University of Glasgow | Department of Computing Science

Evaluating Compact Routing Algorithms on Real-World Networks

Project Dissertation

Graham Mooney

A dissertation submitted in part fulfilment of the requirement of the Degree of Master in Science at The University of Glasgow

April 2010

Abstract

Compact routing has shown promise for reducing the forwarding state in Internet-like graphs, without badly impacting traffic flows. This dissertation compares two such compact routing algorithms on real Internet snapshots from router data across 12 years. The results indicate that these algorithms behave consistently over time, and exhibit extremely small forwarding tables with very low path inflation.

Acknowledgements

Colin Perkins and Stephen Strowes for their support, feedback, discussions, helpful pointers, field trips, anecdotes, and, of course, the initial idea for this project.

The departmental support staff for their help through out the year with machine upkeep, and allowing access to copious amounts of server space, hard drives, and docks; their assistance was invaluable. The Embedded, Networked and Distributed Systems group for their help and support through out. The Algorithms group for sanity checking any graph theory and algorithms I produced.

My fellow MSci and MRes students for their company, help, and generally making this year bearable. Finally, the Level 4 and 3 students for their patience while I stole their CPU cycles, RAM modules, and hard drive space.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Internet is composed of a collection of networks that co-operate to provide end-to-end communication for clients. These networks exchange routing information by taking part in a distributed calculation known as a routing protocol. The Border Gateway Protocol (BGP) is the current routing protocol for the inter-domain Internet. This protocol is designed to route packets using the shortest possible path. The Internet is increasing in size resulting in routing table sizes also increasing. This increase is exacerbated by business practices such as traffic engineering and multi-homing. This increase in routing state is beginning to causing hardware problems at present, which will only become worse as the Internet continues to grow.

Solutions are being devised that address the problem of increasing routing table sizes. One such solution is compact routing. Compact routing reduces table sizes substantially at a cost: compact routing may introduce paths longer than the shortest path. A key question to answer is whether this increase in path length is acceptable.

Compact routing schemes have previously been simulated on power-law random graphs [27, 8, 12, 15]. Results have been promising, with path stretch (the multiplicative factor that represents path length experienced divided by shortest possible path) values of ∼1.2, however compact routing has have not been simulated on real world Internet graph representations. This thesis will simulate compact routing on snapshots of the Internet AS topology graph taken between 1998 and 2009. This will demonstrate the viability of compact routing for routing on static Internet topology snapshots, and determine how the performance of these compact routing algorithms varies over these twelve years of data.

Two compact routing algorithms in particular will be analysed, the Thorup-Zwick (TZ) [39] and Brady-Cowen (BC) [8] compact routing schemes. These have been chosen due to their varying approaches to achieving sub-linearly scaling routing tables with minimal path stretch.

The contributions of this work are a program capable of pre-processing a graph to produce routing tables and node labels as defined by the Thorup-Zwick compact routing scheme. Similarly, API's have been developed to aid in the pre-processing of a graph for use with the Brady-Cowen scheme. This API includes components for extra-edge detection, spanning tree construction, connected component detection, and calculation of proximity-preserving labels. Additionally, this work has constructed a distributed network simulator capable of utilising the Thorup-Zwick compact routing scheme. Furthermore, a Brady-Cowen compact routing scheme simulator was implemented capable of simulating the Brady-Cowen algorithm.

This work also contibutes to the state-of-the-art by providing detailed analysis of the Thorup-Zwick compact routing scheme. The effect of reduced routing table size on path stretch for real-world Internet snapshots is determined. The landmark selection algorithm for Thorup-Zwick is analysed to see its effect on routing table size variance, as well as path stretch. Finally, the landmark set sizes and path stretch of the Thorup-Zwick compact routing scheme is analysed over a period of twelve years (1998 to 2009) to determine the potential of this scheme in the future.

This dissertation assumes that the reader has some background information on shortest path routing algorithms, primarily distance vector algorithms. A knowledge of graph theory and its notation is beneficial, particularly for the description of the Brady-Cowen algorithm (Chapter 6).

The remainder of this dissertation is structured as follows: information on the current inter-domain routing protocol (BGP) and problems associated with it are described in chapter 2; related work and literature surrounding compact routing are provided in chapter 3; chapter 4 on compact routing, routing simulation, and the simulator used to perform experimentation; detailed descriptions of the Thorup-Zwick and Brady-Cowen compact routing schemes will be given in chapters 5 and 6, respectively; the details of the simulations to be perfomed, as well as determination of parameters for the compact routing schemes are presented in chapter 7; the analysis of the Thorup-Zwick landmark selection algorithm is provided in chapter 8; analysis of the Throup-Zwick and Brady-Cowen compact routing schemes over the tweleve Internet snapshots are presented in chapter 9; a comparison of Throup-Zwick and Brady-Cowen to BGP is presented in chapter 10; and finally, chapter 11 provides a conclusion to this dissertation, presenting future work and a summary of the results found.

Chapter 2

Background

This Chapter provides background information about the structure of the Internet and how inter-domain routing is currently achieved in Section 2.1. Section 2.2 specifies problems that this existing architecture is experiencing; solutions to these problems will be discussed in Chapter 3.

2.1 Network Structure

Communications between hosts over the Internet is possible with the use of the Internet Protocol (IP). IP addresses under the classless inter-domain routing (CIDR) scheme are split into two parts, the network and the host. Separation of an IP address into its component parts is done using bit mask that is associated with the IP address. To retrieve part the network information from an IP address a bitwise 'and' operation between the IP address and the bit mask is performed. A prefix is a combination of the network component of an IP address along with the length (i.e., number of bits) of the network component of that IP address. Figure 2.1 shows the IP address, network bit mask, and the resulting prefix for a computer on the Department of Computing Science (DCS) network. CIDR allows for multiple machines or addresses to be represented by a single prefix, for example, 4,096 IP addresses are represented by the prefix shown in Figure 2.1. The ability to represent multiple addresses, as well as the merging of smaller prefixes into a larger prefix, is called aggregation.

10000010110100011111000000010101	130.209.240.21	An IP address
11111111111111111111000000000000	20-bit network mask	A bit mask
The resulting prefix:		
10000010110100011111000000000000	130.209.240.0/20	A prefix

Figure 2.1: IP addresses, bit masks, and CIDR notation

For communication to be possible in the Internet, packet switching hardware (routers) in the network must know where, in reference to the current router, the destination of the IP packet is located. These routers are required to produce from the state they maintain the link on which to send this packet. This type of routing is known as 'next hop' routing: a router does not need to know every step in how to get to a destination, only the next step in routing this packet to its destination. Routers get the required state for routing by participating in a distribute algorithm known as a routing protocol.

The Internet is made up of many networks, or domains, which are independently owned, and maintained. Each independently managed network is called an Autonomous System (AS): "a set of routers that has a single routing policy, that run under a single technical administration. To the outside world, the entire AS is viewed as a single entity" [19]. Communication within an AS is known as intra-domain communication, while communication between two or more ASes is known as inter-domain communication.

Since ASes operate independently, the way they interact with each other is determined by business policy. The relationships between ASes falls into one of three categories: customer-to-provider (the customer is charged for the traffic that travels between it and its provider); peer-to-peer (both ASs agree to provide transit for traffic originating from either AS and their customers); and sibling-to-sibling (a peer-to-peer like relationship where data originating from providers is also exchanged). These relationships can be inferred from BGP data using heuristics [17].

It is believed that the Internet's topological structure exhibits a 'power-law' distribution (few networks with high connectivity; many networks with low connectivity), this graph type is known as a power-law graph or 'scale-free' graph [27]. Within the Internet, there are ASes with as many as 2000 connections to other ASes, however, the majority of the ASes maintain one or two connections. ASes can be classified into three types: transit AS (two, or more, connections comprising of zero or more provider(s) or peer(s) links and at least one customer link); singly-homed stub networks (one connection to its sole provider), or multi-homed stub networks (two, or more, connections to at least two different providers). Figure 2.2 provides an illustration of these different types. Any AS that must maintain a full routing table and therefore, has no need for a default route entry (0.0.0.0/0[1]) in their table, is said to belong in the default free zone (DFZ).

The Border Gateway Protocol (BGP-4) [35] is the current inter-domain routing protocol. AS border routers (a router at the edge of the AS's administrative domain) exchange reachability information about destinations they can reach, or have previously heard about from another AS, while revealing no information about the internal structure of the AS. Routers participating in the routing algorithm maintain a Routing Information Base (RIB) and a Forwarding Information Base (FIB). The RIB contains all prefixes the router has learnt while participating in BGP, the next hop to that prefix, as well as a list of AS corresponding to how this prefix reached the current router. The RIB can contain multiple entries for the same prefix[2]; the routing information which the RIB supplies to the FIB is

[1]Matches all IP addresses.

[2]This is because the RIB consists of three parts: RIB-In (contains all paths learnt); RIB-Out (the paths this router will advertise to its neighbours); and the RIB itself (the entries this router has selected for the FIB).

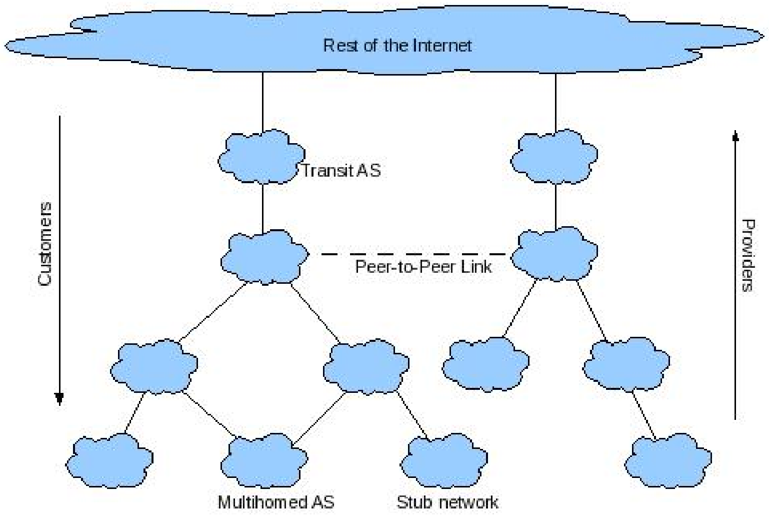


Figure 2.2: AS connectivity example

decided by the AS's local policy. The FIB contains prefixes and next hop information, as supplied to it by the RIB, and performs packet forwarding when a packet arrives at the router. The FIB is known colloquially as the 'forwarding table' and the RIB as the 'routing table'.

The FIB and RIB within a router participating in BGP-4 (or BGP-speaker) use CIDR prefixes to allow aggregation of multiple destinations to one routing table entry. When a packet arrives at a router longest-prefix-matching is performed within the FIB to decide on the next hop for this packet. This is necessary as the FIB will contain multiple entries that match the current destination, to break this tie the entry with the longest prefix is selected as it is a more specific advertisement than the others. For example, if the destination of a packet is 4.1.3.2, the FIB may retrieve two entries, 0.0.0.0/0 and 4.0.0.0/8. The longest prefix here is 8 bits long, and thus 4.0.0.0/8 would be selected and the corresponding next hop would be used to route the packet towards 4.1.3.2.

BGP belongs to the path-vector class of routing algorithms, and is designed to calculate the shortest path to a destination. However, it can be the case that the shortest path between two nodes is not utilised due to politics or business relationships. For example, an AS, a, may have a contractual agreement with one of its neighbours, b, that states it must pay for any traffic that is sent between them. There are two potential routes from AS a to a destination prefix, d: the shortest path to d which traverses b, and another longer path that does not traverse b. Due to AS a being able to choose the RIB route selection policy they may choose to utilise the path that avoids AS b in order to save money, resulting in the shortest path from a to d not being used. This manipulation of BGP state is known as traffic engineering (TE). Traffic engineering is not to benefit routing ability, but to benefit the business running the AS. Multihoming is also a business related practice with the primary goal to increase the ASes network durability and connectivity in case of a link failure. Abley et al. [3] discuss the practices and reasoning behind multihoming in more detail.

2.2 Problems

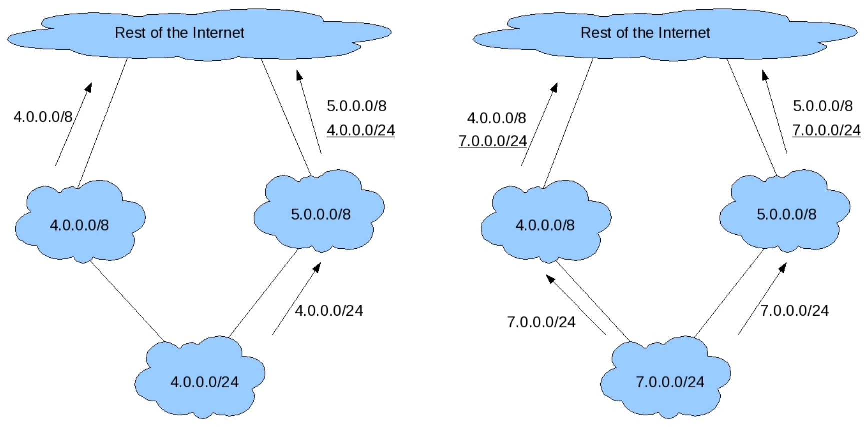
From a routing perspective, AS business decisions can have the undesirable effect of increasing path length (the distance between source and destination) or path stretch (a multiplicative factor which represents path length experienced divided by shortest possible path). On top of this path length and path stretch increase, there may be the additional overhead of more entries in routing tables. For example, traffic engineering is normally performed by manually inserting BGP entries for the desired prefix(es) that need to be redirected due to AS business relationships[3]. Traffic engineering within an AS (intra-domain) have no effect on the global routing system, as these will not be disclosed when advertising routes to other AS. However, if a collective of ASes choose to perform traffic engineering on the inter-domain level, this will result in additional, globally viewable, de-aggregated prefixes (unnecessary (from the routing perspective), more specific entries in the routing table), resulting in growing DFZ routing tables.

The Internet Assigned Numbers Authority (IANA) allow for two types of address allocation to an AS: provider aggregatable (PA), and provider independent (PI). PA space is assigned form a provider AS to its customers, the provider AS is able to advertise one aggregated prefix using BGP to allow traffic to reach both it and its customers. PI space is assigned independent of an AS's provider, and as a result may be from a different prefix range, making it non-aggregateable by this AS's provider requiring the provider to advertise both its own and its customers prefixes.

Another cause of increasing routing table sizes in the DFZ is AS multihoming. A multi-homed AS requires all its providers to advertise the prefixes it serves to allow traffic to be routed via any of its providers. The number of additional advertisements and routing table entries is depending on the type of address allocation for the multi-homed AS, as well as the type of service required by that AS. A multihoming AS with prefixes assigned from PA space, can result in either n or $n-1$ additional advertisements, where n is the number of providers. If the AS's reason for multihoming is redundancy, $n-1$ advertisements are required (one for every other provider). However, if the AS is requesting to be able to send and receive data over all links at once, n advertisements are required to ensure that one path is not selected over another due to that path having a longer prefix. Should the AS's prefixes be assigned from PI space, there is always n advertisements, regardless of type of service, due the each provider having to advertise a prefix outwith its prefix range. Figure 2.3 provides illustration of the differences between a PA and PI prefix allocation with respect to advertisements when an AS multi-homes.

A single-homed (i.e., not multihomed) AS using prefixes not derived from that of its provider's prefixes (e.g., from PI space) will require the provider to advertise the single-homed AS's prefix range, causing additional entries in routing tables.

[3]Another method of intra-domain traffic engineering makes use of multi-protocol label switching (MPLS) [36]



(a) Multihomed AS acquires its addresses from PA space, resulting in one additional advertisement (only if multihoming is used for redundancy, otherwise 2 advertisements are propagated)

(b) Multihomed AS acquires its addresses from PI space, resulting in two additional advertisements

Figure 2.3: Advertisements resulting from a multihomed AS using PA and PI space

This advertisement is necessary so that the single-homed AS can receive data.

Routing tables will naturally grow as more IP addresses are assigned to the ASes of the Internet, each new prefix results in a new entry in the routing tables of routers within the DFZ. The deployment of IPv6 only leads to exacerbate the routing table growth as the new addresses are utilised in the Internet, as these addresses utilise the same routing architecture. Figure 2.4 shows the relative increase of each prefix length over time. All prefixes combined are represented by the top most line, while the '/24' prefix is represented by the second most.

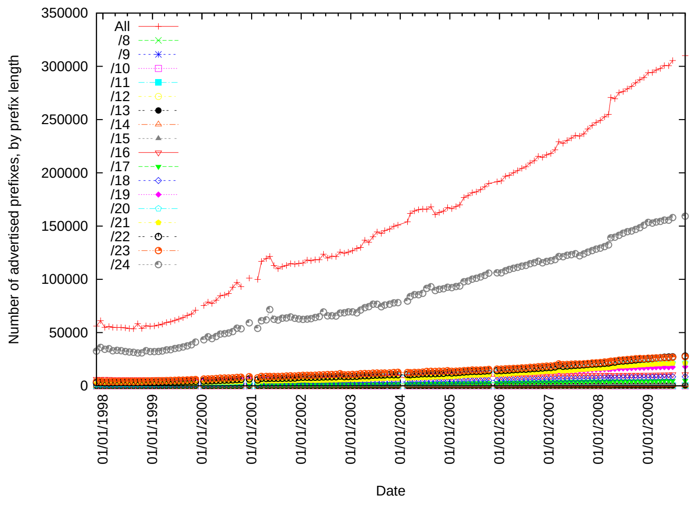


Figure 2.4: RIB entries by prefix length over time

Increasing routing table sizes are beginning to cause a problems for a few ASes, however, it will become a wide-spread cause for concern in for almost all ASes in the near future [29]. Increased routing table sizes impact of FIB size, which in turn causes increased access time when performing longest-prefix routing (as all entries must be compared). Administrators of AS within the DFZ are concerned over increasing running costs for critical routers within their infrastructure (due to upgrades to memory capacity). For these AS it is already becoming an issue.

The increase in routing state not only affects the memory requirements of a router, but the processing requirements also. Routers must recalculate the routes within their RIB when they receive a BGP *update* message from another BGP speaker. BGP update messages are normally sent when there is a change in the network topology (i.e., a link failure) requiring some routers to route traffic differently, or when a prefix is advertised or withdrawn. However, due to router misconfiguration it is possible that a router frequently sends out update messages advertising the availability, or lack thereof, of a route. The misconfigured router may also become indecisive between two or more potential routes to a prefix, producing an update message each time it changes its route choice - this is known as route flapping. Regardless of the source, these frequent update messages cause problems to routes in the DFZ when routing tables get large. Upon reception of an update message, these routes must recalculate the routes for all prefixes - in early routers this could result in packets being un-routeable as the router would not forward packets while calculating the FIB. The rate of BGP update and withdrawal messages is known as churn, and is believed to be increasing [20]. Huston [20] states that router processing time due to an update message can last on "order of hundreds of seconds" at its worst. Huston believes that for updates to be acceptable the worst case convergence time across the network should only be a few seconds.

The increasing stress inflicted on routing hardware, in terms or memory and computational requirements, should be a cause for concern within the networking field. Researchers ([20, 29]) have specified requirements for future Internet routing architectures; they highlight the most important requirement to be routing scalability. Many researchers have published papers on potential Internet architectures that constrain routing table sizes, limit convergence times and lower update communication cost. These potential architectures will be highlighted and discussed in the next chapter.

Chapter 3

Related work and literature

Chapter 2 provided information on the current Internet structure, AS business practices, and the problems experienced by routers in the Internet due to increasing routing table sizes. This chapter will highlight and discuss solutions which may to solve this routing scalability problem. The approaches taken by these researchers fall into three general categories: incremental (or 'dirty slate') solutions; 'clean slate' Internet architectures; and compact routing. Incremental solutions look to start with the existing Internet architecture and propose additions and alterations that would increase, normally only temporarily, the scalability of BGP. Clean slate solutions aim to provide a scalable architecture by creating an entirely new scheme or protocol. Compact routing is a subset of the clean slate solutions, provided here in more detail, as it is the focus of this project.

This chapter is structured as follows: Section 3.1 covers incremental changes to BGP; Section 3.2 highlights some alternative architectures; Section 3.3 details the progression compact routing schemes; finally, Section 3.4 provides a summary of the three types of solutions.

3.1 Incrementally deployable solutions

Incremental solutions are designed to extend the life of the existing inter-domain routing protocol, BGP, as well as the hardware that implement that protocol. These solutions are not long-term solutions to the routing scalability problem, they are instead useful temporary solutions to allow more time to be spent on developing and deploying a new architecture for routing. The two approaches that this section will mention aim to reduce the stress incurred on either the FIB or the RIB of routing hardware.

ViAggre (Virtual Aggregation) [6] is an intra-domain approach for reducing the size of the FIB in an AS's routing hardware. ViAggre achieves FIB size reduction by using 'aggregation points', these are routers that advertise highly aggregated prefixes to the internal routers of the AS. This results in the FIB of every non-edge router within the AS sending packets for a specified destination address (say 4.2.1.3) to a aggregation point that is responsible for that address prefix (e.g., 4.0.0.0/7). When the packet reaches the aggregation point responsible for that prefix, the aggregation point tunnels the packet to the AS's egress point for this destination by encapsulating it using MPLS [36]. This tunnelling is required as all non-edge routers will send the packet towards an aggregation point - creating a packet loop. Figure 3.1 shows this process diagrammatically.

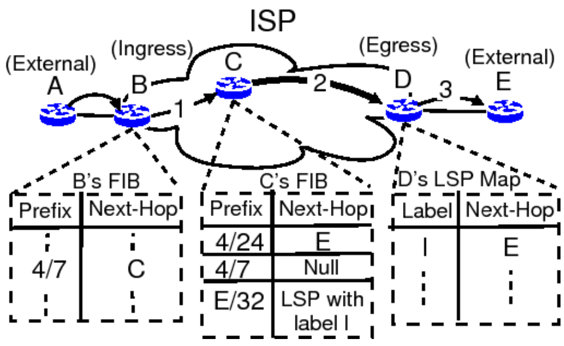


Figure 3.1: Path of packets destined to prefix 4.0.0.0/24 (or, 4/24) between external routers A and E through an ISP with ViAggre. Router C is an aggregation point for virtual prefix 4.0.0.0/7 (or, 4/7). Acquired from [6].

The inclusion of aggregation points into the AS reduces the number of FIB entries in non-edge routers by allowing them to only maintain the necessary forwarding state to route packets to aggregation points (and a few other prefixes, e.g., local addresses). Edge router also benefit from a reduction in FIB state if they utilise the MPLS label to determine the port or link out of the AS if, and only if, it removes the MPLS header prior to forwarding the packet to the neighbouring AS.

A benefit of ViAggre's design is that it can be deployed within an AS by upgrading existing routing hardware with new software. It also has the benefit of providing the implementing AS with immediate benefit (the reduction of FIB state) and so does not rely on other AS's co-operating – AS co-operation is difficult to achieve due to the politics and business arrangements involved. Should an AS choose to deploy ViAggre, this decision is not visible on the inter-domain level, and results in no change to inter-AS communication.

Due to the nature of ViAggre, packets travelling through an AS implementing it, may incur some path stretch, the authors provide evidence to show that this path length increase is low in practice. They authors also note that existing traffic engineering techniques introduce some path stretch while a packet is traversing the infrastructure. ViAggre provides an intra-domain solution to the routing scalability problem; the authors have also suggested this scheme could be extended to allow multiple AS's co-operating to achieve addition state reduction. However, this scheme has proven controversial when discussed in RRG meetings[**?**, **?**], with AS administrators showing heavy opposition to this proposal.

An alternative approach to reducing routing state is to reduce the RIB size. Forgetful routing [22] takes this approach, it reduces the number of alternative routes for the one prefix in the RIB by allowing a router to delete, or forget, the redundant alternative route, thus, freeing memory on the routing hardware. The authors explain that it is possible to delete this information as any alternative route will have advertised to the AS from a neighbouring AS. This alternative route will be that neighbours best or optimal route to that prefix. Should a link failure cause an update message that requires the AS to route using an alternative route, it is able to retrieve an alternative route from its neighbours by utilising the BGP route refresh capability [10] to request a BGP data from a BGP speaker. Upon receipt of the routing information from its neighbouring routers, this router will be able to calculate a new route to the prefix that required updating. It is possible that while waiting for the response to the route refresh message, and while calculating the new route, that packets will not be able to reach their destinations due to old or missing FIB entries.

Forgetful routing reduces RIB memory utilisation at the expense of additional communication costs, and potential loss of packets. It introduces extra network traffic when performing a route refresh, as well as incurring a time delay while it awaits the response. This approach therefore increases the convergence time of routers as they now need to fetch routing information as well as process it. Forgetful routing does not offer a solution to router churn, and has no influence on the frequency of update messages. Router churn will be a serious problem for routers implementing forgetful routing, as they will have to fetch data frequently from neighbours.

Another RIB reduction proposal from O'Dell is called 8+8 [31]. It uses 'aggressive aggregation' to reduce the number of prefix entries in routing tables. This Internet draft is for IPv6 addresses, it suggests that the address space be split up into 2 components: an end system identifier and a routing goop. The routing goop part is equivalent to the network part of a CIDR IPv4 address, however, due to the design specified by O'Dell the routing goops are strictly hierarchically ordered to form a tree structure. This is the key feature that allows for 8+8 to perform aggressive aggregation, as a node only requires routing information of each node above and below it in its sub-tree. The end system identifier is split into 2 parts, the host address and the network address. The host address can be an IPv4 address, a Media Access Control (MAC) address, or Internet Engineering Task Force (IETF) nodeID, and represents a physical device. The network address part simply identifies the network, or sub-network, that a host is attached to. This scheme makes edge routers responsible for appending the routing goop to every packet as it leaves the source network, this has two advantages, firstly, it is possible to correctly identify the source routing goop of a packet (i.e., the destination routing goop cannot be spoofed) and secondly,support for multihoming is trivial and results in no de-aggregation of prefixes in the global routing tables. Multihoming is achieved by selecting which edge router a packet uses as an egress point, and that edge router deals with the routing goop labelling.

A proposal currently being discussed in IETF meetings is the Locator/ID Seperation Protocol (LISP)[16]. This proposal makes use of the locator/identifier split model and aggressive aggregation to produce smaller routing tables. The model uses endpoint identifiers and routing locators to provide the necessary locator/identifier split. To allow this separation, a distributed mapping of endpoint identifiers to routing locators is required. How this distributed mapping is implemented is still being debated, with many suggestions being put forward [28, 21]. This protocol achieves smaller routing tables by performing aggressive aggregation on routing locators, which is the only information BGP speakers will route on. An AS's egress routers are responsible for being able to map the destination endpoint identifier to a routing locator for the destinations network. This allows the benefit of simple end-site relocation (simply change the routing locator for that endpoint) and support for multihoming [34]. This approach is inspired by 8+8, however, LISP is implemented by encapsulation, instead of packet content manipulation.

3.2 Clean-slate Internet architectures

Instead of researching ways to prolong the life of BGP, some researchers have produced solutions that start from a clean-slate. This allows them to produce highly scalable designs. However, many of these designs do not allow for a smooth transition from the current Internet architecture to the new architecture, something that is required for these schemes to be deployed. Regardless of this weakness, these designs are worth investigating.

The current inter-domain routing protocol, BGP, uses path-vector routing to advertise prefix reachability information. Behrens and Garcia-Luna-Aceves [7] offer an alternative approach called link-state vectors. This approach takes the ideas from two existing types of distributed routing information exchange, link-state announcements and path vectors. By combining these, the authors have designed an architecture they believe reduces the effect or route flapping. Link-state announcements suffer less from route flapping, however, do not scale well due to the broadcasting nature of the protocol. To solve this broadcast problem, the authors have instead included link-state information into vectors, removing the need to broadcast. This approach allows for a dampening on the effect of churn, however, it does not solve the routing scalability problem in its entirety.

During the early expansion of the ARPANet, researchers were concerned over routing table sizes in their routers. Kleinrock and Kamoun [24] derived an aggregation technique to reduce the required routing state in the routers. The technique was known as hierarchical routing, and is the same technique used in CIDR and current BGP routers. It works by aggregating or grouping nodes that are further away into one entry, allowing correct routing with lower table sizes. This technique is the core way to reduce routing table sizes in schemes like 8+8, and ISLAY.

ISLAY [23] is a clean-slate approach that defines the Internet topology in terms of new structures called aggregates and destinations. Destinations, as expected, represent the end-points in the network. Aggregates are groupings of destinations and other aggregates. Reachability information is exchanged between aggregates stating which aggregates and destinations they can reach. ISLAY uses this routing information to fill the FIB for each destination within an aggregate, resulting in the FIB not benefiting from any size reduction. This proposal can support multihoming, different protocols (IPv4 and IPv6), and is independent of the underlying protocol addresses (name independent). This scheme is similar to LISP, however, unlike LISP, ISLAY is not currently incrementally deployable.

Hybrid link-state path-vector (HLP) [38] is a protocol that utilises AS relationships to create a routing structure. This approach creates a collection trees using AS customer-provider relationships, each tree communications changes in network topology via link-state announcements. Network topology changes that affect other trees can be communicated via path-vector, however, other trees are only notified of a topology change should that change be considered substantial. A change is considered substantial if the change exceeds a specified threshold that is determined on a per tree basis. For example, it is possible that a link failure happens in a tree, however, there is an alternative root that maintains connectivity – there is no need to notify the other trees. If, however, the tree could not find an alternative root, it would notify the other trees of the failure, allowing them to update their routing tables to reflect the failure. This decision to suppress updates reduces the effect of route flapping, as the router causing the problem will only affect the link-state information of the tree in which it is a member; all other trees remain unaffected by the misconfigured router.

A contrasting approach to those mentioned above, is routing on flat labels (ROFL) [9] which uses a flat name-space rather than a hierarchical name-space. ROFL assigns a new name to every node that joins the network, and routes information to nodes based on this name. ROFL is a routing overlay network, and closely resembles a distributed hash table. Nodes in ROFL are arranged in a ring with each node having a predecessor and a successor (the nodes arithmetically before and after it). Routing is achieved by comparing the current node name with that of the destination and forwarding the packet to either the successor or the predecessor. This process is repeated until the packet reaches its destination. To increase routing efficiency shortcuts across the ring are created, this can significantly reduce the number of nodes in the overlay network that a packet is processed by. However, because the overlay network does not reflect the underlying structure of the network, it is possible that a node and its predecessor or successor be separated by substantial distance, increasing the path traversed by a packet. This approach, by its definition, has a split between identifiers and locations, however, requires no mapping between them (LISP requires a mapping). This approach also supports host mobility and multihoming (achieved by giving a node multiple names).

3.3 Compact routing

Some researchers are developing routing schemes that have routing tables that are sub-linear in size, grow sub-linearly as new nodes are added, and route using packet headers poly-logarithmic in size. Routing schemes of this achieve these properties are known as compact routing schemes. This routing table size reduction is achieved at the expense of increasing path stretch between certain nodes. However, this path stretch is bounded. Gavoille and Gengler [18] proved that minimum value of maximum stretch for sub-linearly scaling routing tables is three. It is therefore possible to have a routing scheme that guarantees that path stretch of a pair of nodes will never exceed three, and still maintain sub-linear routing tables. It is not possible to have a maximal path stretch of 2, or 1, with sub-linear routing tables.

For the duration of this section, unless otherwise stated, n is the number of nodes in the network or graph. Stretch remains as defined in Section 2.2 – a multiplicative factor representing the length of the path travel divided by shortest possible path. Stretch-3 indicates a path three times larger than the shortest possible path. A compact routing scheme is said to be universal, if it can provide routing on any graph type (for example, grid, tree, power-law).

Cowen [13] was responsible for the first universal stretch-3 compact routing scheme. The routing scheme utilises the concept of global landmarks and local neighbourhoods. Routing is performed by routing a packet to the nearest landmark to the destination and then onwards via the local neighbourhood to the destination. This idea is analogous to the postal system. Figure 3.2 shows a node, u, sending a packet to another node, v, in a different neighbourhood by sending it via the local landmark, $L(v)$. The routing table of a node is constructed by maintaining next hop information for its neighbours and the global landmarks, this is all the information required as routing state. A nodes neighbourhood is defined as the k closest nodes to that node (where k is an arbitrary positive integer). Landmark set selection is determined using a greedy approximation to dominating set construction, in such a way to facilitate routing table sizes not exceeding $O(n^{2/3})$ (i.e., number of landmarks + a nodes neighbourhood size $\leq O(n^{2/3})$). Cowen notes that it should be possible to create a stretch-3 compact routing scheme with routing tables of $O(n^{1/2})$ size, and presents this as an open problem to the community.

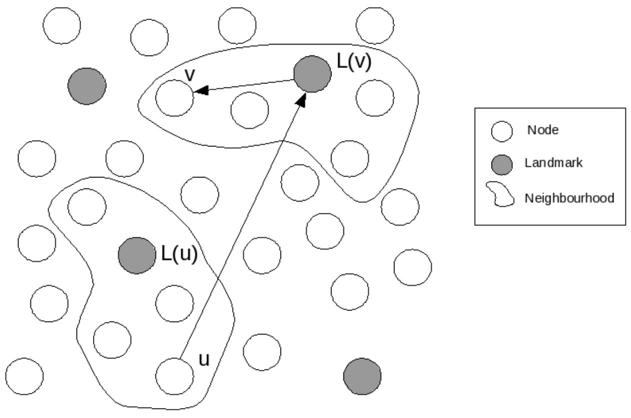


Figure 3.2: Compact routing example

Thorup and Zwick [39] addressed this open problem by producing a universal compact routing scheme with worst-case $O(n^{1/2})$ sized routing tables. This reduction in routing table state is achieved by changing the landmark selection scheme to an iterative process that randomly selects nodes as landmarks. Initially nodes are randomly selected from the whole graph, but in subsequent iterations only nodes whose neighbourhood (cluster in Thorup-Zwick terminology) exceeds a specified threshold are candidates for selection. A nodes neighbourhood is no longer its k nearest nodes, it is instead, any node that the distance between that node and its landmark is larger than the distance to this node. The landmark selection algorithm ensures, due to the neighbourhood threshold size, that routing tables for nodes in this scheme do not exceed $O(n^{1/2})$. They provide proof of this result within the paper.

In addition to providing a universal compact routing scheme with $O(n^{1/2})$ maximum routing state, Thorup and Zwick presented a specialised compact routing scheme for trees. This tree routing scheme has constant time ($O(1)$) forwarding and extremely compact header sizes. These compact headers are achieved by using bit manipulation to encode routing information from the source to the destination. This scheme, despite its being a tree only routing algorithm, is used to construct a universal compact routing scheme by Brady and Cowen [8]. Finally, to round off the spectrum of compact routing, Thorup and Zwick present a universal stretch-$2k-1$ scheme that maintains $O(n^{1/k})$ sized routing tables, where k is any integer greater than 1. This scheme achieves stretch-$2k-1$ by utilising a handshaking procedure between the source and destination nodes. If this handshaking is not performed, the scheme dissolves to stretch-$4k-3$. The authors leave the open problem of achieving a stretch $2k-1$ algorithm without use of a handshaking process.

The participants of an Internet Architecture Board (IAB) workshop for routing and addressing [29] heavily encourage that any future Internet architecture provide a separation between the locator of a destination and its identifier. Arias et al. [5] present a name-independent compact routing scheme with $O(n^{1/2})$ routing table sizes. Mapping between a nodes identifier and its locator is facilitated by use of a distributed identifier to locator dictionary. Nodes are arranged (clustered) such that they can provide a mapping between an identifier and a locator for any node in their cluster. To perform a identifier to locator mapping, a node first queries its local cluster for this mapping. Should this query fail, the scope of the query is gradually expanded until the query is answered. A response is guaranteed as eventually the scope of the query will encompass the entire graph. To route from a source to a destination, the source node first queries for queries fro the nodes locator. This locator is then used to route using a scheme resembling both the Cowen and Thorup-Zwick universal schemes. However, this scheme allows for packet header contents to be changed to allow for the addition of topology-dependent routing information to be added at intermediate nodes. This allows for nodes to utilise the shortest path, should an intermediate node know this information. This name-independent scheme achieves routing table sizes of no more than $O(n^{1/2})$, however has a worst case stretch of five.

A name-independent compact routing scheme that achieves both $O(n^{1/2})$ routing tables and stretch-3 is a scheme from Abraham et al. [4]. This scheme is a landmark based compact routing scheme, however, it uses a graph colouring approach to produce landmarks. The 'colour' of a node can be determined by hashing the node's name. Nodes of the same colour are then grouped into sets. There are two constraints on this grouping, there can not be more than $2\sqrt{n}$ nodes in one colour-set, and every node must have a node of every colour within its neighbourhood. A node's neighbourhood is defined as the k nodes nearest to it, much like the Cowen scheme [13]. A node's routing tables comprise of all nodes within its neighbourhood, and all nodes of the same colour. Routing between two nodes of the same colour is achieved by querying the local routing table and sending accordingly. Routing between nodes of different colours is achieved by querying the local routing table for a neighbourhood node of the same colour, the packet is then sent to that node, routing then proceeds as defined for routing to a node of the same colour.

Both the Thorup-Zwick [39] and Abraham [4] scheme previously mentioned have a mathematical worst case of stretch 3. Krioukov et al. [27] performed experiments on random synthetic power-law graphs using the Thorup-Zwick scheme. They produce formulae for the statistical analysis of the various stretch values possible within these graphs. They also perform simulations of the Thorup-Zwick algorithm on these power-law random graphs (PLRGs). The simulations result in an average stretch of 1.1, i.e., very close to the shortest possible path on average. This suggests it may be possible to have small routing tables with very low stretch impact if applied to the Internet topology. Krioukov et al. strongly suggest that future work within the compact routing area should evaluate the performance of the Thorup-Zwick scheme on 'realistic power-law networks'. The also note that for the advancement of compact routing, dynamic low stretch routing algorithms must be developed.

Brady and Cowen [8] present a compact routing scheme specifically designed for power-law graphs. Their scheme involves use of multiple trees spanning the entire graph. Labeling and routing in this scheme is performed using a slightly modified version of the Thorup-Zwick compact tree routing scheme. The authors prove this scheme to have a worst case stretch that is *additive*. If the shortest possible path between two nodes is 3 hops in length, traffic between these two nodes in an *additive* stretch-3 scheme would exhibit a worst possible path length of 6. The authors also suggest a hybrid scheme merging this additive stretch scheme with the Thorup-Zwick universal compact routing scheme. This merging is performed by performing the necessary calculations for both schemes; the routing table of a node is then determined to be the best possible route that either scheme provides. The resultant scheme is guaranteed to perform better than either scheme on their own, as their will be some paths that are shorter in one scheme than the other, and vice versa. The authors demonstrate this by simulating the Thorup-Zwick, Brady-Cowen, and the hybrid TZ-BC scheme on PLRGs.

It has been previously mentioned that there exist both name-dependent (e.g., Throup-Zwick and Cowen) and name-independent (e.g., Abraham and Arias) schemes that guarantee a worst case (multiplicative) stretch of 3. Krioukov et al. [26] perform experiments between name-dependent and name-independent compact routing schemes to see how the average stretch of each type compare. The authors perform simulations on the Thorup-Zwick, Brady-Cowen, TZ-BC, and Abraham schemes. The result show that name-independent schemes consistently perform worse, on average, than name-dependent schemes. The authors use their result to deduce that compact routing is a more scalable alternative to the general category of locator identifier split schemes. The authors also present the argument that it is not possible to reduce communication costs of a routing protocol (for example, updates or withdrawal messages) to below linear with respect to the network size, and as such, believe that the future of routing protocols relies on *convergence free routing* architectures. Convergence free routing architectures are based on the small world principle[1] [30]. Small worlds and the Internet share similar properties as they both exhibit a power-law. A convergence free routing architecture would allow for routers to successfully perform their task regardless of how much information they have on the network topology as a whole, i.e., it would be possible to route correctly with a partial view of the network.

All previously mentioned compact routing schemes are only viable when applied to the static graph model (i.e., graphs that do not change). This is a major limitation as real-world networks are subject to interruption, and failures. Korman [25] claims to have developed a tree compact routing scheme that is resilient to the addition and removal of nodes with up to degree 2. Korman provides details of the node addition process, however defers the details on node deletion to a future publication.

Following on from earlier work, Enachescu et al. [15] have derived alternative landmark selection algorithms for stretch-3 universal compact routing schemes. The authors have attempted to integrate knowledge of the Internet topology into the landmark selection algorithm to reduce the average stretch. Their resultant landmark selection schemes result in an average routing table size of $O(n^{1/2})$ while maintaining the stretch-3 upper bound. This routing table size is not as good as the Thorup-Zwick scheme, however, the merits of their research is due to their experimental method. Enachescu et al. simulated these new landmark selection schemes over representations of the real-world Internet graph, as well as power-law random graphs. Unfortunately their results to not supply the average path stretch of their algorithms.

Another attempt to use the Internet topology to affect the landmark selection algorithm is demonstrated by Chen at al. [12]. Their algorithm only selects landmarks from what they define to the Internet 'core'. This results in stub networks being ineligible for landmark selection. The authors also evaluated the landmark selection algorithm by simulation on real-world Internet graphs provided by CAIDA [1]. The result of evaluating their landmark selection scheme on the Internet graph was an average stretch of 1.28. This is worse than the Thorup-Zwick scheme's average stretch on power-law random graphs of 1.1, however, the authors suggest a rigorous average stretch analysis of the Thorup-Zwick routing scheme on real-world Internet graphs to provide comparison to their scheme. The authors also propose similar experiments be performed to the Brady-Cowen scheme also.

3.4 Summary

This chapter has highlighted the several different approaches to solving the routing scalability problem that exists in the current Internet architecture. Researchers have suggested proposals designed to prolong the current Internet architecture, however, since these solutions only prolong the life of the current architecture, they only allow other researchers more time to design and evaluate new architectures to replace the current architecture. New architectures include ideas like LISP, HLP, ROFL, and compact routing. These architectures target different sources of the

[1]The small worlds principle is synonymous with the phrase 'six-degrees of separation'.

problems that plague the current architecture. Compact routing looks to target the routing table size by reducing the amount of state each router requires. This is not just a one time routing state reduction; compact routing algorithms are designed to manage growing networks such that the routing state grows at a sub-linear rate with respect to the new additions. For this reason, compact routing appears to be able to make the biggest impact to the routing scalability problem. However, compact routing is not without its flaws.

Compact routing is only a viable option on the static graph model, which is not representative of the Internet. Any compact routing scheme that is destined to become a deployable inter-domain routing protocol, must be able to handle changes in the network topology. Krioukov et al. believe that compact routing does not go far enough, and should instead attempt to integrate some degree of convergence free routing, in order to become truly scalable with respect to routing convergence, as well as routing table size.

Regardless of these points, experiments performed on power-law random graphs have produced favourable results in terms of routing table sizes and path stretch. However, these experiments have not be performed thoroughly on real-world Internet graphs. This is a large milestone for compact routing should it strive to be a candidate for the inter-domain routing protocol; it must first prove its worth. The next chapter outlines how we will attempt to reconcile this point.

Chapter 4

Evaluating compact routing on real-world networks

This chapter discusses the potential approaches for evaluating compact routing algorithms on real-world graphs. It will critique each approach, and select the most viable, providing a strong justification for why this approach is feasible.

4.1 Potential approaches

From the papers outlined in the compact routing section (3.3) of the related work chapter (3), the approaches used by the various authors fell into to broad categories: mathematical analysis, and simulation.

Mathematical analysis is used by many authors to prove the bounds of their algorithms. Cowen [13], Thorup and Zwick [39], Arias et al. [5], and Abraham [4] all preformed this analysis to prove routing table sizes, path stretch bounds and other properties of their algorithms on generic graphs. Krioukov et al. [27] provided a mathematical analysis of the Thorup-Zwick algorithm on power-law graph models. Their analysis allowed for analysis of path stretch on graphs of this type. This approach is close to being able to answer our research question, however, Krioukov et al. highly recommend performing simulations to validate this information [27]. For this reason, as well as the fact that repeating this mathematical analysis will yield no new results. Further, the mathematical models cannot represent the real interconnections of the Internet, a real concern as the aim is to look at the practical aspects of the routing schemes not the theoretical as would be provided by the model. Mathematical analysis is therefore not the best approach, albeit a possible one.

Routing simulators, if provided with an accurate snapshot of the Internet graph, would provide the information to answer the research statement. Brady and Cowen [8], Chen et al. [12], and Enachescu et al. [15] have all used simulations on synthetic power-law random graphs to provide stretch information for graphs of this type. Chen et al. went one step further by utilising Internet snapshots provided by CAIDA [1]. Routing simulators are computationally intensive software programs, often requiring vast amounts of memory. Strowes and Perkins [37] estimate a 32000 node graph simulating BGP-like protocols would require of 120GiB of RAM. This would normally require extremely expensive hardware to achieve, however, Strowes and Perkins have produced a simulator designed to work over a distribute cluster allowing normal computing hardware when networked to perform the simulation.

Provided with the simulator from Strowes and Perkins, as well as acquiring some Internet snapshots from CAIDA [1] it will be possible to perform an experiment that will answer the research questions. The simulator will be open-source software once completed and is designed to be a platform for experimental protocols, allowing for easy integration of those protocols into the simulator.

4.1.1 Internet snapshots

ASes are businesses, and often consider their network's routing state and configuration to be proprietary. It is, however, possible to derive Internet AS-level connectivity by peering with an existing AS who will exchange all their BGP state to the observer, but will advertise no prefixes for that observer. Thus, the observer achieves a full view of the network from that peering location. CAIDA snapshots [1] are the aggregate of multiple such views.

BGP data acquired from peering points, like all BGP data, does not contain information on the relationships between ASes. To derive this information heuristics must be researched. Dimitropoulos et al. [14] and Gao [17] have been developing such heuristics. Dimitropoulos et al. have developed a heuristic that is capable of inferring 94.2% of the AS business relationships in the Internet graph. The authors evaluated the heuristic by presenting the derived relationships to AS administrators and asked them to validate the information. The authors have provided an additional contribution; the output of the heuristic is freely available online for use in research [1]. This information when fed to the distributed simulator would provide a test bed for this research.

Some AS connectivity is hidden from BGP advertisements (for example, sibling-to-sibling links), therefore the inferred AS relationship graph is not complete. Research is currently in progress to increase the accuracy of the inferred graph by including these missing links. Chen et al. [11] are utilising a modified peer-to-peer bit torrent client to observe links that are utilised in sending traffic, but not advertised. Unfortunately, this data has not been released yet, and as such the CAIDA Internet snapshots will be used.

4.2 Summary

A distributed simulator utilising real-world Internet AS-level snapshots from CAIDA will simulate the Thorup-Zwick and Brady-Cowen compact routing schemes. Despite the missing edges in the snapshots, the accuracy of these results will exceed those of a mathematical analysis, and are the best option for answering the research question. The experiment will require the implementation of the

Thorup-Zwick and Brady-Cowen algorithms along with their integration with the distributed simulator. Descriptions of these algorithms follow in Chapter 5 for Thorup-Zwick and Chapter 6 for Brady-Cowen.

Chapter 5

Thorup-Zwick (TZ) compact routing

The Thorup-Zwick universal[1] compact routing scheme employs use of 'landmark'[2] nodes and node 'clusters' to achieve routing between source-destination pairs. Landmarks are globally known nodes, and as such are contained within in the routing table of every node in the graph. A node's cluster comprises of any nodes which it requires next hop information about in order to complete routing of any packets toward that destination. Section 5.1 provides a description of the Thorup-Zwick scheme, section 5.2 explains the implementation process, and section 5.3 summarises this chapter.

5.1 Description

Section 5.1.1 provides detail on how landmarks are determined and the importance of a node's clusters. Section 5.1.2 details the routing procedure of the algorithm. Finally, section 5.1.3 provides a discussion of this scheme.

5.1.1 Landmarks and clusters

The TZ scheme makes use of landmark nodes and node clusters. How these landmarks and clusters are calculated is described here.

Previous attempts to determine landmarks by Cowen relied upon creating a dominating set, using these nodes as landmarks. In the Thorup-Zwick scheme landmarks are selected randomly with a uniform probability from, initially, all nodes. Once these landmarks are selected, each node in the graph is assigned the landmark closest to it. Each node's cluster is then calculated based on these distance, and the distance of all other nodes to the landmark set.

[1]Can be applied to a graph regardless of its structure. For example, trees, or grids

[2]There is no universally acknowledged name for these nodes. Thorup and Zwick [39] refer to these nodes as 'centers'.

```
algorithm landmark(G, s)
landmarkSet = $\emptyset$
potentialLandmarks = V   // V = vertices of G

while potentialLandmarks $\neq \emptyset$ do
{

  landmarkSet $\leftarrow$ landmarkSet $\cup$ sample(potentialLandmarks, s
     )
  // function sample() randomly selects s nodes to become
     landmarks
  for all w $\in$ V do
    // determine every nodes cluster
    Cluster(w) $\leftarrow$ { v $\in$ V | $\delta$(w,v) < $\delta$(LandmarkSet, v)}}
  //if a node's cluster is too large, we must select more
     landmarks
  potentialLandmarks $\leftarrow$ { w $\in$ V | $|Cluster(w)|$ > 4n/s }
}
return landmarkSet
```

Listing 5.1: The TZ landmark selection algorithm [39]

Should this nodes cluster exceed a specified limit that node is then considered a candidate for becoming a landmark in the next iteration of the landmark selection. This algorithm continues until all nodes have a cluster size not exceeding the limit. The threshold and the probability of a node being selected as a landmark are dependent on a value s, such that $1 \leq s \leq n$, where n is the number of nodes in the graph. The probability that a node is selected as a landmark is $s/|potential\ landmarks|$, and the limit for a nodes cluster size is $4n/s$. Thorup and Zwick recommend a value of s such that $s = \sqrt{(n/log\ n)}$, allowing for a maximum routing table size of $6\sqrt{(n\ log\ n)}$ comprising maximum cluster size of $4\sqrt{(n\ log\ n)}$ and maximum landmark set size of $2\sqrt{(n\ log\ n)}$.

Clusters are required to ensure packets can be routed from a landmark to the destination node, without this cluster information packet loops would occur between landmarks and the first hop from that landmark to the destination. This would result in any node not directly connected to a landmark being unable to receive data. This is shown diagrammatically in Figure 5.1, if nodes do not maintain information relating to their cluster, a packet loop may occur between nodes a and c for any packet destined for node d (follow arrows 'step 1' and 'step 2a'). If instead cluster information is maintained at nodes c and b, packets addressed to node d will be correctly forwarded from node c, and b, to d (arrows 'step 1' and 'step 2b').

The cluster of a node is constructed using the equation specified in Eq. 5.1. This equation states that a node v is in w's cluster if w is closer to v than v is to its landmark. The bunch of a node ($B(v)$) is the inverse of its cluster, as shown using the Eq. 5.2. The bunch of a node is simply any node closer to v than v is to its

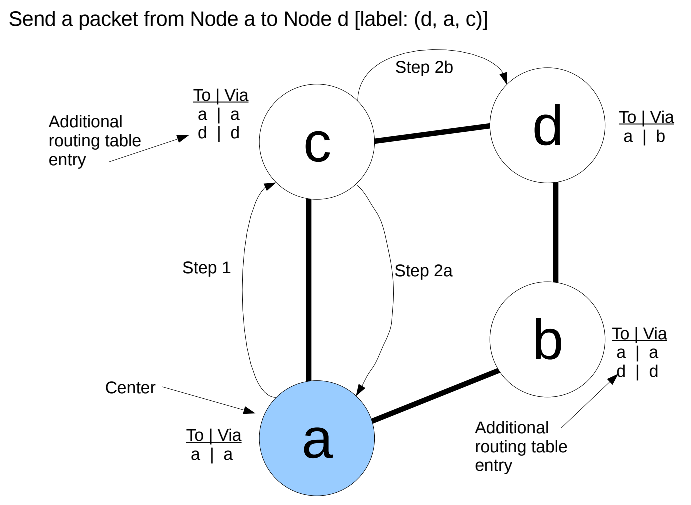

Send a packet from Node a to Node d [label: (d, a, c)]
Step 2b
To | Via
a | a
d | d
Additional routing table entry
c
d
To | Via
a | b
Step 1
Step 2a
b
To | Via
a | a
d | d
Center
To | Via
a | a
a
Additional routing table entry

Figure 5.1: Why do we need clusters?

landmark. The bunch formula is useful for proving properties of the TZ scheme, as well as calculating a nodes cluster (see Section 5.2.2 for details).

$$C(w) = \{v \in V | \delta(w, v) < \delta(A, v)\} \tag{5.1}$$

$$B(v) = \{w \in V | \delta(w, v) < \delta(A, v)\} \tag{5.2}$$

5.1.2 Routing

To route using the TZ scheme you must know the destination's name, the destinations landmark, and the port that landmark uses to route traffic towards the destination; these three components make up the 'label' of a node. Routers within the network make routing decisions based entirely on these labels. For example, in Figure 5.1 the label of node d is (d, a, c) where d is the destination, a is d's landmark, and the next hop for traffic to d is node c. The TZ scheme is a labeling scheme as the nodes in the network are renamed or labeled for the purpose of routing correctly.

To route from a source, a, to a destination, b, using the TZ scheme proceeds as follows: procure the name of b's landmark node, l, route packet towards l (possible due to all nodes containing next hop information for all landmarks). Upon reaching l the packet is routed towards b (possible as nodes in-between l and b will contain next hop information for b). Upon reaching b routing is complete, and the packet is successfully delivered. This is the simplest scenario for this routing scheme resulting in the worst case path stretch on this transmission, reasons for this will be covered in detail in Section 5.1.1. This routing procedure is analogous how the postal service works. Mail for the recipient is delivered to the destinations local sorting office (the landmark), before being routed locally to the destination address.

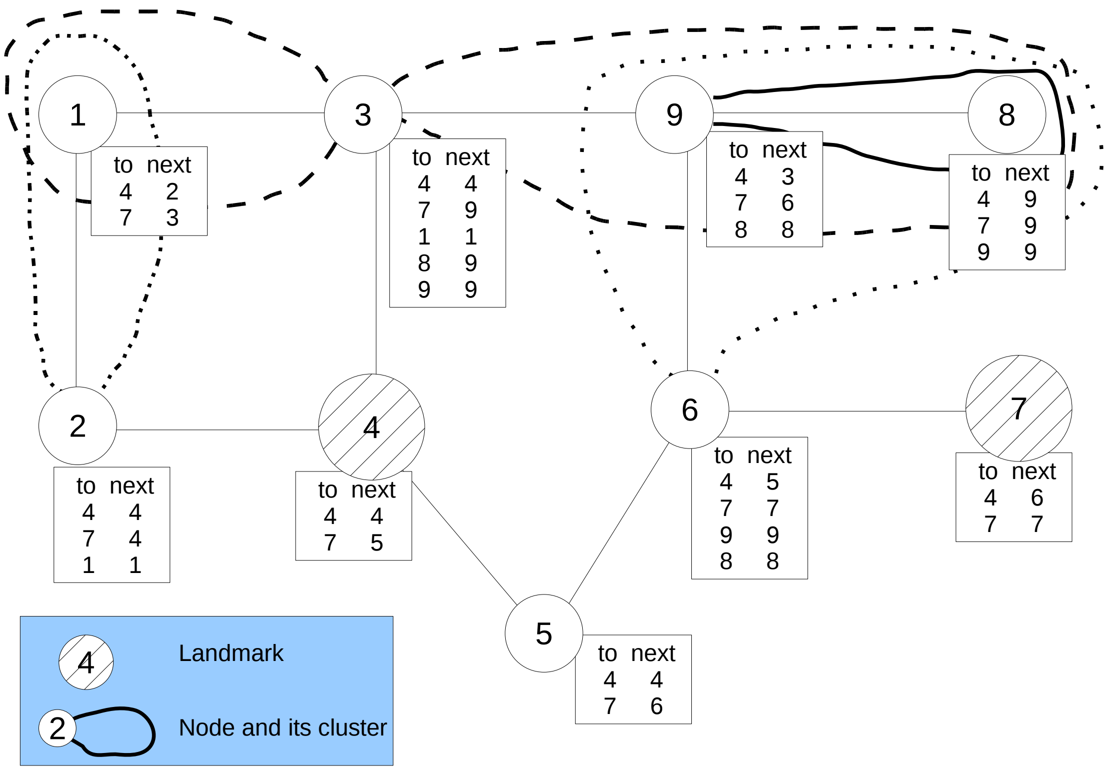


Figure 5.2: A 9 node graph showing TZ routing table information.

Routing via a landmark then onward to the destination can increase the distance travelled by a packet. There is a limit imposed by this due to the nature of this routing procedure, no Thorup-Zwick path is ever more than 3 times the shortest path between a and b (Stretch 3). This is proved mathematically using the triangle inequality and symmetry by Thorup and Zwick [39].

On receiving a packet with label $(x,\ land(x),\ port(land(x), x))$ routing decisions made by a node, say y, are shown in Listing 5.2.

5.1.3 Discussion

Within the Thorup-Zwick scheme, nodes are sent from a source to a landmark then onwards towards the destination, with a worst possible path stretch of 3. Clusters not only allow packets to be routed from a landmark to their destination, but also allow for paths shorter than source to destination via a landmark to be discovered. To illustrate this, consider a packet with source node 4, destined for node 8 (label: (8, 7, 6)) in Figure 5.2. The routing decision at 4 (send toward the landmark (7)) results in the packet being sent to 5. Node 5 follows the same strategy and sends to node 6. Node 6 contains node 8 within its cluster, and routes using this information, similarly for node 9. As you see, the packet destined for node 8 from node 4, never reached the landmark node 7. It is possible that a packet can be sent from a source to a destination without interacting with a landmark node, it certain situations the path exhibited by a packet within the Thorup-Zwick

```
if y = x:
  Reached destination: Done.
else:
  if x is a landmark or in y's cluster:
    look it up x in local routing table and forward it
  else:
    Extract the landmark information from label. [land(x)]
    if y = cent(x):    // y is the landmark of x
      forward using the label's port information. [port(
         land(x), x)]
    else:
      look up cent(x) in local routing table and forward it
```

Listing 5.2: Routing decision in the TZ scheme

routing scheme may match the shortest possible path between the source and the destination (i.e. Stretch-1).

It is possible for the paths exhibited between two nodes to follow different routes depending on the direction travelled. Consider node 2 and node 8 in Figure 5.2. Source 2, destination 8 follows path: $2 \to 4 \to 5 \to 6 \to 9 \to 8$ (length 5); meanwhile, source 8, destination 2 follows path: $8 \to 9 \to 3 \to 4 \to 2$ (length 4). These paths are not identical, in fact, it is possible to have a source-destination pair such that traffic travelling in one direction exhibit stretch-1, while traffic travelling in the opposite direction exhibit stretch-3.

5.2 Implementation

Implementation of the TZ compact routing scheme followed three distinct stages that lead on from one another. These are distance calculations, landmark selection, and simulation of the routing mechanism. These will be summarised in the following sections.

5.2.1 Distance information

It is important that every node know the distance from itself to every other node for the process of calculating next hop information and calculating a nodes bunch and cluster. As such an implementation of Dijkstra's algorithm and an adaption of Breadth First Search were used to calculate distances from each node to all other nodes in the graph, as well as recording nodes traversed to reach that node.

The distance calculation results in a file formatted as shown in Figure 5.3.

This distance information file is read in by the landmark selection algorithm and the routing simulator by the instance of this AS to allow it to complete the next hop information in its routing table.

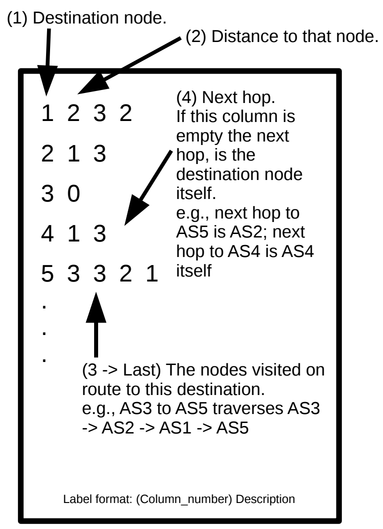

(1) Destination node.
(2) Distance to that node.
1 2 3 2
2 1 3
3 0
4 1 3
5 3 3 2 1
(4) Next hop. If this column is empty the next hop, is the destination node itself. e.g., next hop to AS5 is AS2; next hop to AS4 is AS4 itself
(3 -> Last) The nodes visited on route to this destination. e.g., AS3 to AS5 traverses AS3 -> AS2 -> AS1 -> AS5
Label format: (Column_number) Description

Figure 5.3: The distance file (.dij) file format.

```
for w in V:
  for v in W:
    if (dist(w,v) < dist(land(v), v)):
      w.addToCluster(v)
```

Listing 5.3: Calculating clusters using clusters (see Eq. 5.1)

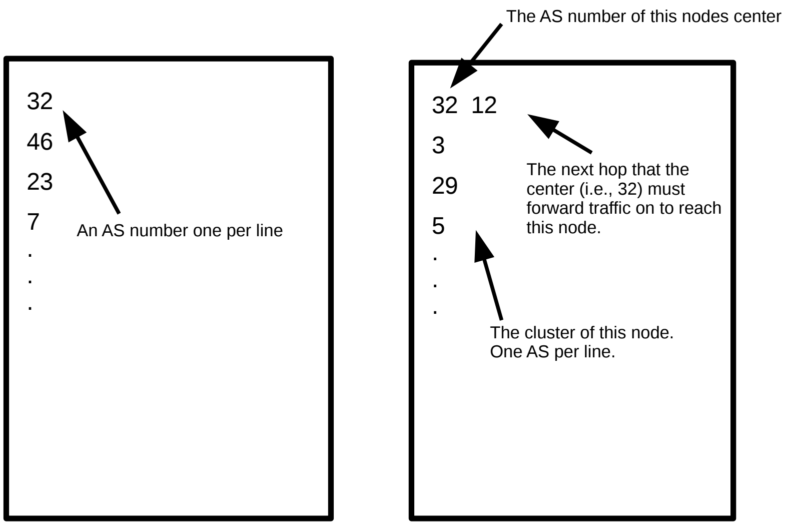
5.2.2 Landmarks algorithm

The landmark selection algorithm follows the pseudo-code outlined in Figure **??**. An interesting take on this algorithm that allows a noticeable performance increase in the clustering section is the use of the bunches equation, rather than the clusters equation. Using the clusters equation results in code resembling Listing 5.3 which is $O(n^2)$ in time complexity, where n is the number of nodes in the graph. However, utilising the fact that the inverse of a bunch is a nodes cluster, we can produce code resembling Listing 5.4 which results in $O(nm)$ where m is between $0 \leq m \leq n$. As the algorithm proceeds m is guaranteed to substantially smaller than n, as more and more landmarks are added resulting in smaller clusters. The bunch of a node is easily computed as a 'filter' operation on all nodes to produce a list of nodes with distance less than the distance between this node and its landmark. This filter operation in many programming languages, for instance Scala, Python and Haskell.

The landmark selection algorithm results in two types of files, one global file representing the landmark set, and one file per node in the graph, representing that nodes landmark, landmark port, and the members of its cluster. The format of these files are specified in Figure 5.4.

```
for v in V:
  bunch = v.getBunch()
    for w in bunch:
      w.addToCluster(v)
```

Listing 5.4: Calculating clusters using bunches (see Eq. 5.2)



32
46
23
7
.
.
.
An AS number one per line
The AS number of this nodes center
32 12
3
29
5
.
.
.
The next hop that the center (i.e., 32) must forward traffic on to reach this node.
The cluster of this node. One AS per line.

(a) The Center file format (one global file). This file represents the ASs that have been randomly selected to be landmarks.

(b) A cluster and center information file (one per AS). This file represents one nodes center and cluster information.

Figure 5.4: Center and cluster file formats

5.2.3 Routing

To modify the existing network simulator to preform compact routing involved modifying the existing routing decision components, and the addition of a mechanism to read in the pre-computed routing table information from disk.

The routing decision was changed to code implementing the pseudo-code from Listing 5.2. The routing table itself was constructed by reading in the distance information and cluster information files for the AS this instance represents, as well as the global landmarks file. The routing information then consists of one routing entry for each cluster member and each landmark, next hop information is procured from the distance information file by using the 4^{th} column for each AS, as highlighted in Figure 5.3.

Control of the test traffic for the simulator is controlled from one host. This host is responsible for parsing the source and destination (represented as a label) from within a file, and then informing the source AS's instance to send data to the destination. The AS instance will then consult the routing information it has and route accordingly. Look-up of the host responsible for a particular AS is done by consulting the master *NameServer* and maintaining a cache of recently used AS to host mappings to increase the throughput of the *TrafficGenerator* class. The traffic file is sorted numerically by AS number to allow for dispatching of all traffic descriptions for a particular AS to be completed at once. This provided a substantial decrease in processing time.

5.3 Summary

The Thorup-Zwick compact routing scheme is a landmark based routing scheme. It utilises landmark nodes as well as clusters to construct a worst-case stretch-3 routing algorithm. The pre-processing for the Thorup-Zwick scheme, landmark selection algorithm, was implemented alongside a simulator capable of routing this compact routing scheme.

Chapter 6

Brady-Cowen (BC) compact routing

This chapter describes, in detail, the Brady-Cowen tree based compact routing scheme. Section 6.1 explains the concepts used by the Brady-Cowen algorithm, as well as the other schemes it utilises to produce the final *additive* stretch routing scheme. Section 6.2 details the implementation of the Brady-Cowen algorithm to be used within the simulations.

6.1 Description

The Brady-Cowen routing scheme creates several spanning trees out of a given graph. These trees are then processed using a routing algorithm provided by Thorup and Zwick [39] for very efficient routing in tree graphs. Each tree is also processed such that the label assigned to a node contains information to allow the distance to any other node to be calculated. Each of these steps, and how they related to routing, are explained in the remainder of this section.

6.1.1 Routing table construction

The Brady-Cowen creates a forest of spanning trees, and creates one routing table entry per tree. To create this forest (denoted τ) they first creating a spanning tree that encompasses the entire graph (G) originating from the node with the highest connectivity (broken lexicographically if there are multiple candidates). This node of highest connectivity is denoted by h, and the spanning tree with h as the root is denoted T_0. The remaining trees of the forest, τ, are constructed by use of the concept of a d-core (I), d-fringe (F), connected components and finally, 'extra' edges (e_G, e_F, *or* e_I, depending on where these extra edges are located),.

First, d is any positive, even integer. The *d-core* of a graph consists of any node, and any edges between those nodes, at most distance $d/2$ from h, the highest degree node. This results in a core of diameter d. The *d-fringe* of a graph consists of any node not in the d-core, and all edges between those nodes.

When considering the fringe and the edges it shares in common with the spanning tree T_0, there will be nodes that are physically connected in the graph, however are no longer connected in the fringe due to the spanning tree being a subset of all edges in the full graph. A collection of nodes that are connected in the original graph but are now disjoint have edges reinstated to form a *connected component*. However, the missing edges of a connected component are restored such that connectivity to all nodes in the connected component is possible, but there are no cycles within that connected component of the graph, i.e., it must also form a tree. Finally, *extra edges* are any edge within a connected component that would cause it to become acyclic.

Figure 6.1 provides a diagrammatic view of the concepts of the d-core, d-fringe and connected components.

The Brady-Cowen scheme then creates one tree per connected component, and one spanning tree per extra edge, such that the tree originates from that edge. Should the number of extra edges be less than five, Brady and Cowen highly recommend using a heuristic: If there are less than five extra edges, randomly select up to five edges from fringe to act as extra edges. This heuristic allows for lower stretch paths to be achieved than if only the tree T_0 and connected components are used, as they proved within the Brady-Cowen paper.

The forest, τ, is simply: the initial spanning tree, T_0; all the spanning trees derived from the extra edges, or the heuristic; and all the trees formed by connected components.

These trees are used to provide the routing within the Brady-Cowen scheme, only after being processed with the Thorup-Zwick tree compact routing scheme [39], and Peleg's Proximity-Preserving labels [33]. These concepts are detailed below.

6.1.2 Thorup-Zwick compact routing scheme for trees

The Thorup-Zwick tree compact routing scheme [39] is a re-labeling scheme that can achieve constant time forwarding of packets. The key to its efficiency is this labeling process. This section outlines this labeling scheme, as well as the information required by a router and packet headers utilising this scheme.

This scheme utilises a variable b, which can be any positive integer. This variable is used in determining if a node is consider a light node or a heavy node (alternatively, a light or heavy child). Each node has a weighting (s_v), that is the number of descendants it has, including itself. A node is considered heavy if it shares at least $1/b$ nodes in common with its parent. For convenience the root of the tree is considered to be heavy. Each node has a light level, this is the number of light nodes in between the root and the current node, itself included.

Given this information, nodes are given a label by traversing light nodes before heavy nodes using depth-first search ordering.

The routing table contained within a node is made up of the following parts. Its label as determined by depth-first search (v); its light level (l_v); the number of its largest descendant (f_v); the number of its first heavy child (h_v); the number of

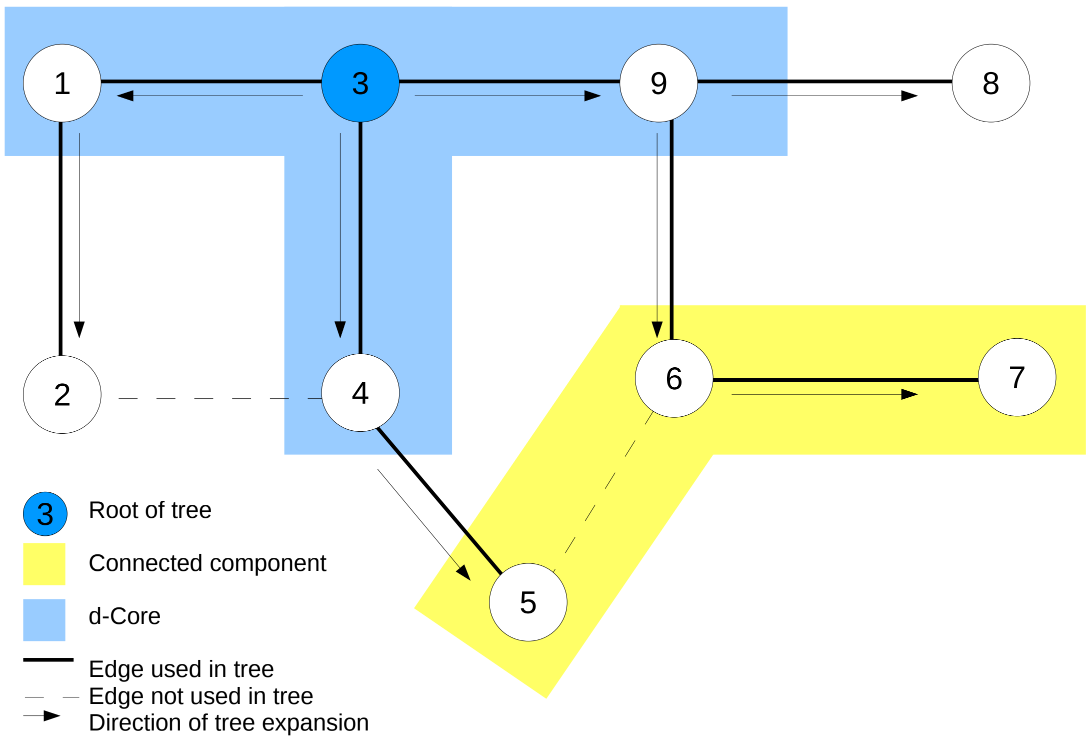


Figure 6.1: A 9 node graph showing BC concepts

heavy children and an enumeration of those heavy children encapsulated into one array (H_v); and finally, a list of ports to the heavy children contained in another array (P_v). The correspondence between the heavy children array and the port array is such that H[1] and P[1] give the name and port number of the same heavy child, H[2] and P[2] are the same, and so forth; while H[0] contains the number of heavy children, and P[0] is the port information to reach this nodes parent.

A packets header in this scheme is the destinations label (v), and the port information to be used at each node that needs to forward this packet to a light child (the array, L_v). Should a node, when making a routing decision require to pass to a light child, the node will use its light level (l_v) as an index into the array of port information contained in the packet header. This will result in the forwarding port information being acquired.

The routing process in its entirety is described as pseudo-code in Listing 6.1.

From the process listed, it is clear to see that a nodes routing decision is clearly independent of the size of the network, resulting in a constant time forwarding algorithm.

6.1.3 Proximity-preserving labels

The Peleg proximity-preserving labeling scheme [33] is a processing step that produces labels for nodes in a tree, such that the node can be identified uniquely,

```
Packet destined for v with header (v, Lv) arrives at node w
    with routing table (w, fw, hw, Hw, Pw, lw)

if w = v:
   done // we are the destination
else:
   if v is not in the range (w, fw):
      // it is not one of w's descendants
      forward it to the parent using Pw[0]
   else if v is in the range (hw, fw):
      // it is a descendant of one of w's heavy children
      Search Hw for the appropriate heavy child,
      and use the corresponding entry in Pw to forward it.
   else:
      // it is one of our light children
      Use Lv[lw] to retrieve the port from the packet
         header,
      to forward to the correct light child
```

Listing 6.1: TZ in Trees routing decision

and more importantly, distance information can be extracted from the label to determine the distance between that node and any other node in the tree.

The scheme is a repeated two stage process, first find the *separator* of a tree, then perform a partial labeling of all the nodes. The separator of a tree is the node that if it were to be removed from the tree, creates multiple sub-trees containing less than half the nodes of the original fully connected tree. The next iteration of the labeling process is performed on the resultant sub-trees created from removing the separator. A node is considered fully labeled once it has been processed as a separator, and the labeling process is complete when all nodes are fully labeled.

On finding the separator, each node is assigned a tuple containing: the separator's name; the distance of the current node from the separator; and which sub-tree this node is in (the separator receives tree number zero, while the remaining trees are arbitrarily numbered). For example, should the separator of the tree be node a, the tuple assigned to a is $(a, 0, 0)$; A node b of distance 2 away from node a, could be assigned the tuple $(a, 2, 3)$ assuming that the separator a caused at least 3 sub-trees to be formed.

A nodes label is simply all of the tuples assigned to it from the start of the labeling process until this node itself is selected as the separator, maintained in order of occurrence.

Given two nodes' labels it is possible to calculate the distance between those nodes. This is possible as each label contains the distance information to the separator during each stage of the de-construction. To calculate the distance between any two nodes we simply need to find a point in time when they were on different sides of a separator or, when one node is the separator. These are easily determined;

```
Given a Label(u) and Label(v), where Label(foo) = list of (
   separator, distance, sub-tree) tuples.

if size of Label(v) = 1:
   return distance field in first tuple of Label(u)
if size of Label(u) = 1:
   return distance field in first tuple of Label(v)

otherwise:
   examine the tuple at the head of both Label(u) and Label
      (v)
   if u and v are in different sub-trees:
      return the sum of the distance fields

   if u and v are in the same sub-tree:
      recursively call this function using the tail of both
          Label(u) and Label(v), returning the result
```

Listing 6.2: The Peleg distance calculation

either case occurs when the nodes are assigned different sub-tree number during an iteration. Once we find this iteration, we simply add the distance fields from the tuples originating from that iteration of the labeling process. Listing 6.2 provides pseudo-code for the recursive distance calculation function.

6.1.4 Routing table construction, revisited

The first stages of processing the graph was to create a forest of trees, $\tau = T_0, T_1...$. Each of these trees is subjected to processing by the Thorup-Zwick tree compact routing scheme, and the proximity-preserving labeling scheme. As previously stated, each node will contain a routing entry for every tree it is a member of. This entry is simply the tree number, Thorup-Zwick routing table appended with the proximity-preserving label. The labels used for routing in the Brady-Cowen scheme are concatenation of the tree number, the Thorup-Zwick label and the proximity-preserving label from every tree.

6.1.5 Routing

The Brady-Cowen scheme provides routing services by using the routing scheme provided by Thorup-Zwick (Listing 6.1). To allow for this, the source of a packet must calculate the best tree to use for the transfer. The proximity-preserving labels allow for a source to calculate the distance between the source and the destination using the local routing table and the destinations label. The source then selects the tree that has the shortest distance traversed from the trees the source and destination share. Once this tree is selected, the traffic is routed using the Thorup-Zwick routing table for the selected tree. Only the source of a packet need perform the calculation, the intermediate nodes and the destination only require the tree number and the Thorup-Zwick label corresponding to that tree, maintaining the constant time forwarding.

6.2 Implementation

Implementation of the Brady-Cowen compact routing scheme can be dissected into finding the initial spanning tree, gathering the *d*-core, and therefore the fringe. Processing of the fringe to determine connected components and extra edges. If necessary, utilisation of the heuristic is applied to ensure sufficient number of trees. Finally, pre-processing is finished by applying the compact routing for trees by Thorup-Zwick and the proximity-preserving labels. All of the key implementation details of the stages are outlined in the following sections.

6.2.1 Spanning tree

The construction of spanning trees is performed by used of a breadth-first search approach, returning the tree as a result. This function is heavily utilised: initially to construct the first spanning tree, and then the core itself; to determine and process connected components; and generate the spanning trees cause by extra edges or the Brady-Cowen heuristic. The fringe of the graph is determined by performing the set difference between the graph and the nodes in the core.

6.2.2 Connected components and extra edges

Connected components are determined by taking the fringe generated by the process mentioned above, and iterating through the nodes creating spanning trees until all nodes have been included in a spanning tree, or have been determined to be a lone node just off the core. The spanning trees created are the tree used for each connected component, and will require processing by Thorup-Zwick and the proximity-preserving schemes.

Extra edges are any edges between nodes in a connected component that are not in the spanning tree created while determining the component. To determine these extra edges, the edges used in the tree are removed from a representation of the connected component, the remaining edges are automatically considered a extra edge. Since starting a spanning tree using this edge, is the same as if a spanning tree is created starting from a node connected to this edge, a node from one end of the edge is remembered and a spanning tree will be created from this point.

6.2.3 Thorup-Zwick tree routing

All the spanning trees created in the processes highlighted above are then processed using the Thorup-Zwick scheme as discussed in Section 6.1.2. This involves the use of a depth-first search and labeling recursive function. Unfortunately the implementation of this function is not 'tail-recursive', the last line of the function is not a recursive call. Should this function be tail-recursive the scala compiler would create iterative code for this function – reducing memory footprint in terms of stack utilisation. At present this code when utilised requires a very generous allocation of stack memory from the Java Virtual Machine. A modification to this code would be to either make the function tail recursive, if this is at all possible, or to produce an iterative counter part.

The result of this function is the fully specified Thorup-Zwick in trees routing table and routing label for every node in the specified tree.

6.2.4 Proximity-preserving labels

The proximity-preserving labels function makes interesting use of a post-order traversal and knowledge of the number of nodes in a graph. Each child after being traversed will inform the parent of the number of descendants it has, allowing the parent node to label each edge with the number of nodes that can be reached using that edge. In addition to this, once all children have been processed, it is possible for the parent node to determine how many nodes are reachable using the edge to its parent.

On completion of the traversal every node has edge weightings for every edge connected to it. The separator is determined easily now. The node with the minimum difference between all its edge weights is the separator. In the case of two nodes being candidates for the separator role, the first one processed is used.

The traversal process is then repeated for each sub-tree produced by using the separator from the previous iteration. The process ends when there are no more sub-trees.

The runtime of the processing steps mentioned in the past sections is heavily influenced by the number of connected components and extra edges detected. As will be illustrated in the d-core analysis in Section 7.1.2, the runtime to process a graph can be in the order of minutes, or may exceed 2 weeks.

6.2.5 Routing

The modification required to complete a Brady-Cowen simulator were similar to that for the Thorup-Zwick simulator. Each AS would have to read in its routing table from the disk, this required modification to the `AutonomousSystem` class. The simulator must be able to make routing decisions on packets, this was performed by creating a new class to hold routing table information as well as perform the routing logic outlined in Listing 6.1. Finally, the simulator must also be able

```
// tree_num (name, largest_descendant , largest_heavy ,
   heavy_array , port_array , light_level) peleg_label ...
0 (6302,6302,6303,[0],[209],2) (701,2,449), (209,1,237),
   (26570,0,0)
1 (11800,11800,11801,[0],[209],3) (20485,7,1), (1239,2,604)
   , (209,1,225), (26570,0,0)
```

Listing 6.3: An example routing table for a node in 2 trees

```
// tree_num (name, light_path) peleg_label ...
0 (6302,[209,26570]) (701,2,449), (209,1,237), (26570,0,0)
1 (11800,[1239,209,26570]) (20485,7,1), (1239,2,604),
   (209,1,225), (26570,0,0)
```

Listing 6.4: An example routing labels for a node in 2 trees

to send traffic, this required the generation of a new `TrafficGenerator` class.

To facilitate an AS to read its routing table from the disk, the necessary files must be distributed across the network. This required modification of shell scripts to distribute the files required by a specific AS to the machine running that AS instance. After distribution, parsing of the files is all that is required by the `AutonomousSystem` class. The file format for a routing table is specified in Listing 6.3.

The routing decision was implemented as described by the Thorup-Zwick scheme in Listing 6.1, there were no significant deviations from the routing decision described.

The traffic generation required reading of a specific traffic file, and communication with the source AS of the traffic. This was performed by performing a look up in the `NameServer` and communicating directly with the machine responsible for that AS. The source AS is then supplied with the label representing the destination. The source AS then performs its routing decision based on its routing table and the supplied label.

Listing 6.5 shows an example traffic file entry. This states that AS1 should send a packet using tree 0 to TZ label (6302,[209,26570]), the remaining tuples are the Peleg proximity-preserving label which are not required for routing, however, are left in for sanity checking purposes.

The Brady-Cowen simulator benefited from the speed-up mentioned in the Thorup-Zwick routing section (Section 5.2.3). In fact, the Brady-Cowen simulator is the reason the optimisation was discovered; its input is sorted by default. The traffic file for the Brady-Cowen simulator does not require a return packet to be inserted to test the reverse path; in Brady-Cowen the reverse path will always equal the forward path, they will be routed through the same tree.

```
// source_AS tree_num (destination_name, light_path)
   peleg_label...
1 0 (6302,[209,26570]) (701,2,449), (209,1,237),
   (26570,0,0)
```

Listing 6.5: An example traffic entry used by the TrafficGenerator

6.3 Summary

The Brady-Cowen compact routing scheme utilises the Thorup-Zwick compact routing scheme for trees, as well as the Peleg proximity-preserving labels. The Brady-Cowen scheme creates a collection of trees spanning various areas of the graph, then applies these two algorithms to those trees to provide routing services. Determination of which tree provides routing is calculated from the label and a nodes local routing table, the tree that produces the shortest path according to the proximity-preserving labels is utilised for traffic. This process was implemented into a Scala program to allow for this pre-processing to take place, before being used within our simulator.

Chapter 7

Experiment details

The experiment aims to evaluate the performance of the Thorup-Zwick and Brady-Cowen compact routing algorithms on snapshots of the Internet AS topology taken between the years 1998 and 2009. Primarily, the experiment looks to see the effect on routing table sizes and path stretch when running the Thorup-Zwick and Brady-Cowen compact routing schemes on these topology snapshots, as compared to BGP routing using shortest paths..

The fundamental variables of the experiment are the choice of compact routing algorithm and the topology snapshot being simulated. The Thorup-Zwick and Brady-Cowen compact routing schemes have been discussed in chapters 5 and 6.

The topology snapshots are acquired from CAIDA [1]. This data is provided as annotated pairs of AS numbers, representing an edge between those two AS as well as the inferred relationship between the ASes (i.e., peer-to-peer, customer-to-provider). This data is collected by aggregating the views observed from multiple peering points within the Internet. Heuristics are then applied to it to determine the AS relationships [14]. This data has its limitations, it is not possible to view all edges within the Internet topology due to ASes not advertising all paths available to them. Attempts have been made to find and add these additional edges to snapshots by utilising peer-to-peer clients [11].

The Thorup-Zwick algorithm has an element of variability within it, namely the landmark selection algorithm. Due to the random nature of this algorithm, some pre-experiment analysis to determine if there are patterns in the selection of the landmark set will be performed. This will allow for analysis into the ability of the landmark selection algorithm to cause bias. The results of this analysis are presented in Section 7.1.1.

The Brady-Cowen algorithm has two variables defined within it. Brady-Cowen recommend a value of $b = 2$ for the Thorup-Zwick compact routing in trees algorithm. However, they do not recommend the value of d. They do state that the size of d-core should be chosen to ensure less than ten extra edges in the d-fringe. To this extent, some experimentation with different values of d is required to determine the extra edge count of the fringe. Details of this analysis can be found in Section 7.1.2.

To further evaluate the effects of the landmark selection algorithm, fifteen different landmark sets from the same snapshot (March 2004) will be compared to determine if the landmark selection algorithm can affect the stretch values as well as the routing table sizes. Fifteen landmark algorithm instances will provide enough output data to observe patterns emerging between the instances. The evaluation will be done by simulating the fifteen landmarks sets generated on the same snapshot, with identical traffic sent through each instance. The traffic sent through the simulator will be structured such that every node will send to 1% of the the remainder of the graph. The 1% of traffic from each node will be randomly selected before the running of any landmark set instance. Each landmark set instance will then be subjected to the exact same pairs. A packet returning in the opposite direction will then be sent. This will allow for a comparison of the forward versus reverse stretch of each AS pair, since it is possible that these paths are different in the Thorup-Zwick scheme.

Only 1% of the possible destinations within a snapshot are simulated as this should allow for a preliminary view of the performance of the Thorup-Zwick and Brady-Cowen compact routing schemes within the time constraints of the project. With this quantity of traffic it is possible to simulate both algorithms on the twelve yearly snapshots. If the project were substantially longer in duration, all possible pairs of source and destinations would be simulated to provide a full analysis of the compact routing schemes on these snapshots.

After performing the analysis of the potential effects of the landmark selection algorithm on any results, it will be possible to use other snapshots to determine the stretch values. These snapshots will be subjected to similar tests. Each node will sent to 1% of the remainder of the graph, and a response packet will be sent back. The Brady-Cowen scheme will have the same experiment performed on it, using the same pairs for that year, however, only traffic in one direction need be simulated as the forward and reverse paths in the scheme will utilise the same tree for routing. By performing this analysis on the different snapshots it will be possible to observe the performance of both schemes in relation to how the Internet has grown and changed over the 1998 to 2009 time period.

Finally, as a contrast to randomly selecting destinations, an experiment will be conducted on graph snapshots such that every AS will send data to ASes derived from the list of the top 100 websites in terms of traffic volume. This list is acquired from Alexa (`http://www.alexa.com/topsites`). The conversion from URL to IP prefix and AS number was performed by querying the Domain Name System (DNS), and use of the whois service provided by Team Cymru (`http://www.team-cymru.org/Services/ip-to-asn.html`). This will provide an alternative view of how the Internet would be affected by compact routing deployment by simulating using more realistic traffic patterns.

7.1 Understanding compact routing parameters

The following sections discuss the results gathered before the simulations began. Section 7.1.1 details the results of performing multiple landmark selection runs on the 20040301 CAIDA AS-level Internet snapshot. This aimed to find an patterns in terms of the selected landmarks. Section 7.1.2 details the analysis required to find suitable values for d to obtain less than ten extra edges in the d-fringe of the various snapshots.

7.1.1 Landmark selection algorithm for the Thorup-Zwick algorithm

The landmark selection process was completed 161[1] times on the 2004 CAIDA AS-level snapshot. This is a sufficient number of runs to see if a pattern is emerging from the landmark sets calculated, as well as being possible within a constrained time frame. The March 2004 snapshot was selected for its size, it is not extremely large, and it is not too small. This allows for a compromise in processing time and the results acquired from the analysis.

From the landmarks selected a handful of ASes were frequently selected as a landmark node. These nodes had node degrees substantially higher than the mean node degree. The mean node degree for the 2004 snapshot is 4.5 neighbours, nodes that are frequently selected by the landmark selection algorithm exhibit a node degree considerable higher than this average (normally exceed 100 neighbours). This is an interesting, non-obvious side-effect of the landmark selection algorithm itself which helps to explain the low stretch factors observed later in section 9.1.1. The reason they are so frequently chosen is as follows: every node within the graph is a candidate for landmark selection; later iterations of the landmark selection algorithm only choose landmarks from the nodes that exceed the cluster constraint (i.e., the nodes cluster side exceeds the value $4n/s$); nodes with high node degree are most likely to exceed this value.

Figure 7.1 shows the relationship between node degree and frequency of being chosen as a landmark. It is clear that nodes with high degree stand a greater chance of being selected as a landmark. Nodes that appear in greater than 20% of the landmark sets have node degree greater than 100 (with a few exceptions). The landmark selection algorithm does still have a high percentage of variance in landmark set, of the 16655 possible nodes in the 2004 snapshot, 7747 different nodes were selected. However, there are certain nodes that are selected a substantial amount of the time, that suggests that the variation between the 15 landmark sets we have choose to simulate will be minimal. The results of that simulation are deferred until chapter 8.

The nodes selected most frequently (i.e, in more than 80% of the landmark selection instances) by the landmark selection algorithms are shown in Table 7.1. This is a superset of the ASes known to be "Tier 1" AS [2]. A Tier 1 AS is defined as an AS that does not act as a customer for any other AS, as a result they make all of their data exchanges with no cost incurred. This list is heavily debated, with many people refuting the idea of a Tier 1 AS outright. As such, acquiring a list

[1]150 calculations dispatched for this experiment, 4 failures due to machines being rebooted. The 146 successful runs combined with 15 for the landmarks experiment that had already been performed produces 161 landmark sets.

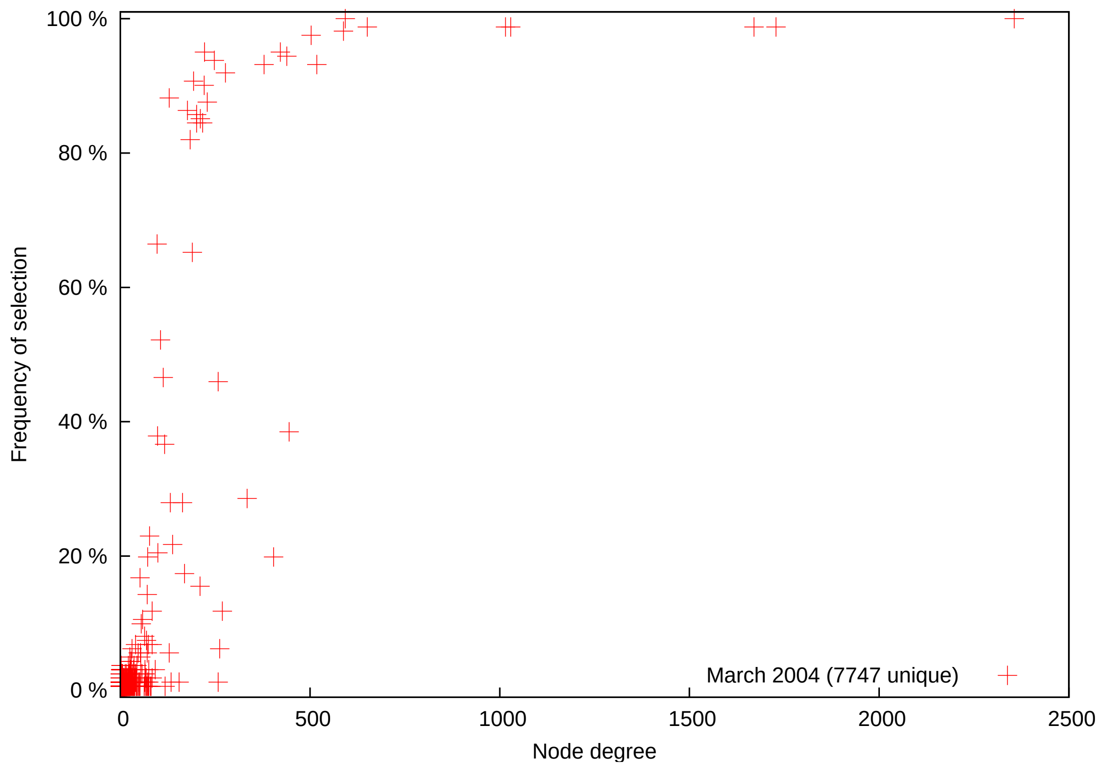


Figure 7.1: Occurrence rate of AS in landmark set versus node degree

of Tier 1 ASes from a reliable source is difficult. Table 7.2 contains a list of Tier 1 AS as acquired from Wikipedia (as also used by Oliveira et al. [32]). As can be observed, the frequently selected landmarks by the landmark selection algorithm closely match up with the list of so called Tier 1 AS.

7.1.2 Deriving values of 'd' for the Brady-Cowen algorithm

The Brady-Cowen compact routing scheme requires that the d-fringe of a graph contain no more than 10 extra edges. However, the value of d required to achieve this aim is unknown as it is dependent on the topology of the network. As such, core and fringe size analysis has been performed on all twelve snapshots from 1998 to 2009 to find the number of extra edges contained within that snapshot at various values of d. The aim is to find the value of d that results in less than ten extra edges in each snapshot. The results of these analysis can be found in the tables below. As stated in the Brady-Cowen description (chapter 6) d must be a positive even number, it represents the diameter of the core. This diameter is constructed by including every node up to $d/2$ distance from the highest degree node.

AS Number	AS Name or Description (* indicates ‘Tier 1’ AS)
701	UUNET - MCI Communications Services, Inc. d/b/a Verizon Business *
2914	NTT-COMMUNICATIONS-2914 - NTT America, Inc. *
7018	ATT-INTERNET4 - AT&T WorldNet Services *
3549	GBLX Global Crossing Ltd. *
3356	LEVEL3 Level 3 Communications *
209	ASN-QWEST - Qwest Communications Company, LLC *
1239	SPRINTLINK - Sprint *
3561	SAVVIS - Savvis *
6461	MFNX MFN - Metromedia Fiber Network
3303	SWISSCOM Swisscom (Switzerland) Ltd
1299	TELIANET TeliaNet Global Network *
13237	LAMBDANET-AS European Backbone of LambdaNet
4637	REACH Reach Network Border AS
702	AS702 Verizon Business EMEA - Commercial IP service provider in Europe
4589	EASYNET Easynet Global Services
3257	TINET-BACKBONE Tinet SpA
3491	BTN-ASN - Beyond The Network America, Inc.
12956	TELEFONICA Telefonica Backbone Autonomous System
6762	SEABONE-NET Telecom Italia Sparkle
6939	HURRICANE - Hurricane Electric, Inc.
5511	OPENTRANSIT France Telecom - Orange - Worldwide IP Backbone
6453	GLOBEINTERNET TATA Communications *
286	KPN KPN Internet Backbone
6539	GT-BELL - Bell Canada
174	COGENT Cogent/PSI
2497	IIJ Internet Initiative Japan Inc.

Table 7.1: List of ASes selected frequently (>80% occurrence rate in landmark sets) by the landmark selection algorithm

AS number	AS Name or Description
7018	AT&T
3549	Global Crossing (GBLX)
3356	Level 3 Communications (L3)
1299	TeliaSonera International Carrier
2914	NTT Communications (Verio)
209	Qwest
1239	Sprint
6453	Tata Communications (formerly Teleglobe)
701 (702, 703)	Verizon Business (formerly UUNET) (also owned by Verizon)
3561	Savvis

Table 7.2: List of Tier 1 AS

March 1998

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	702	20.42%	Excessive	194
4	2214	64.41%	856	20
6	3220	93.68%	190	0
8	3415	99.35%	21	0
10	3436	99.97%	1	0
12	3437	100.0%	0	0

March 1999

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	1067	22.52%	Not yet started	-
4	3310	69.86%	1092	13
6	4528	95.56%	189	0
8	4721	99.64%	16	0
10	4737	99.97%	1	0
12	4738	100.0%	0	0

March 2000

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	1518	21.85%	Not yet started	-
4	4951	71.26%	Not yet started	-
6	6651	95.73%	275	0
8	6928	99.72%	19	0
10	6947	100.0%	0	0

March 2001

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2258	21.79%	Not yet started	-
4	7349	70.92%	Not yet started	-
6	9937	95.89%	386	1
8	10335	99.73%	26	0
10	10361	99.99%	1	0
12	10362	100.0%	0	0

March 2002

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2583	19.97%	Not yet started	-
4	9290	71.83%	Not yet started	-
6	12449	96.25%	397	7
8	12889	99.65%	39	0
10	12929	99.96%	4	0
12	12933	100.0%	0	0

March 2003

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2514	16.76%	Not yet started	-
4	9791	65.29%	Not yet started	-
6	14124	94.18%	813	2
8	14963	99.77%	31	0
10	14994	99.98%	2	0
12	14996	100.0%	0	0

March 2004

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2357	14.15%	Not yet started	-
4	10295	61.81%	very, very large	120
6	15715	94.35%	876	1
8	16614	99.75%	40	0
10	16654	99.99%	1	0
12	16655	100.0%	0	0

March 2005

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2333	12.22%	Not yet started	-
4	11496	60.22%	Still running (> 2 weeks)	-
6	17847	93.48%	1102	2
8	18995	99.50%	92	0
10	19088	99.98%	2	0
12	19090	100.0%	0	0

March 2006

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2371	10.97%	Not yet started	-
4	12369	57.26%	Still running (> 2 weeks)	-
6	20022	92.70%	large amounts	3
8	21462	99.37%	131	0
10	21593	99.97%	5	0
12	21598	100.0%	0	0

March 2007

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2348	9.60%	Not yet started	-
4	12925	52.85%	Still running (> 2 weeks)	-
6	22529	92.12%	many	5
8	24275	99.26%	165	0
10	24442	99.95%	10	0
12	24452	99.99%	2	0
14	24454	100.0%	0	0

March 2008

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2481	9.06%	Not yet started	-
4	14231	52.01%	Still running (> 2 weeks)	-
6	25534	93.33%	1678	2
8	27246	99.59%	110	0
10	27356	99.99%	1	0
12	27357	100.0%	0	0

March 2009

d	*d*-Core size	% of nodes in core	Num of connected components	Num of extra edges
2	2304	7.36%	Not yet started	-
4	21153	67.63%	Still running (> 2 weeks)	-
6	30007	95.93%	1171	0
8	31201	99.75%	68	0
10	31269	99.97%	8	0
12	31277	100.0%	0	0

From the tabular results above, as well as the graphical representation in Figure 7.2, it can be seen that all AS-level snapshots exceed ten extra edges (as defined by the Brady-Cowen scheme, chapter 6) when the number of nodes within the core is less than approximately 70% of the graph's size. Each graph snapshot falls below the 70% threshold at d=4. To fulfil the requirement that the extra edge count of the fringe should be less than 10, d must be greater than 4.

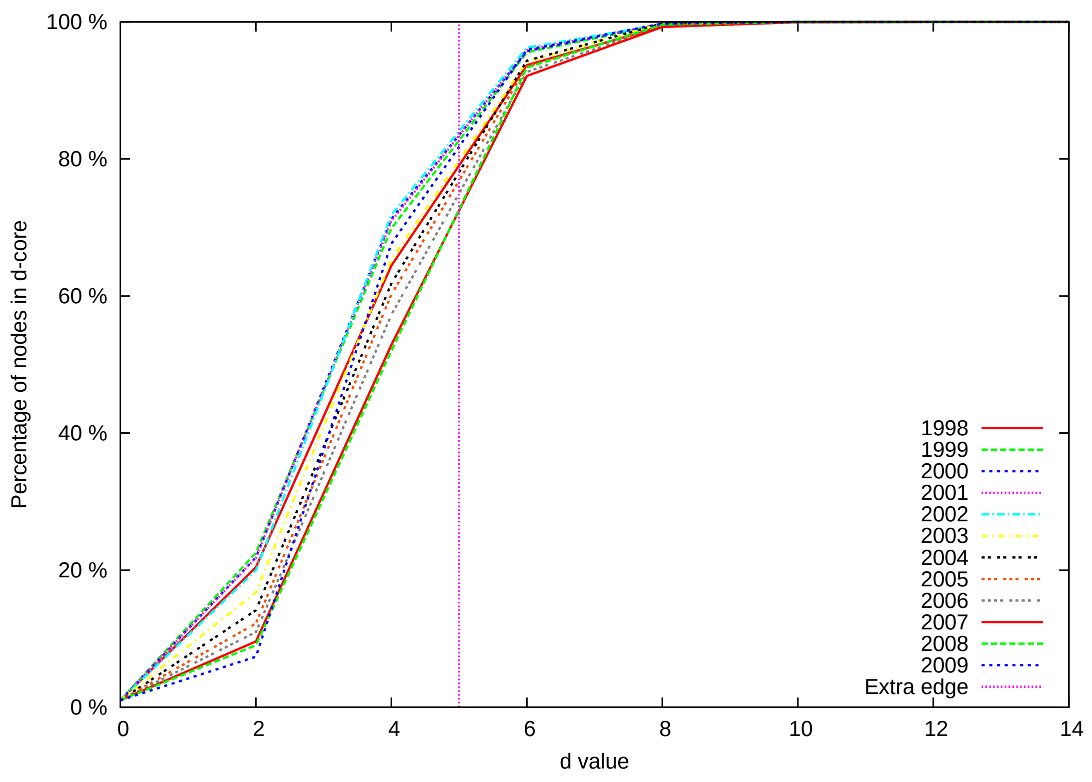


Figure 7.2: Core size with relation to extra edge count

Many of the snapshots for the value d=6 have zero extra edges, this is disadvantageous in terms of the stretch exhibited by packets in the system. With no extra edges, only the trees within connected components and the single spanning tree encompassing the entire graph are used for routing. This has the effect of turning the Internet into a tree, causing any source-destination pair with the source and destination in different connected components having to traverse the core – incurring a path increase of d hops (the size of the core). This situation is the reason for the existence of the Brady-Cowen heuristic (section 6.1). This heuristic adds extra spanning trees originating from within the fringe, allowing for alternative routes between connected components, some that may by-pass the core completely.

The value of d=6 with the use of the Brady-Cowen heuristic, will provide the test bed for our experiments. This will guarantee at least five extra edges, but no more than ten, the range Brady and Cowen determine this scheme incurs small stretch while maintaining small routing tables.

7.2 Summary

Our experiment will run several simulations of both the Thorup-Zwick and Brady-Cowen compact routing schemes on snapshots of the Internet AS-level graph. These Internet graphs have been gathered from CAIDA [1] such that they span twelve years (1998 to 2009). Analysis of the parameters of the compact routing algorithms revealed a pattern within the Thorup-Zwick landmark selection algorithm, and also determined that a value of d=6 is ideal for the Brady-Cowen simulator for all snapshots.

Chapter 8

Thorup-Zwick landmark selection

This chapter presents further analysis of the landmark selection algorithm utilised by the Thorup-Zwick routing scheme. This was done by running the landmark selection algorithm on the March 2004 snapshot fifteen times, these landmark sets were then simulated ensuring the same traffic was simulated between simulations. This allows for the determination of how the landmark selection algorithm affects the observed path stretch and routing table sizes. The March 2004 snapshot was chosen as it was the smallest snapshot available at that time, while still providing sufficient quantity of nodes to provide room for the landmark selection algorithm to made different decisions. Fifteen instances of the landmark selection were run, this decision is a compromise between time constraints and the statistical power of the data.

The results of each snapshot is presented as six graphs representing: (a) the path stretch incurred by all packets; (b) and (c) the subset of paths that exhibit stretch-3 and stretch-2; (d) the longest path witnessed in the simulator; (e) a comparison of the stretch incurred on the forward packet versus the reverse packet; and finally, (f) the routing table sizes of all nodes within the snapshot.

8.1 Stretch

The effect of path stretch is determined utilising graphs (a) through (e) on the various snapshots. These will now be discussed.

The inclusion of the path stretch incurred by all packets us shown in subfigure (a) of figures 8.2 through 8.2. These are plotted on a log scale to allow all possible stretch values to be witnessed on the graph. This information allows for the calculation of mean path stretch within the snapshot, a primary research goal. It allows for clear illustration of how that mean path stretch was derived, as well as allowing for the percentage/number of packets exhibiting a specific stretch value to be determined. The favourable outcome for a graph of this type is many paths exhibiting stretch-1 while very few exhibit stretch-3.

As can be observed from the graphs, there is a significant skew in the graph towards low stretch values. Stretch-1 is the most prevalent value within every graph, with Stretch-1.x values being the next popular stretch values. Very few packets experience a path stretch of 3, in fact, it is the stretch value with the least amount of packets (ignoring stretch values not observed at all). This pattern is good, this shows a distinct skewing of stretch values towards stretch-1. This results in many packets exhibiting shortest path or near shortest path routing in the compact routing scheme. Further analysis of the stretch-3 paths follows.

Subfigure (b) in figures 8.2 through 8.2 represents the stretch-3 paths observed within the simulator, providing information on their shortest path length. This is an analysis of the worst-case performance of the Thorup-Zwick scheme, showing how often it occurs and what path lengths have been affected by a three times multiplicative increase. The ideal for this graph would present very few paths exhibiting this stretch, and if included, a small shortest path length to minimise the effect of a three times larger path.

The results show that the shortest path length for the packets exhibiting stretch-3 are only 1 or 2 hops long. This stretch increase causes them to exhibit path length of 3 or 6. This is very good, as if a larger original path was subjected to stretch-3 the distance traversed through the network would be undesirable, however, path lengths of 3 or 6 hops are acceptable. The largest number of packets exhibiting this stretch factor is visible in figure 8.4(b), approximately 180 packets are stretch-3; in comparison to the 5.5 million packets simulated this value is minuscule. Overall, the 15 landmark sets perform similarly producing similar quantities of the various stretch values. The similarity can be further depicted in figure 8.1 which shows the mean path stretch of each landmark set side by side.

The paths exhibiting stretch-3 with original shortest path 1 can be adapted to produce stretch-1 paths by utilising an addendum to the Thorup-Zwick scheme produced by Krioukov et al. [26]. This modification states that a node will contain routing information for all directly connected neighbours. This would ensure that all pairs of nodes with a shortest path length of 1 utilise this link, providing stretch-1 paths. There are two possible stretch values affected by this addendum, stretch-3 paths (previously covered) and stretch-2 paths.

To fully witness the effect of the Krioukov et al. addition to the TZ scheme, paths exhibiting stretch-2 have been plotted alongside their original shortest paths in subfigure (c) of figures 8.2 to 8.2. Any stretch-2 paths with original path length of 1, can be removed by utilising the modification.

The stretch-2 profiles of the various snapshots show very little variation in general. As can be shown from the graphs it is possible to remove approximately 250 stretch-2 paths from this graph if the Krioukov et al. modification is employed. This totals approximately 350 paths could be removing in total (when combined with the stretch-3 path information) allowing the mean path length to be lowered slightly.

Another potential worst-case aspect of the compact routing schemes is visible by viewing the largest path length encountered within the simulator subfigure (d) of figures 8.2 to 8.2. The longest paths graph represents all the packets that experienced this largest path and shows the shortest possible path for those packets. Ideal behaviour of the compact routing schemes would result in only paths with large shortest paths appearing in this graph. This would show that larger shortest paths do not exhibit stretch-3, larger shortest paths exhibiting large multiplicative stretch increase would be a very negative aspect of these schemes.

The longest paths observed within the simulations are either 10 to 11 hops long, these are comprised of original shortest path lengths of 7, 8, or 9. This is stretch-1.5 in the worst case (11/7). This is result is good as there are small shortest path lengths contained within the data set which would represent a path being significantly stretched (ie stretch of 2 or 3). Figure 8.9(d) shows that landmark set 9 has a substantially more paths exhibiting the longest observed path, this is clearly affects the mean stretch value of the data set as can be shown in figure 8.1. However, the effect of this is minimal when considering the difference. Landmark set 9 has a mean stretch of 1.11, while most other landmark sets produce a mean stretch of approximately 1.08. The larges paths within the 2004 graph is a 10 hop path, the result of applying either mean stretch value is a increased path length of 11. This shows that this difference is not significant in practical terms.

Due to its design paths within the Thorup-Zwick scheme are not symmetric, the path from node a to node b is not necessarily the same as the path between node b and node a. As a result, the path stretch incurred by traffic be be larger in the forward direction than in the reverse, or vice versa. It could also be possible for a packet to exhibit stretch-1 paths both ways, or even stretch-3 in both directions. The stretch comparison graphs, subfigure (e) in figures 8.2 to 8.2 are a visual representation of the path stretch incurred by a packet. Ideally, the stretch experienced by a packet will be small in both directions, or at least show a trend toward the majority of traffic exhibiting this trend. Large amounts of paths exhibiting stretch-3 both directions would be a problem for the compact routing schemes.

The comparison of forward to reverse stretch shows that the majority of packets exhibit small stretch values in both directions, a small proportion of pairs exhibit a larger path in one direction, and approximately 30 paths are unfortunate to experience stretch-3 in both the forward and reverse directions. This clustering effect toward bottom left corner (i.e., low stretch both directions) shows that the stretch experienced by the majority of the network is low.

8.2 Routing tables

The routing table sizes of the landmark sets are presented in subfigure (f) of figures 8.2 to 8.2. This graph depicts the routing tables within the snapshot, sorted by largest routing table to smallest routing table. Smaller routing table sizes are the core reason for the existence of compact routing. This graph will demonstrate the smaller sizes of the routing tables, and allow for a distribution to be seen visually. The nodes with the smallest routing table size are the landmarks selected by the landmark selection algorithm, every node higher than the points on the far right of the graph are non-landmark nodes with a cluster. The larger the cluster the further to the left of the graph the node will be. An ideal for this graph is low routing table sizes in general, as well as many nodes with near

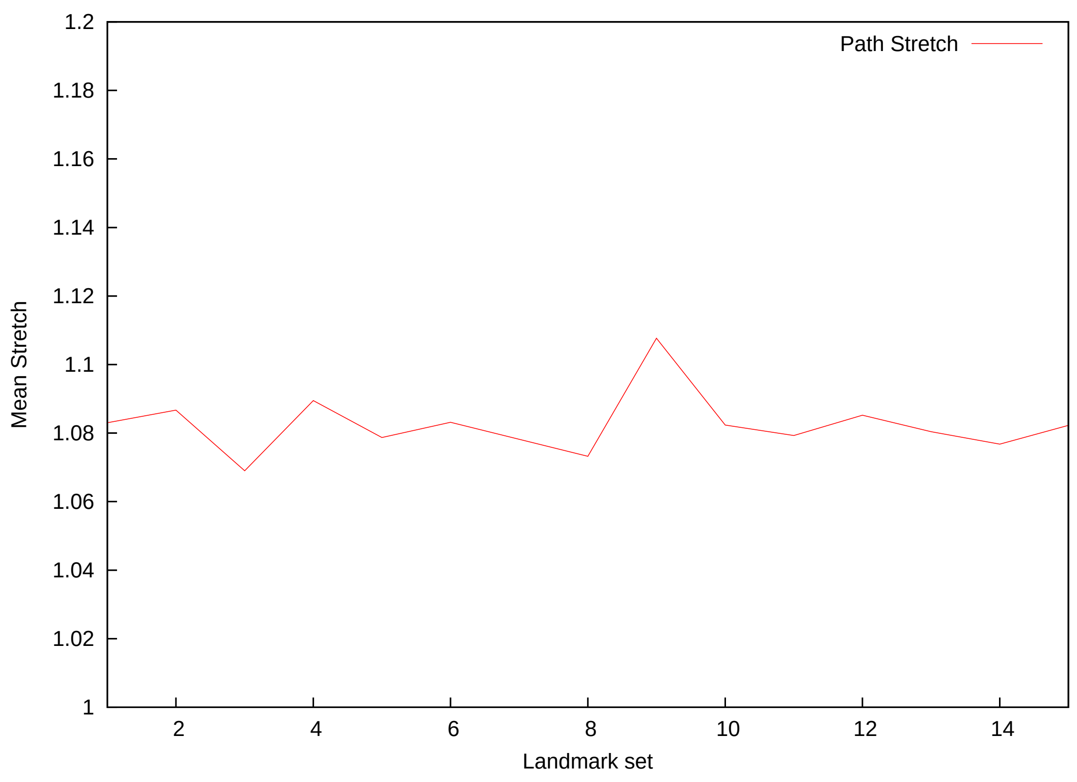


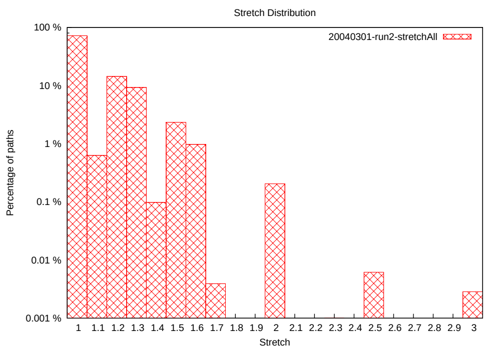
Figure 8.1: Mean path stretch for each landmark set

minimum routing table sizes. This graph when observed with the path stretch graph will allow for clear analysis of the benefits (or lack of) of compact routing.

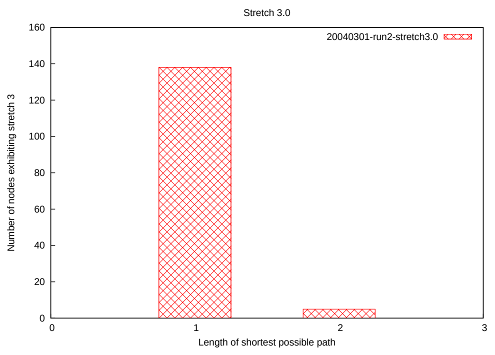
The routing table graphs presented show a large amount of nodes with small routing tables containing approximately 100 entries. This is very small, the entire graph contains 16,655 nodes, which if represented by a distance vector routing algorithm would require 16,655 entries in every node. Across the landmark sets, there are only a hand full of nodes with larger than average routing table sizes. The average of the largest routing table size is approximately 360, with the largest routing table being witnessed in landmark set 9 (Figure 8.9(f)) at 649 entries. The larger routing table in landmark set 9 is off set by the fact that this landmark set has fewer landmarks, resulting in rest of the nodes having fewer routing table entries - it is a counterbalancing effect. Even largest routing tables, which only affect a few nodes, are substantially smaller than that required by the distance vector approach. The variation in routing table sizes across the landmark selection algorithm instances is very small if the entire graph is considered as a whole.

8.3 Summary

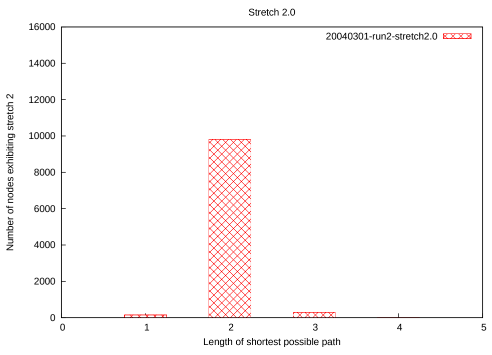
It can be seen that there is a minor variation in the stretch values and the routing table sizes between the different landmark sets calculated. Each sample instance demonstrates a dominance of stretch-1 paths, with many pairs exhibiting symmetrical path stretch. There is also very few stretch-3 paths. The largest paths observed consist of pairs of nodes with shortest paths of considerable length being subjected to low stretch values. The variation between landmark sets is not substantial, and it will be possible to select one landmark set per snapshot to evaluate the performance of the Thorup-Zwick compact routing scheme without unfairly biasing the results.



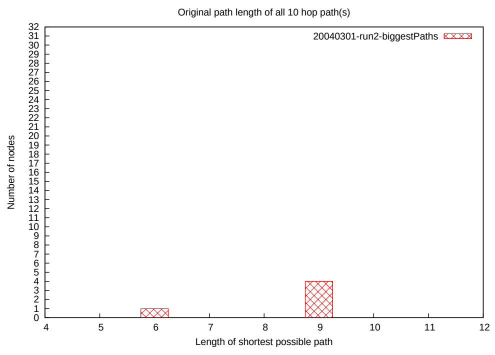
(a) Path stretch (log scale)



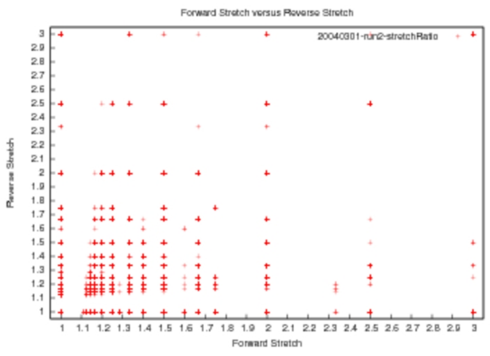
(b) Stretch 3 original path comparison



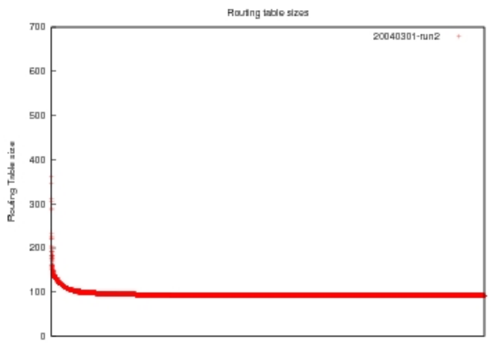
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

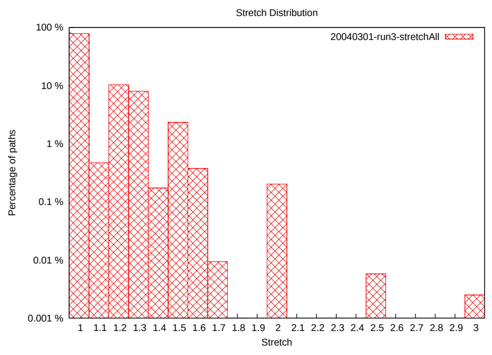


(e) Stretch comparison forward path versus reverse path

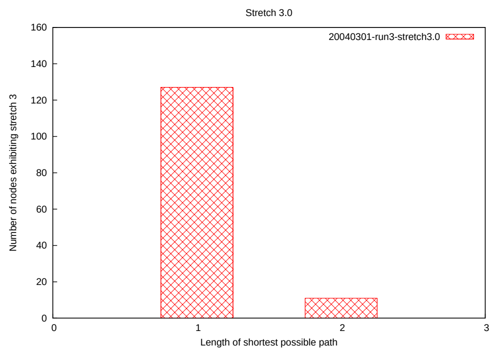


(f) Routing table sizes

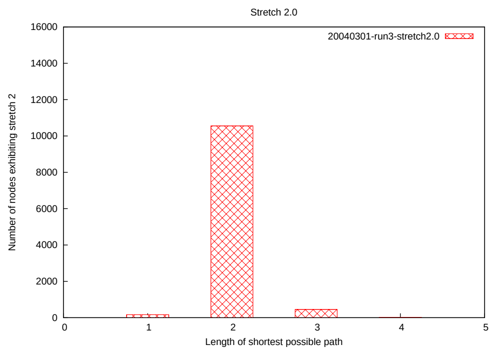
Figure 8.2: TZ results on the March 2004 snapshot using landmark set 2



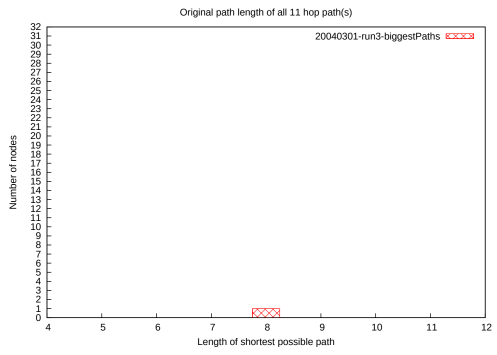
(a) Path stretch (log scale)



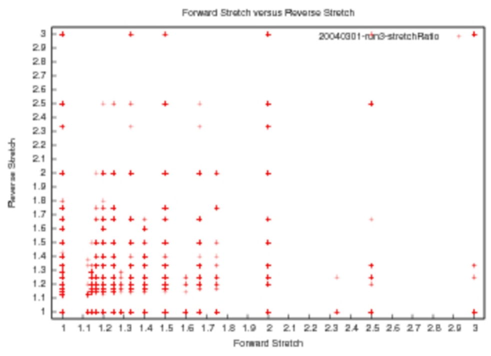
(b) Stretch 3 original path comparison



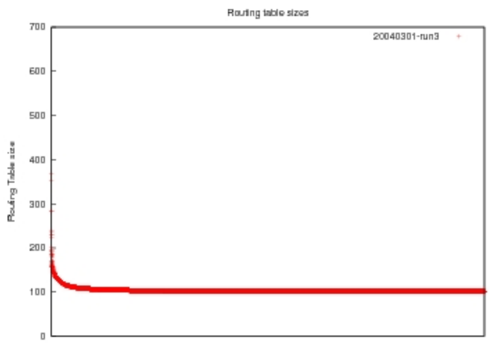
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

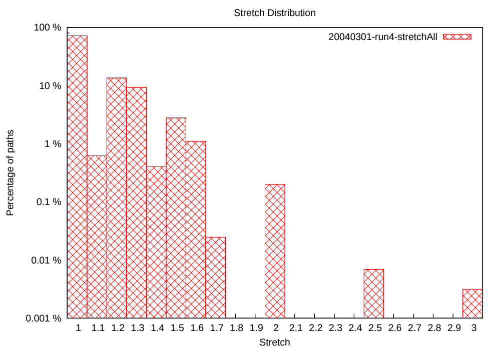


(e) Stretch comparison forward path versus reverse path

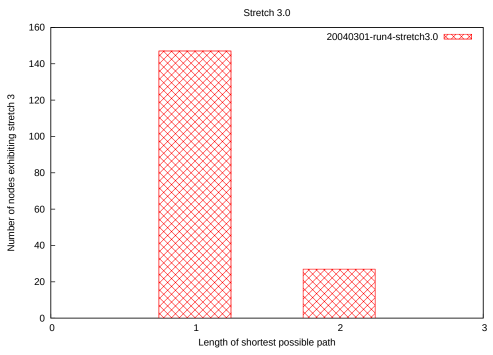


(f) Routing table sizes

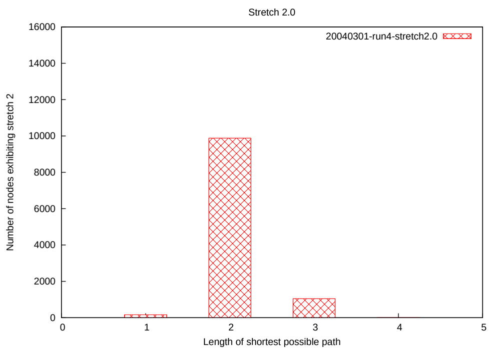
Figure 8.3: TZ results on the March 2004 snapshot using landmark set 3



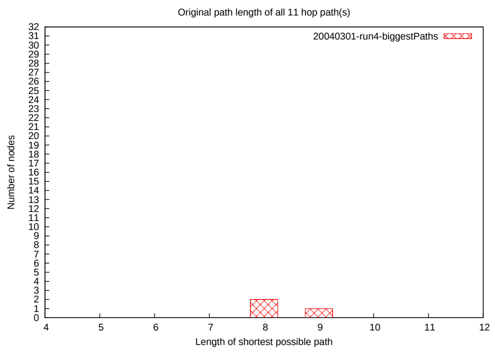
(a) Path stretch (log scale)



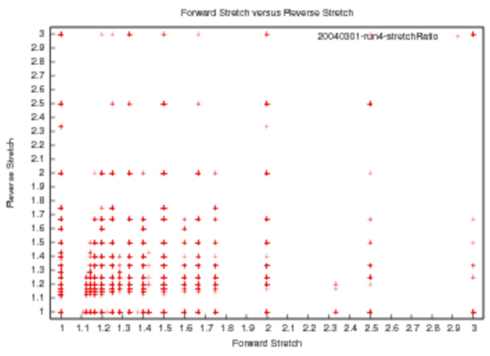
(b) Stretch 3 original path comparison



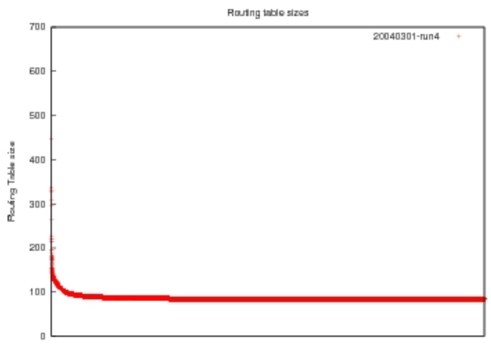
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

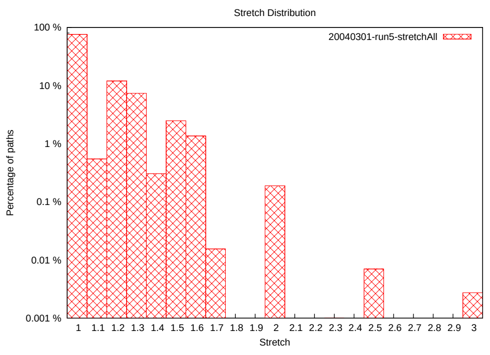


(e) Stretch comparison forward path versus reverse path

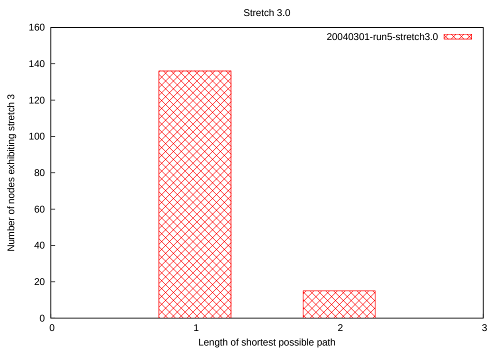


(f) Routing table sizes

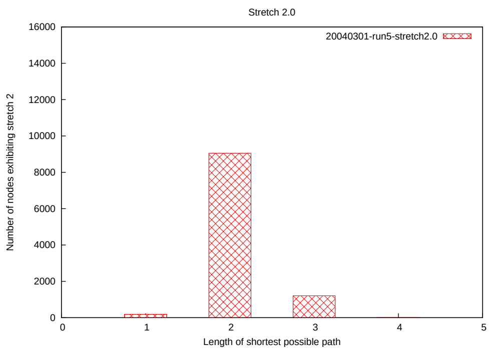
Figure 8.4: TZ results on the March 2004 snapshot using landmark set 4



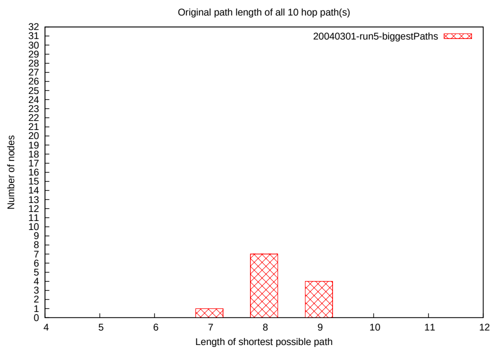
(a) Path stretch (log scale)



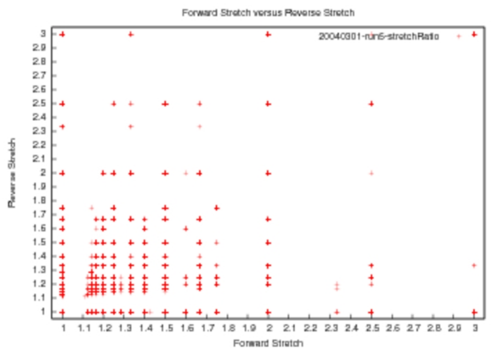
(b) Stretch 3 original path comparison



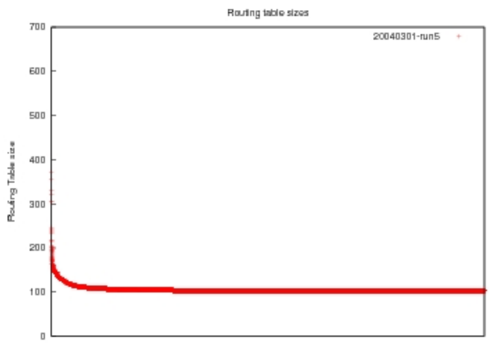
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

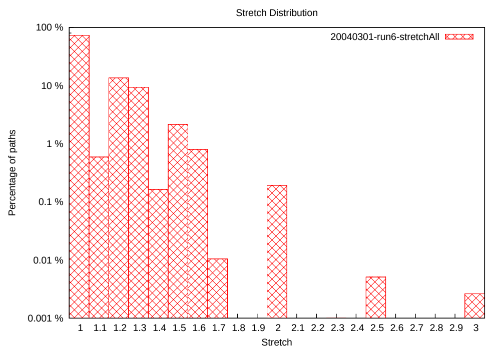


(e) Stretch comparison forward path versus reverse path

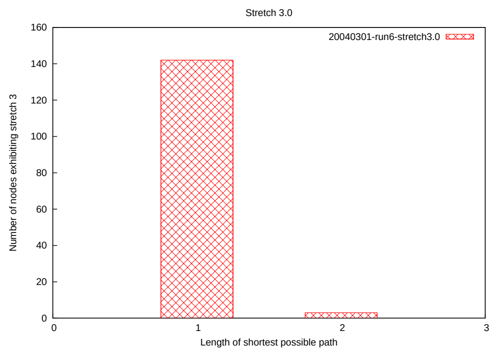


(f) Routing table sizes

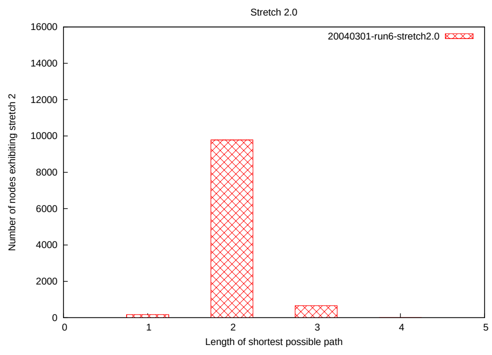
Figure 8.5: TZ results on the March 2004 snapshot using landmark set 5



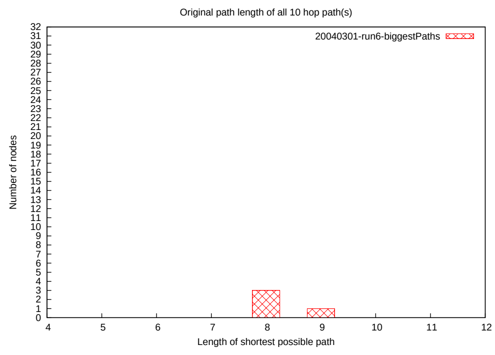
(a) Path stretch (log scale)



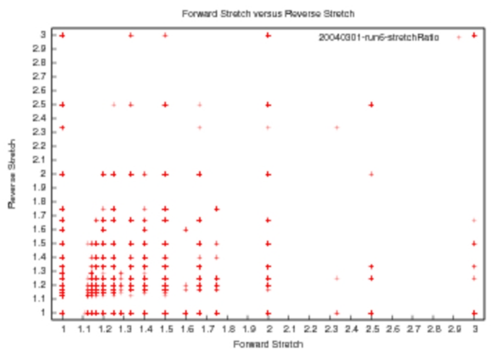
(b) Stretch 3 original path comparison



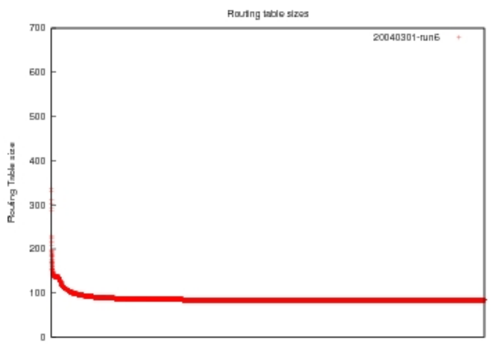
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

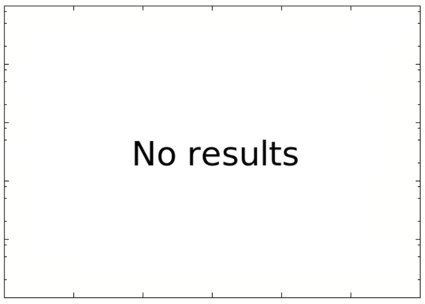


(e) Stretch comparison forward path versus reverse path

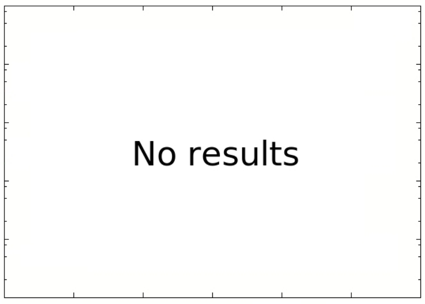


(f) Routing table sizes

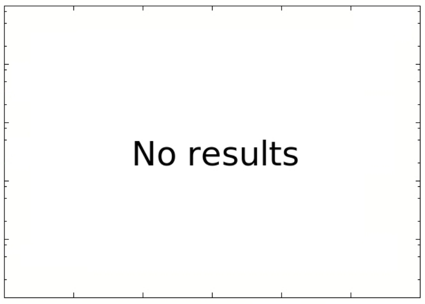
Figure 8.6: TZ results on the March 2004 snapshot using landmark set 6



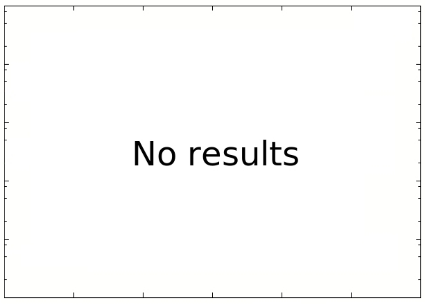
(a) Path stretch (log scale)



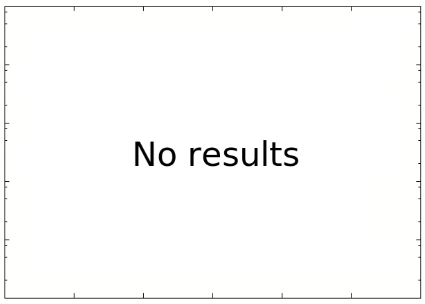
(b) Stretch 3 original path comparison



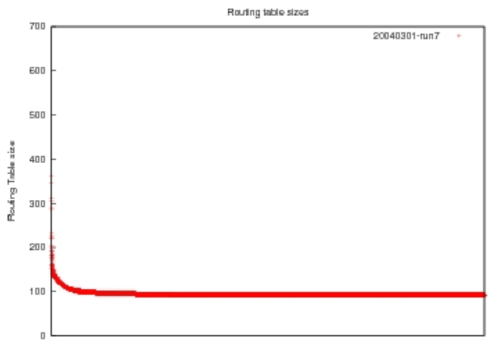
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

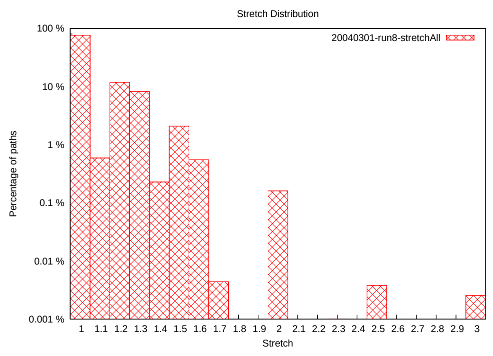


(e) Stretch comparison forward path versus reverse path

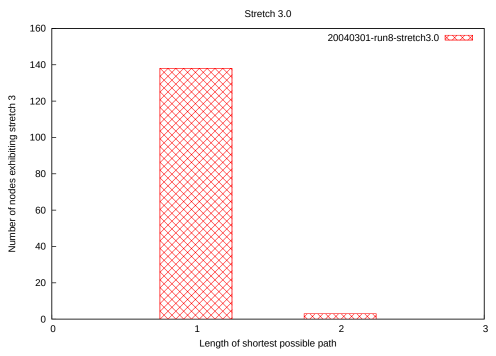


(f) Routing table sizes

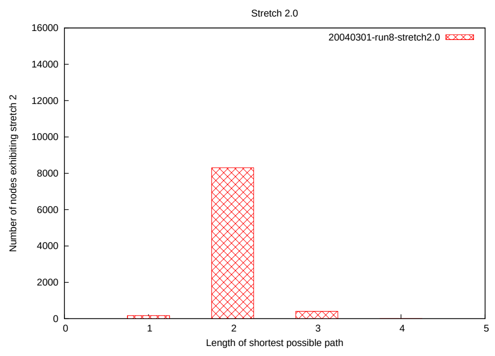
Figure 8.7: TZ results on the March 2004 snapshot using landmark set 7. Stretch data (a) through (e) corrupted while on storage media.



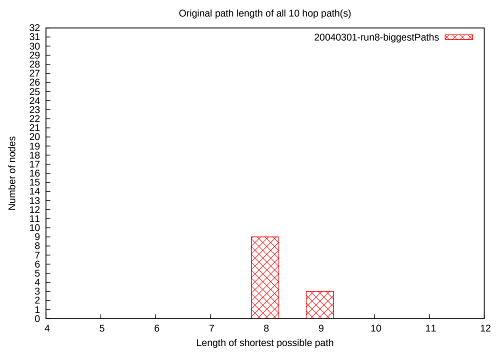
(a) Path stretch (log scale)



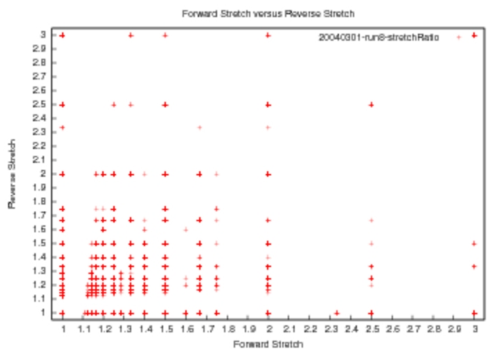
(b) Stretch 3 original path comparison



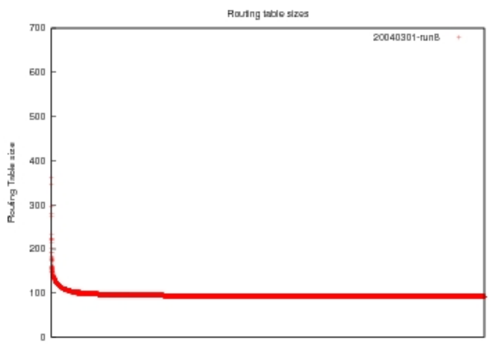
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

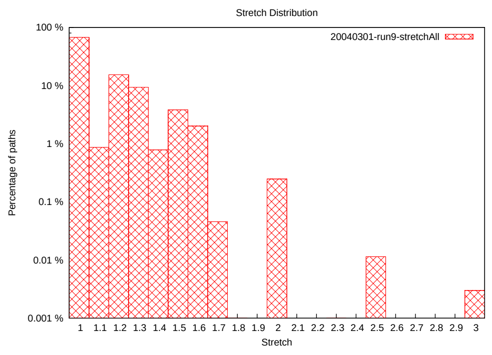


(e) Stretch comparison forward path versus reverse path

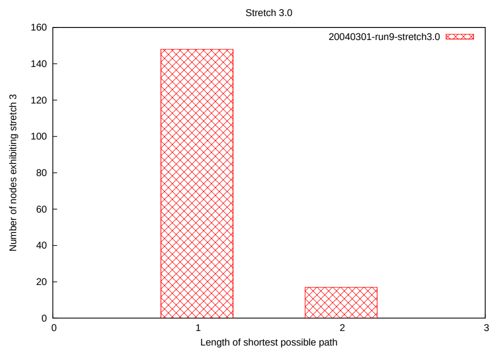


(f) Routing table sizes

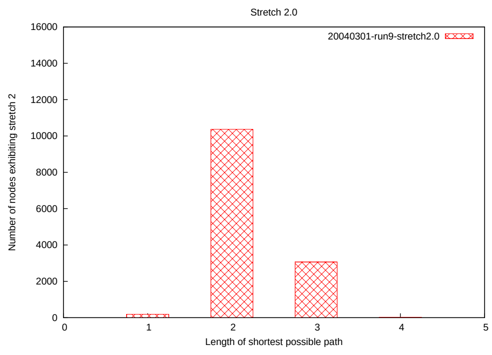
Figure 8.8: TZ results on the March 2004 snapshot using landmark set 8



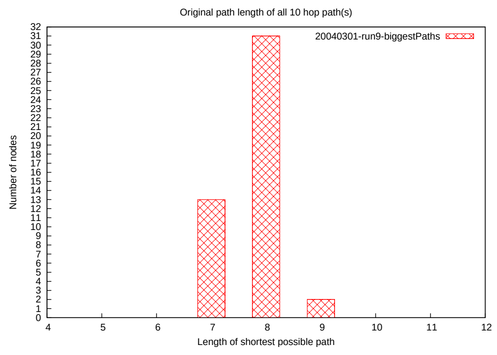
(a) Path stretch (log scale)



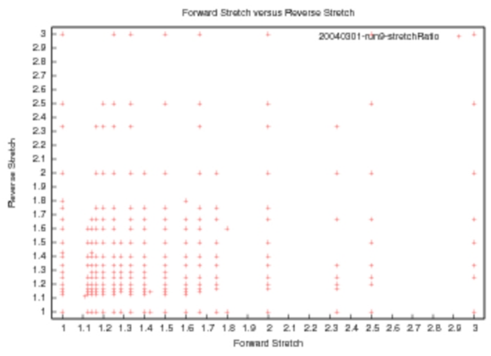
(b) Stretch 3 original path comparison



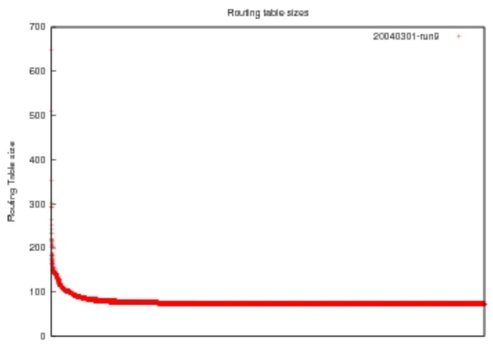
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

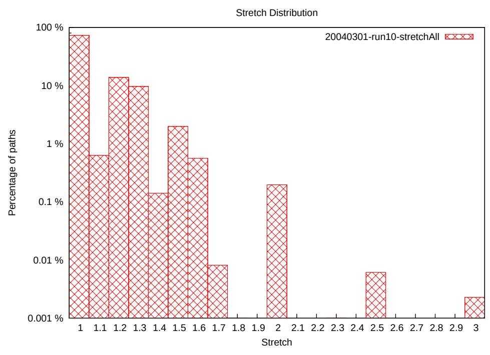


(e) Stretch comparison forward path versus reverse path

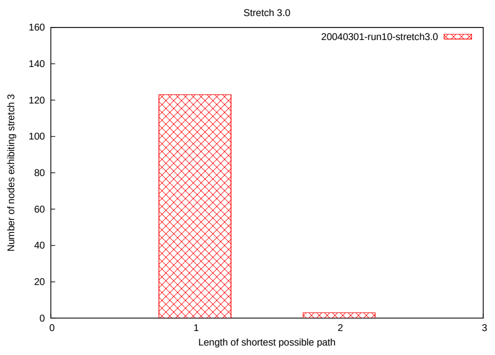


(f) Routing table sizes

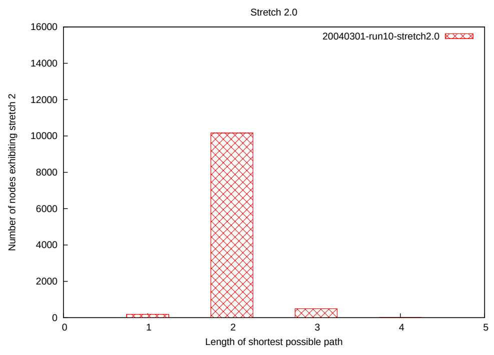
Figure 8.9: TZ results on the March 2004 snapshot using landmark set 9



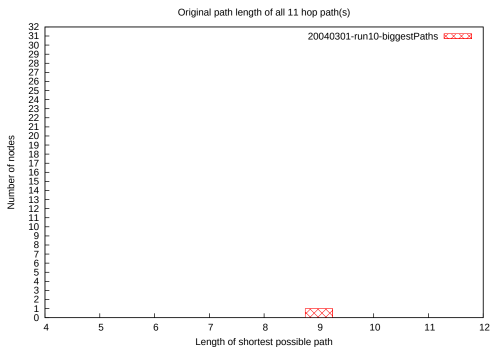
(a) Path stretch (log scale)



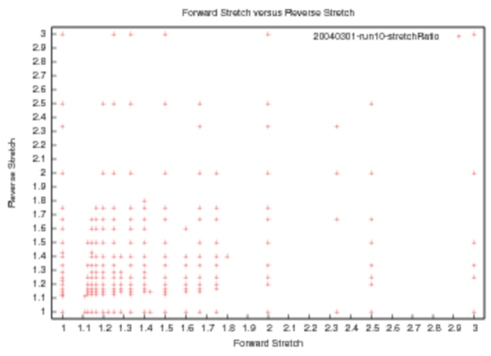
(b) Stretch 3 original path comparison



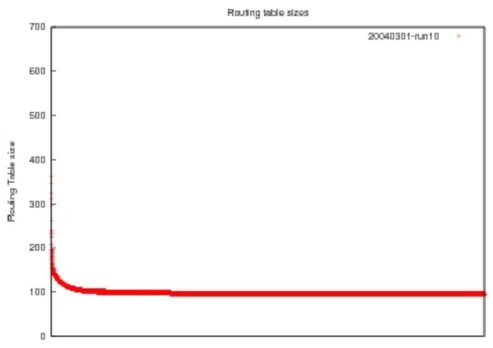
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

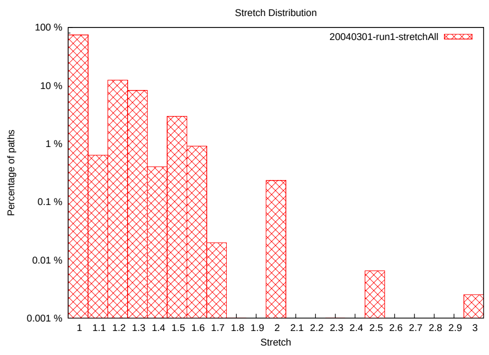


(e) Stretch comparison forward path versus reverse path

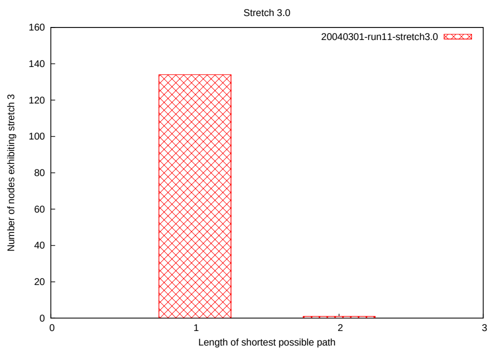


(f) Routing table sizes

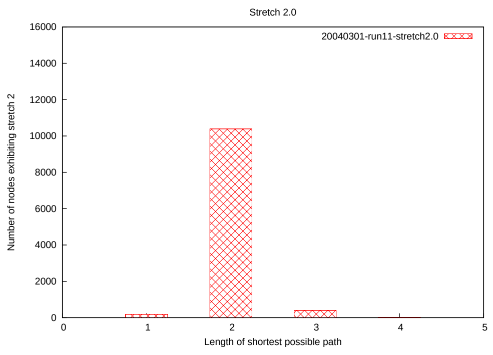
Figure 8.10: TZ results on the March 2004 snapshot using landmark set 10



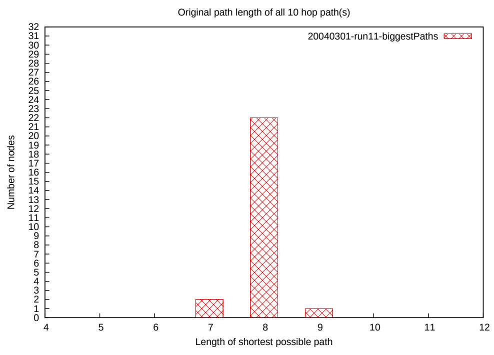
(a) Path stretch (log scale)



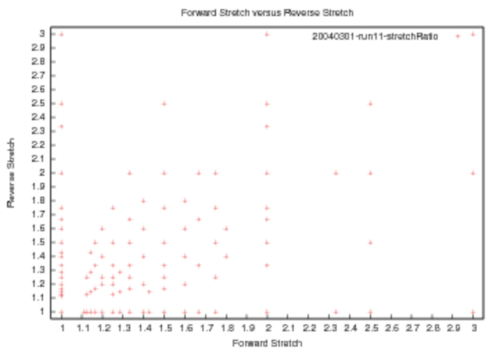
(b) Stretch 3 original path comparison



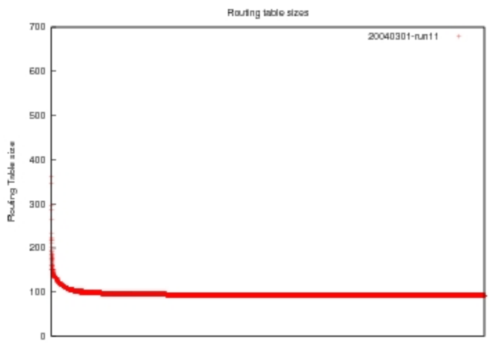
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

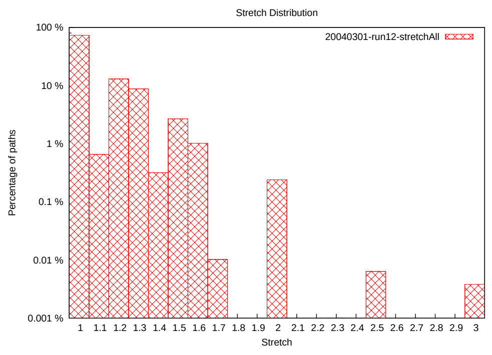


(e) Stretch comparison forward path versus reverse path

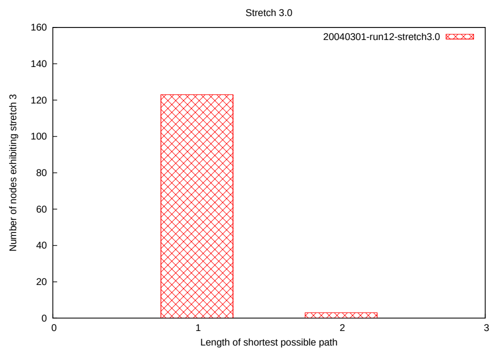


(f) Routing table sizes

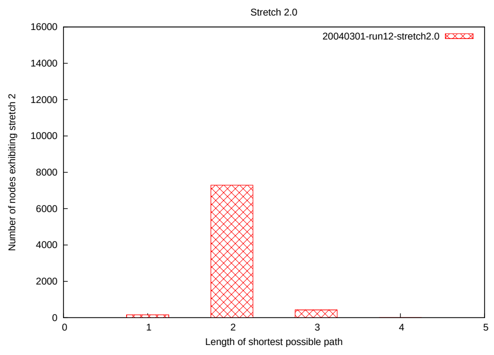
Figure 8.11: TZ results on the March 2004 snapshot using landmark set 11



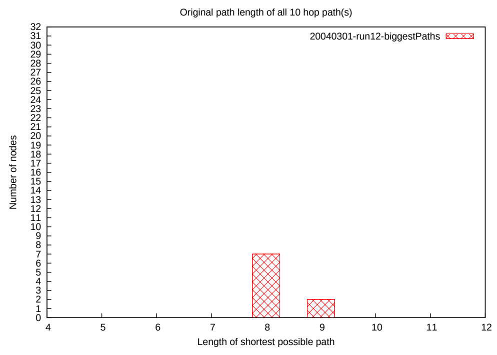
(a) Path stretch (log scale)



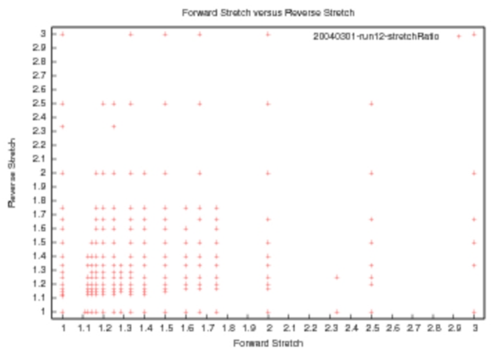
(b) Stretch 3 original path comparison



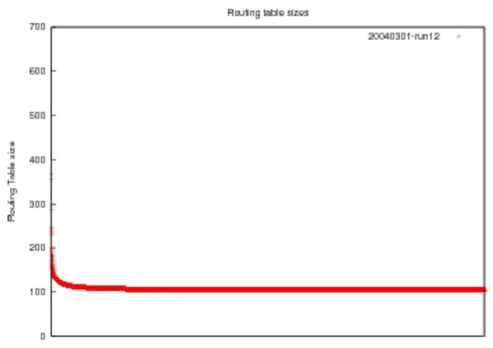
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

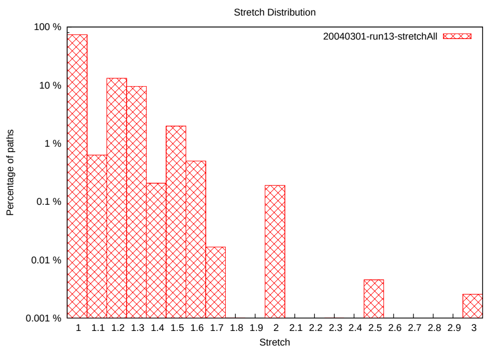


(e) Stretch comparison forward path versus reverse path

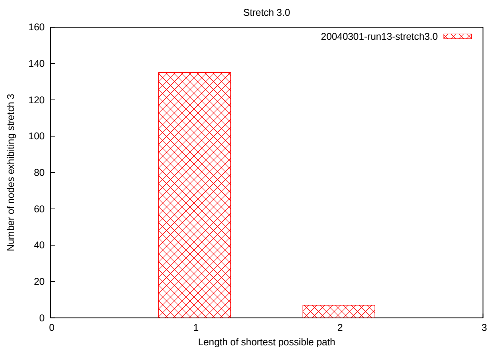


(f) Routing table sizes

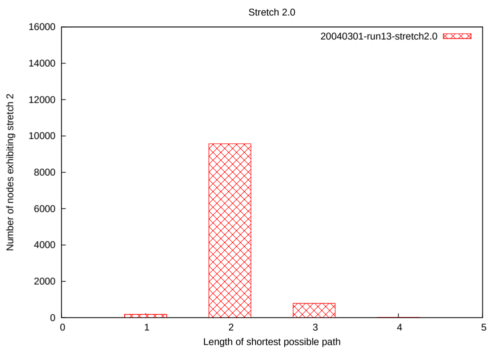
Figure 8.12: TZ results on the March 2004 snapshot using landmark set 12



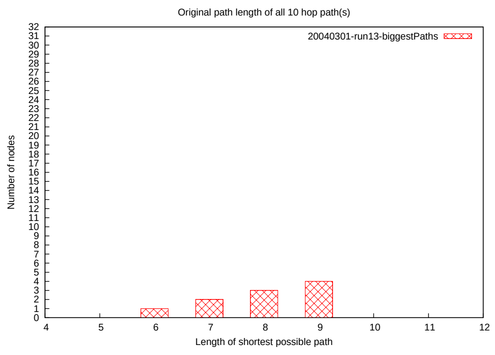
(a) Path stretch (log scale)



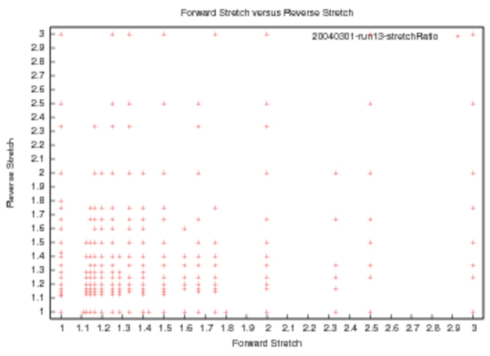
(b) Stretch 3 original path comparison



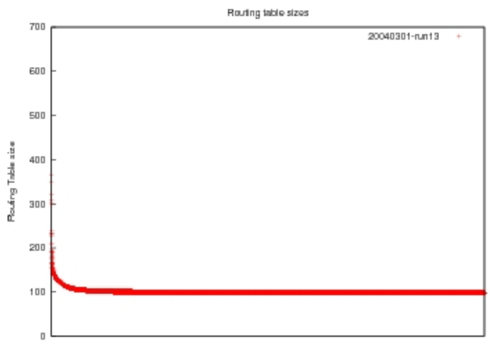
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

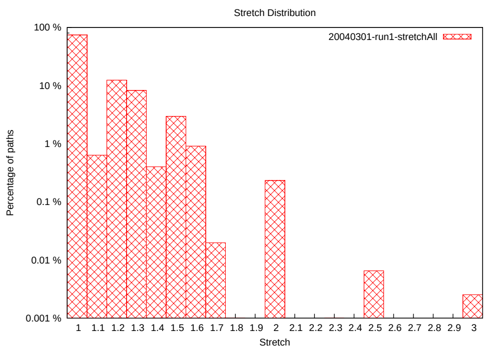


(e) Stretch comparison forward path versus reverse path

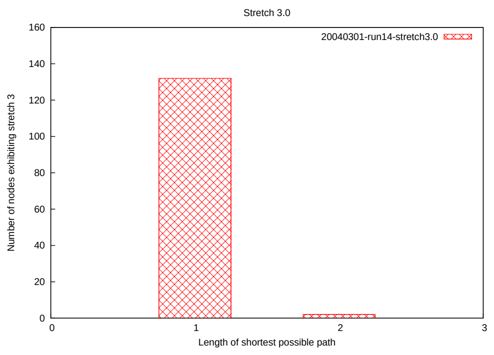


(f) Routing table sizes

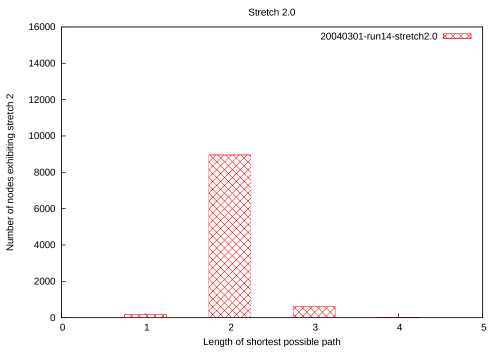
Figure 8.13: TZ results on the March 2004 snapshot using landmark set 13



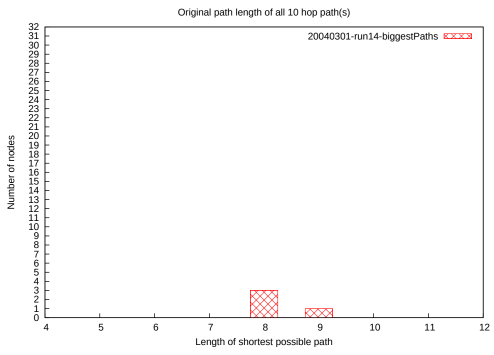
(a) Path stretch (log scale)



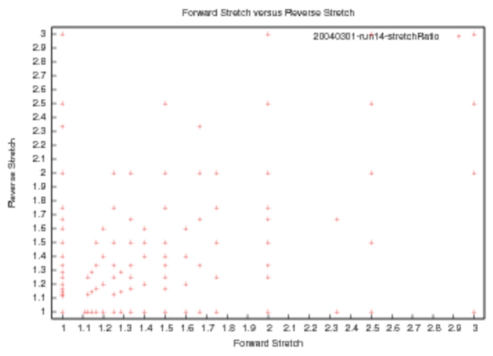
(b) Stretch 3 original path comparison



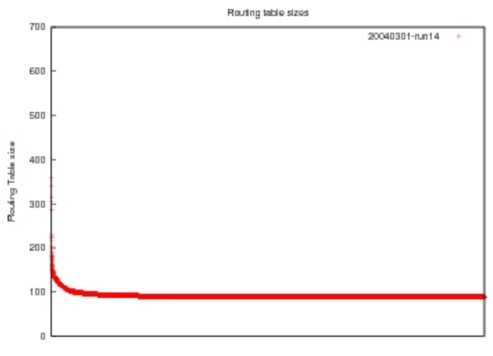
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

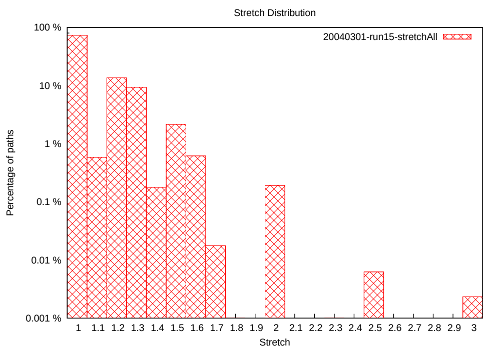


(e) Stretch comparison forward path versus reverse path

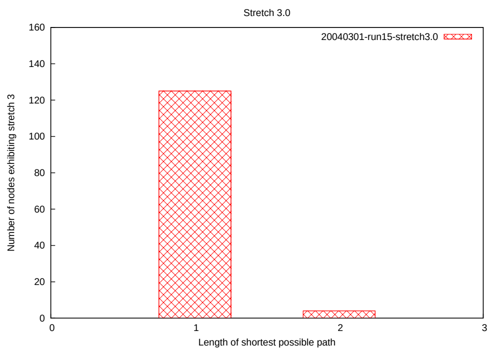


(f) Routing table sizes

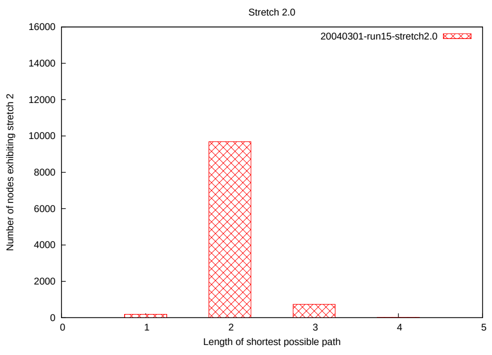
Figure 8.14: TZ results on the March 2004 snapshot using landmark set 14



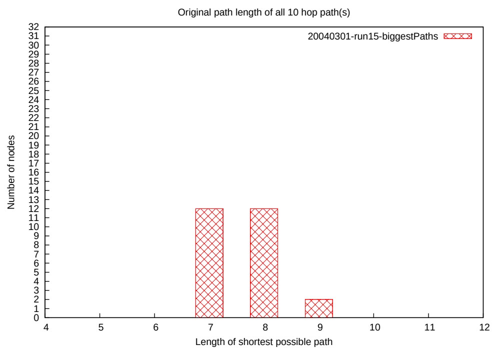
(a) Path stretch (log scale)



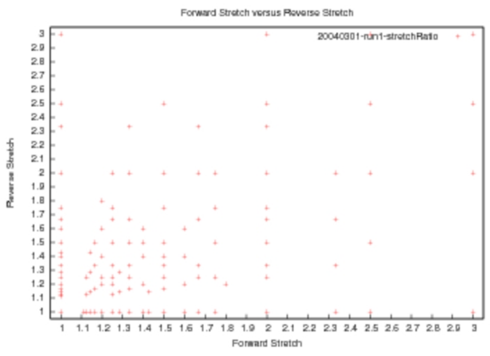
(b) Stretch 3 original path comparison



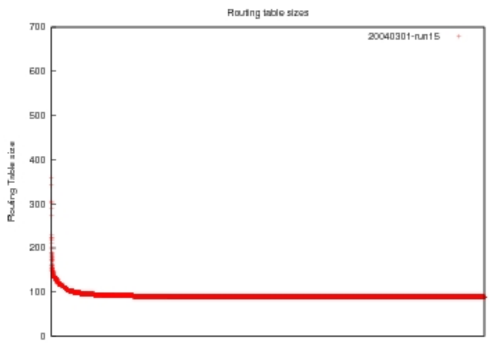
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length



(e) Stretch comparison forward path versus reverse path



(f) Routing table sizes

Figure 8.15: TZ results on the March 2004 snapshot using landmark set 15

Chapter 9

Compact routing over time

This chapter presents the results of simulating the Thorup-Zwick and Brady-Cowen compact routing schemes on real-world AS-level Internet snapshots provided by CAIDA [1]. The results with respect to Thorup-Zwick is presented in section 9.1, and Brady-Cowen will be discussed in section 9.2.

9.1 Thorup-Zwick

The evaluation of the Thorup-Zwick compact routing scheme on the yearly snapshots will be presented in a format resembling that of chapter 8.

9.1.1 Stretch

The effect of stretch will be determined using subfigures (a) through (e) on each of the figures, i.e., similar to that of section 8.1.

The stretch distribution graphs can be found in subfigure (a) in figures 9.1.4 through 9.1.4. These show the same trend witnessed in the landmark selection experiment. The most prevalent stretch value is stretch 1, by a significant margin. Stretch-1.x values are the next most frequently observed, with stretch-3 being very low ($< 0.01\%$ rate of occurrence on all graphs). The results are very good; resulting in a low average stretch of approximately 1.09. This can be witnessed more clearly in figure 9.1, where it is also possible to see the variation of compact routing as graph size increases.

Subfigure (b) in figures 9.1.4 through 9.1.4 depicts the number of stretch-3 paths within the snapshots, providing the original shortest path for comparison purposes. It can be seen that the graphs exhibit stretch-3 (the worst case) on paths of original shortest path length 1 or 2. This is good as it ensures there are not very large paths within the graph; path lengths of 3 and 6 are acceptable. A notable exception to this is in figure **??** where there are 15 instances of a path of length 12 due to a shortest path of length 4 being subjected to a stretch of 3. This is undesirable, due to the large path length involved, however, due to the corruption of the longest path information (Figure 9.10(d)) we cannot prove this.

Similarly as was observed in the landmark selection, the various snapshots over time produce some paths that exhibit stretch-2 that can be removed by use of the Krioukov et al. [26] addendum to the Thorup-Zwick scheme. This addendum continues to lower the mean path stretch of the network, however slight that change may be. This can be observed in subfigure (c) in figures 9.1.4 through 9.1.4.

The biggest paths observed on the snapshots showed a mixture of results, see subfigure (d) in figures 9.1.4 through 9.1.4. Most snapshots had biggest paths of approximately 11 hops. These consisted of shortest path lengths of 7, 8, or 9, producing a over all worst stretch of 1.5. This is similar to the results from the landmark analysis (chapter 8). However, various snapshots produced interesting results. The 1998 snapshot had a worst path length of 12 based on a stretch-3 increase on a path of length 4, similarly for the 2007 snapshot, a path was elongated to 15 hops from shortest path length of 5 hops. The 2003 graph observed its longest path as a stretch-2 increase on a a path of length 5. These stretch values would prove inconvenient for the source and destination in question, however each snapshot only had one such path of this type. To contrast this, the 2009 snapshot had its longest path consisting of a stretch-1 path of length 11, showing that there is no elongation to this already long path.

The comparison of forward versus reverse stretch is presented in subfigure (e) of figures 9.1.4 through 9.1.4. As was observed in the landmark selection experiment, the majority of paths observe small stretch in both directions, with the the most common being stretch-1 both ways. There are a few outliers once again, that exhibit either stretch-3 both ways, or a larger stretch in either the forward or reverse directions. The clustering effect of the points show that most points exhibit very low stretch.

9.1.2 Routing tables

The routing table sizes for each snapshot are presented in subfigure (f) in figures 9.1.4 through 9.1.4. As the snapshots progress from 1998 to 2009, the average routing table size increases very slowly. The 1998 snapshot has an average routing table size of 40 entries, while the 2009 graph has 140. However, in this time the graph size increased from 3437 nodes to 33103 nodes. The routing tables have increase in size by 3.5 times; while the graph has 9.5 times larger. This shows the sub-linearly scaling nature of the compact routing scheme. Small routing tables, and the sub-linear scaling, provide a very strong reason for utilisation of compact routing schemes on very large graphs.

The routing tables for the Internet snapshots over time follow the pattern outlined in Section 8.2. Many nodes have small routing table sizes, while a minority maintain larger routing table sizes. In comparison to the routing tables produced by a distance vector algorithm, these larger routing tables are actually tiny.

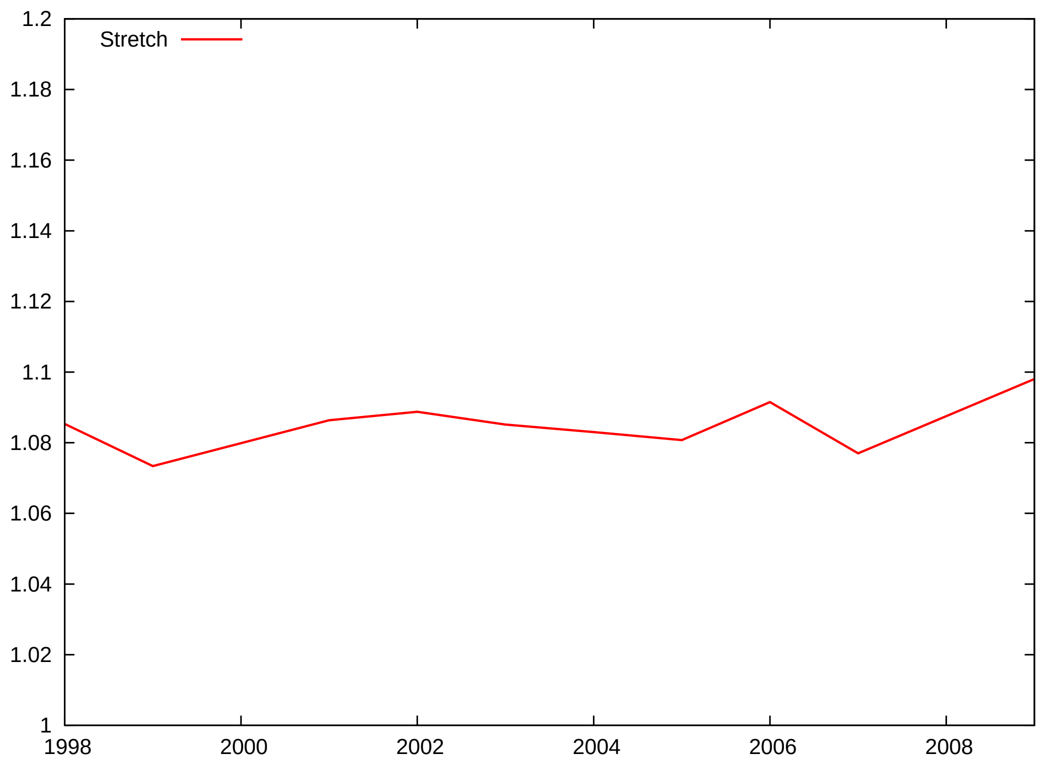


Figure 9.1: Average path stretch over time for the Thorup-Zwick compact routing scheme

9.1.3 Popular Traffic

This experiment was never performed. Unfortunately time constraints did not allow enough time for the popular traffic files to be processed by the simulator. This experiment is deferred for future work.

9.1.4 Thorup-Zwick over time

The utilisation of the snapshots between the years 1998 to 2009 allowed for us to observe how the Thorup-Zwick compact routing scheme reacts to larger graphs in terms of stretch and routing table size. The use of the snapshots over a period of twelve years also allows for us to see how compact routing may react in the future.

Figure 9.1 demonstrates the mean stretch values of the Thorup-Zwick compact routing scheme over time. It is clear to see that although the value fluctuates, the scheme provides stretch of approximately 1.09 consistently over the years. This is achieve while maintaining slowly growing routing tables, as previously mentioned the routing table over the twelve year period has increase by 100 entries. This shows that the Thorup-Zwick compact routing scheme has great scalability properties, while exhibiting very low stretch.

9.2 Brady-Cowen

The Brady-Cowen scheme implementation is mostly complete, but a bug in the pre-processing code led to incorrect results. The Peleg proximity-preserving labels [33] utilised by the simulator are not always correct. This invalidates the results of any runs of the Brady-Cowen scheme simulator.

This bug has been identified and can be easily resolved. The pre-processing program does not correctly generate the resultant sub-trees after choosing a separator, causing the separator to be contained within at least one sub-tree. Since this separator is not removed, and its connectivity data still exists, it is selected as the separator in the next iteration. However, on this iteration it is properly removed. This results in incorrect Peleg proximity-preserving labels due to the repeated separator being inserted to the nodes label and routing table.

The fix involves ensuring proper disconnection of the chosen separator from the current working representation of the graph. After fixing of the bug, it would be possible to re-process the Internet snapshots to produce correct routing table and node label information. This process is time consuming (O(days)), and so this analysis is left as future work.

9.2.1 Stretch

The results for this section were affected by pre-processing problems, and could not be run on the simulator. This experiment is deferred for future work.

9.2.2 Routing tables

When processed by the Brady-Cowen scheme every snapshot, except for one (March 2002), was given five additional spanning trees, either due to extra edges or use of the heuristic. The March 2002 snapshot is the exception, it contained seven extra edges. When combined with the original spanning tree, as well as the trees from connected components, this resulted in the snapshots having routing tables of size 6 or 7 (8 or 9 in the 2002 snapshot). Nodes located within the core of the scheme will have a routing table of the lower bound (6 or 8), while nodes located within a connected component will contain a higher number of entries (7 or 9). Table 9.1 presents the percentage of nodes in each snapshot with respect to each routing table size.

Due to an implementation decision, the early version of the program that supplied the data for Table 9.1 considered nodes in the fringe with connections only to the core, to be a connected component of size one. This approach does not encompass the real definition of a connected component, and as such the values presented are lower and upper bounds for the minimum and maximum table sizes, respectively.

The routing table size of the Brady-Cowen scheme is unchanging over the various snapshots, consistently precessing worst-case routing table sizes of 7 entries. This provides this scheme with unprecedented scalability properties as more nodes are

Snapshot	Nodes with routing table size 6 (or 8*) (minimum)	Nodes with routing table size 7 (or 9*) (maximum)
1998	93.68%	6.32%
1999	95.56%	4.44%
2000	95.73%	4.27%
2001	95.89%	4.11%
2002	96.25%*	3.75%*
2003	94.18%	5.82%
2004	94.35%	5.65%
2005	93.48%	6.52%
2006	92.70%	7.30%
2007	92.12%	7.88%
2008	93.33%	6.77%
2009	95.93%	4.07%

Table 9.1: Routing table sizes for the Brady-Cowen scheme

added to the graph.

9.2.3 Popular Traffic

This experiment was never performed. Unfortunately time constraints did not allow enough time for the popular traffic files to be processed by the simulator. This experiment is deferred for future work.

9.2.4 Brady-Cowen over time

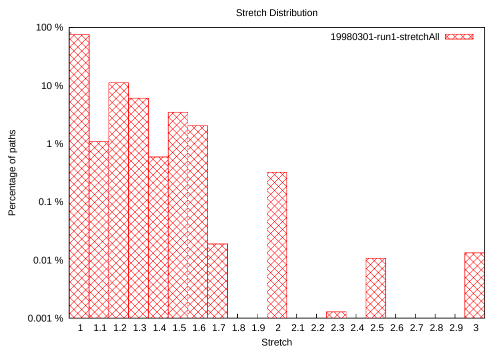
In terms of routing table sizes the Brady-Cowen is clearly unaffected by the increasing size of the Internet AS-level graph. This scheme consistently presents routing table sizes of six or seven entries regardless of snapshot for the value d=6, with the sole exception of the 2002 snapshot which requires up to nine routing table entries. This trait of the Brady-Cowen scheme suggests it has very strong scalability properties.

Unfortunately due to the failure of the pre-processing stage of the Brady-Cowen scheme there are no results for the stretch over time for this scheme. This is work that is heavily suggested as useful future work.

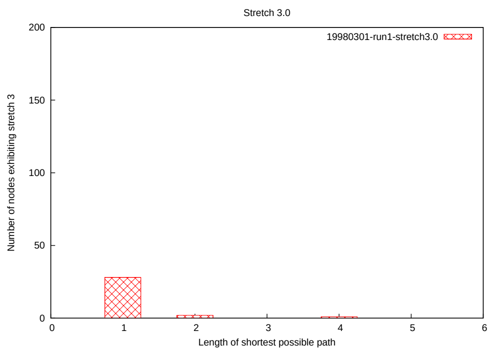
9.3 Summary

The Throup-Zwick algorithm appears to exhibit very low average stretch, a result similar to experiments performed on power law random graphs [27, 8, 12, 15]. Routing table sizes experienced by nodes within the Thorup-Zwick compact routing scheme are small containing on average only 1% of the graph to maintain correct forwarding and routing. Stretch-3 paths are rarely encountered, and many of them can be eliminated with use of the Krioukov addendum to the Thorup-Zwick scheme. Pairs of nodes with largest shortest path sizes appear to be treated fairly by the Thorup-Zwick scheme, with the large paths only experiencing stretch increases of 1.3.

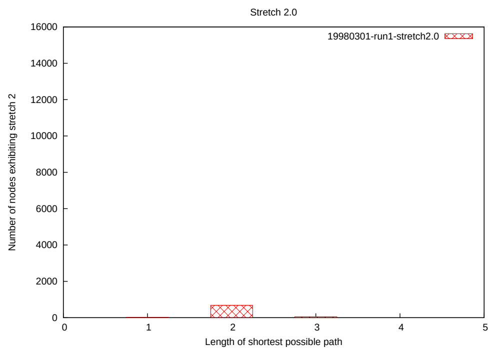
Unfortunately, due to problems encountered, the path stretch for Brady-Cowen cannot be analysed at this point. Routing table sizes are very promising, and paired with low stretch, this scheme could prove very favourable.



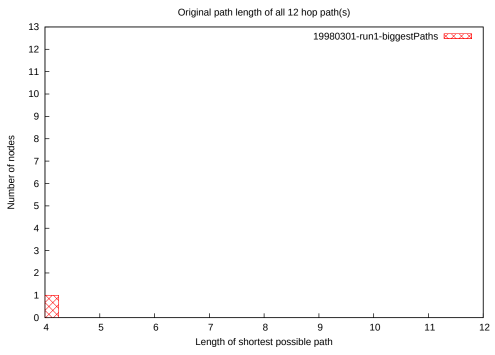
(a) Path stretch (log scale)



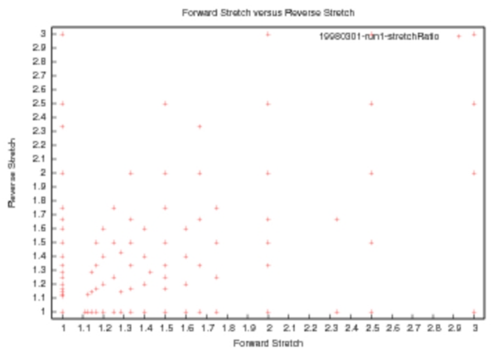
(b) Stretch 3 original path comparison



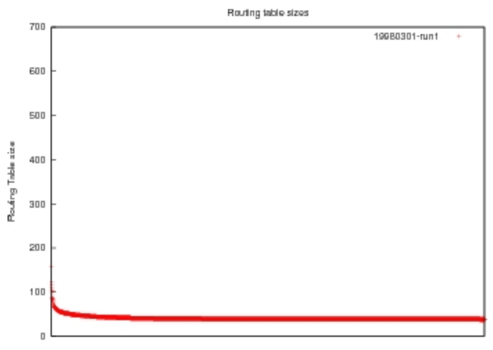
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

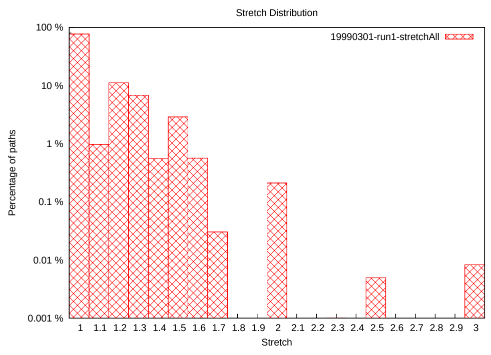


(e) Stretch comparison forward path versus reverse path

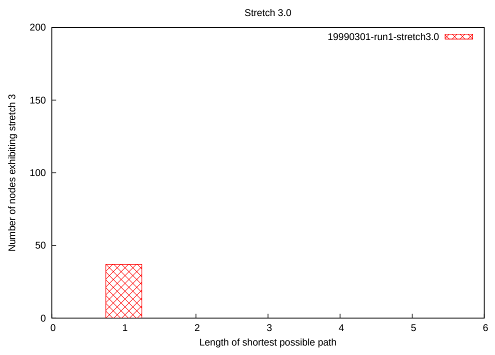


(f) Routing table sizes

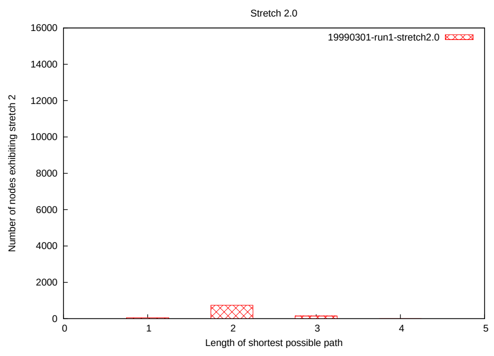
Figure 9.2: TZ results on the March 1998 snapshot



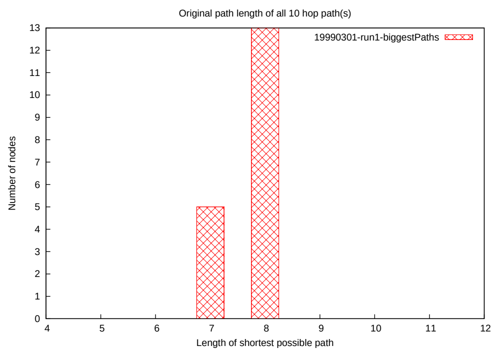
(a) Path stretch (log scale)



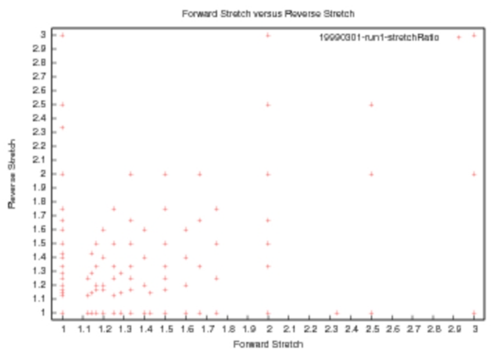
(b) Stretch 3 original path comparison



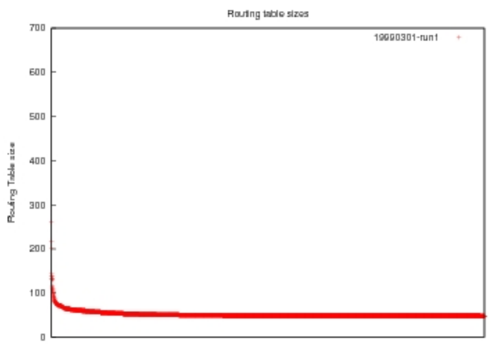
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

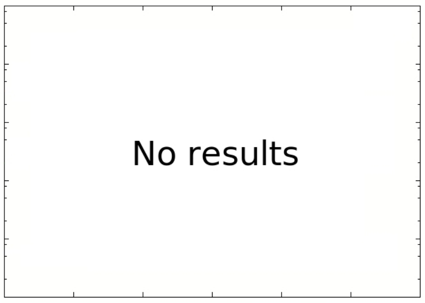


(e) Stretch comparison forward path versus reverse path

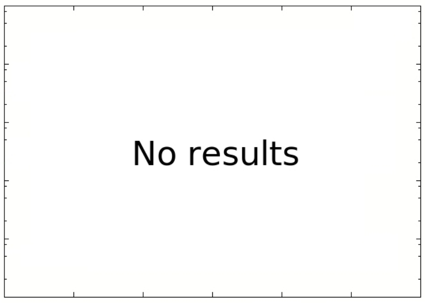


(f) Routing table sizes

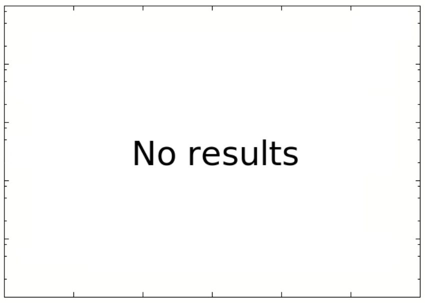
Figure 9.3: TZ results on the March 1999 snapshot



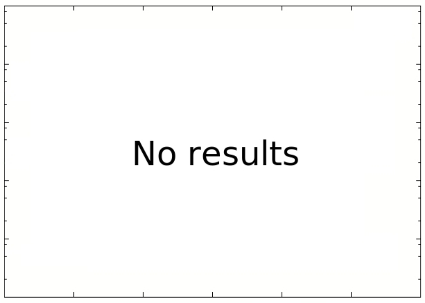
(a) Path stretch (log scale)



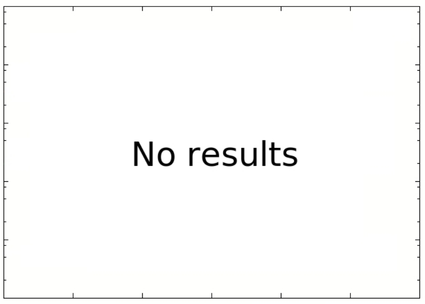
(b) Stretch 3 original path comparison



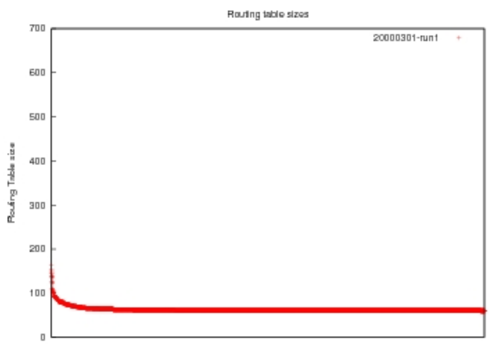
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

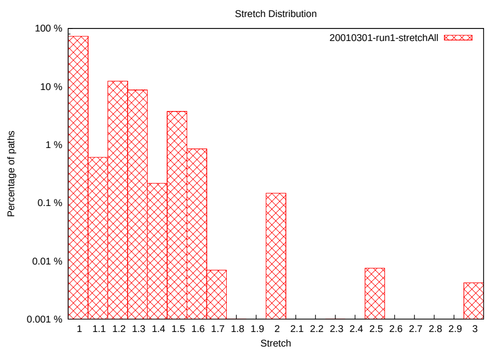


(e) Stretch comparison forward path versus reverse path

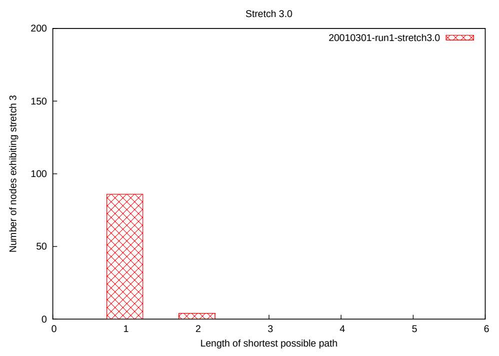


(f) Routing table sizes

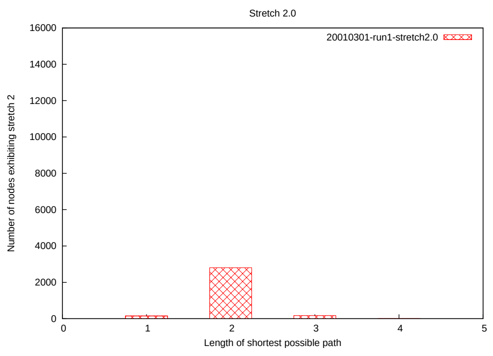
Figure 9.4: TZ results on the March 2000 snapshot. No stretch results as this snapshot causes the class loader to misplace essential files required by the simulator.



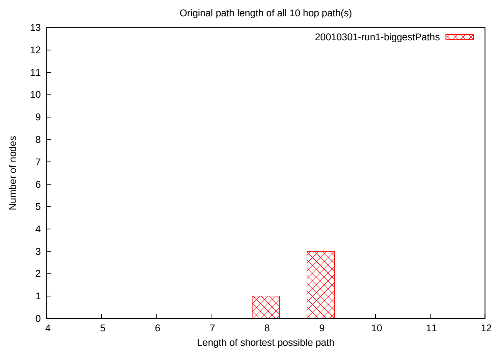
(a) Path stretch (log scale)



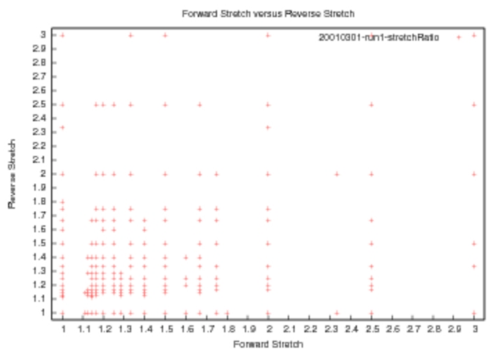
(b) Stretch 3 original path comparison



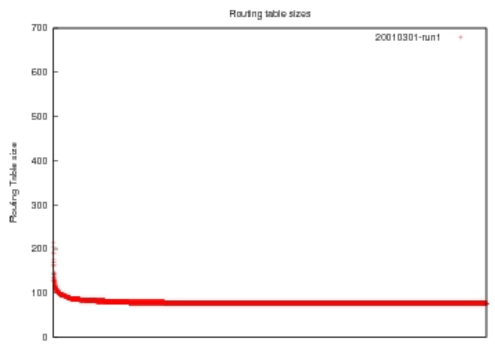
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

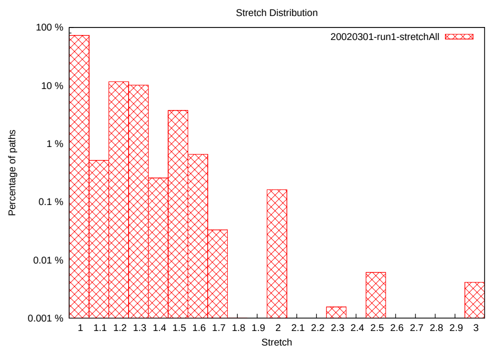


(e) Stretch comparison forward path versus reverse path

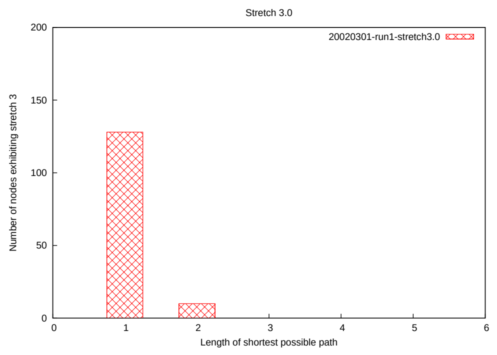


(f) Routing table sizes

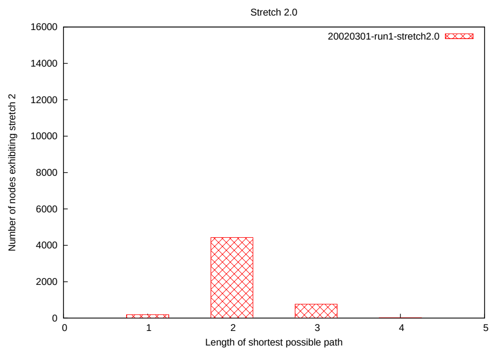
Figure 9.5: TZ results on the March 2001 snapshot



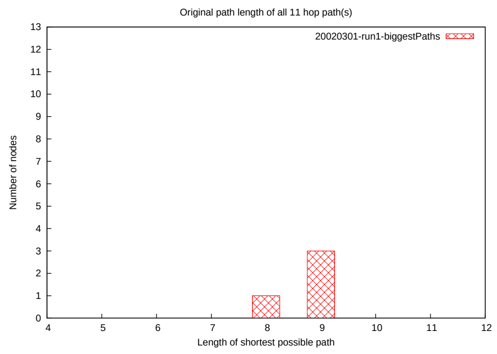
(a) Path stretch (log scale)



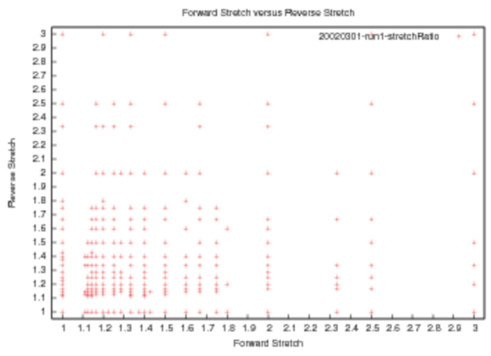
(b) Stretch 3 original path comparison



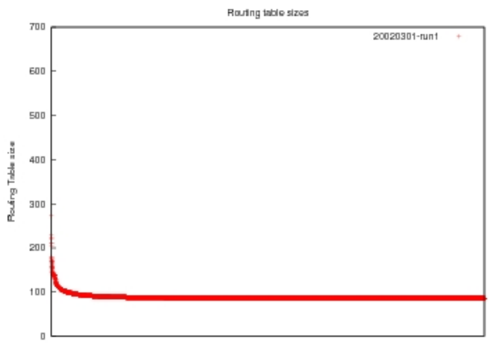
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

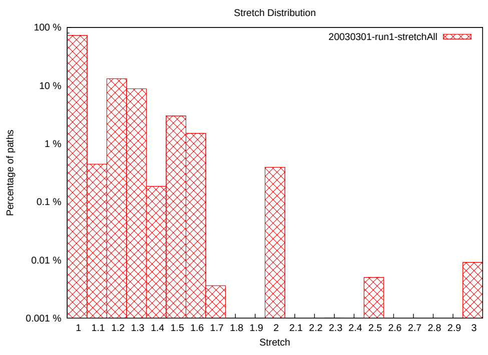


(e) Stretch comparison forward path versus reverse path

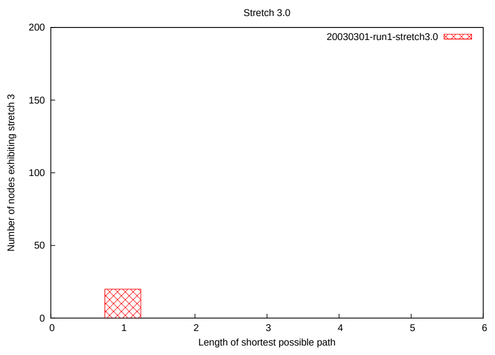


(f) Routing table sizes

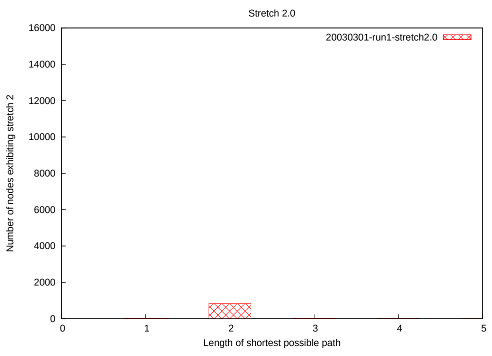
Figure 9.6: TZ results on the March 2002 snapshot



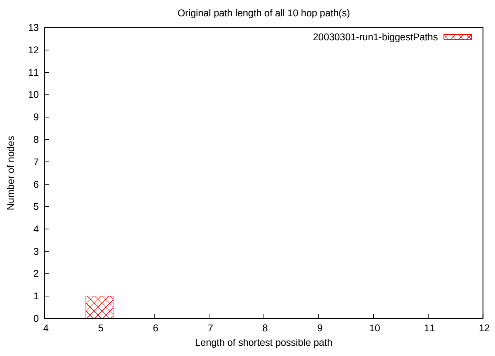
(a) Path stretch (log scale)



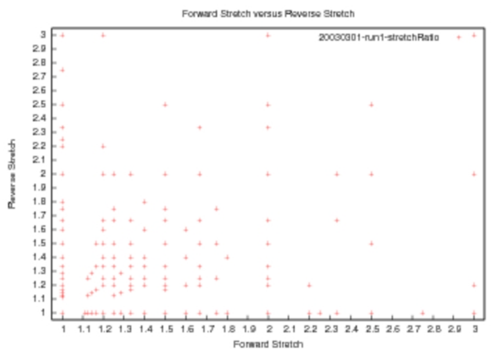
(b) Stretch 3 original path comparison



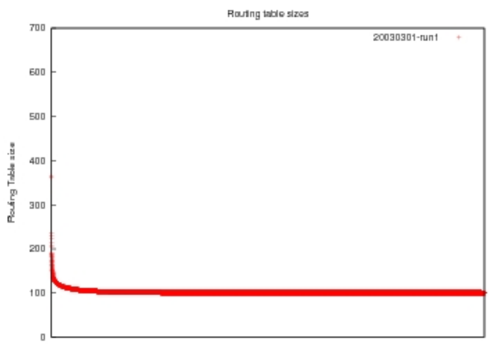
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

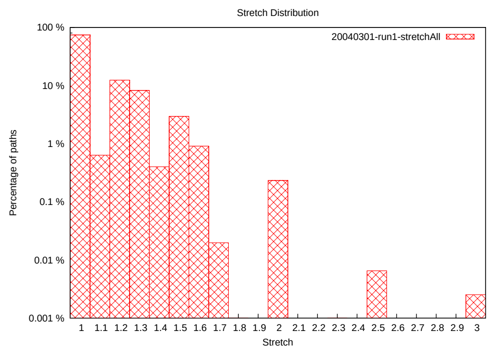


(e) Stretch comparison forward path versus reverse path

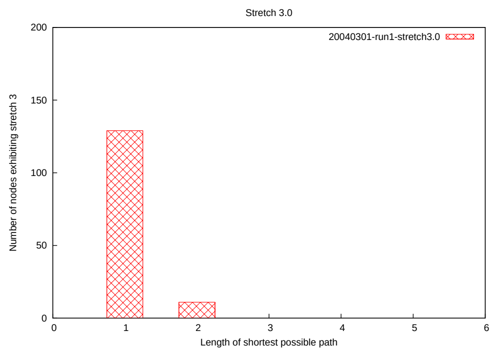


(f) Routing table sizes

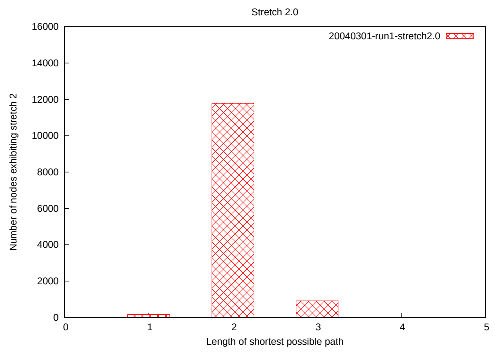
Figure 9.7: TZ results on the March 2003 snapshot



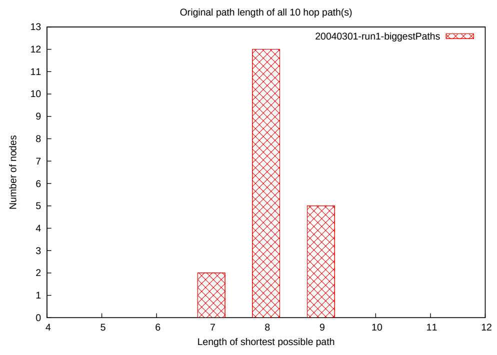
(a) Path stretch (log scale)



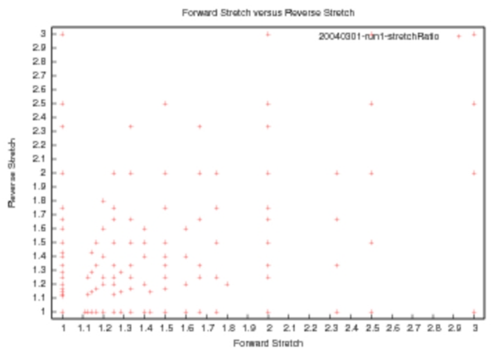
(b) Stretch 3 original path comparison



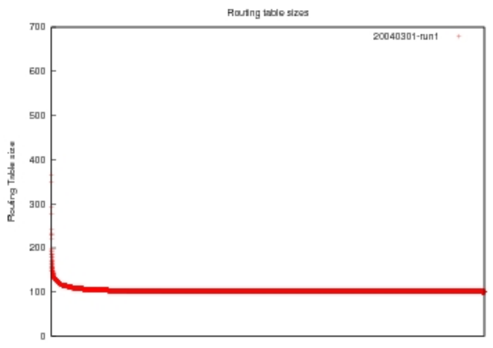
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

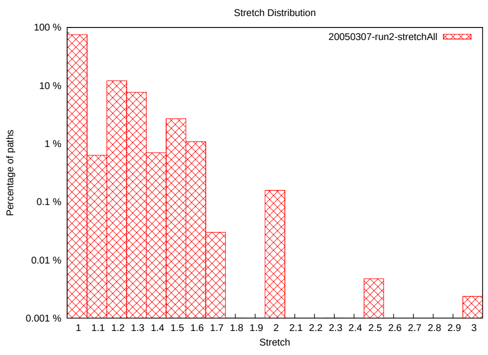


(e) Stretch comparison forward path versus reverse path

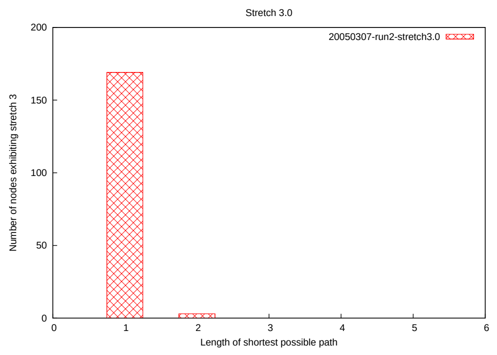


(f) Routing table sizes

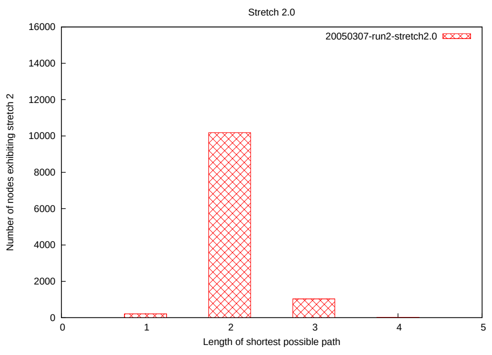
Figure 9.8: TZ results on the March 2004 snapshot using landmark set 1



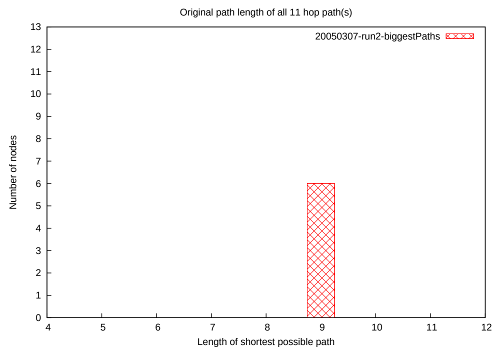
(a) Path stretch (log scale)



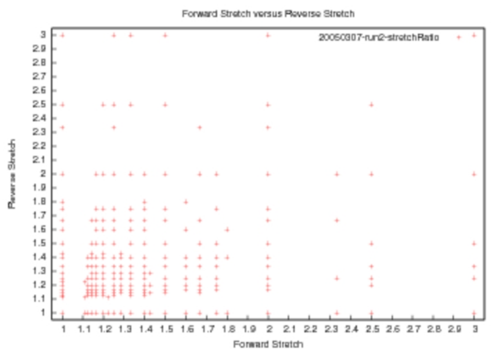
(b) Stretch 3 original path comparison



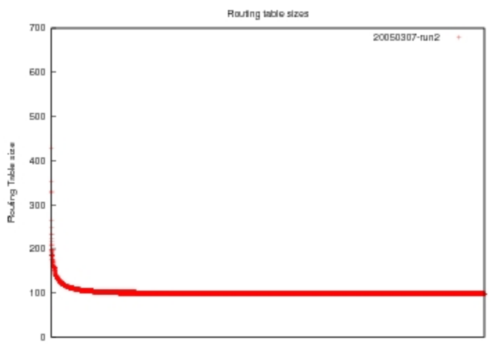
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

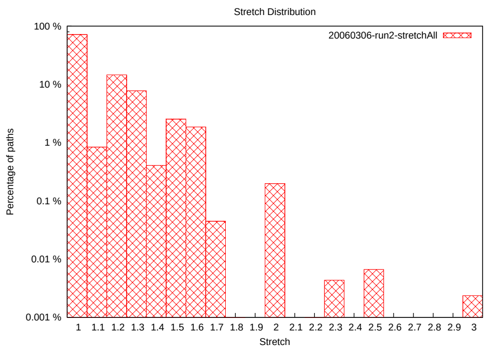


(e) Stretch comparison forward path versus reverse path

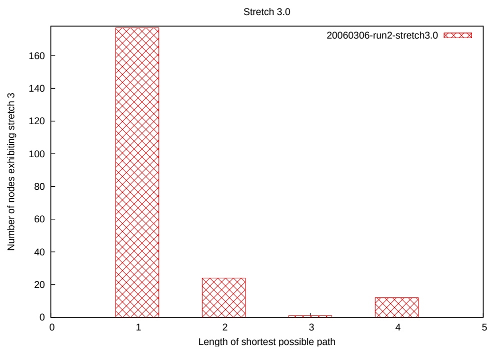


(f) Routing table sizes

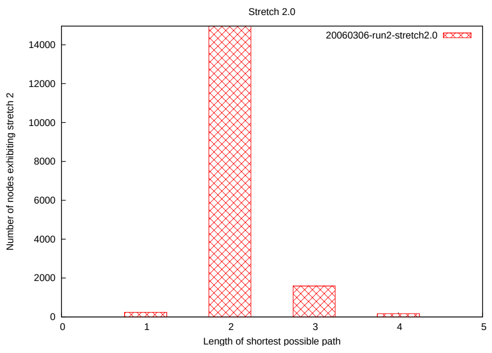
Figure 9.9: TZ results on the March 2005 snapshot



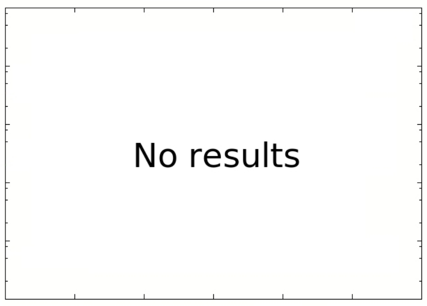
(a) Path stretch (log scale)



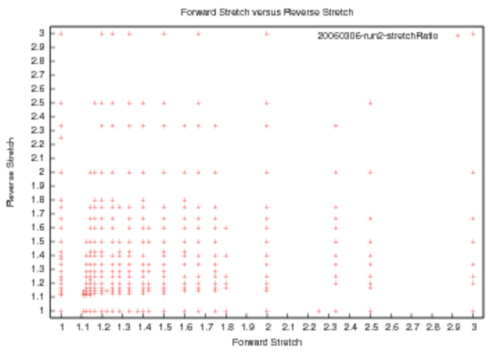
(b) Stretch 3 original path comparison



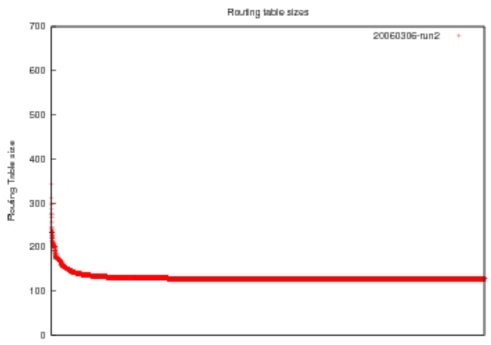
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

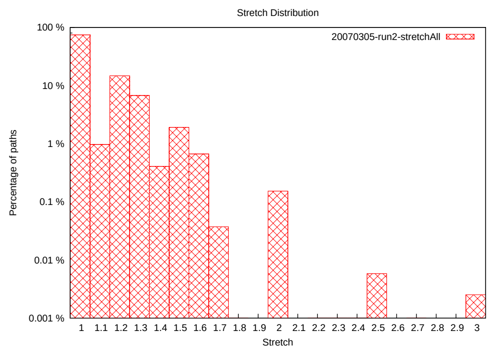


(e) Stretch comparison forward path versus reverse path

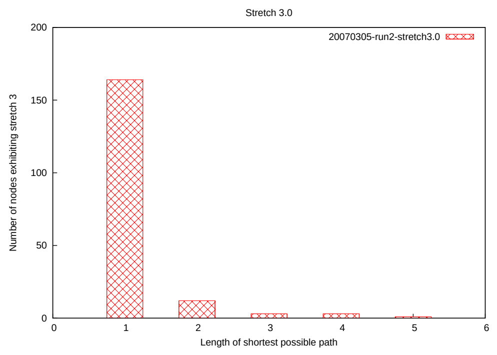


(f) Routing table sizes

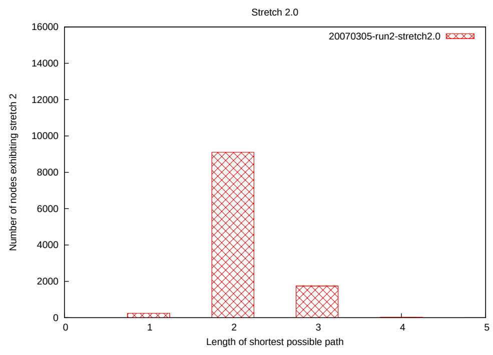
Figure 9.10: TZ results on the March 2006 snapshot. No longest path result due to corrupt result file.



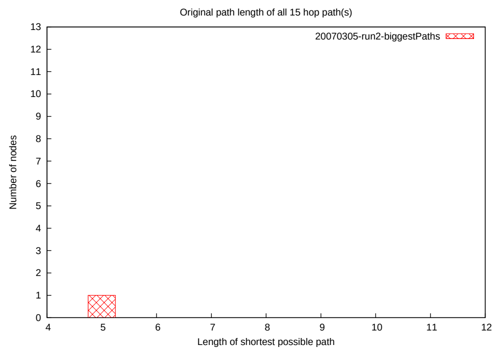
(a) Path stretch (log scale)



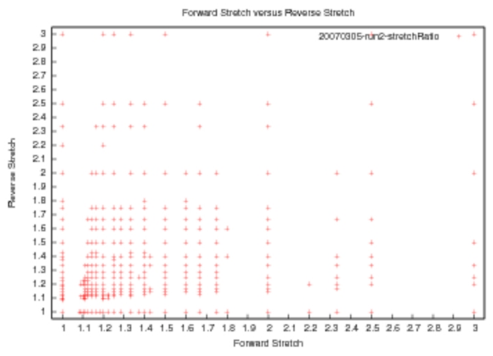
(b) Stretch 3 original path comparison



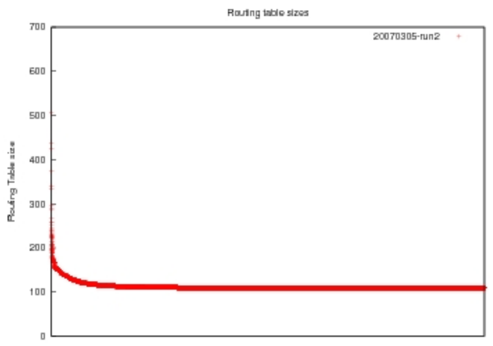
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

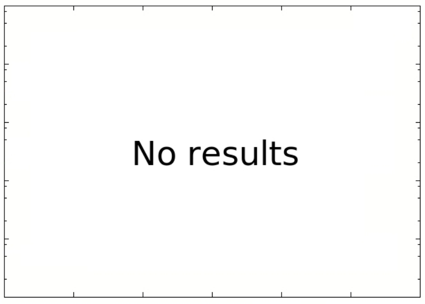


(e) Stretch comparison forward path versus reverse path

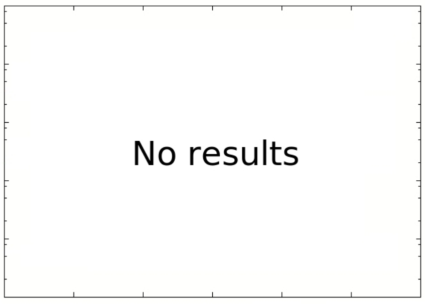


(f) Routing table sizes

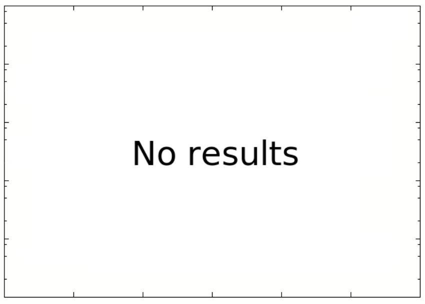
Figure 9.11: TZ results on the March 2007 snapshot



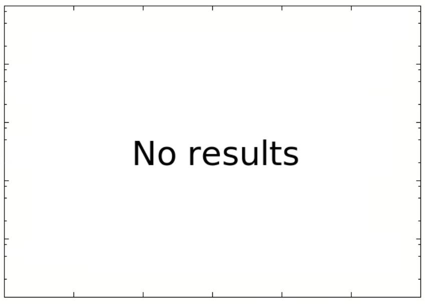
(a) Path stretch (log scale)



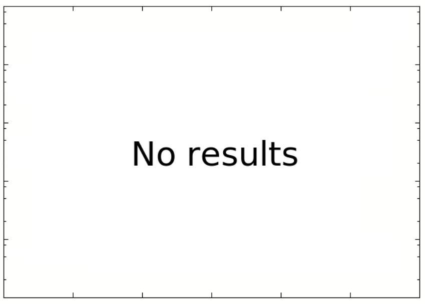
(b) Stretch 3 original path comparison



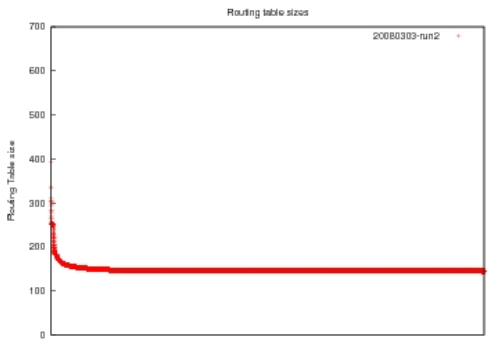
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length

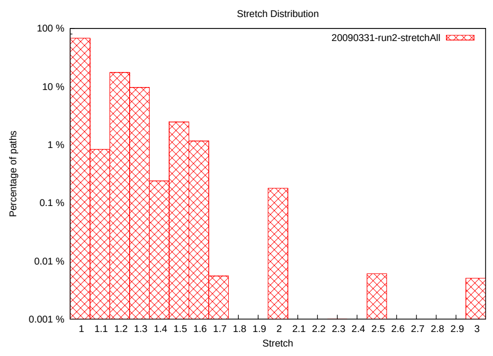


(e) Stretch comparison forward path versus reverse path

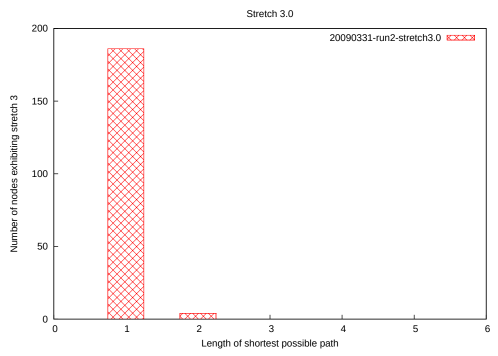


(f) Routing table sizes

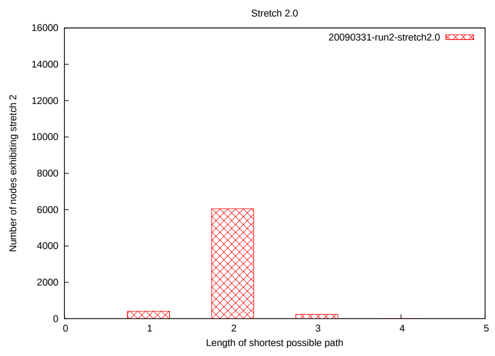
Figure 9.12: TZ results on the March 2008 snapshot. No stretch results due to simulator failing to initialise with this dataset



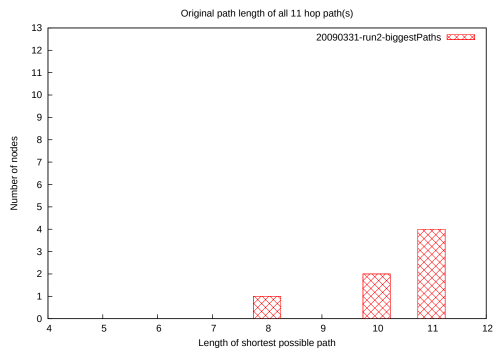
(a) Path stretch (log scale)



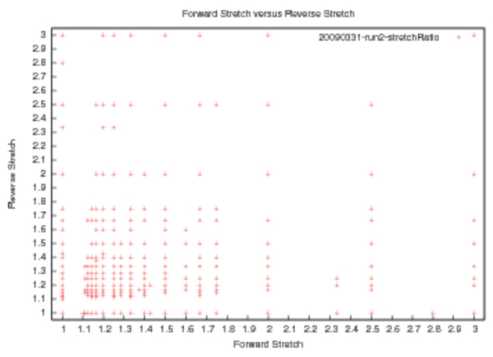
(b) Stretch 3 original path comparison



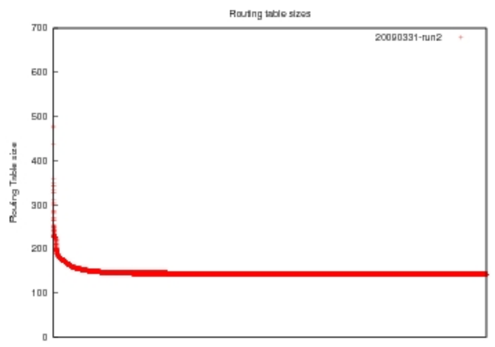
(c) Stretch 2 original path comparison



(d) Longest path versus the original path length



(e) Stretch comparison forward path versus reverse path



(f) Routing table sizes

Figure 9.13: TZ results on the March 2009 snapshot

Chapter 10

Comparison of algorithms

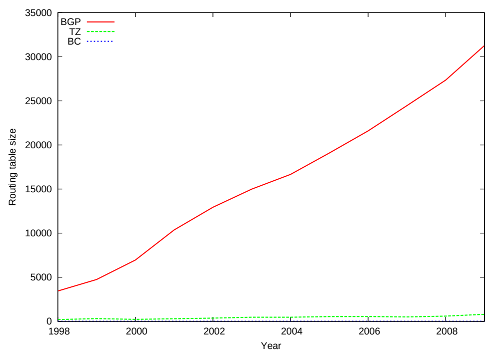
The previous chapter presented results of simulating the Thorup-Zwick and Brady-Cowen compact routing schemes on snapshot of the Internet AS topology. These schemes show potential for being very scalable inter-domain routing protocols. This chapter provides a comparison of the Thorup-Zwick and Brady-Cowen schemes with BGP, the current inter-domain routing protocol.

BGP is currently experiencing large routing tables due to de-aggregation of prefixes and the address space's natural growth [29]. This thesis aimed to explore the ability of compact routing to reduce routing table size. Figure 10 shows the routing table sizes over time for the Thorup-Zwick and Brady-Cowen compact routing schemes as well as the BGP. The BGP as previously mentioned is a prefix based routing scheme, for the purposes of comparison Figure 10 has the routing table sizes of BGP if it utilised AS numbers instead. This dissolves BGP to a distance vector algorithm, from its original form as a path vector approach.

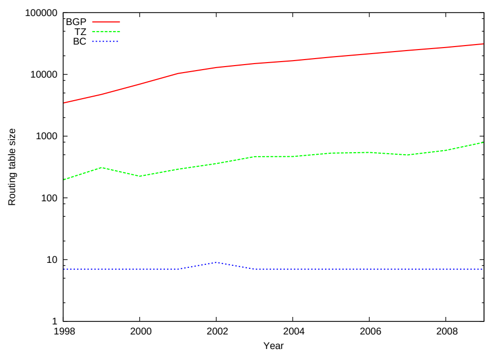
As can clearly be seen in Figure 10.1(a), the routing table sizes of BGP have increased linearly reaching a size of 31277 entries in March 2009. The table size for the Thorup-Zwick algorithm fluctuates as time progresses but maintains much smaller routing tables consistently over the years. The Thorup-Zwick algorithm in March 2009 produces a worst-case routing table size of 476 entries (1.5% of the size of BGP). The Brady-Cowen scheme routing table sizes can be witnessed in Figure 10.1(b). These routing tables do not fluctuate, but instead maintain a consistent worst-case routing table of size seven entries (0.02% of the size of BGP). The year 2002 provides an exception this, when the maximum routing table sizes were 9 entries in size.

This information clearly presents the routing state savings possible by utilisation of compact routing schemes. Both compact routing schemes produce routing tables substantially smaller than that of BGP, with the Brady-Cowen compact routing scheme coming out on top with the smallest routing table sizes. Although this comes at a cost of higher complexity than the Thorup-Zwick scheme.

The compact routing schemes provide this substantially lower routing table size at a cost, increasing path length of traffic traversing the network. However, as can be shown in Figure 9.1 the Thorup-Zwick compact routing scheme exhibits an average path stretch of approximately 1.1. This stretch value is low, in fact it is most likely



(a) Routing table size



(b) Routing table size (log scale)

Figure 10.1: Routing table size over time BGP, TZ and BC

on par with the stretch of BGP. Packets in BGP are subjected to the local policy of each AS which may choose to route a packet on a sub-optimal path, increasing the path length on that packet. However, without simulation of the BGP on the snapshots for 1998 to 2009, we cannot provide direct comparison. Similarly for the Brady-Cowen compact routing scheme. Stretch comparison between the three schemes mentioned here is highly recommended as future work.

The routing table size reduction offered by compact routing schemes is certainly substantial, and is a favourable trait of these schemes. These schemes show resilience to the AS-level graphs increasing in size and provide consistently low routing table sizes over time; demonstrating the ability of compact routing schemes to provide sub-linearly scaling routing tables.

Chapter 11

Conclusion

This dissertation has presented two compact routing schemes, Thorup-Zwick and Brady-Cowen, it has detailed their operation and the stages necessary to implement these schemes. Two distributed simulation environments were created to allow the analysis of routing table sizes and path stretch of these compact routing schemes. The results show that routing table sizes due to these compact routing algorithms are significantly smaller than a shortest path routing approach (e.g., distance vector), as well as exhibiting in small mean path stretch of approximately 1.1. This is a significant finding, allowing a substantial reduction in routing state while incurring only a small side-effect on path stretch. The routing tables scale sub-linearly with respect to graph size increases. These factors suggest compact routing is a favourable candidate for a scalable routing protocol.

11.1 Future Work

Unfortunately due to time constraints it was not possible to perform simulations on the Brady-Cowen compact routing scheme. This is future work that would benefit the advancement of the compact routing field. It would allow for comparison with the Thorup-Zwick compact routing scheme as well as other potential routing protocols. Simulation of BGP path stretch relative to the shortest distance on the unannotated graph, to allow for a direct comparison is also recommended.

Within this project, it was possible to evaluate the effect of the Thorup-Zwick landmark selection algorithm on path stretch and routing table sizes, as it could be a potential area of bias. The Brady-Cowen compact routing scheme also has a random element to it; the Brady-Cowen heuristic randomly picks up to five edges from the fringe should less than five extra edges be detected within the fringe. This random element has the ability affect on the results of simulating the Brady-Cowen compact routing scheme. As such, analysis of the effects of the Brady-Cowen heuristic is recommended before performing a comprehensive study of the Brady-Cowen scheme.

Another aspect of the experiment that could not be performed due to time constraint was the use of popular websites to produce more realistic traffic within the simulators (see chapter 7). This would allow for analysis of how frequently used paths contend with path stretch. If frequently utilised paths exhibit large stretch this degrades the through-put of the Internet, and would be a disadvantage towards the use of compact routing schemes. Ways to avoid this large increase in path stretch would have to be devised to allow a favourable outcome for compact routing.

The experiment performed for this dissertation analysed the longest path witnessed within the simulator as a metric to determine the effect of path stretch. Another approach to determining the effect of path stretch would be to observe the stretch exhibited by the source-destination pairs that exhibit the largest shortest path within the graph. A large multiplicative stretch on these already large paths would be detrimental to the use of compact routing. This test case must be analysed to see if it is a problem.

The landmark set is determined by a the landmark selection algorithm. This algorithm is random in nature, and as such may have sub-optimal spread of the landmarks. It may be the case that several landmarks are placed close together on the first iteration of the landmark selection algorithm. This results in additional routing table entries in all nodes. From a theoretical perspective, it would be interesting to see if it is possible to reduce a landmark set after all the cluster constraints have been satisfied. This would reduce the routing table size in a large quantity of nodes, however, would such a change radically affect path stretch? A heuristic or algorithm to reduce the landmark set would have to be devised, this may prove challenging.

The compact routing schemes utilised in this dissertation are valid only within the static graph model. The real-world Internet does not follow this model; node and link failures can happen at any time. As such, for compact routing to be deployed on the inter-domain scale, it would have to be able to deal with such failures, as well as the addition of more nodes and links. As such future work in the compact routing field could research ways to make existing compact routing schemes fit for purpose, or research new compact routing schemes to perform this task. This is a substantial amount of work.

Another potential weakness in compact routing is its inability to allow businesses to implement policy within their domain. AS administrators require that they can perform traffic engineering, multi-homing, and other requirements. A future route for research would allow for the policy to be integrated into a compact routing scheme, or again, a new scheme devised that allows this policy. This is another requirement should compact routing wish to become a potential inter-domain routing protocol. This is substantial work.

11.2 Summary

Compact routing schemes show great potential for reducing routing table sizes, as well as maintaining a sub-linear growth of the routing tables as graph size increases. This has been demonstrated by simulating two compact routing schemes by Thorup-Zwick and Brady-Cowen on real-world Internet AS-level topology snapshots over twelve years. This resulted in both schemes producing very small routing tables in comparison to the currently operating inter-domain protocol. This routing table state reduction comes at the cost of increase path lengths for packets, however, this path stretch has been shown to be consistently very low in simulations. Further investigation into the feasibly of compact routing schemes as a replacement to the current inter-domain routing protocol is recommended due to the merits previously highlighted.

Bibliography

[1] The CAIDA AS relationships dataset. http://caida.org/data/active/as-relationships/, April 2010.

[2] Tier 1 network – wikipedia. http://en.wikipedia.org/wiki/Tier_1_network, April 2010.

[3] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill. IPv4 multihoming practices and limitations. RFC4116, Internet Engineering Task Force, July 2005.

[4] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. *ACM Transactions on Algorithms*, 4(3):1–12, 2008.

[5] M. Arias, L. J. Cowen, K. A. Laing, R. Rajaraman, and O. Taka. Compact routing with name independence. In *SPAA '03: Proceedings of the 15th annual ACM symposium on Parallel algorithms and architectures*, pages 184–192, San Diego, California, USA, June 2003. ACM.

[6] H. Ballani, P. Francis, T. Cao, and J. Wang. Making routers last longer with ViAggre. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 453–466, Berkeley, CA, USA, 2009. USENIX Association.

[7] J. Behrens and J. J. Garcia-Luna-Aceves. Distributed, scalable routing based on link-state vectors. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 136–147, London, United Kingdom, August/September 1994. ACM.

[8] A. Brady and L. J. Cowen. Compact routing on power law graphs with additive stretch. In *ALENEX '06 Proceedings of the 8th workshop on Algorithm engineering and experiments*, pages 119–128, January 2006.

[9] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. ROFL: routing on flat labels. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 363–374, Pisa, Italy, September 2006. ACM.

[10] E. Chen. Route refresh capability for BGP-4. RFC2918, Internet Engineering Task Force, September 2000.

[11] Kai Chen, David R. Choffnes, Rahul Potharaju, Yan Chen, Fabian E. Bustamante, Dan Pei, and Yao Zhao. Where the sidewalk ends: extending the Internet as graph using traceroutes from P2P users. In *CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 217–228, Rome, Italy., December 2009. ACM.

[12] W. Chen, C. Sommer, S. Teng, and Y. Wang. Compact routing in power-law graphs. In *Proceedings of the 23rd international symposium on Distributed computing*. Springer-Verlag LNCS 5805, September 2009.

[13] L. J. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 38(1):170–183, 2001.

[14] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, kc claffy, and G. Riley. AS relationships: inference and validation. *ACM SIGCOMM Computer communication review*, 37(1):29–40, 2007.

[15] M. Enachescu, M. Wang, and A. Goel. Reducing maximum stretch in compact routing. In *INFOCOM '08: The 27th conference on Computer communications*, pages 336–340, Phoenix, Arizona, April 2008. IEEE.

[16] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis. Locator/ID separation protocol (LISP). Internet-draft, Internet Engineering Task Force, January 2010. Work in progress.

[17] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, December 2001.

[18] C. Gavoille and M. Gengler. Space-efficiency for routing schemes of stretch factor three. *Journal of Parallel distributed computing*, 61(5):679–687, 2001.

[19] B. Halabi. *Internet Routing Architectures*. Cisco Press, 1997. ISBN: 1562056522.

[20] G. Huston. Commentary on inter-domain routing in the Internet. RFC3221, Internet Engineering Task Force, December 2001.

[21] L. Iannone and O. Bonaventure. On the cost of caching locator/id mappings. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, USA, December 2007. ACM.

[22] E. Karpilovsky and J. Rexford. Using forgetful routing to control BGP table size. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–12, Lisboa, Portugal, December 2006. ACM.

[23] F. Kastenholz. ISLAY a new routing and addressing architecture. Internet-draft, Internet Engineering Task Force, May 2002. Work in progress.

[24] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks, performance evaluation and optimization. *Computer networks*, 1(3):155–174, January 1977.

[25] A. Korman. Improved compact routing schemes for dynamic trees. In *PODC '08: Proceedings of the 27th ACM symposium on Principles of distributed computing*, pages 185–194, Toronto, Canada, August 2008. ACM.

[26] D. Krioukov, k c claffy, K. Fall, and A. Brady. On compact routing for the Internet. *ACM SIGCOMM Computer communication review*, 37(3):41–52, 2007.

[27] D. Krioukov, K. Fall, and X. Yang. Compact routing on Internet-like graphs. In *INFOCOM '04. 23rd annual joint Conference of the IEEE Computer and communications societies*, pages 209–219, Hong Kong, PR China, March 2004.

[28] L. Mathy and L. Iannone. Lisp-dht: Towards a dht to map identifiers onto locators. In *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–6, Madrid, Spain, December 2008. ACM.

[29] D. Meyer, L. Zhang, and K. Fall. Report from the IAB workshop on routing and addressing. RFC4984, Internet Engineering Task Force, September 2007.

[30] S. Milgram. The small world problem. *Psychology Today*, 1:61–67, May 1967.

[31] M. O'Dell. 8+8 an alternative addressing architecture for IPv6. Internet-draft, Internet Engineering Task Force, October 1996. Work in progress.

[32] 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*, 18(1):109–122, February 2010.

[33] D. Peleg. Proximity-preserving labeling schemes and their applications. In *WG '99: Proceedings of the 25th international workshop on Graph-theoretic concepts in computer science*, pages 30–41, Ascona, Switzerland, June 1999. Springer-Verlag.

[34] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure. Evaluating the benefits of the locator/identifier separation. In *MobiArch '07: Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving Internet architecture*, pages 1–6, Kyoto, Japan, August 2007. ACM.

[35] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). RFC4271, Internet Engineering Task Force, January 2006.

[36] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC3031, Internet Engineering Task Force, January 2001.

[37] S. D. Strowes and C. Perkins. Towards a graph simulator for analysis of inter-domain routing and forwarding behaviour. Submission to IEEE International Conference on Communications '10.

[38] 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 *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 13–24, Philadelphia, Pennsylvania, USA, August 2005. ACM.

[39] M. Thorup and U. Zwick. Compact routing schemes. In *SPAA '01: Proceedings of the 13th annual ACM symposium on Parallel algorithms and architectures*, pages 1–10, Crete Island, Greece, July 2001. ACM.