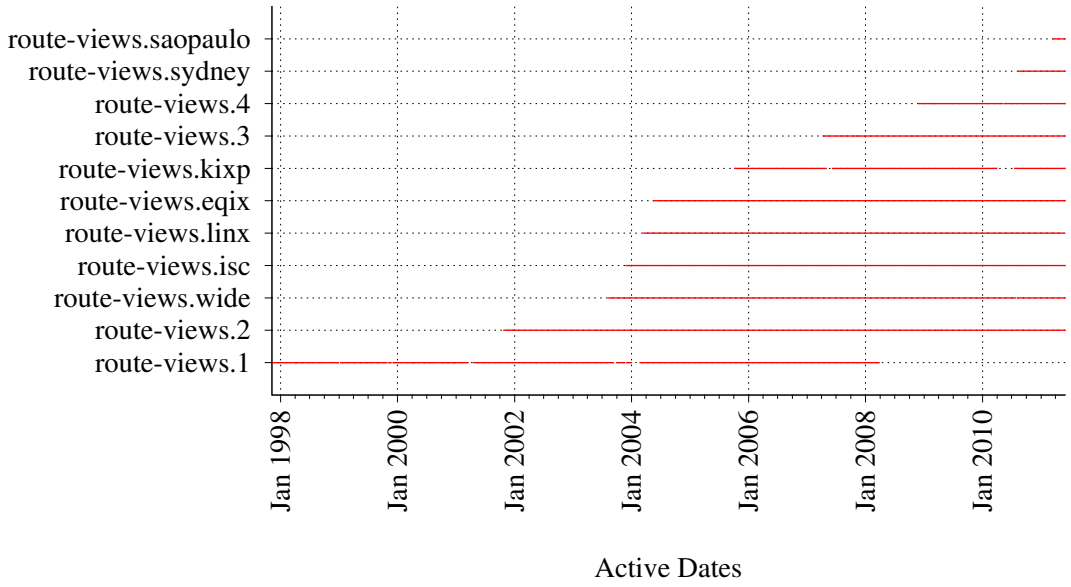


Figure A.1: Dates the various IPv4 collectors were active.

bgpdump tool[2]. BGP tables store local preferences, selected routes given several paths to choose from, and other important data for local decision making, but the tables in essence make available prefixes and the AS paths traversed to reach the collector.

Human-Readable Cisco Data

The Cisco BGP dump data format is intended to be human readable, and has changed over time. An example of the data is as follows:

```
*  130.209.0.0      194.85.4.55                    0 3277 3267 3343 25462 786 i
*                   134.222.85.45                  0 286 786 i
*                   129.250.0.171          0       0 2914 6453 786 i
*>                  195.219.96.239                 0 6453 786 i
*  130.210.0.0/17   194.85.4.55                    0 3277 3216 3549 7018 2386 i
*                   134.222.85.45                  0 286 3549 7018 2386 i
*                   129.250.0.171          1       0 2914 7018 2386 i
```

Thus, the data presents certain subtle difficulties: some lines contain paths but refer to a prefix displayed on a previous line; some lines contain old, pre-CIDR advertisements, which I translate into CIDR-prefixes; some lines contain a marker at the start of the line indicating that it is the preferred route for a given advertisement, and so forth. These must be handled carefully to produce a correct listing, mapping prefixes to paths, while ensuring that no prefixes or path hops are accidentally removed.

All Cisco data is passed through an AWK script that maintains enough state as it parses each line to handle the format. In the above example, 130.209.0.0 would be mapped to

[2] `http://nms.lcs.mit.edu/software/bgp/bgptools/`

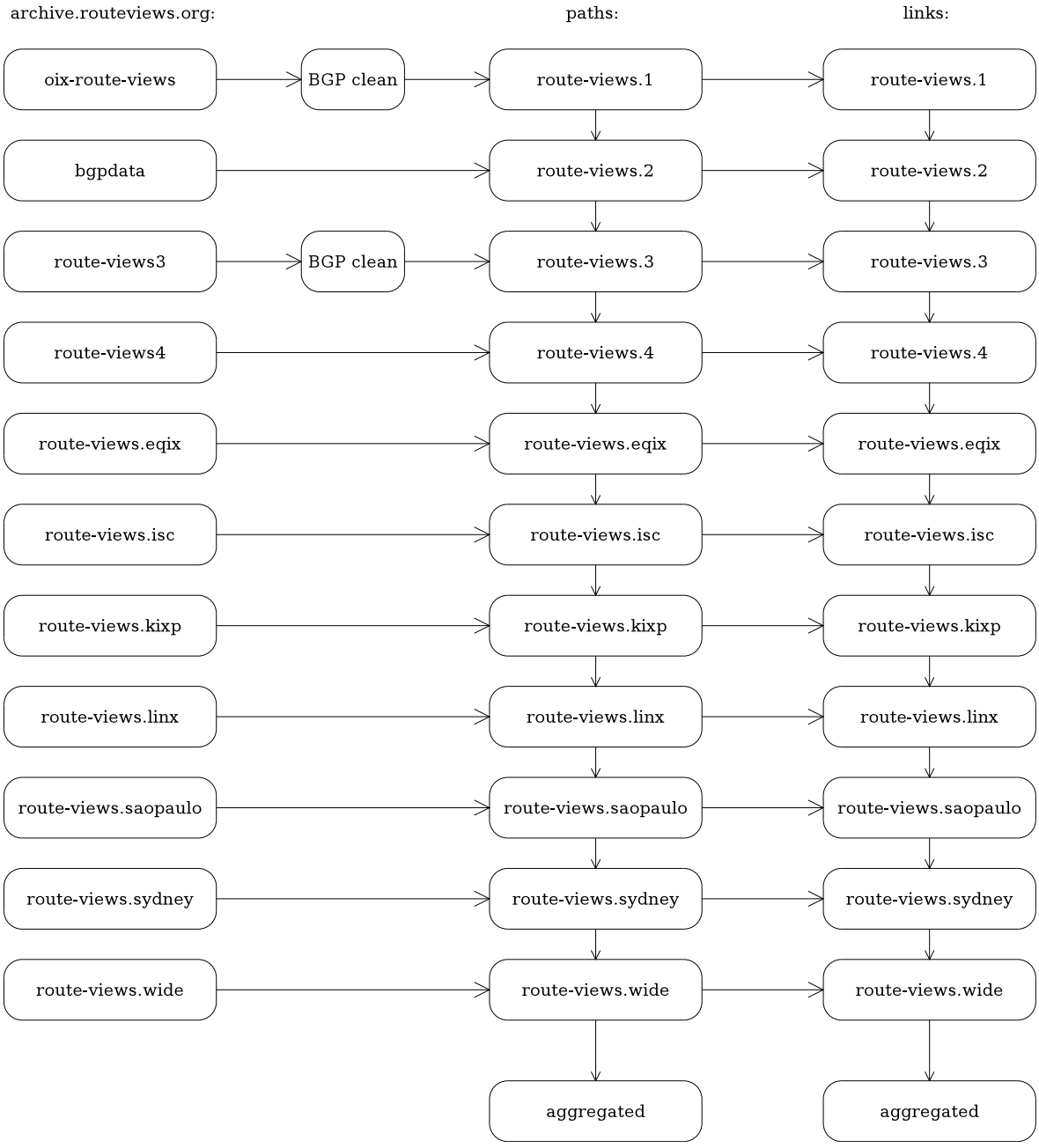


Figure A.2: Data processing pipeline.

130.209.0.0/16, and would be output next to each of the four AS paths listed. Irregular additional columns must also be treated with care.

Some classful addresses are listed (130.209.0.0 in the example above) that are mapped to their CIDR equivalent: prefixes in the range [0.0.0.0 through 126.255.255.255] are mapped to /8s; prefixes in the range [128.0.0.0 through 191.255.255.255] are mapped to /16s; prefixes in the range [192.0.0.0 through 223.255.255.255] are mapped to /24s, to match the block sizes defined in [118].

A.1.2 MRT Data

MRT format table dumps, once passed through *bgpdump*, are simpler to handle. The data is presented delimited by a known special character, as follows:

```
TABLE_DUMP|1195087535|B|147.28.7.2|3130|130.209.0.0/16|3130 1239 1299 786|IGP|147.28.7.2|0|0|3130:380|NAG||
TABLE_DUMP|1195087535|B|167.142.3.6|5056|130.209.0.0/16|5056 1239 1299 786|IGP|167.142.3.6|0|0||NAG||
TABLE_DUMP|1195087535|B|134.222.87.3|286|130.209.0.0/16|286 786|IGP|134.222.87.3|0|0|286:18 286:19 286:29 286:800 286:888 286:3044 286
TABLE_DUMP|1195087535|B|81.209.156.1|13237|130.210.0.0/17|13237 3320 7018 2386|INCOMPLETE|81.209.156.1|0|0|3320:1840 3320:2020 3320:902
TABLE_DUMP|1195087535|B|134.222.85.45|286|130.210.0.0/17|286 3549 7018 2386|INCOMPLETE|134.222.85.45|0|0|286:18 286:19 286:29 286:888 2
TABLE_DUMP|1195087535|B|216.218.252.164|6939|130.210.0.0/17|6939 3549 7018 2386|IGP|216.218.252.164|0|0||NAG||
```

From this data, it is easy to output a list of prefixes and paths using standard tools, such as *AWK* by splitting out the sixth and seventh columns.

A.1.3 Prefixes

The list of prefixes visible in tables dumped by different routers at the same time will largely contain the same prefixes, save for a small number of prefixes that differ due to BGP update propagation delays or varying policy in different regions [116].

To generate the list of prefixes observed, the prefixes from each BGP snapshot for each day are concatenated, and all duplicates are removed. All RFC 1918 [119] prefixes, and other reserved prefixes [120] are also removed (local configurations mean that some private prefixes may be visible in the table dumps). All IPv6 addresses are also removed.

Figure 2.5 (page 17) shows the total number of prefixes visible in the data that were advertised on the dates covered in this dissertation.

A.1.4 AS Paths

The list of paths visible in a table is dependent on where the router is located. By aggregating all the paths observed in the tables available from different vantage points, more valid AS

paths are revealed. In effect, more collectors makes for a more complete sampling of the AS graph.

The paths are stored as they are observed in the tables, without special processing. Paths from all available tables for a given date are concatenated and then duplicates are removed prior to storage.

A.1.5 AS Links

The AS paths stored in Section A.1.4 are used to derive neighbouring ASes. An AS path indicates that a valid path exists for traffic to a given prefix from the current location. Thus, each AS hop in the path can be considered an adjacency in a logical AS graph of networks. The AS path does not reveal the nature of the business relationship between ASes, if there is one at all, merely that some business relationship exists allowing transit of data.

Techniques based on node-degree exist to infer AS relationships [38, 43, 44, 42, 40, 45], but the output of such algorithms is difficult to evaluate.

To construct a set of accurate AS links from AS paths, there is a set of special cases that are treated as follows:

AS Sets: AS sets aggregate multiple advertisements from distinct ASes into one single advertisement for further propagation, placing an obvious *AS set* into the AS path indicating the origins of the pre-aggregated advertisements. Retaining the AS set of origins allows loop detection to function as normal. AS sets are indicated by curly brackets, the ASes in the set being contained within a comma-separated list. Sometimes, perhaps due to a configuration issue or upstream filtering, an AS set will contain only one AS. In this instance, I assume this represents an AS adjacency, and simply strip the brackets. Otherwise, I assume no adjacencies between the AS set and neighbouring ASes, because there is no way to determine the internal structure of the set. AS sets are extremely rare, appearing in fewer than 1% of paths in current BGP tables.

32-bit AS Numbers: 32-bit AS numbers can be printed in multiple formats [121]. The most common is *asdot* (for example, 65546 is 1.10, *i.e.*, $(1 * 65536) + 10)$), and for clarity I translate all these to *asplain* (*i.e.*, to 65546). (Additionally, *asdot* notation often looks like an RTT value, which makes testing the processes described later in Section A.2 difficult.) Once expanded, all 32-bit AS numbers are treated like any other AS number.

AS23456: AS23456 is a special AS number used by a router when a neighbouring router is incapable of handling a 32-bit AS number. AS23456 does not represent an actual AS. 16-bit AS numbers form a proper subset of the 32-bit AS number space (the upper

16 bits are set to zero), but 32-bit AS numbers from outwith this set do not have a direct 16-bit mapping. BGP routers with such a 32-bit number, when communicating with another BGP router that cannot handle 32-bit numbers, identify themselves as AS23456 [122].

AS23456 is ignored when building links from path data. The two ASes on either side of AS23456 are not assumed to be directly connected. To build AS23456 into the graph model would potentially create a high-degree node that is actually the aggregation of many smaller networks in reality.

AS58368 – 65535: AS numbers in this range are private AS numbers, not to be propagated across the Internet. They can be used internally in complex networks subdivided into multiple networks that communicate via BGP, or they can be used by a network to connect to its ISP. Assuming the network only has one ISP, the ISP advertises the customer's address space on its behalf, and so a private AS number can be used by the customer.

These AS numbers are ignored when building links from path data. The two ASes on either side of AS numbers in this range are not assumed to be directly connected.

Having filtered all of the special cases listed above, the remaining AS adjacencies are output as a list of AS number pairs, each pair indicating a link exists between the ASes. Only one link per pair is output.

A.2 Traceroute Data

To simulate the routing protocol in Chapter 6, I require latency values to augment the AS graph model. The CAIDA Archipelago (Ark) Measurement Infrastructure [123] archives round-trip times to the rest of the Internet by performing mass traceroutes from multiple locations. Traceroute probes can be used to actively infer latency and connectivity characteristics of the network. I use this data in this section to derive reasonable delay parameters for routing simulations.

Ark operates by taking the current address space from the BGP tables hosted by Route Views, and then splitting into a list of $/24$'s to be individually probed. This is a massive task, which is subdivided into eighteen monitors, each of which is a member of one of three teams. Each team is given a portion of the full IPv4 address space. Monitors within the team initiate a traceroute to each of those blocks, choosing the destination address at random from within each block. Each team operates independently of each other, and each full scan of the address space it is given is called a *cycle*. Each cycle takes 2-3 days to complete.

The resulting archived data contains, for each destination selected, as many router hops as responded or a marker indicating that a hop was silent. (That is, even if the destination does not respond, RTT information is stored for all router hops that did.) The Paris traceroute [124] that is used to conduct these measurements exposes more paths than standard traceroute, by modifying bits in packet headers to increase the likelihood of on-path load balancers choosing alternative paths to the same destination.

I collected one traceroute cycle per week from each team, starting from 2 January 2011, up to 19 May 2011. By selecting a range of cycles, AS coverage will be greater because the destination address is chosen randomly within each $/24$, and so over time the number of responding hosts within each block will increase. Also, traceroutes selected over a longer time period will reveal alternate paths, and the accumulation of data should be exhaustive enough that many measurements across each AS will be taken.

A.2.1 IP to AS Mapping

To build a model of AS latency using IP router-level measurements, the IP addresses must be mapped onto the AS number of the network they belong to. To map IP addresses onto AS numbers, I took the full list of BGP advertisements from 21 May 2011 to represent the list of advertised prefixes visible upon completion of the final traceroute cycle.

To map IP addresses onto AS numbers is a two-stage process. First, the IP address must be matched against the longest matching prefix. Next, the prefix must be matched against the origin AS. This mapping is performed for all IP addresses visible in the traceroute data. In the event of no matching prefix for an IP address, or a prefix mapping onto multiple origin ASes, the AS hop is marked as "NA".

Handling IXPs

Traceroute paths reveal inter-AS connectivity on the forwarding path. For most addresses observed, it is possible to map the address directly to its origin AS. Many ASes connect to each other using Internet Exchange Points (IXPs) as a common physical location for interconnection. IXPs effectively offer a layer-2 substrate that BGP peers connect to, allowing configuration of the layer-3 connectivity to allow a BGP session to be created. For the layer-3 connectivity, a common address range is used within the IXP, which is used on the IXP-side of all BGP speakers co-located at the IXP (see Figure A.3 for an illustrative example).

Although a traceroute packet will be oblivious to the layer-2 substrate, the traceroute will emit a packet with a TTL that expires at the router on a path exiting the IXP. When the TTL expires, routers must respond with an ICMP Time Exceeded message with the packet source

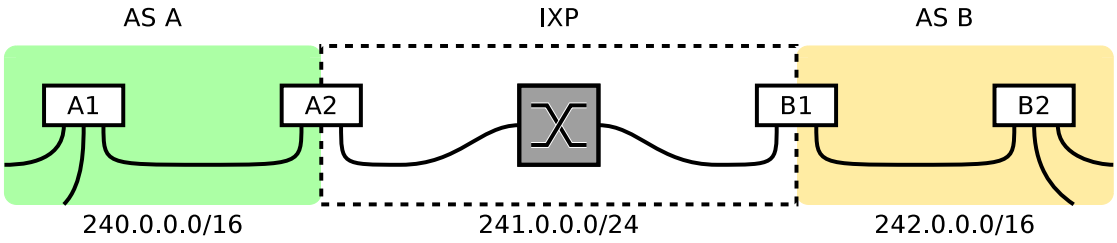


Figure A.3: Typical IXP structure.

set to the interface on which the original packet arrived. Thus, an IP address from the shared (IXP) prefix will be the source address in the packet, and therefore addresses appear in the traceroute data that are often not publicly routable. For the same reason, private address blocks can be observed in traceroute data.

To identify IXPs, on 3 June 2011 I scraped a list of known IXP address blocks from two sites: Packet Clearing House (`pch.net`) and PeeringDB (`peeringdb.com`), similar to the work done in [125]. These list 326 and 257 prefixes respectively, providing 443 unique prefixes in total.

Given these prefixes, I add them to the existing list of prefixes derived from the BGP data to create a "complete" list of prefixes. All IXP prefixes are marked with the identifier "IXP" instead of an AS number, and also all the private address blocks specified by RFC 1918 [119] are marked with the identifier "PRV". Additional information on deriving AS level data from Traceroute data is available in [126, 127].

AS Coverage

Figure A.4 indicates the proportion of ASes that are visible in the BGP data as the traceroute data accumulates.

The visibility of ASes relies on the paths taken through ASes, and the IP address targetted at the destination. Since destination addresses are chosen at random from within $/24$s, it is possible to target an address not in use and, therefore, get no response. Thus, as more traceroute data is accumulated, more ASes become visible. Figure A.4a shows the total number of ASes observed in each snapshot (the figure may increase over time because the prefix list I used for the mapping was built from the prefixes visible the end of all the traceroute cycles). Figure A.4b indicates the total number of ASes visible as the traceroute data was accumulated.

Similarly with AS paths, the Paris traceroute will help expose multiple paths. More traceroute data reveals a greater number of AS paths. Figure A.4c shows the proportion of the ASes and AS links visible in the traceroute data as the data accumulated. The figure shows that approximately 20% of the ASes visible in the BGP data for 8 May 2011 are not visible

in the traceroute data at the end of all cycles, and approximately 40% of the links are also missing. How these missing ASes and links are handled are described is Section A.2.2.

A.2.2 Determining AS Graph Latency Values

Building a latency model for the AS graph is not trivial, but in order to simulate routing updates propagating across the AS topologies used in Chapter 6 I require some measure of latency. The obvious mechanisms to measure latency are:

1. Latency through an AS (measuring entry and exit timestamps).
2. Latency between pairs of adjacent ASes (measuring exit then entry timestamps).

These are reasonably easy to derive by subtracting the ingress RTT value from the egress RTT value for each AS observed on path (for a measure of intra-AS latency), and subtracting the egress RTT for one AS from the ingress RTT for the next (for a measure of inter-AS latency).

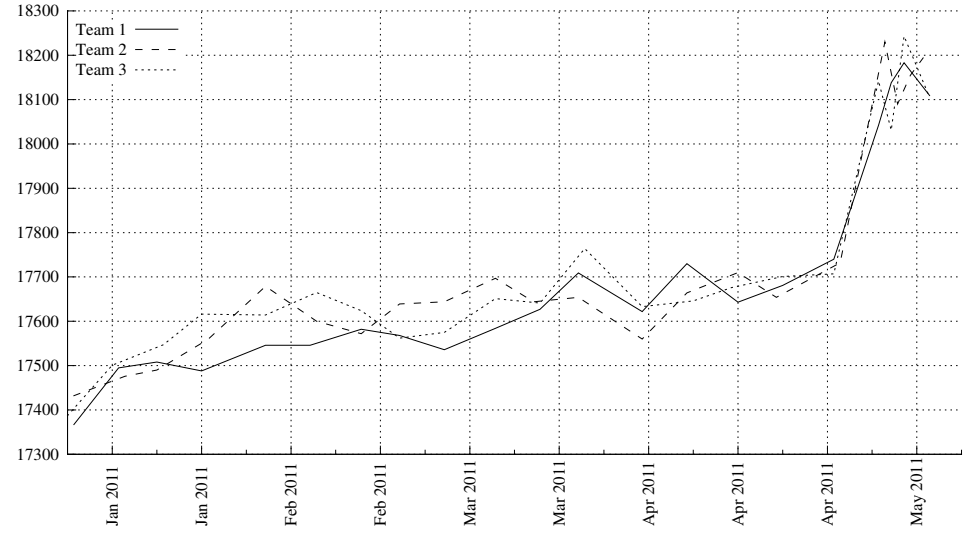
There exists a proportion of RTT values that are negative, and there are also large positive outliers. To alleviate this issue, the values must be filtered. Figure A.5 shows the distribution of observed maximal latencies to all IP addresses monitored, prior to deriving inter-AS and intra-AS latencies, with no negative latencies shown. The most common range is between 0 and 1ms, observed on 291174 (1.07%) traces. There appears to be distinct clustering of RTT values beneath 60ms, around 140ms, and around 310ms.

Distributions of intra-AS latencies are shown in Figure A.6, with some negative values shown to demonstrate the distribution. Figure A.6a shows the result of the data if the end of the AS path is assumed to be a full traversal of the destination AS, but in actuality the path will terminate somewhere within the AS (quite possibly near the ingress boundary). Figure A.6b shows the resulting distribution of intra-AS traversal times if the destination AS on the AS path is ignored completely. The latter of these is more useful as an aggregate measure of traversal time and it is used as the traversal time of an AS if it is available; if it is not available, then a value from the more complete dataset is used.

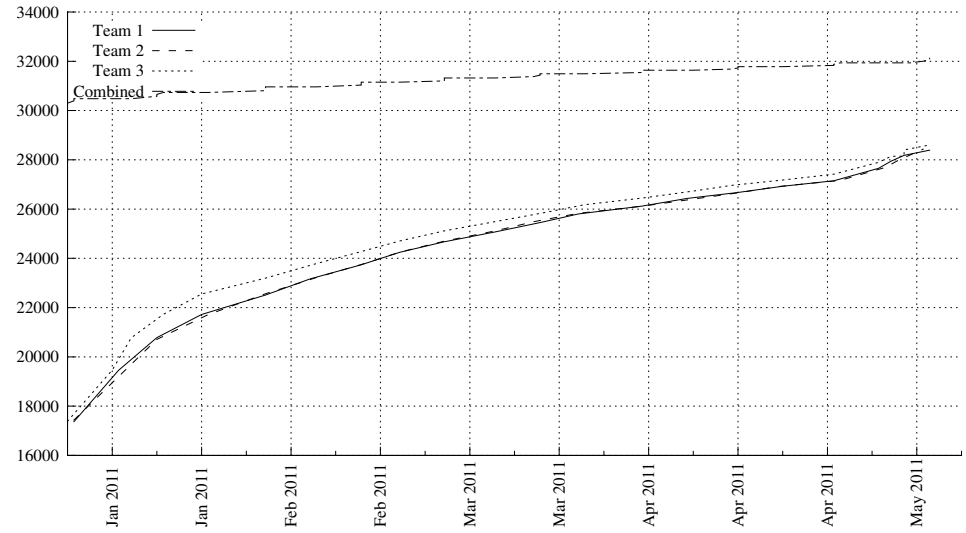
Distributions of intra-AS latencies are shown in Figure A.7, with some negative values shown to demonstrate the distribution.

A.2.3 Default Values

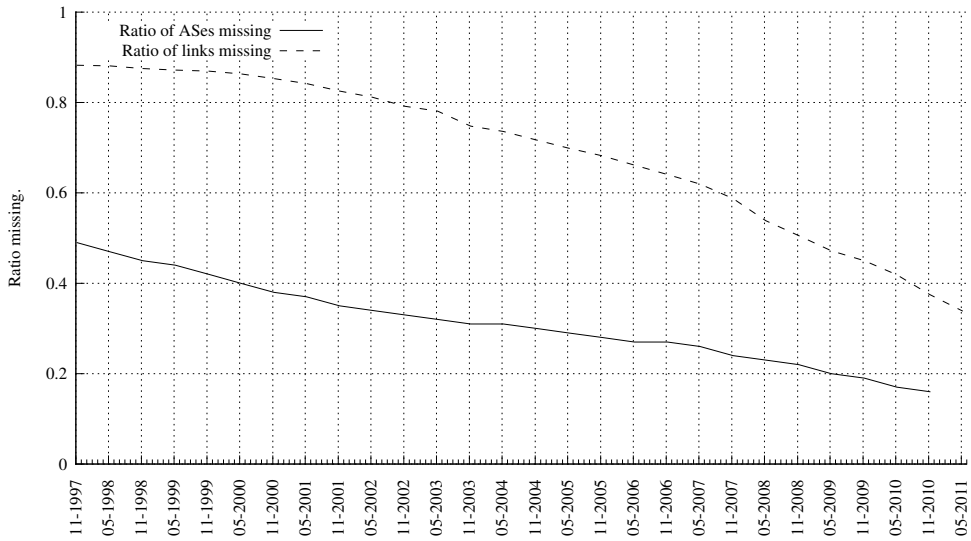
As noted in Section A.2.1, there are ASes and links missing from the traceroute data. To handle these, median values are taken from the intra-AS transit distribution (with the end of



(a) Absolute AS coverage, by Ark team, for each date.



(b) Cumulative AS coverage.



(c) Ratio of missing ASes

Figure A.4: Traceroute AS visibility

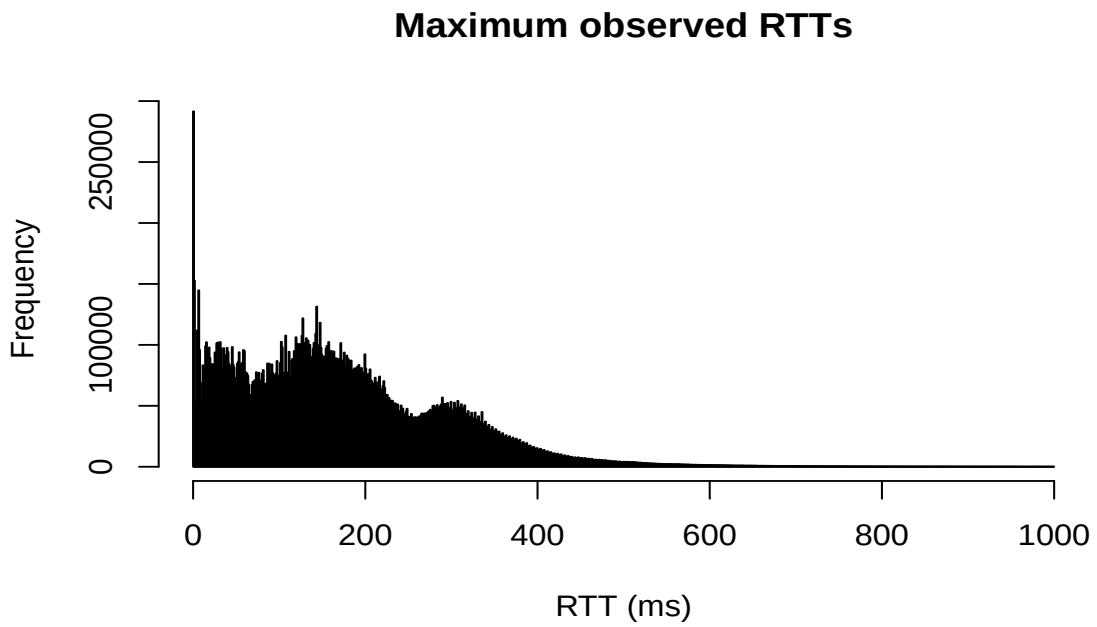
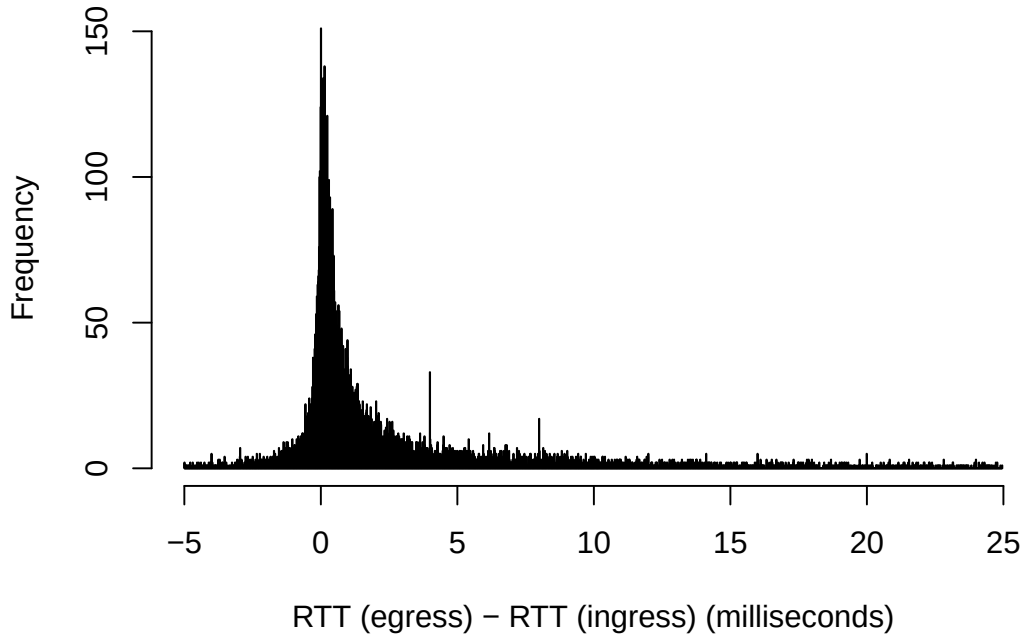


Figure A.5: The distribution of observed round-trip times in the Ark data. This shows the distribution of observed RTTs to the target, not intermediate hops. The data is clipped, with no results longer than 1 second shown. Some 401626 traces (1.48%) demonstrated an RTT greater than 1 second.

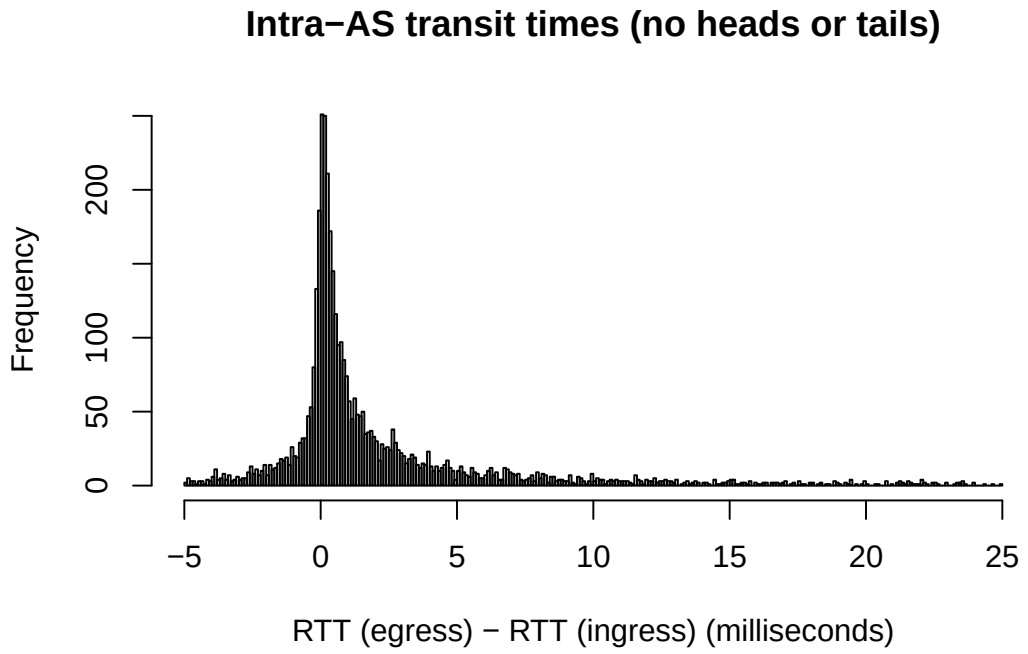
paths removed as in Figure A.6b) and the inter-AS data. The median values are 1.32ms and 3.74ms respectively.

A.3 Summary

This appendix has outlined the processing pipelines for the various data sources used in this dissertation. In particular, it has clarified how the data is cleansed, and how the data is aggregated, to form the basis for the experiments and analysis presented in Chapters 2, 4, 5 and 6.

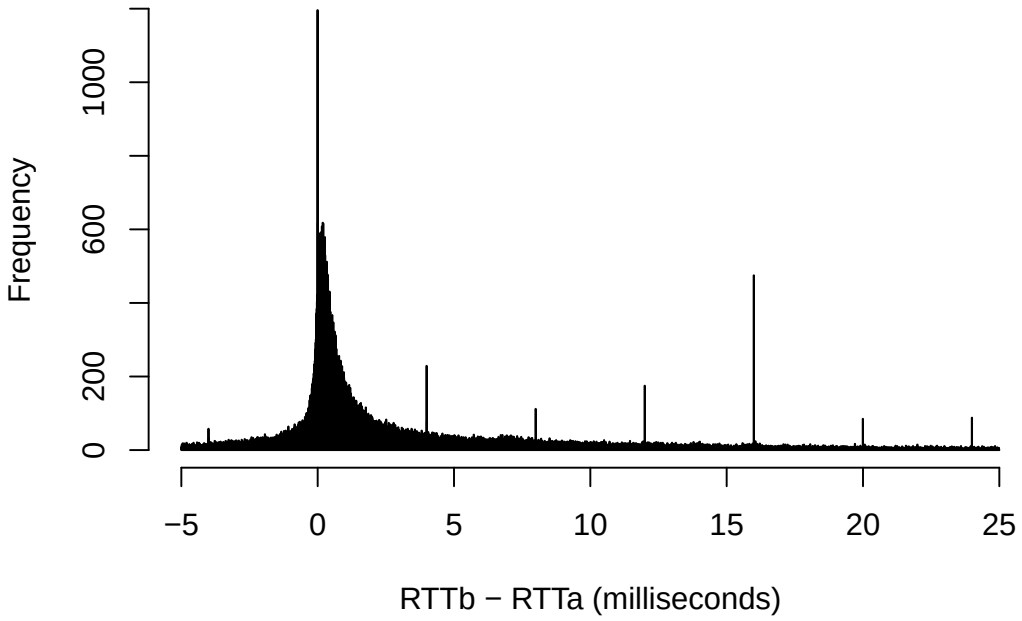


(a) Distribution of intra-AS latencies.



(b) Distribution of intra-AS with heads and tails of paths removed.

Figure A.6: Intra-AS latency distributions.



(a) Distribution of inter-AS latencies.

Figure A.7: Inter-AS latency distributions.

Appendix B

Simulation

To generate the results required for Chapters 4, 5 and 6, simulation software was built. Specifically, Chapters 4 and 5 simulate the forwarding algorithm for the TZ and TZ_k compact routing algorithms respectively, and Chapter 6 simulates the propagation of routing state using the decentralised TZ network protocol. This appendix provides an overview of the simulation software that was built to perform the evaluations detailed in those chapters.

The simulator is implemented as a distributed time-based event simulator. Distribution of the simulation is primarily logistical: there is copious computation power available in the labs at Glasgow in the form of undergraduate workstations, each with a dual-core Athlon (each core 64-bit maximally clocked at 2.7GHz) with 4GB RAM. There are 116 of these hosts available.

The primary need to distribute the simulation is memory usage, although there are obvious gains in parallelisation. Memory is an issue for simulation of larger networks. The largest of the graphs simulated for this dissertation is the 37,943 node AS graph from 8 May 2011. Each simulated node has its own local state, and each physical host has its a queue of pending events that must be stored until the appropriate simulation time. Memory requirements, especially for Chapter 6 which requires queueing events to retain causal ordering, are not trivial.

This appendix is structured as follows:

Section B.1: contains a brief overview of related simulation work.

Section B.2: provides an overview of the various components in the simulator, and the interactions between them.

Section B.3: describes the simulator's traffic generation.

Section B.4: describes the routing state propagation, how messages are ordered, and how causality is maintained.

Section B.5: summarises the appendix.

B.1 Background & Requirements

Discrete event simulation software packages exist, but generally they focus on details of protocol implementations (*i.e.*, they aim for compliant TCP stacks with implementations of multiple TCP algorithms, or they aim to provide realistic BGP implementations). Many of these simulation environments are not designed to be distributed across multiple hosts. One of the simulation environments closest to the requirement of simulating large topologies is C-BGP, a simulator designed to model and analyse the flow of traffic *internal* to large AS networks [128]. C-BGP is not distributable, however. Other simulators such as the commonly-used *ns-2* and *ns-3* also exist, but these aim to solve a different problem: they are designed to carefully emulate protocol behaviour rather than abstract away details to simulate the large-scale simulations required for this dissertation.

This dissertation evaluates routing algorithms on topologies derived from real data, by using archived BGP data to build a model of an AS graph that features ASes as nodes with unweighted, unannotated links between them. Considering that an AS represents a routing domain with a unified policy, this is sufficient model for the simulation of graphs on this scale. Fine-grained BGP emulation is not necessary, especially when the detailed configuration of each AS is not known. The requirements for this dissertation are two-fold:

- The simulator needs to be able to evaluate the forwarding algorithm of the TZ compact routing algorithm. Given correct routing state, which in Chapters 4 and 5 is the output of centralised computation and in Chapter 6 is the output of distributed computation, the forwarding algorithm must be executed by generating traffic and logging the lengths of the paths traversed. Calculating TZ path lengths analytically is not possible, because the algorithm can "shortcut" away from the longest path by not utilising the landmark, as described in Section 4.2.1.

- For Chapter 6, the simulator needs to be able to simulate the progression of routing state through the network as time advances, using inter-AS and intra-AS latencies to dictate when messages arrive at an AS and when they are emitted from an AS. That is, a requirement is the scheduling and timing of events to mimic network delay.

In the final simulator design, these two components are distinct. The traffic forwarding simulation is more parallelisable than the routing state propagation, as the latter involves timing constraints to maintain causality. All components have been implemented in Scala, aside from the supporting shell scripts described in Section B.2.2, and all simulations were run on the lab hosts already mentioned. The software was written in Scala for several reasons:

- Scala's actor model allows for message passing between independent actors, which maps well onto the simulation of distributed nodes exchanging routing state.
- Scala's remote actors allow near-transparent distribution of actors onto multiple physical hosts.
- The language is succint, improving readability. It runs on the JVM, and offers various functional constructs that reduce programmer error, with higher-order functions that operate over the full membership of a collection, removing the need for manual looping.

Regarding distribution of simulations, there are two broad approaches for distributed discrete event simulation: optimistic simulators, that assume largely independent operation between the distributed components with the facility to rollback after an assumed state turned out to be incorrect (*e.g.*, Virtual Time: [129]), and pessimistic simulators, that use coordination mechanisms between distributed components to ensure ordering of event processing. The trade-offs between these types of simulations depend on the type of simulation being constructed; if rollbacks are infrequent, then optimistic distributed simulators can offer performance gains over pessimistic simulators [130].

The simulation software constructed for this dissertation is a pessimistic distributed event simulator, primarily on the basis that it is easier to argue the correctness of the code if all actions are causally ordered.

B.2 Software Components

The simulator software has several key components that are referenced in Sections B.3 and B.4. Most components are generic and don't contain any code specific to the routing algorithms being evaluated; only some of the components contain such logic or state.

The main components that are effectively generic infrastructure that could facilitate other simulation types are:

Controller: The centralised controller, which retains the current timeframe of the full simulation. There is one Controller per simulation.

HostController: The host-level controller, which retains the current timeframe of the AutonomousSystems it hosts. There is one or more HostControllers per simulation.

EventQueueManager: Maintains an ordered list of events received, and dispatches them to AutonomousSystems in-order. There is one EventQueueManager per HostController.

NameServer: A basic name mapping service that performs certain key functions: first, it provides a rendezvous point for AutonomousSystems located on different hosts, and can be looked up the first time an AutonomousSystem attempts to communicate with another; second, it provides the mock DNS service for AutonomousSystems to use during simulation of routing state and packet forwarding. There is one NameServer per simulation.

Components that contain code specific to the simulation are:

AutonomousSystem: Which maintains its local forwarding and routing state, and runs the routing and forwarding algorithms according to this state. There are one or more AutonomousSystems per simulation.

AutonomousSystems also maintain:

- Message counts: update message counts; landmark message counts; "DNS" message counts.
- Landmark table: a mapping between landmark AS numbers and a set of valid paths to that destination.
- Routing table: a mapping between ordinary ASes and a set of valid paths to that destination.
- Forwarding table: a mapping from all stored destinations to the next hop.

TrafficGenerator: Accepts a list of *source* and *destination* pairs to generate traffic. There is one TrafficGenerator per simulation.

Each of these components is intended to run on an independent host, but that is not a requirement: a simulation can be run on one host, with one Controller, one NameServer, one HostController, and one TrafficGenerator. There is no advantage to running multiple HostControllers on the same physical host.

B.2.1 Logging

Logging is provided by the Controller and the HostControllers hierarchically, and objects log to their direct parent (*i.e.*, each HostController logs with the Controller; each AutonomousSystem logs with its HostController.)

There are three logging message types:

- LogMsg(**source**: *Int*, **tick**: *Int*, **message**: *String*)

  This is the standard logging message for an AutonomousSystem to the HostController. The *source* identifies the AS number, the *tick* field indicates the current clock counter at that AS, and the *message* field is the message to be logged. The messages are formatted by the HostController and written to a local, simulation-specific, log file.

- LogMsgStr(**source**: *String*, **tick**: *Int*, **message**: *String*)

  This is an alternative logging message for an AutonomousSystem to the HostController. The *source* can be used to indicate, for example, the TZData traffic covered in Section B.3. As before, the *tick* field indicates the current clock counter at that AS, the *message* field is the message to be logged, and the messages are formatted by the HostController and written to a local, simulation-specific, log file.

- CtrlLog(**source**: *String*, **message**: *String*)

  This is the standard logging message for a HostController to the primary Controller. The *source* field carries the hostname the HostController is running on, and the *message* field is the message to be logged.

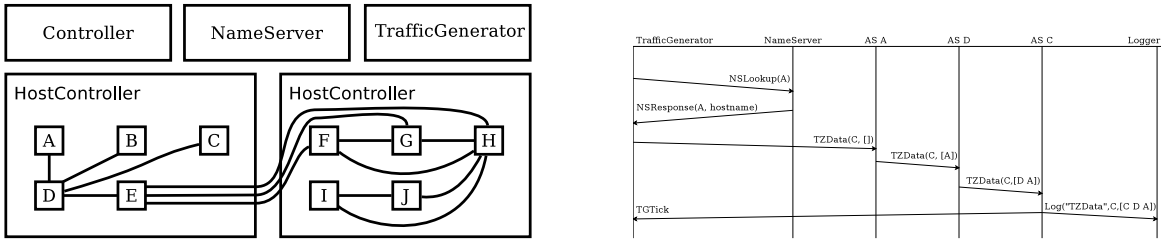
All messages are logged to tab-delimited files, containing the time in milliseconds that the message was processed by the Logging component, the *source* of the message, the simulation time from which the message was emitted (logged by HostControllers only), and the message itself in the final field.

B.2.2 Glue

There are supporting scripts to locate running hosts, launch a simulation, run multiple simulations in sequence, and kill a simulation. The Controller is capable of detecting completion and invoking the kill script itself.

Primarily, these are:

runsim.sh: Accepts as input the name of the graph to simulate, the type of simulation to run (protocol state propagation, or traffic forwarding), and the size of the host pool to utilise. It locates as many hosts as are required that are not currently in use (performing basic checks for host liveness, no active users, etc). Once the host set is discovered, it splits up the input graph into equally-sized subgraphs, distributes each subgraph to one of the hosts, and initiates a HostController on each host. The HostController reads the configuration file locally, and creates as many AutonomousSystems as are required.



(a) Simulator Instance: Traffic Generation.

(b) Sequence diagram for simulated traffic messages passing through the simulator.

Figure B.1: Simulating traffic forwarding.

A unique simulation ID is generated by this script that is known by all components to uniquely identify the running simulation. The simulation ID is embedded into configuration files and log files, and consists the timestamp of the simulation's creation, the name of the graph being evaluated, the hostname of the Controller node, and a random 5-digit number. Each component writes its process ID to a file on its local host such that all processes can be located later.

killsim.sh: Accepts as input the simulation ID to terminate. On the Controller node, important information such as the set of hosts utilised will have been written to file by the *runsim.sh* script, and the script can access each of these hosts in sequence and, using the unique simulation ID, locate the running processes for this simulation and kill all processes.

After killing the processes, the log files stored on each host are copied back to the controller node, and compressed for storage or subsequent processing.

B.3 Simulating Packet Forwarding

The simulation of forwarding paths is required to generate the path stretch results presented in Chapters 4, 5, and 6. The key components required to facilitate the simulation are shown in Figure B.1a, which indicates the AS links for the network presented in Figure 5.1a (page 60). In this example, the AutonomousSystems are split across two hosts, and so there is one Controller, one NameServer, one TrafficGenerator, and two HostControllers. The AutonomousSystems are linked directly to each other.

The TrafficGenerator is provided with a list of source-destination pairs on initiation. Traffic for the traffic generator is simply a whitespace-separated list of AS pairs detailing sources and destinations. Once all AutonomousSystems have been instantiated and have connected to each other, the Controller is notified, which notifies the TrafficGenerator to begin. The sequence of events for a simulated packet from node *A* to node *C* is shown in Figure B.1b, with time advancing from the top to the bottom of the figure. The sequence is as follows:

- The TrafficGenerator sends a lookup request for the host that *A* is located on to the the simple NameServer mechanism, which responds with the hostname.
- The TrafficGenerator then sends a TZData message to *A*. *A* is unaware of the existence of the TrafficGenerator.
- *A* receives the message, and processes it using the TZ forwarding algorithm and its local routing state. It emits a TZData message to its neighbour, *B*, with the stored path state updated.
- *B* receives the message, processes it using the TZ forwarding algorithm and its local routing state, and emits a TZData message to *C*.
- *C* receives the message, and processes it using the TZ forwarding algorithm. Since *C* is the destination, it does two things: it logs the successful arrival of a TZData message and the full path traversed, and also indicates to the TrafficGenerator that the message has arrived at its destination.

The full specification for a TZData message is as follows:

- TZData(**source**: *Int*, **dest**: *Int*, **landmark**: *Int*, **lm_nh**: *Int*, **ttl**: *Int*, **payload**: *List[Int]*)

The fields are populated as follows:

source: Selected by the TrafficGenerator, and not modified by the simulator once set.

dest: Selected by the TrafficGenerator, and not modified by the simulator once set.

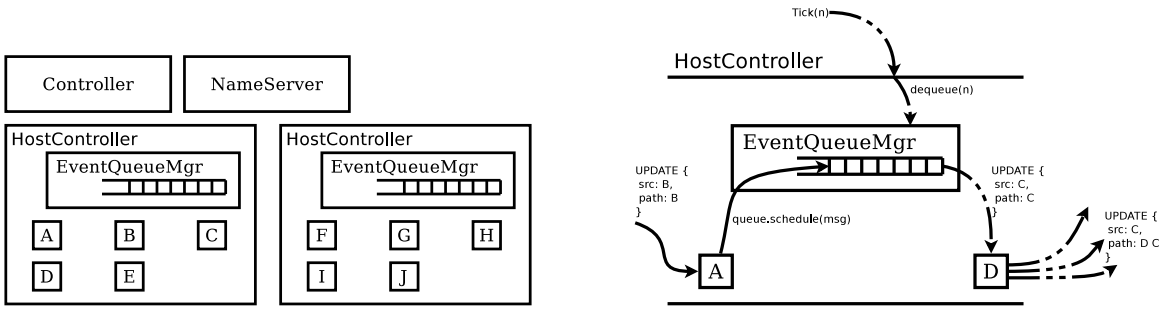
landmark: In Chapters 4 and 5, this value is pre-computed by the TZ compact routing algorithm and is stored. The landmark value is provided to the TrafficGenerator.

In Chapter 6, the landmark field is set to zero by the TrafficGenerator. If, at the source AutonomousSystem, the destination is unknown, that AutonomousSystem queries the NameServer for the set of landmark nodes that the destination has registered, and selects one of the nearest of that set.

lm_nh: In Chapters 4 and 5, this value is pre-computed by the TZ compact routing algorithm and is stored. The lm_nh value is provided to the TrafficGenerator.

In Chapter 6, this field set to zero, and is unused.

ttl: The *ttl* field is set initially to 64, and decremented by each node the packet traverses. (As shown in Figure 2.3a on page 13, the largest diameter on any of the paths observed is 19, and so the TTL should never be reached if stretch-3 routing is adhered to.)



(a) Simulator Instance: Routing.

(b) Sequence diagram for messages passing through a host controller.

Figure B.2: Simulating routing state propagation.

payload: Stores the path traversed by the messages. Each AS visited prepends its AS number to this list.

At any node, if an appropriate next hop is known, the forwarding algorithm send the *TZData* directly to the next AutonomousSystem actor. The forwarding algorithm logs any errors: if a TZData message loops, or if the local forwarding state has no valid next hop for the packet. If the message has reached its destination, then it logs its source, destination, path length, and the list of hops visited. No TZData messages on any of the results presented in this dissertation generated any errors.

B.4 Simulating Routing State Propagation

The simulation of routing state propagation is required to generate the routing state overhead results presented in Chapter 6.

The key components required to facilitate the simulation of routing state propagation are shown in Figure B.2a. For clarity the direct links between AutonomousSystems are not shown in this figure.

AutonomousSystems exchange messages directly with each other. To ensure the correct sequencing of events, however, they do not *act* on any received messages immediately, aside from lodging the timestamp of receipt with the EventQueueManager. The EventQueueManager holds the message for the AutonomousSystem until it is time to react. The AutonomousSystems keep local counts of messages processed.

The important message types are as follows:

- Landmark(**seq**: *Int*, **src**: *Int*, **path**: *List[Int]*)
- Update(**seq**: *Int*, **src**: *Int*, **lmDist**: *Int*, **path**: *List[Int]*)

These represent the message types used for advertising nodes to the network and updating advertisements, as discussed in Section 6.2. The equivalent withdrawal messages are not covered, as dynamic networks are not simulated in this dissertation and therefore nodes do not leave the network, and a node's landmark status does not change once it is set.

These messages are contained within another message type:

- SimMsg(**sendtime**: *Int*, **recvtime**: *Int*, **seq**: *Int*, **msg**: *Message*)

B.4.1 Message Ordering

The AS delay model used in this dissertation was described in Section 6.4 and Section A.2. To simulate routing with delays, messages must be queued and executed in order to maintain causality.

The time model described in this Appendix is pessimistic, *i.e.* it operates in lock-step and doesn't optimistically predict future states. The centralised Controller holds the primary clock.

The processing of messages is shown in Figure B.2b. This figure indicates two distinct phases:

- The solid arrows indicate an AutonomousSystem *A* receiving a *SimMsg* message containing an *UPDATE*, which the AS passes to the local EventQueueManager for queueing according to the *recvtime*. The contents of the message are not processed by the AutonomousSystem when the message is received.
- The dashed arrows indicate the advancement of the simulation clock and the processing of a message. The message is distinct to the one queued by *A*, above. Here, the Controller sends a clock *tick* n to all Host Controllers to instruct them to run any events scheduled for that time stamp. The HostController receives the *tick*, and instructs the EventQueueManager to dequeue all events for n, which then sends each message to the AutonomousSystem that registered the message to act upon it. In this case, node *D* acts upon an *UPDATE* message it had received from node *C*, which *D* decides to propagate to its neighbours.

Thus, AutonomousSystem actors exchange routing messages directly between each other, but lodge inbound messages with the event queue manager on the local host when they arrive. No message can be acted upon until the master clock at the Controller reaches that time stamp.

Once the event queue manager has depleted the queue of all messages for this timestamp, the HostController responds to the central Controller to state that it is done. Once the central Controller receives responses from all HostControllers, it queries each host for its next available timestamp, prior to dispatching a new *tick* to run all events with the next lowest timestamp present in the simulation.

No message generated during timestamp t can be queued for timestamp t or any time prior to t. All new events scheduled at time t must be scheduled for *at least* timestamp $t + 1$, so causality is maintained. When a node propagates a message, it increments its local counters indicating the number of messages it has propagated, and logs the total to the local logging service. By logging the total counts as the simulation progresses, cumulative message counts can be generated easily.

B.5 Summary

This section has briefly covered the primary components of the simulator used to run the simulations used to generate the results used in Chapters 4, Chapter 5, and Chapter 6.

The simulator is a custom built, distributed time-based event simulator, written in Scala. It is highly decentralised, and can be deployed across many physical hosts to alleviate the memory requirements inherent in running a large event-based simulation.

Bibliography

[1] J. Hawkinson and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)," Internet Requests for Comments, RFC Editor, RFC 1930, March 1996. http://www.rfc-editor.org/rfc/rfc1930.txt

[2] X. Cai, J. Heidemann, B. Krishnamurthy, and W. Willinger, "Towards an AS-to-organization map," in *Proceedings of the Tenth Annual ACM Internet Measurement Conference (IMC)*, November 2010.

[3] "University of Oregon Route Views Project Archive," http://archive.routeviews.org/.

[4] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task Force, Jan. 2006. http://www.ietf.org/rfc/rfc4271.txt

[5] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1654 (Proposed Standard), Internet Engineering Task Force, Jul. 1994, obsoleted by RFC 1771. http://www.ietf.org/rfc/rfc1654.txt

[6] E. Rosen, "Exterior Gateway Protocol (EGP)," RFC 827, Internet Engineering Task Force, Oct. 1982, updated by RFC 904. http://www.ietf.org/rfc/rfc827.txt

[7] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy," RFC 1519 (Proposed Standard), Internet Engineering Task Force, Sep. 1993, obsoleted by RFC 4632. http://www.ietf.org/rfc/rfc1519.txt

[8] J. Moy, "OSPF Version 2," RFC 2328 (Standard), Internet Engineering Task Force, Apr. 1998, updated by RFC 5709. http://www.ietf.org/rfc/rfc2328.txt

[9] D. Oran, "OSI IS-IS Intra-domain Routing Protocol," RFC 1142 (Informational), Internet Engineering Task Force, Feb. 1990. http://www.ietf.org/rfc/rfc1142.txt

[10] M. Caesar and J. Rexford, "BGP Routing Policies in ISP Networks," *IEEE Network*, vol. 19, no. 6, pp. 5–11, 2005.

[11] A. Dhamdhere and C. Dovrolis, "Ten years in the evolution of the internet ecosystem," in *Proceedings of the Tenth Annual ACM Internet Measurement Conference (IMC)*, October 2008.

[12] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2005.

[13] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law Relationships of the Internet Topology," in *Proceedings of the ACM SIGCOMM 1999 Conference*, September 1999.

[14] P. Mahadevan, D. Krioukov, M. Fomenkov, X. Dimitropoulos, k. c. claffy, and A. Vahdat, "The Internet AS-Level Topology: Three Data Sources and One Definitive Metric," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 17–26, 2006.

[15] H. A. Simon, "On a Class of Skew Distribution Functions," *Biometrika*, vol. 42, no. 3-4, pp. 425–440, 1955.

[16] A. Medina, I. Matta, and J. Byers, "On the Origin of Power Laws in Internet Topologies," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 2, pp. 18–28, 2000.

[17] R. Pastor-Satorras, A. Vázquez, and A. Vespignani, "Dynamical and Correlation Properties of the Internet," *Physical Review Letters*, vol. 87, no. 25, November 2001.

[18] M. Fayed, P. Krapivsky, J. W. Byers, M. Crovella, D. Finkel, and S. Redner, "On the emergence of highly variable distributions in the autonomous system topology," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 41–49, 2003.

[19] L. Kleinrock and F. Kamoun, "Hierarchical Routing for Large Networks," *Computer Networks*, vol. 1, no. 3, pp. 155–174, 1977.

[20] T. Bu, L. Gao, and D. Towsley, "On characterizing BGP routing table growth," *Computer Networks*, vol. 45, no. 1, pp. 45–54, 2004.

[21] L. Cittadini, W. Muhlbauer, S. Uhlig, R. Bush, P. Francois, and O. Maennel, "Evolution of Internet Address Space Deaggregation: Myths and Reality," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1238–1249, October 2010.

[22] X. Zhao, D. J. Pacella, and J. Schiller, "Routing Scalability: An Operator's View," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1262–1270, October 2010.

[23] k. c. claffy, "Workshop on Internet Economics (WIE2009) Report," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 56–59, 2010.

[24] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger, "Towards capturing representative AS-level Internet topologies," *Computer Networks*, vol. 44, no. 6, pp. 737–755, 2004.

[25] P. Mahadevan, D. V. Krioukov, M. Fomenkov, B. Huffaker, X. A. Dimitropoulos, K. C. Claffy, and A. Vahdat, "Lessons from Three Views of the Internet Topology," *Computing Research Repository (CoRR)*, August 2005.

[26] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang, "The (in)Completeness of the Observed Internet AS-level Structure," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 109–122, February 2010.

[27] M. Roughan, S. J. Tuke, and O. Maennel, "Bigfoot, Sasquatch, the Yeti and Other Missing Links: What We Don't Know About the AS Graph," in *Proceedings of the Eighth Annual ACM Internet Measurement Conference (IMC)*, October 2008.

[28] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush, "10 Lessons from 10 Years of Measuring and Modelling the Internet's Autonomous Systems," *IEEE Journal on Selected Areas in Computing Science*, vol. 29, no. 9, pp. 1810 – 1821, 2011.

[29] B. Zhang, R. Liu, D. Massey, and L. Zhang, "Collecting the internet AS-level topology," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 1, pp. 53–61, January 2005.

[30] R. V. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang, "In Search of the Elusive Ground Truth: The Internet's AS-level Connectivity Structure," in *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2008.

[31] H. Haddadi, D. Fay, A. Jamakovic, O. Maennel, A. W. Moore, R. Mortier, M. Rio, and S. Uhlig, "Beyond Node Degree: Evaluating AS Topology Models," *Computing Research Repository (CoRR)*, July 2008.

[32] R. G. Clegg, C. D. Cairano-Gilfedder, and S. Zhou, "A Critical Look at Power Law Modelling of the Internet," *Computer Communications*, vol. 33, no. 3, pp. 259–268, 2010.

[33] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy, "Lord of the links: a framework for discovering missing links in the internet topology," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 391–404, April 2009.

[34] R. V. Oliveira, B. Zhang, and L. Zhang, "Observing the Evolution of Internet AS Topology," in *Proceedings of the ACM SIGCOMM 2007 Conference*, August 2007.

[35] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao, "Where the Sidewalk Ends: Extending the Internet AS Graph Using Traceroutes From P2P Users," in *Proceedings of the 2009 CoNEXT Conference*, December 2009.

[36] H. Haddadi, S. Uhlig, A. Moore, R. Mortier, and M. Rio, "Modeling Internet Topology Dynamics," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 65–68, March 2008.

[37] S. Shakkottai, M. Fomenkov, D. V. Krioukov, R. Koga, and K. C. Claffy, "Evolution of the Internet AS-Level Ecosystem," *Computing Research Repository (CoRR)*, 2006.

[38] L. Gao, "On Inferring Autonomous System Relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, December 2001.

[39] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points," in *Proceedings of the Twenty-First Annual IEEE INFOCOM*, June 2002.

[40] X. Dimitropoulos, D. Krioukov, B. Huffaker, k. claffy, and G. Riley, "Inferring AS Relationships: Dead End or Lively Beginning," in *4th International Workshop on Efficient and Experimental Algorithms*, May 2005.

[41] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, k. claffy, and G. Riley, "AS Relationships: Inference and Validation," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 29–40, 2007.

[42] J. Xia and L. Gao, "On the Evaluation of AS Relationship Inferences," in *IEEE Global Telecommunications Conference, 2004 (Globecom)*, December 2004.

[43] T. Erlebach, A. Hall, and T. Schank, "Classifying Customer-Provider Relationships in the Internet," in *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN)*, November 2002.

[44] G. D. Battista, M. Patrignani, and M. Pizzonia, "Computing the Types of the Relationships Between Autonomous Systems," in *Proceedings of the Twenty-Second Annual IEEE INFOCOM*, April 2003.

[45] G. D. Battista, T. Erlebach, A. Hall, M. Patrignani, M. Pizzonia, and T. Schank, "Computing the Types of the Relationships between Autonomous Systems," *IEEE/ACM Transactions on Networking*, 2007.

[46] G. Huston and G. Armitage, "Projecting Future IPv4 Router Requirements from Trends in Dynamic BGP Behaviour," in *Proceedings of the Australian Telecommunication Networks and Applications Conference, 2006 (ATNAC)*, December 2006.

[47] K. Fall, G. Iannaccone, S. Ratnasamy, and P. B. Godfrey, "Routing Tables: Is Smaller Really Much Better?" in *Proceedings of the Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, October 2009.

[48] D. R. Morrison, "PATRICIA — Practical Algorithm To Retrieve Information Coded in Alphanumeric," *Journal of the ACM*, vol. 15, no. 4, pp. 514–534, October 1968.

[49] M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, "Small forwarding tables for fast routing lookups," in *Proceedings of the ACM SIGCOMM 1997 Conference*, September 1997.

[50] P.-C. Wang, C.-T. Chan, S.-C. Hu, Y.-C. Shin, and Y.-C. Chen, "Hardware-based IP routing lookup with incremental update," in *Proceedings of the Ninth International Conference on Parallel and Distributed Systems (ICPADS)*, December 2002.

[51] H. Narayan, R. Govindan, and G. Varghese, "The impact of address allocation and routing on the structure and implementation of routing tables," in *Proceedings of the ACM SIGCOMM 2003 Conference*, August 2003.

[52] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proceedings of the Seventeenth Annual IEEE INFOCOM*, April 1998.

[53] W. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, 1995.

[54] S. A. McKee, "Reflections on the Memory Wall," in *Proceedings of the 1st Conference on Computing Frontiers*, April 2004.

[55] B. Carpenter, R. Atkinson, and H. Flinck, "Renumbering Still Needs Work," RFC 5887 (Informational), Internet Engineering Task Force, May 2010. http://www.ietf.org/rfc/rfc5887.txt

[56] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," RFC 4423 (Informational), Internet Engineering Task Force, May 2006. http://www.ietf.org/rfc/rfc4423.txt

[57] E. Nordmark and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6," RFC 5533 (Proposed Standard), Internet Engineering Task Force, Jun. 2009. http://www.ietf.org/rfc/rfc5533.txt

[58] R. Stewart, "Stream Control Transmission Protocol," RFC 4960 (Proposed Standard), Internet Engineering Task Force, Sep. 2007. http://www.ietf.org/rfc/rfc4960.txt

[59] M. O'Dell, "GSE - An Alternative Architecture for IPv6," February 1997. http://tools.ietf.org/id/draft-ietf-ipngwg-gseaddr-00.txt

[60] D. Farinacci, V. Fuller, D. Oran, and D. Meyer, "Locator/ID Separation Protocol (LISP)," March 2009. http://tools.ietf.org/id/draft-farinacci-lisp

[61] D. Massey, L. Wang, B. Zhang, and L. Zhang, "A Scalable Routing System Design for Future Internet," in *IPv6'07: Proceedings of the ACM SIGCOMM workshop on IPv6 and the Future of the Internet*, August 2007.

[62] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure, "Evaluating the Benefits of the Locator/Identifier Separation," in *Proceedings of 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*, August 2007.

[63] H. Ballani, P. Francis, T. Cao, and J. Wang, "Making Routers Last Longer with ViAggre," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2009.

[64] D. D. Clark, K. Sollins, J. Wroclawski, and T. Faber, "Addressing Reality: An Architectural Response to Real-World Demands on the Evolving Internet," in *Proceedings of the ACM SIGCOMM workshop on Future Directions in Network Architecture (FDNA '03)*, August 2003.

[65] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet," in *Proceedings of the ACM SIGCOMM 2002 Conference*, August 2002.

[66] R. Braden, D. Clark, S. Shenker, and J. Wroclawski, "Developing a Next-Generation Internet Architecture," ISI White Paper, Tech. Rep., July 2000. http://www.isi.edu/newarch/DOCUMENTS/WhitePaper.pdf

[67] X. Yang, D. Clark, and A. W. Berger, "NIRA: A New Inter-Domain Routing Architecture." *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 775–788, August 2007.

[68] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, "HLP: A Next Generation Inter-domain Routing Protocol," in *Proceedings of the ACM SIGCOMM 2005 Conference*, August 2005.

[69] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "ROFL: Routing on Flat Labels," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, October 2006, pp. 363–374.

[70] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual Ring Routing: Network Routing Inspired by DHTs," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 351–362, October 2006.

[71] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in G Major: Designing DHTs with Hierarchical Structure," in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, March 2004.

[72] C. Gavoille and M. Gengler, "Space-Efficiency for Routing Schemes of Stretch Factor Three," *Journal of Parallel and Distributed Computing*, vol. 61, no. 5, pp. 679–687, 2001.

[73] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269 – 271, 1959.

[74] R. Bellman, "On A Routing Problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87 – 90, 1958.

[75] D. Peleg and E. Upfal, "A tradeoff between space and efficiency for routing tables," in *Proceedings of the twentieth annual ACM Symposium on Theory of computing (STOC'88)*, New York, NY, USA, May 1988.

[76] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, "Improved routing strategies with succinct tables," *Journal of Algorithms*, vol. 11, no. 3, pp. 307 – 341, 1990.

[77] B. Awerbuch and D. Peleg, "Routing with Polynomial Communication-Space Trade-Off." *SIAM Journal of Discrete Mathematics*, vol. 5, no. 2, pp. 151 – 162, 1992.

[78] T. Eilam, C. Gavoille, and D. Peleg, "Compact routing schemes with low stretch factor (extended abstract)," in *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, June 1998.

[79] L. J. Cowen, "Compact Routing with Minimum Stretch," *Journal of Algorithms*, vol. 38, no. 1, pp. 170–183, 2001.

[80] M. Thorup and U. Zwick, "Compact Routing Schemes," in *Proceedings of the Thirteenth ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, July 2001.

[81] D. Krioukov and k. c. claffy, "Toward Compact Interdomain Routing," *Computing Research Repository (CoRR)*, August 2005.

[82] D. Krioukov, K. Fall, and X. Yang, "Compact Routing on Internet-Like Graphs," in *Proceedings of the Twenty-Third Annual IEEE INFOCOM*, March 2004.

[83] D. Krioukov, k. claffy, K. Fall, and A. Brady, "On Compact Routing for the Internet," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, July 2007, pp. 41–52.

[84] R. Agarwal, P. B. Godfrey, and S. Har-Peled, "Approximate distance queries and compact routing in sparse graphs," in *Proceedings of the Thirtieth Annual IEEE INFOCOM*, April 2011.

[85] Y. Mao, F. Wang, L. Qiu, S. Lam, and J. Smith, "S4: Small State and Small Stretch Compact Routing Protocol for Large Static Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 761–774, 2010.

[86] M. Enachescu, M. Wang, and A. Goel, "Reducing Maximum Stretch in Compact Routing," in *Proceedings of the Twenty-Seventh Annual IEEE INFOCOM*, April 2008.

[87] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup, "Compact name-independent routing with minimum stretch," *ACM Transactions on Algorithms*, vol. 4, no. 3, pp. 1–12, 2008.

[88] D. Krioukov, F. Papadopoulos, A. Vahdat, and M. Boguñá, "On curvature and temperature of complex networks," *Physical Review E*, vol. 80, no. 3, September 2009.

[89] A. Jamakovic, P. Mahadevan, A. Vahdat, M. Boguñá, and D. V. Krioukov, "How small are building blocks of complex networks," *Computing Research Repository (CoRR)*, August 2009.

[90] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá, "Hyperbolic geometry of complex networks," *Physical Review E*, vol. 82, no. 3, 2010.

[91] D. Krioukov, F. Papadopoulos, M. Boguñá, and A. Vahdat, "Greedy Forwarding in Scale-Free Networks Embedded in Hyperbolic Metric Spaces," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 2, pp. 15–17, 2009.

[92] M. Boguñá, F. Papadopoulos, and D. V. Krioukov, "Sustaining the Internet with Hyperbolic Mapping," *Nature Communications*, vol. 1, p. 62, 2010.

[93] A. Brady and L. Cowen, "Compact Routing on Power Law Graphs with Additive Stretch," in *Proceedings of the Workshop on Algorithm Engingeering and Experiments (ALENEX'06)*, January 2006.

[94] W. Willinger, R. Govindan, S. Jamin, V. Paxson, and S. Shenker, "Scaling Phenomena in the Internet: Critically Examining Criticality," *Proceedings of the National Academy of the United States of America (PNAS)*, vol. 99, no. 1, pp. 2573 – 2580, February 2002.

[95] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing," RFC 4984 (Informational), Internet Engineering Task Force, Sep. 2007. http://www.ietf.org/rfc/rfc4984.txt

[96] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, October 1999.

[97] D. Peleg, "Proximity-Preserving Labeling Schemes and Their Applications," in *Graph-Theoretic Concepts in Computer Science*, ser. Lecture Notes in Computer Science, 1999, vol. 1665, pp. 30–41.

[98] G. Mooney, "Evaluating Compact Routing Algorithms on Real-World Networks," Master's thesis, Deptartment of Computing Science, University of Glasgow, 2010.

[99] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, "A model of Internet topology using k-shell decomposition," *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, vol. 104, no. 27, pp. 11 150–11 154, July 2007.

[100] W. Chen, C. Sommer, S.-H. Teng, and Y. Wang, "Compact Routing in Power-Law Graphs," in *Proceedings of the 23rd International Symposium on Distributed Computing*, September 2009.

[101] "The CAIDA AS Relationships Dataset," http://www.caida.org/data/active/as-relationships/.

[102] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, "K-Core Decomposition of Internet Graphs: Hierarchies, Self-Similarity and Measurement Biases," *Networks and Heterogeneous Media*, vol. 3, no. 2, June 2008.

[103] U. Kang, S. Papadimitriou, J. Sun, and H. Tong, "Centralities in Large Networks: Algorithms and Observations," in *Proceedings of the Eleventh SIAM International Conference on Data Mining*, 2011.

[104] S. B. Seidman, "Network Structure and Minimum Degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983.

[105] V. Batagelj and M. Zaversnik, "An O(m) Algorithm for Cores Decomposition of Networks," *Computing Research Repository (CoRR)*, October 2003.

[106] V. Batagelj and M. Zaversnik, "Generalized Cores," *Computing Research Repository (CoRR)*, February 2002.

[107] B. E. Carpenter, "Observed Relationships between size Measures of the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 2, pp. 5–12, 2009.

[108] Y. Zhang, R. Oliveira, and H. Z. L. Zhang, "Quantifying the Pitfalls of Traceroute in AS Connectivity Inference," in *Proceedings of the Passive and Active Measurement Conference*, ser. Lecture Notes in Computer Science, no. 6932, April 2010.

[109] S. D. Strowes, G. Mooney, and C. Perkins, "Compact Routing on the Internet AS-Graph," in *14th IEEE Global Internet Symposium*, April 2011.

[110] S. Gangam and S. Fahmy, "Distributed Partial Inference under Churn," in *13th IEEE Global Internet Symposium*, March 2010.

[111] V. Fuller, D. Farinacci, D. Meyer, and D. Lewis, "LISP Alternative Topology (LISP+ALT)," September 2011. http://tools.ietf.org/html/draft-ietf-lisp-alt

[112] W. Mühlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig, "Building an AS-topology model that captures route diversity," in *Proceedings of the ACM SIGCOMM 2006 Conference*, September 2006.

[113] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proceedings of the ACM SIGCOMM 1999 Conference*, September 1999.

[114] G. Rétvári, A. Gulyás, Z. Heszberger, M. Csernai, and J. J. Bíró, "Compact Policy Routing," in *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, 2011.

[115] T. G. Griffin and J. L. Sobrinho, "Metarouting," in *Proceedings of the ACM SIGCOMM 2005 Conference*, August 2005.

[116] H. Yan, B. Say, B. Sheridan, D. Oko, C. Papadopoulos, D. Pei, and D. Massey, "IP Reachability differences: Myths and realities," in *14th IEEE Global Internet Symposium*, April 2011.

[117] L. Blunk, M. Karir, and C. Labovitz, "Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format," RFC 6396 (Proposed Standard), Internet Engineering Task Force, Oct. 2011. http://www.ietf.org/rfc/rfc6396.txt

[118] J. Postel, "Internet Protocol," RFC 791 (Standard), Internet Engineering Task Force, Sep. 1981, updated by RFC 1349. http://www.ietf.org/rfc/rfc791.txt

[119] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918 (Best Current Practice), Internet Engineering Task Force, Feb. 1996. http://www.ietf.org/rfc/rfc1918.txt

[120] M. Cotton and L. Vegoda, "Special Use IPv4 Addresses," RFC 5735 (Best Current Practice), Internet Engineering Task Force, Jan. 2010. http://www.ietf.org/rfc/rfc5735.txt

[121] G. Huston and G. Michaelson, "Textual Representation of Autonomous System (AS) Numbers," RFC 5396 (Proposed Standard), Internet Engineering Task Force, Dec. 2008. http://www.ietf.org/rfc/rfc5396.txt

[122] Q. Vohra and E. Chen, "BGP Support for Four-octet AS Number Space," RFC 4893 (Proposed Standard), Internet Engineering Task Force, May 2007. http://www.ietf.org/rfc/rfc4893.txt

[123] "The CAIDA Archipelago Measurement Infrastructure," http://www.caida.org/projects/ark/.

[124] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proceedings of the Sixth Annual ACM Internet Measurement Conference (IMC)*, October 2006.

[125] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: mapped?" in *Proceedings of the Ninth Annual ACM Internet Measurement Conference (IMC)*, November 2009.

[126] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz, "Towards an Accurate AS-level Traceroute Tool," in *Proceedings of the ACM SIGCOMM 2003 Conference*, August 2003.

[127] K. Xu, Z. Duan, Z.-L. Zhang, and J. Chandrashekar, "On Properties of Internet Exchange Points and Their Impact on AS Topology and Relationship," in *Networking*. Springer, 2004, pp. 284–295.

[128] B. Quoitin and S. Uhlig, "Modeling the routing of an autonomous system with C-BGP," *IEEE Network*, vol. 19, no. 6, pp. 12–19, November 2005.

[129] D. R. Jefferson, "Virtual time," *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 3, pp. 404–425, 1985.

[130] F. Quaglia, V. Cortellessa, and B. Ciciani, "Trade-off between sequential and time warp-based parallel simulation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 8, pp. 781–794, August 1999.