Tapio Partti

Improving Internet Inter-domain Routing Scalability

Faculty of Electronics, Communications and Automation

Supervisor:

Prof. Raimo Kantola

Instructor:

Lic. (Tech.) Hannu Flinck

Aalto-yliopisto
Teknillinen korkeakoulu

AALTO-YLIOPISTO
TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Tapio Partti

Työn nimi: Internetin alueiden välisen reitityksen skaalautuvuuden kehittäminen

Päivämäärä: 21.2.2010 Kieli: Englanti Sivumäärä: 9+79

Elektroniikan, tietoliikenteen ja automaation tiedekunta

Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotekniikka Koodi: S-38

Valvoja: Prof. Raimo Kantola

Ohjaaja: TkL Hannu Flinck

Internetin alueiden välisen reitityksen skaalatuminen on nähty ongelmallisena jo vuosia. Sen välittömiä seurauksia ovat operaattoreille aiheutuvat suuret kustannukset ja tilanteen pahentuessa myös Internetin kasvamisen vaarantuminen. Tähän mennessä toteutetut parannukset ovat vain lieventäneet ongelmia tai viivästyttäneet niiden ilmenemistä. Reitittimien muistia pidetään suurimpana lähitulevaisuuden haasteena, koska reitittimien täytyy pystyä ohjaamaan sisääntulevat paketit nopeasti kohti jotakin jopa useista sadoista tuhansista verkoista.
Tämä opinnäytetyö esittelee ongelma-alueen tunnistamalla suurimmat ongelmat sekä niiden perimmäiset syyt, ja kartoittaa näiden ongelmien tärkeyttä. Parannusmenetelmistä muutama oleellisin on esitelty ja analysoitu. Syvempi analyysi kohdistuu työssä ennen kaikkea reititinten muistintarvetta pienentävään Virtual Aggregation -menetelmään, jonka kantavana ideana on sallia virtuaalisten IP-osoiteprefixien käyttö yksittäisten verkkojen sisällä. Työ esittelee myös uuden tavan muodostaa ja käyttää näitä virtuaalisia IP-osoiteprefixejä ja vertailee sitä muihin nopean muistin tarvetta vähentäviin menetelmiin simuloimalla näitä Sprint operaattorin verkkotopologiassa. Simulointitulosten perusteella esittämämme menetelmä kykenee huomattaviin muistisäästöihin välttäen samalla joitain Virtual Aggregation -menetelmästä löydetyistä ongelmista. Menetelmän jatkokehitystä on myös mietitty simulointien pohjalta.

Avainsanat: operaattoreiden välinen reititys, BGP, skaalautuvuus, aggregointi, Virtual Aggregation, FIB Aggregation

AALTO UNIVERSITY SCHOOL OF SCIENCE AND TECHNOLOGY	ABSTRACT OF THE MASTER'S THESIS

Author: Tapio Partti

Title: Improving Internet Inter-domain Routing Scalability

Date: 21.2.2010 Language: English Number of pages: 9+79

Faculty of Electronics, Communications and Automation

Department of Communications and Networking

Professorship: Networking and Communication Code: S-38

Supervisor: Prof. Raimo Kantola

Instructor: Lic. (Tech.) Hannu Flinck

For years inter-domain routing scalability has been seen as a problem which increases ISPs costs and may even decelerate the growth of the Internet. Few improvements have been made over the years, but they have only delayed the issue. Router memories (i.e. FIBs) are the most critical concern as they have to be fast and ever larger to handle great amounts of packets to possibly hundreds of thousands of networks.
This thesis introduces the problem set by identifying the main issues and their root causes, as well as present analysis on their criticality. The improvement mechanisms are also considered by introducing and comparing few most relevant proposals. Deeper study and analysis in this thesis focuses on Virtual Aggregation which allows networks to individually lower their routers' memory load via the use of virtual IP address prefixes. Also, a new solution for allocating Virtual Prefixes and aggregation points for them is introduced and compared against other FIB shrinking mechanisms using extensive simulations on Sprint topology. As a result, the new solution is identified to save FIBs considerably while avoiding some drawbacks found in Virtual Aggregation. Further improvements to the mechanism are also considered although not tested.

Keywords: Internet inter-domain routing, BGP, scalability, aggregation, Virtual Aggregation, FIB Aggregation

Esipuhe

Tämän opinnäytetyö on tehty Nokia Siemens Networks:n ja TEKES:n tuella, osana TIVIT Future Internet -projektia.

Haluan kiittää Professori Raimo Kantolaa ja ohjaajaani Hannu Flinckiä hyvästä ohjauksesta ja neuvoista. Kiitoksen ansaitsevat myös useat ihmiset Future Network Architectures ryhmässä NSN:llä. Erityisesti Petteri Pöyhönen, Jarno Rajahalme ja Johanna Heinonen ovat neuvoillaan ja kommenteillaan vaikuttaneet tämän työn syntymiseen. Työn mahdollistamisesta haluan kiittää koko NSN:ää ja lähintä esimiestäni Jari Lehmusvuorta. Lopuksi haluan kiittää vielä perhettäni ymmärtäväisyydestä ja tuesta, jota olen heiltä saanut työn tekemisen aikana.

Espoo, 21.2.2011

Tapio Partti

Foreword

This thesis was supported by Nokia Siemens Networks and TEKES as part of the Future Internet TIVIT project.

I would like to thank Professor Raimo Kantola and my instructor Hannu Flinck for great guidance and instructions. Several people from Future Network Architectures team at NSN have also my gratitude. Especially Petteri Pöyhönen, Jarno Rajahalme and Johanna Heinonen have influenced the outcome of this thesis with their advices and comments. For enabling me to this thesis I thank the entire NSN and my direct manager Jari Lehmusvuori. Finally, I would like to thank my family for understanding and support I have got during working with the thesis.

Espoo, 21.2.2011

Tapio Partti

Table of contents

Abbreviations

AIS	Aggregation in Increasing Scopes
ALT	ALternative Topology
API	Application Programming Interface
APR	Aggregation Point Router
AS	Autonomous System
ASBR	Autonomous System Border Router
ASN	Autonomous System Number
BB	BackBone
BGP	Border Gateway Protocol
CIDR	Classless Inter-Domain Routing
CoA	Care of Address
DFRT	Default Free Routing Table (global routing table)
DFZ	Default Free Zone
DNS	Domain Name System
DPPS	Default Popular Prefix Selection
DRAM	Dynamic Random Access Memory
eBGP	external BGP
EID	Endpoint IDentifier
ETR	Egress Tunnel Router
FIB	Forwarding Information Base
HA	Home Agent
HIP	Host Identity Protocol
IAB	Internet Architecture Board
IANA	Internet Assigned Numbers Authority
iBGP	internal BGP
ICT	Information and Communication Technologies
IETF	Internet Engineering Task Force
IGP	Internal Gateway Protocol
ILNP	Identifier-Locator Network Protocol
IP	Internet Protocol
IPPS	Including Popular Prefix Selection
IPSec	IP Security architecture
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IRTF	Internet Research Task Force
ISP	Internet Service Provider
ITR	Ingress Tunnel Router
L2	Layer 2
L3	Layer 3
LIR	Local Internet Registry
LISP	Locator Identifier Separation Protocol
LSP	Label Switched Path
MIPv4	Mobile IPv4

MIPv6	Mobile IPv6
MPLS	Multi-Protocol Label Switching
MRAI	Minimum Route Advertisement Interval
NAT	Network Address Translation
NEMO	NEtwork MObility
NIR	National Internet Registry
PA	Provider Aggregated
PC	Personal Computer
PE	Provider Edge
PHP	Penultimate Hop Popping
PI	Provider Independent
PIM-SM	Protocol Independent Multicast – Sparse Mode
POP	Point Of Presence
PP	Popular Prefix
PPPS	Picky Popular Prefix Selection
RIB	Routing Information Base
RIPE	Réseaux IP Européens (European IP Networks)
RIR	Regional Internet Registry
RLDRAM	Reduced Latency DRAM
RLOC	Routing LOCator
RR	Route Reflector
SRAM	Static Random Access Memory
TCAM	Ternary Content-Addressable Memory
TCP	Transmission Control Protocol
TE	Traffic Engineering
VA	Virtual Aggregation
VFT	Virtual Forwarding Table
VP	Virtual Prefix
VPN	Virtual Private Network
VRF	VPN Routing and Forwarding

1. Introduction

Only two decades ago nobody could predict how important Internet will be for so many people and businesses already at the beginning of the 21st century. Internet has grown with a remarkable speed for the last fifteen years and has shaped the society fundamentally by enabling global communications and businesses even for common citizens. It has played a crucial role in building today's information society by revolutionizing the way information is managed and exchanged.

Taking into account that Internet originates from the trusted University world, it is a small miracle that it even works as well as it does in our world of conflicting interests. Yet at the same time, its open design has made it so flexible and popular. During the last two decades Internet has faced many challenges which have usually been solved only incrementally. Today the challenges are perhaps more severe and versatile than ever before which, if left unsolved, will slow down Internet's further growth, make it too expensive and disallow the creation of some wanted new services.

There are some fundamental limitations in the design of several key protocols which dictate the way routing and addressing is done in the Internet. One of the most problematic and important is the Internet Protocol (IP). Its first widely used version (version 4) has too small address space to allow every network interface in every device to have a globally unique IP address while even more fundamental problem is the way addresses are used in Internet in general. In road network addresses indicate only locations, but in Internet they also specify hosts' identities. This makes mobility and many other functions hard and costly. Moreover, it exacerbates the creation of a scalable routing system which can handle billions of users cost efficiently. The Internet inter-domain routing scalability problem is most acute on the routers which cannot keep up with the constantly rising number of routing entries. Forwarding tables are often the bottlenecks because they must reside on fast memory in order to not adversely delay packet forwarding.

It is indeed the scalability which has been seen as a problem for many years. Some enhancements have been deployed along the years, but they have only pushed the problem slightly into the future. But, if the problem has been identified years ago, why is it so hard to fix? The reasons are exactly the same which have made Internet so important: size and openness. Internet is approximated to currently have about two billion users [1] and thousands of Autonomous Systems (AS) which offer their customers the connectivity to the Internet and participate in the inter-domain routing at the heart of the Internet. By considering that most ASs are private companies which have business responsibilities, it is easy to understand that they will probably never share a common view on what if anything should be done to correct large scale problems. Most actions would also require changes on many devices, which ask for a great amount of coordination, work and money.

In an attempt to address the scalability and other issues in Internet inter-domain routing, Internet Research Task Force (IRTF) asked the research community to bring forth their suggestions on how the problems should be solved. By the summer 2010 they received several proposals out of which two were selected as recommendations for Internet Engineering Task Force (IETF) to work on: 1) Evolution Towards Global Routing Scalability (Evolution) as a short term and 2) Identifier-Locator Network Protocol (ILNP) as a more final solution. Address renumbering needs also to be considered together with the ILNP. In addition to the IRTF recommended proposals Locator Identifier Separation Protocol (LISP) is entitled to get significant attention as well because it is driven by a major player in the industry and may thus end up being used.

The objectives of this thesis are to introduce Internet inter-domain routing along with its many challenges and the possible future solutions to the reader as well as to evaluate the capabilities of Virtual Aggregation (VA) and deepen the understanding on it as a central part of the Evolution proposal. The assets of Evolution are on its approach to solve the problems. Therefore, it is a more plausible solution to be deployed than many of the other proposals which design a novel new architecture and lack real incentives for early adopters. VA is a network wide mechanism which uses rather short Virtual Prefixes and tunnels to sacrifice packet latencies in minor extent in order to drastically diminish FIBs on possibly all of the routers.

This thesis does not only introduce and analyze what others have done before, but also identifies a previously undocumented issue in VA and describes a new way of using it. To be able to base the analysis on something concrete, a simulation environment resembling the Sprint network from 2002 is constructed with Matlab. Various simulations are run on that environment to show the benefits and drawbacks in using VA, the relevance of the previously undocumented problem and the feasibility of the new design.

This thesis continues from here by first briefly going through some relevant background information related to Internet routing in Chapter 2. Chapter 3 then properly specifies the research question by elaborating on a number of inter-domain routing challenges and analyzing their core reasons. Chapter 4 introduces three different approaches (LISP, Evolution and ILNP) from the larger set of proposals which aim to alleviate many of the problems described in Chapter 3. The functionality of Virtual Aggregation is considered in Chapter 5 along with its strong and weak properties. From Chapter 6 onwards the focus is more on what new this thesis brings and what kind of studies have been conducted. More specifically Chapter 6 introduces a new approach in using VA while Chapters from 7 to 9 elaborate on the conducted simulations on VA, present the acquired results, and discuss their meaning and relevance. Finally, Chapter 10 concludes this thesis by summing up what it has presented, how the proposed new mechanism could be further improved and how this all relates to building the future Internet.

2. Background

To be able to understand issues in Inter-domain Routing, some background information is needed. This Chapter briefly introduces how addressing and routing is done in the Internet by explaining IP addressing, the inter-domain routing protocol called Border Gateway Protocol (BGP), and the basic architecture in IP routers.

2.1 IP Addressing and Aggregation

Starting from 1993 a historical Internet Protocol version 4 (IPv4) addressing scheme with three network classes [2] was replaced by Classless Inter-domain Routing (CIDR) [3], which allows any sized prefixes to exist. With this change Internet could keep up with the increasing popularity among especially medium sized networks for quite long time. Still, researchers envisioned that IP addresses will eventually run out and only half a decade from CIDRs deployment next IP version (version 6) [4] was promoted to gradually replace its predecessor. Among many improvements, it promised to have vastly greater address space with different scopes to help in scaling the routing. Part of the CIDR and IPv6 is the strategy of aggregating multiple longer prefixes into shorter ones within topological limits. The idea is to hide overly specific information about distant networks to help scale the global routing system. Only the Internet Service Provider (ISP) knows its clients specific prefixes, which effectively lowers the amount of prefixes propagated over the Internet.

As many of us know, the deployment of IPv6 has not proceeded as anticipated and even today only a small portion of Internet traffic is carried on it. One of the main reasons is the lack of deployment incentives for those who already have routable IPv4 addresses. Also, IPv6 requires changes to every node and end-host in the network which has furthermore diminished its attractiveness. Instead of turning into IPv6, people have invented ways of using IPv4 addresses prudently. Network Address Translation (NAT) [5] devices allow the use of private addresses to connect to Internet and reduces the amount of required global addresses. They map between local (private) and global IP addresses with the help of port numbers allowing multiple end hosts to use one global IP-address.

2.2 Border Gateway Protocol

Internet consists of over thirty thousand independently operated interconnected Autonomous Systems (AS) [6] which exchange routing information with their adjacent ASs by using Border Gateway Protocol [7]. BGP is a policy constrained path vector protocol operating both within an AS and between adjacent ASs. Accordingly, the protocols are called internal BGP (iBGP) and external BGP (eBGP). In principal, iBGP forms one coherent view of all known prefixes between the internal BGP routers while eBGP optimizes the routing on inter-domain scale, letting business and political reasons to ultimately dictate the course of action. On operational level, BGP routers advertize policy matching best routes to external neighbors and re-advertize eBGP and customer routes to all internal BGP routers. Means for doing this are the update and withdrawal messages that can be sent between adjacent BGP routers. The actual implementation on how routes are preferred over each other is quite complex and varies over router vendors and ISPs. Nevertheless, the longest matching rule is the most important factor, having the property of guaranteeing loop free route formation. [7]

The basic iBGP approach is to run it in a full mesh in order to avoid loops that would occur if iBGP learned routes would be re-advertized via iBGP. Alternative solutions like Route Reflectors (RR) [8] or confederations [9] can enhance the scalability. Another way of managing the vast amount of control plane signaling (i.e. BGP messages) and routing information load inside an AS is to use core-edge topologies and allow routing information to spread only where needed. Instead of advertizing all the known routes, default routes can be used. This is a possible approach e.g. on internal customer or external peer facing Provider Edge (PE) routers, where most of the routes and traffic go via the core. Peer is another AS for which there is a mutual agreement to share connectivity to own and customer networks, but not to locations learned from third party ASs.

2.3 IP Router Functionality

To be able to handle the vast amount of packet formatted data as quickly as possible and to be able to offer routing and other services, routers perform a clear separation of data and control plane functionalities. Routing protocols etc. operate on control plane and there are typically many Routing Information Bases (RIB) for different protocols and purposes. Figure 1 outlines a typical IP router architecture.

BGP as an extreme case has one RIB for usable routes (Loc-RIB) and two for each BGP neighbor connection; Adj-RIB-In for received routes and Adj-RIB-Out for advertized routes. Before all the relevant routing information is delivered to the data-plane memories, a set of all the known routes is gathered into the main RIB. The selected best routes from this are then inserted into the Forwarding Information Base (FIB) that is also copied to fast line-card memories (dFIB). On the same fast memories resides also other information (such as packet filtering rules and different counters) which has to be accessed at near line speed [10]. Both RIB and FIB memories are always limited, but because operations on RIBs do not have to be as quick as on FIBs and because slower memory is less expensive, memory bottleneck on routers is usually on the FIB.

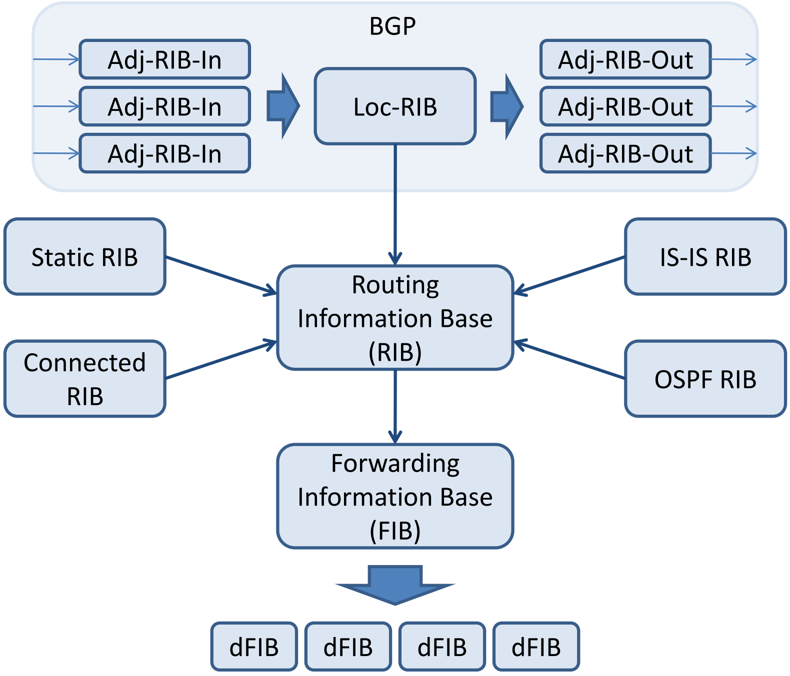


Figure 1: IP router architecture.

3. Inter-domain Routing and Addressing Related Challenges

Current inter-domain routing architecture has many issues [11], [12], [13] and [14] that can be seen on economic, political and technical aspects. For example, ISPs can for some extent affect each others' operations and there are serious concerns about the scalability of the global routing system. This Chapter introduces and discusses these and other main issues in global routing.

3.1 Operational Environment

Internet can be seen as an ecosystem where its maintenance and development is a shared responsibility of the internet community, consisting of all the parties who make their living around it. ISPs are at the heart of it making the direct profit out of it. They are typically in a rivaling relationship with each other and try to benefit at each other's expense. Businesswise it is therefore not very feasible to invest in the Internet with its holistic operation in mind. ISPs rather try to boost their capabilities in comparison to other players in the field and most commonly are not willing to take the risk of investing in a solution that would be beneficial only after the majority of ISPs have adopted it [14]. A good example of this is the IPv6 deployment challenge.

Furthermore, when one ISP decides to change or enlarge its network, it is not the only one who has to pay for it. Other ASs can be affected with increased routing table sizes, BGP signaling load and data traffic load. Mutual agreements between ASs should direct the induced costs to whoever benefits from the growth of data traffic. However, for other costs there are no mutual agreements since inter-domain control plane is very global in nature and ISPs have not been willing or able to take these costs into account.

The problem in the operational environment is a very serious one and if not properly addressed it may lead to the tragedy of commons [15], where a common good (Internet) is rendered useless in the face of overuse by each individual (ISP) trying to maximize its gain without considering the whole. Fortunately, ongoing market consolidation may mitigate the problem because a fewer number of larger players will more probably agree on the problem and have willingness and power to address it than the current incoherent group of different sized ISPs.

3.2 Running Out of Free IPv4 Addresses

IP address assignments are coordinated by Internet Assigned Numbers Authority (IANA) together with Local, National and Regional Internet Registries (LIR, NIR and RIR). IANA hands over /8 prefixes to RIRs when they come close to depleting their existing prefix space and RIRs further coordinate the prefix allocations towards end organizations such as ISPs and in minor volume other parties. Some institutions are also holding excessively large IPv4 address blocks because they were not seen as a scarce resource in the early days of the Internet.

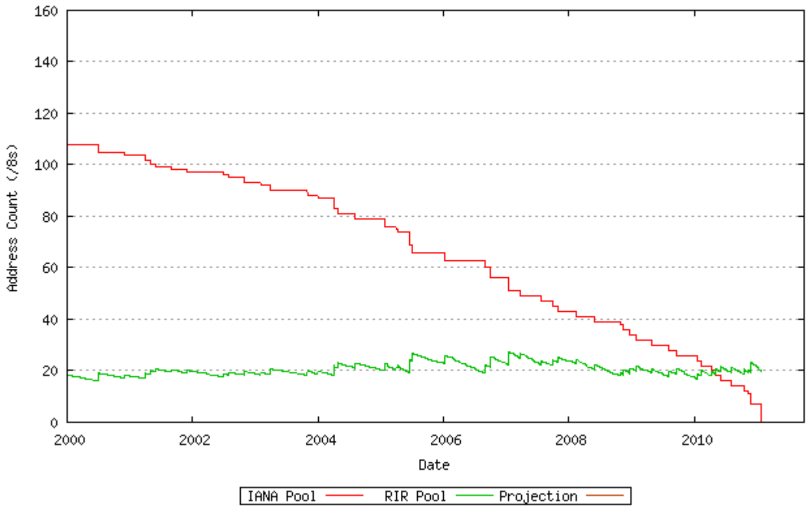


Figure 2: IANA and RIR address pool size as time series. [16]

32-bit long IPv4 address space, containing roughly 4.3 billion addresses would as such suffice for a long time, but because of the unequal address allocation and the technical properties of the global routing system, the depletion of free IPv4 addresses will happen in the near future. As Figure 2 shows, IANA has already run out of free address blocks in February 2011 and other organizations will gradually follow [16]. Thus, some may be denied new addresses already in 2011 while somewhere else in the world free addresses may be available for some years.

The problem has been known for some time and it has been fought mainly with two different approaches: limiting IP address usage with Network Address Translation (NAT) [5] and by introducing IPv6 [4] as an alternative addressing scheme. As explained in Section 2.1, NAT devices map between local and global IP addresses allowing multiple end hosts to use one global IP-address. This naturally lowers the need for globally unique IP addresses, but on the downside NATs violate the end-to-end connectivity principal which was originally one of the main ideas in the Internet. More importantly NATs induce problems on many services and to the protocol level operation behind them, although the severity depends on the NAT type.

IPv6 [4] has a 128-bit address space, effectively increasing the amount of unique addresses in comparison to IPv4. It also improves many other features, but this all comes at the expense of required router and end system support. While the support has widely been there for years, the usage of IPv6 has just recently shown some increase [17]. Although the amount of traffic or advertized prefixes on IPv6 is still just a fraction of that on IPv4, it is foreseeable that IPv6 becomes more popular when there are no longer any free IPv4 address blocks available. Still, there is the challenge of dual-stacking, i.e. operating these two protocols simultaneously with the purpose of allowing IPv4 and IPv6 hosts to communicate with each other [12].

3.3 Growth of the Global Routing Table

3.3.1 Overview

Before going in to the details of global routing table growth and its semantics, the term itself should be specified. Global routing table (also called as Default Free Routing Table - DFRT) is a theoretical table on a router where all the advertized BGP routes are visible, i.e. default or some other local aggregate routes are not used to reach each and every network on the Internet. The set of ASs which are connected in this manner are said to be on a Default Free Zone (DFZ), although strictly speaking no single AS holds all the routes due to route filtering and global routing dynamics. ASs are commonly classified in a loose tier hierarchy, where only tier-1 ASs (only a few) do not have to pay anyone for knowing routes to every network in the Internet. Instead they make peering arrangements with each other to get the connectivity information. Tier-2 and tier-3 ASs pay for some or all routes accordingly and seek peering agreements when possible and advantageous to both parties. Recently, large content providers and Content Distribution Networks (CDN) have made direct connections to many customer networks (tier-1 – tier-3) getting to the same level with tier-1 providers in terms of inter-network connections [18]. Traditionally DFZ consisted of mainly tier-1 ASs, but with the growing interest to multihoming [19] DFZ may have to reach even tier-3 networks today. Multihoming, i.e. getting connectivity to the Internet through multiple ISPs not only removes the single point of failure between the end site and the ISP, but allows also one to use load-sharing as well as policy and performance based routing.

The issue of routing tables growing faster than the fairly priced router hardware [11], [13] has been on discussions for few decades now. Figure 3 depicts how the number of active BGP entries in IPv4 DFZ FIB has increased from 1994 to June 2010, showing rather stable super-linear increase pattern since 2002. Currently there are already over 350000 prefixes and if the DFZ continues to grow similarly in the future, its size will reach the milestone of 0.5 million prefixes in around 2015. The number of entries in IPv4 DFZ RIB is already nearly eleven millions. [17]

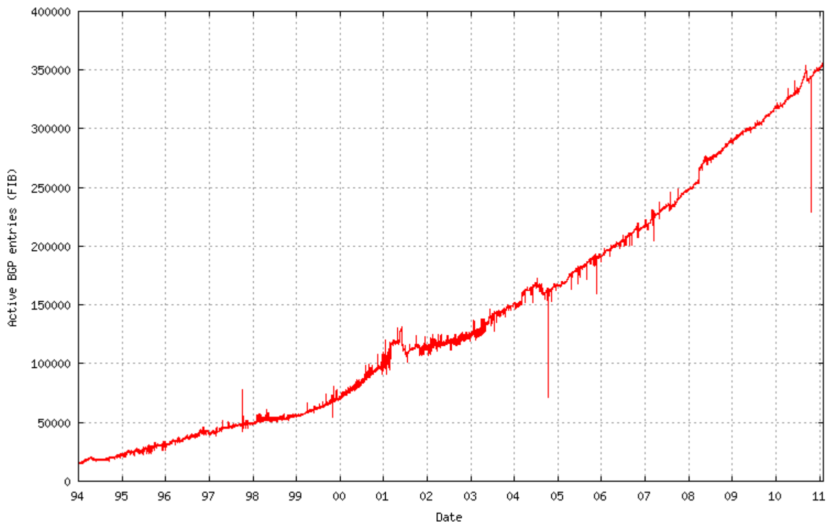


Figure 3: Number of prefixes in IPv4 DFZ FIB from 1994 to February 2011. [20]

Many factors have contributed to the DFZ FIB growth behavior and continue to do so while some new trends may further accelerate its growth rate [12], [17]. The reasons behind global routing table growth are introduced and discussed in the following subsection to give a better understanding of the specific factors contributing to it and to elaborate the overall situation. The actual relevance of this issue is explained in Sections 3.4 and 3.5 because it is one of the main contributors to the problems of routers running out of fast memory and of routing instability.

3.3.2 Reasons for increasing number of routing table entries

Ideally for routing, ISPs would form a topological tree structure where each would hold only a few prefixes to minimize both the control plane load and the FIB size on BGP routers. In reality, topology is dictated by history and business reason and ISPs are typically forced to hold several prefixes. Nevertheless, effort has been put to keep the amount of advertized prefixes relatively small by assigning large enough consecutive address blocks and by using aggregation where appropriate [3].

For significantly better topological aggregability, the relation between organizations and IP addresses should be loosened. As IP addresses serve and have always served as both the endpoint identifier in transport layer and the locator in routing [21], they are assigned organizationally following the former semantic. However, to get IP addresses to follow the real topology, they should be assigned according to the locator property. This clearly brings about an incongruity and therefore a split of semantics would be beneficial in accomplishing an aggressively aggregating global routing system. The split would also be beneficial for other purposes, such as end host mobility. [11]

As free address space becomes smaller, the remaining free prefixes come smaller and more scattered in the address space, forcing their users usually to advertize each new prefix separately instead of growing existing ones. When networks are split or merged, it may also increase the amount of prefixes in routing tables because address renumbering is costly and there may not be enough consecutive address space left to fit all the new hosts. Furthermore, it is feared that ISPs will start to trade with rather small prefixes when RIRs run out of addresses [22]. The scarce IPv4 address space is therefore problematic as such, increasing the amount of prefixes in routing tables for multiple reasons. [12]

As the adoption of IPv6 goes further, the new addresses from IPv6 will also increase the routing tables as the two IP versions are most certainly dual-stacked on the routers, inducing as big as 50 percent increase to FIB sizes [12]. If IPv6 would be the only addressing scheme, then there would naturally be no address space scarcity inherent problems. Unfortunately, this is not a probable scenario in the near future, because IPv4-only hosts are predicted to exist for several centuries ahead and many years could pass even for an IPv6-only core to materialize. In fact, a likely scenario is to have IPv4 in the core and IPv6 at the edge with support to both IP version end hosts.

One of the biggest reasons for DFZ growth is the growing use of multihoming [23]. To get its benefits, additional routes must be installed into the routing tables (DFZ) [12]. More specifically two kinds of addresses can be used to enable multihoming: Provider

Independent (PI) and Provider Aggregated (PA) [19]. PI addresses are harder to get and they are assigned through IANA and different registries. They form typically rather small prefixes and are spread through all the used ISPs. Sometimes PI addresses are acquired even though not immediately needed for multihoming, because it gives the site address space freedom from any single ISP. This on the other hand allows organizations to more easily put ISPs out to tender and reduce their ICT (Information and Communication Technologies) expenses. PA addresses are acquired from one of the ISPs address blocks and although belonging to the shorter prefix in that ISP, the PA block must be spread as such to the DFZ through every used ISP. If this would not be the case, the site would not be truly multihomed because the longest matching rule would always favor the more specific route. [12]

ISPs wish also to have some control over how data is forwarded in their network to balance the load percentage on each link, to be able to use cheaper routes and to adjust the traffic to match peering relations as well as to apply policies. This is called Traffic Engineering (TE) and it must be applicable for both incoming and outgoing traffic. For outgoing TE, IGP means are sufficient because the egress point is determined by the preferability of intra-domain routes. On the contrary, the only means to get control over incoming traffic is to advertize prefixes in smaller units via BGP and by specifying how each prefix is advertized. Consequences for the routing table size can vary greatly depending on the used granularity, i.e. the number of routes needing differentiated treatment. [12]

3.4 Limitations on Routers' Memory Capacities

3.4.1 Fast Memory Consumption

As discussed in Section 3.3, DFZ FIB will probably continue to grow about 10 percent per year if not even faster, causing routers eventually to run out of FIB space. Among others, people on Internet Architecture Board (IAB) identified this to be an acute problem and one of the main concerns regarding today's Internet routing system in their workshop in 2007[11]. In many cases the DFZ growth, however, is not the only if even the main concern. In addition to external (DFZ) routes, an ISP must also store internal and VPN customer routes. In a large ISP, the number of internal customer site routes can be close to that of the global routes [12], but in a small ISP, their share may be negligible. Depending again on the ISP, the VPN share of the total fast memory consumption on PE routers can be even an order of magnitude larger than the external routes' share [24]. That being said an explanation of what constitutes to the huge number of VPN routes is in order.

Each VPN customer must have its own addressing and routing, separated from the other VPNs and from the Internet traffic. Typical ISP VPN services include layer 2 (L2) and layer 3 (L3) MPLS VPNs, where the former provides routing information across the different sites of a VPN and the latter logical wires between the sites. In both cases MPLS is used to logically connect PE routers leaving the core (i.e. P routers) to only perform simple label based forwarding. L2 solution maps circuit IDs (e.g. VLAN IDs) to MPLS labels on PE routers and stores it on their Virtual Forwarding Tables (VFTs).

L3 solution on the other hand stores the routing information in separate VPN Routing and Forwarding (VRF) tables for each VPN customer. Considering that an ISP can have thousands, or even tens of thousands of VPN customers, this alone explains part of the memory needs. But what really makes the memory need so high, is the cumulative amount of routes each L3 VPN customer advertizes to all of its sites. As an example, having a thousand VPN customers each advertizing thousand prefixes will result to a total of one million VPN prefixes in VRFs. And as Kim, C. et al. illustrates [10], a single L3 MPLS VPN customer can quite easily account for more than ten thousand routes while many of the large and medium sized businesses have MPLS VPNs in place or are planning to have them in the near future.

As explained in Section 2.1, the number of prefixes in DFZ is heavily scaled down by aggregating customer site routes on ISP level. Moreover, many ISPs use default routes where possible in order to effectively shrink the number of external or internal routes on some of their routers. However, the same approach is not applicable for the VPN routes, since every customer must have its own routes which are originated on different parts of the topology. Further, because only PE routers are aware of these routes, there are no clear aggregation points in the network. Then, what can be done is to have the VPN customer serving PE to be different from the one that has the Internet routes, so that DFZ and VPN routes do not reside on the same PE router. Doing so does not still necessarily separate the Internet connectivity from the VPN customers as default routes can be used to forward Internet packets where they belong to.

3.4.2 Memory Capacity Estimations

Commodity memories like DRAMs (Dynamic Random Access Memory) used in PCs (Personal Computer) and other devices have rather large capacities, are relatively cheap and their advancement is expected to follow Moore’s Law. The law indirectly predicts that memory capacity will be doubled in every two years with unchanged costs [25]. However, the same cannot be said of extremely fast memories, like TCAMs (Ternary Content-Addressable Memory) or SRAMs (Static Random Access Memory) which are used in some switches and routers. They come on little volumes which makes them expensive while their development is typically slower. RLDRAMs (Reduced Latency DRAM) are something newer and in between of SRAMs and DRAMs considering latency times and cost. In terms of (static cost) development speed, they are close to SRAMs.

Building a router with a large FIB is not a problem as such, but building a router that is not overly expensive and has a fast and rather large FIB may be. According to T. Li [26], the development of large and fast enough FIBs with SRAM is lacking behind the growth curve of DFZ, making the router hardware costs to constantly increase. However, he only considered the SRAMs and did not take into account the ISPs internal and VPN routes. Later, K. Fall et al. [27] came to the conclusion that by using RLDRAMs in FIBs it should be possible to build cheap enough routers that can hold at least 5 billion prefixes even today. Juniper for example, which is one of the largest IP router vendors, supported 2 billion IPv4 routes on RLDRAM back in 2007 [13] and claims that it is not a problem to build much larger FIBs too if demand rises.

3.4.3 Is There a Real Problem with Scaling Router Memories?

The role of the DFZ table in outgrowing the FIBs is often exaggerated, because ISPs internal and VPN routes are not taken into account. Nonetheless, studies show that it should be possible to build large enough FIBs with reasonable price, if the need truly exists and the business case makes sense for vendors. Therefore, there should be no problems with newer routers for years to come if the ISPs see it beneficial for investing a little more on routers that have larger FIBs. Older routers, however, are a completely different issue, because back then ISPs and vendors where not so conscious about the upcoming problems in FIB sizes and the technology was not ready to support large and fast memories. It may also be that due to some surge in routing information, some newer routers will face the same problem as the older ones sooner than was expected.

In recent history and in near future, many otherwise perfectly usable routers are either rendered useless or their locations in the ISPs network are changed because of too small FIBs. Locations where large VPN tables must be supported and where most of the DFZ routes have to be stored are the most demanding ones. Therefore the problem with too small FIBs has not vanished completely and there is definitely room for remedy solutions. Taking into account that larger memories usually also consume more energy and require more efficient cooling arrangements than smaller ones, i.e. increase the usage expenditures, a solution which lowers the memory needs instead of growing the memory sizes would be preferred.

3.5 Routing Load and Instability

As briefly explained in Chapter 2, BGP is a path vector protocol that operates via update and withdrawal messages sent between adjacent routers. When something (e.g. a fault or a deliberate action) incurs a best route change or unreachability for some destination in a BGP router, it informs its adjacent BGP routers about the change via the aforementioned messages accordingly. This something can be a rather permanent or an oscillating event, but nevertheless, the change information is propagated everywhere in the Internet where it changes some of the best routes. The propagation does not happen instantly as the information may have to travel thousands of kilometers and pass several routers on the way. Every router has to e.g. update its RIBs, calculate new best routes and send messages onwards to other routers. Thus, it is a typical situation that a router gets the changed information from some of its neighbors significantly sooner than from others and makes the decision of the new best paths based on incomplete routing information. This possibly incorrect information is then advertised further and the Internet for the affected parts may go into an unstable state where best paths are continuously changed. When routers finally have the necessary information and all the correct new best paths have been installed, the network is said to have converged. In case of oscillating routes, it is possible that the route has gained back its original state before some distant ASs have converged and hence the process is further delayed.

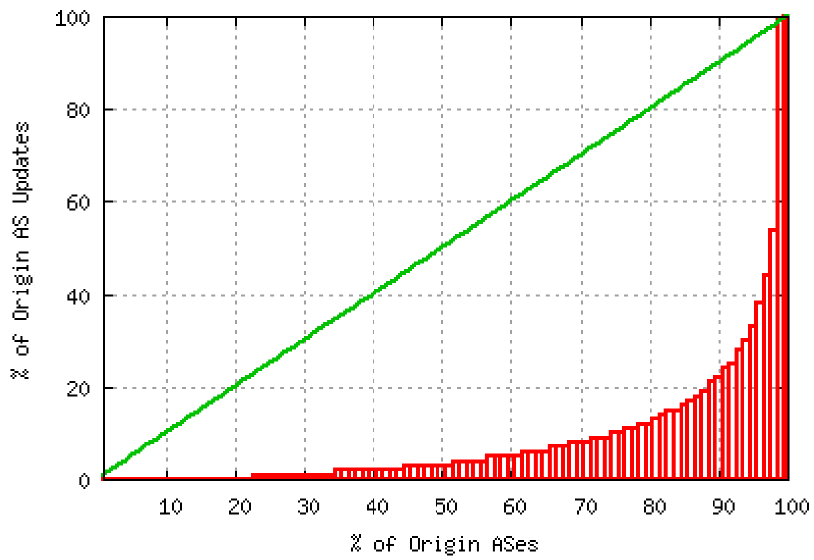


Figure 4: BGP origin AS update cumulative distribution. [28]

A great number of academic studies (e.g. [29], [30] and [31]) have been made around the BGP convergence and instability topic and a few mechanisms have actually been implemented. A default mechanism for reducing the total number of advertisements is the Minimum Route Advertisement Interval (MRAI) timer. It sets the router to wait for some time before advertizing the changed routes so that advertisements are more often based on complete and up-to-date routing information. Typical values for MRAI are 5 and 30 seconds for iBGP and eBGP while more optimal would probably be a fifth of those values [32]. Sadly, it is also common that MRAI is disabled. Another mechanism called route flap damping [33] was widely used during the 00's to prevent oscillating routes from causing frequent update churns. Unfortunately, it turned out to only magnify the problem [31] and hence its use is not recommended anymore [34].

Increasing routing (i.e. control plane) load and convergence time has also been a concern in IETF regarding the current architecture [12]. Routing load is clearly affected by the huge and increasing number of advertized routes as well as the convergence process and the increasing interconnectedness in the Internet. Naturally, each separately advertized route increases among others the complexity of the best path calculations and the chance of some route change happening at any given time. A rather intuitive approach in reasoning taken by Narten [12] would also indicate that the more direct connections ASs have with each others, the more BGP messages are sent and the more processing is needed in each router. Although this is certainly true regarding the events that do not involve a long convergence process, in overall, it is much more dubious because the number of BGP messages sent in the whole Internet greatly increases when an event induces a long converge process.

In fact, Geoff Huston showed in his presentation [17] that interconnectedness actually lowers the overall instability in the Internet. Even though the number of ASs and prefixes in DFZ has increased rather fast (see Figure 3), the number of convergence sequences and the average convergence time has seen only a mild increase. Huston reasons that the major factor in stability is the average AS path length which has stayed rather constant (around 3.5) for years while the amount of prefixes and the interconnectedness has risen. A study carried through by Labovits et al. [35] backs this conclusion by showing that AS path length is the ultimate factor in determining the convergence speed of the BGP. Instability is also heavily concentrated on certain ASs (and therefore prefixes) as can be seen in Figure 4. This further indicates that simply increasing the amount of prefixes does not explicitly lead to significantly increased routing load. It is much about the actual amount of unstable prefixes which has been increasing on a much slower rate than the DFZ FIB. [17]

So, how do different trends contribute to the overall control plane load and instability? Those that simply lower the aggregation rate increase the control plane load and instability for two reasons. They increase the amount of advertized prefixes and make it more probable that information about local changes is spread widely in the Internet. Example of such a trend is the shrinkage of free IPv4 address space. In addition, if more than one prefix covers the same network (as is the case e.g. with TE and dual stacking), a change concerning a single link can cause multiple routes to be changed. The effects of multihoming are even little controversial because on one hand it increases the amount of advertized prefixes and therefore also the routing load, but on the other, it lowers the load by increasing the interconnectedness. Some ISPs also try selfishly to optimize their network performance and usage by changing route advertisements according to traffic patterns causing other ISPs to suffer from increased control plane load.

3.6 Security

Security in Internet has gained much publicity during the recent years and not least because criminals are using Internet in their businesses ever more. Most of this is about hoaxes etc. where Internet is used simply as a tool and medium, but sometimes criminal activity is directly related to Internet infrastructure. Criminals are for example changing routing decisions, braking into systems, overloading networks and servers as well as extorting companies that they will attack its networks or servers if it does not pay large sums of money. However, all security problems cannot be dealt with technical solutions, but a network which does not allow a host to hide its true identity is both possible and a big help in the fight against criminals. End-to-end security is also an important tool as it is a necessary property in many applications over the Internet to implement a reliable communiqué between remote parties.

Traditionally Internet has been very unsecure and many protocols do not secure the information sent between different nodes. Only later on patches and additional features have been introduced to offer security for example on IP packets and BGP connections. With IP, a whole new protocol called IPSec (IP Security Architecture) [36] was introduced to give end-to-end security. Unfortunately, its usage is not always as simple as would be desirable because it requires that each end-point's IP address remains unchanged or that certain special purpose modifications are to be used. Multihoming, private addressing with NATs and IP mobility solutions tend to make IPSec operation more difficult as they change the IP addresses between the end-hosts.

With BGP, if an eBGP connection is not secured, a third party may be able to corrupt or terminate an eBGP session and thus cause harm on inter-domain routing and traffic forwarding. In fact, securing the actual connections is not enough, because legitimate adjacent BGP routers may also behave wrongly either accidentally or intentionally. A very serious misbehavior is the case when an AS falsely announces its neighbors to have better routes to certain prefixes, causing routing to change for possibly large parts of the Internet. This issue is called BGP hijack, because an AS hijacks traffic destined to the affected prefixes. History knows several incidents of it starting from 1997 while the latest was in 2010 with a Chinese ISP [37] which advertised to originate 37000 prefixes that it actually did not originate. Tools already exist to prevent the BGP hijacking, but apparently all ASs do not bother or know how to configure their routers properly.

3.7 Mobility

Originally IP was designed to interconnect fixed networks implemented with different technologies. At that time there were no mobile computers or real reasons for fast mobility and the case when a node changes its network was thought to be handled with local addressing changes. TCP (Transmission Control Protocol) was also built with these assumptions and it was decided that both IP addresses and port numbers would have to remain fixed during a TCP session. Only later on when mobile phones and laptops became common, the need rise for fast mobility: a capability to change network without breaking transport (e.g. TCP) connections. However, as IP addresses are bound to networks and TCP is bound to IP addresses, no simple solution can be adopted to implement mobility. The proposed identifier locator split would widely solve the problem.

Today, mobility is implemented in Internet with Mobile IPv4 (MIPv4) [38], Mobile IPv6 (MIPv6) [39] and Network Mobility (NEMO) [40]. They are all far from optimal solutions as they require special functionality nodes to intermediate the mobility. In principle, a home agent keeps track where the node is moving and tunnels packets sent to it to the mobile node. For doing this, it requires a Home Address (HA) and must map this to Care of Address (CoA) which is the address the mobile gets from the network it is visiting. Disadvantages of such solutions are diversified and include the increased transmission latency, tunneling costs, triangle routing and problems with firewalls and NATs.

3.8 Problem Summarization

A crucial infrastructure for a big part of the world, which Internet has become, faces pressure from many directions. The number of users, the amount of carried data and the variety of required functionalities are constantly rising while the foundations of the system are largely the same they were two decades ago. Because Internet has become ever more important and vast, and because of its distributed nature, changing any basic component in it asks for great and unified endeavors from many central parties. Even though everybody would understand what should be done for the common good, business reasons dictate many actions and if there are not large enough incentives for individual companies to invest in changes, they will never happen.

The most serious concerns today are on the scalability of the routing system which manifests in a few ways: as too large routing tables, as disruptive routing load and as huge energy consumption figures. Memories (especially fast memories) on routers are having tough time on keeping up with the increasing capacity requirements while ever more processing power is needed to keep up with the routing changes and more over to be able to forward traffic to correct outbound links. The direct impact of these is the increased energy requirements which the world has become very concerned in general. Baliga et al. [41] estimates that one to four percentages of the world's energy goes into keeping Internet up and running and argues routers being the largest energy users. An even alarming although questionable estimation carried out by Namiki et al. [42] predicts that energy consumption with current technologies skyrockets with the estimated traffic increase in a few years' time.

The pressure on Internet routing and addressing to better support secondary functionalities like multihoming, traffic engineering, mobility, security and multi-paths can also be seen as an important issue. Many of the problems in this area stem from the dualistic nature of IP addresses: indicating both the location and the identity. Solutions exist to implement the mentioned functionalities but they are not of novel design, simplistic nor very scalable. E.g. multi-homing and traffic engineering are brought in on the expense of increased routing state and mobility as an add-on tunneling solution with poor performance. Time will also tell will the depletion of free IPv4 address space cause major problems or will the adoption of IPv6 go forward as planned and relieve any worries on insufficient number of IP addresses.

All in all, challenges in Internet routing and addressing are many while the tools to drive any major changes are scarce. A best way to solve at least nearly all the mentioned challenges would be to revise the Internet architecture completely. As this is very unlikely due to high change costs and lack of co-operation in the industry, we can only hope to tweak the existing system towards something which is good enough for the time being. Thus, in my personal opinion, the question of how to do the changes is much more important than the seeking of an optimal solution for the problems.

4. Proposed New Solutions for Improving the Inter-domain Routing Scalability

This Chapter introduces three main proposals for alleviating challenges discussed in Chapter 3. None of them address all the aspects, but offer solutions at least for better routing scalability. The purpose is not to extensively cover all the proposals, but rather to introduce samples of different approaches while also considering their implementability. Locator/ID separation protocol (LISP) is the first to be introduced, approaching the problem set with a Map & Encapsulate type of a solution. LISP has gained most interest among the Internet community, although mainly so because it is driven by the world’s largest IP router vendor. The proposal introduction is then continued with IRTFs recommendations for the short and long term solutions [43], i.e. Evolution and Identifier-Locator Network Protocol (ILNP). Evolution is an evolutionary approach of implementing the locator – identifier split with a focus on the most imminent problems at each point in time. ILNP in turn outlines a clean solution for separating locators from identifiers. In the last section of this chapter the positive and negative aspects of each of the three introduced proposals are consider and compared against each others.

4.1 Locator/Id Separation Protocol

Locator/Id Separation Protocol (LISP) is designed to turn the Internet into a significantly better network in terms of scalability. It does this by forming a tunneled virtual network over the site border nodes, which connect ISPs to their client networks. Inside and outside of this tunneled core network IP addresses have different meanings: inside they indicate locations while outside they more or less only mean the host identities. The key idea is to separate the global core network from all other networks so that the number of locations, i.e. routing entries in the core, will scale. A vital policy for simpler implementability is to change as few things as possible. Therefore, in LISP, IP and many other protocols are used without modifications and only relatively few routers will need to change how they operate. End hosts will also stay as they are and IPs are allocated to them as the network owner pleases from the address pool it possesses, although, end site prefix aggregation should still be used. [44]

LISP does not change anything when packets are sent within an individual network. When a host sends an IP packet to a host in another network, in LISP terminology it will have source and destination endpoint IDs (EID) specified with source and destination IP addresses. Destination EID is used for routing inside the local network until the packet arrives to the edge of the tunneled core network. Such a router is called an Ingress Tunnel Router (ITR) because it prepends a LISP header on top of the existing IP packet and sends it through the tunneled core network. Similarly Egress Tunnel Router (ETR) removes the LISP header on the other side of the core network and forwards it on the correct end-site network which will again use the EID for routing. What happens in between the ITR and ETR is how LISP contributes. [44]

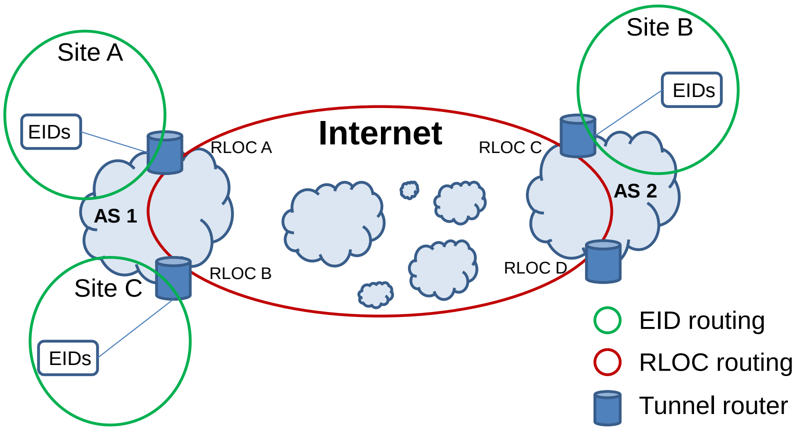


Figure 5: Operation of LISP. Routing is split to core and site areas where different locators are used.

The routing between tunnel routers (e.g. ETRs and ITRs) is based on Routing Locators (RLOC) which are given in the LISP header's IP address fields. RLOCs are assigned to ETRs where they are mapped to EID-prefixes. All tunnel routers have RLOC to EID-prefix mapping tables. Figure 5 visualizes where different address types are used.

ITR now knows the source and destination EIDs by looking at the IP packet headers and uses its own source RLOC when sending packets to an ETR. The destination RLOC is yet to be figured out and to do that ITR sends map-requests via a mapping system and waits for a map-reply from one of the destination site ETRs. ETRs reply on map-requests when the destination EID in the request matches with a prefix in its EID-RLOC mapping database. From the possibly many returned RLOCs ITR will choose one to be used. [44]

Mapping queries between destination EID-prefixes and subsequent RLOCs does not have to be made every time a packet is to be sent through the core network. Instead, the information can be stored on ITRs' local mapping cache which provides fast access times. LISP also supports the use of alternative logical topologies over which the mapping requests can be sent. Such could possibly improve the mapping procedure considerably by lowering the amount of map-request messages and the mapping delay. Several have already been described (e.g. [45], [46], [47]) while the testing is furthest with ALT [45].

4.2 Evolution Towards Global Routing Scalability

Evolution towards global routing scalability (also called as Aggregation in Increasing Scopes (AIS)) approaches the Internet routing scalability problem in an evolutionary fashion. It argues that if early adopters do not have large enough business incentives, no significantly different new architecture or design should be expected to be widely implemented. Moreover, ASs see the scalability problem differently as they come in many flavors. For some the problem is real and they would like to see a major change in the whole Internet as soon as possible, while for others the change would only mean additional expenditures. Authors behind the proposal say that an evolutionary path is therefore more likely to be taken; starting from ASs that are having problems today. Still, to have operators implement some small fixes that help only locally, will not lead to a dramatically better architecture which is more scalable in inter-domain context. However, if the small fixes are designed to lead towards such architecture, a more comprehensive change can be expected to happen. [48]

The proposal presents a five step process, where each step could be deployed in networks where needed, while others can simply ignore it. Each step should fix the most imminent problems at each point in time and give immediate benefits to their users as it is the only way to make it attractive for early adopters. The steps they give and which are introduced here are to give an example that an evolutionary path is possible, not to give an absolute plan which procedures should the steps be composed of. [48]

4.2.1 FIB Aggregation

First step, i.e. FIB Aggregation [49], seeks to reduce the amount of prefixes routers have to keep in their FIBs by removing ineffectual prefixes, combining longer prefixes into shorter ones and otherwise reducing the total amount of prefixes needed to get the same forwarding behavior for known prefixes. Compression can be done in case two or more prefixes use a same link to reach a next hop and they reside conveniently in the prefix tree. FIB Aggregation does not change control plane or RIB in any way, but incurs some latency and other overhead when preferred paths are changed. In FIB Aggregation, each router can be separately set to compress its FIB using different algorithms. These algorithms are divided into four levels in terms of what they will do, while the implementation details and even the scope can vary within a level. [49]

Each FIB compression level will perform everything the lower levels do and add something new. The basic operational depiction of the algorithmic levels is visible in Figure 6. Level 1 removes prefixes that are already covered by their parent with same next-hop (Figure 6a). Level 2 allows sibling prefixes, i.e. prefixes with same non-existent parent, to be combined to their parent (Figure 6b). Level 3 continues here by allowing extra routable space and hence the combination of non-sibling nodes into grandparents and even further up in the prefix tree. However, level 3 algorithm (Figure 6c) must make sure that a shorter prefix with a different next-hop does not cover the affected part of the prefix tree; otherwise routing for some prefixes is altered. This would happen e.g. for Ps siblings when a prefix with next-hop other than P would cover the whole tree on the left side of the Figure 6c. Level 4 is the highest FIB Aggregation level and in addition to everything level 3 does, it allows holes in the aggregation. Figure 6d depicts the behavior with next-hop X prefix. [49]

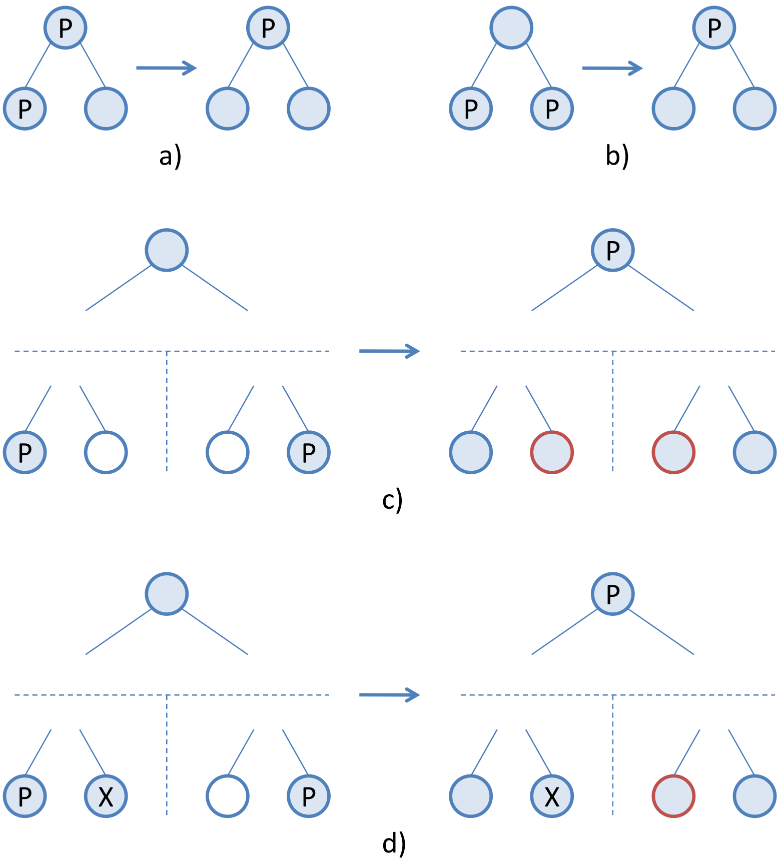


Figure 6: FIB Aggregation levels 1 – 4 (a) – d)). P and X are next-hops of the prefix in question. Circle fill colors indicate the routable space: white is unroutable and blue is routable. Red circle border means that the address space is routable only after FIB compression.

Each aggregation level is expected to give further savings in FIB size compared to the previous levels, but at the same time computation and other overheads will increase. There is therefore a clear tradeoff between the compression efficiency and overheads. Level 3 and 4 algorithms can also increase the routable space which can increase the load in routers and links. This can be seen especially adverse when extra traffic is induced on inter-domain links increasing operators' direct costs. Zhang et al. [49] have done an evaluation on the trade-offs and also considered when and how should FIBs be updated.

4.2.2 Virtual Aggregation

Virtual Aggregation (VA) [50] is properly explained in Chapter 5 and introduced here only briefly to show its role in the evolutionary road. Evolution steps two and three are about Virtual Aggregation in which the aggregation is done in a larger scope consisting of many nodes or even networks. In Virtual Aggregation, Virtual Prefixes (VP) are used to gather traffic from multiple nodes into an Aggregation Point Router (APR) which then forwards it onwards. VPs are shorter than any real prefixes and are advertized by

APRs. Those routers which are not assigned as aggregation points for a certain VP need only to insert route to VP in their FIBs, not prefixes which are covered by that VP. By assigning different routers to aggregation points for different VPs, the FIB size can be reduced on all the routers. Forwarding to and from APRs must happen through tunnels in order to avoid loops.

One of the costs in VA is the increased processing requirements which lead to increased energy consumption and latency (similar to FIB Aggregation). Another cost in using aggregation points is path stretch, which can be controlled by allocating VPs and assigning APRs optimally. In evolution towards global routing scalability step two, VA is applied inside an autonomous system, allowing operators to freely choose whether to use it or not. For further reducing the stretch or the amount of aggregation points, adjacent ASs can direct tunnels to go through each other's networks and avoid doing the aggregation twice (step three). To be able to do this, aggregation routers on each AS needs to be made aware of each other's egress routers for the selected set of prefixes. A simple piggybacking over BGP is proposed, although it will not scale in case the adjacent AS VA is widely used. [48]

4.2.3 Control Plane Scalability

Evolution step four continues fixing the most imminent problem in this five step scenario by introducing a better mapping system for Virtual Aggregation prefix to egress router mappings. With piggybacking, RIBs may start growing substantially when multiple adjacent ASs start exchanging mapping information with each other. Same information will be learned from multiple sources and hence saved multiple times in RIBs. A separate BGP session only between aggregation routers could be controlled with rules to keep only one copy of the mapping information, since the path through which the mapping information was sent is irrelevant. Furthermore, with such a setup in place, many routers would not even have to keep the full RIB, because forwarding is done according to VPs. This would ultimately lead to a separation of transit and edge routing if the majority of ASs would use Virtual Aggregation and a mapping system. This is similar to what e.g. LISP describes.

The final step in this proposal is to protect the routing system from routing churns caused by edge site instabilities [48]; something route flap damping [33] tried to achieve during the 00's. The separation of transit and edge routing should help on this, since it would be easier to keep the dynamics in edge routing out of the mapping system.

4.3 Identifier-Locator Network Protocol

Somewhat similarly as the other two proposals introduced in this chapter and as its name indicates, Identifier-Locator Network Protocol (ILNP) describes an architecture which breaks the address identity in IP to locator and identifier identities. Different from the LISP and evolution towards global routing scalability, it does not revert into using tunnels but implements the separation straight on the protocol headers (such as IPv4 & IPv6 headers). Moreover, ILNP introduces changes to where the identity is bound to. Currently IP addresses are given to interfaces, but ILNP dedicates the identity to nodes. The implications of this are many and some are considered later, but first the functionality of ILNP is explained.

ILNP can support many network layer protocols out of which IPv4 and IPv6 have been specified. The main difference between these is which bits represent which identities. Address field in IPv6 is 128-bits long and can therefore be split to two 64-bit parts while IPv4 needs extra bits to support the 64-bit identity field in ILNP. The extra bits in IPv4 are brought in with a new type of option. The bits in IPv4 which currently indicate the source and destination IP addresses are to be interpreted as locators, i.e. the subnetworks where the source and destination nodes reside. In IPv6 the first 64-bits indicate the locator and the last 64-bits the identity. For compatibility with current architecture, locators should be same as today's IP prefixes, and for the same reason, a special nonce in both IPv6 and IPv4 is used to indicate other nodes that the source is operating as an ILNP node. To allow Fully Qualified Domain Names (FQDN) to be mapped to identifiers and locators instead of addresses, ILNP introduces also new DNS resource records.

Because ILNP changes the DNS and IP protocol level operation, changes have to be made at least on each end node and DNS server. The IP protocol stack is altered slightly which in the worst case would mean that all the applications would also have to be rewritten. Fortunately the situation is not quite that bad as special APIs can be used to do the converting between ILNP network stack and applications which are built to use traditional network stacks. Anyhow, each end node that implements ILNP requires changes which severely diminish ILNP's deployability.

4.4 Proposal Analysis and Comparison

As discussed in Chapter 3, multihoming and traffic engineering have been great sources for increase of routing tables while the depletion of IPv4 addresses is predicted to radically increase the amount of IPv6 prefixes in the near future. Nevertheless multihoming and TE are desired properties in addition to routing security, network robustness and mobility. Thus, the remainder of this section concentrates on explaining how the three proposals affect these and other functionalities and what exactly are the operation and deployment wise pros and cons of each solution.

LISP efficiently reduces the amount of routing state on the Internet core, adequately solving the routing table size related challenges: tough requirements on fast memory and routing load & instability. It also allows one to use PI addresses, traffic engineering and multihoming without adding state to the global routing system. Multihoming is brought in by mapping one EID to multiple RLOCs and TE can be done with the help of TE tunnel routers. A major weakness in the proposal is in the mapping between RLOCs and EIDs. It asks for complexity in the system and induces high latency on initial packets for which there is not already a cache entry; although some argue that the issue is not that bad. For end-host mobility LISP has a special mechanism [51], but it does not natively solve the problem like some other remedy proposals do. Finally, as LISP does not change many of the currently used protocols, it does not improve the security properties either. However, some new security threats are possible, especially along with the mapping system. [43]

Because LISP is driven by a major router vendor Cisco, it has a good chance of becoming the standard in the Internet. The implementation is also only moderately painful as only site border routers need to be modified. But, although LISP authors claim otherwise [43], there are only minor incentives for early adopters as routes for the rest of the Internet have to be handled as is done today. The real gains are to be expected only when a large number of ISPs have adopted it. Additionally there are legitimate concerns whether LISP is really the best option being mainly a business opportunity for its inventor company. On the other side, testing with LISP is quite far and hence many problems have been identified and fixed or at least alleviated. The same may not be true for many other proposals.

Contrary to any other proposal, Evolution focuses on the deployability more than on anything else. It does not even seek to solve all the problems at once, but concentrates on the most urgent issues on each step. Memory size problems are handled first and with the last two steps, the routing load and instability issues are presumably overcome too. However, e.g. mobility and security are not addressed at all, making the solution less complete in comparison to LISP and ILNP.

Deployment of the Evolution starts from single routers and spreads to networks and adjacent networks. While this is an ideal option for individual ISPs and will therefore probably gain some popularity, are there enough incentives for most to ever proceed in the multi-step process? If that will be the case, routing load and instability will never be improved. And further, because ISPs can solve their immediate problems with relatively painless quick fixes, they may not see it beneficial to put effort in adopting some larger change. All in all, IRTF has still promoted Evolution as a short term solution, even though there is a real possibility that the short term solution will become also the long term solution. [43]

ILNP is among these three solutions the best in terms of technical properties. As a fully deployed architecture, it would perform identifier-locator split in a clear manner, gaining all of its advantages. Traffic engineering, multihoming and mobility are naturally there with simple and efficient solutions while routing table and state related problems are clearly diminished by the efficient topological aggregation. Even multipath TCP is easily implemented and there are clear benefits to many other applications which today suffer from the use of NATs. [43]

However, ILNP's Achilles heel is the deployability. There are only minor incentives for the early adopters while the expenses are great and are also targeted to end users. The fact that routers may not have to be changed is a relatively small relief when each and every end node needs to upgrade its IP stack and DNS servers have to be enhanced with ILNP support in order to fully exploit the advantages in the architecture. The suggestion of an API between current IP stack and applications would help a lot, but there is still a big concern whether such an API can be actually deployed on at least most of the end nodes. There are also many concerns related to using DNS in the described manner. Nevertheless, IRTF saw ILNP as a promising and clean new architecture which should be developed further [43].

It is clear that none of the three introduced proposals cover all aspects that users want from the Internet. For example, trust issues are skipped altogether, although it is the basis for reliable communication in the network. They simply assume that such concerns are handled separately e.g. with secure DNS [52] and IPSec [36]. However, some other proposals, like Host Identity Protocol (HIP) [53], do describe a trust model and are in that sense better than any of the proposals introduced in this Chapter.

Comparing the three introduced proposals, one can see a clear relation between the functionality and the deployability. The richer and more efficient the solution is, the harder it is to deploy. Evolution promises only to address some of the perceived issues while its efficiency may even be questioned, but it has immediate benefits to its users, requires no initial co-operation between different parties and to start with, can be deployed on a router granularity. ILNP on the other extreme would seem to help on a variety of current and future challenges, and do so elegantly without resorting to the patching approach. Unfortunately the deployment seems to require too much to ever succeed. Finally, LISP is something in between of the two in terms of functionality and deployability. For this reason, it is clear that none of these are perfect and only future will show how much we can and are willing to put effort in enhancing the global telecommunication system. Of course there are many other options for the future architecture and new ones are developed all the time.

5. Virtual Aggregation

5.1 Overview and Motivation

Virtual Aggregation [50] is part of the Evolution Towards Global Routing Scalability proposal [48] and with it a substantial reduction in routers FIB sizes is sought. VA does not require any major changes to the network, nor does it require co-operation between different ASs (although, some additional benefits may be achieved if adjacent ASs operate VA over AS borders). VA is also applicable with configuration changes only and minimally with very little data plane overhead. It is intended especially for older routers that can support rather limited number of prefixes in their FIBs, effectively prolonging their service time. How much could VA contribute to alleviating the FIB size related scalability challenges in ISPs networks is a more difficult question to deal with. Nonetheless, it is something this thesis tries to find an answer to. Following sections elaborate more closely on how VA works and what kind of positive and negative consequences it has on the network where it is used. Furthermore, simulations described in Chapter 7 try to find out how VA should be used to get the best out of it and how big FIB savings can be expected.

5.2 Operational Description

The fundamental idea in Virtual Aggregation is to aggregate larger than physically aggregateble external prefixes by using tunnels. The aggregates, called Virtual Prefixes (VP), can be freely configured, allowing any sized aggregates to be created. VPs do not even have to be similar across the domain, but can be different on e.g. POP or router basis. In contrast to normal route aggregation, VA does not impact control plane operation, i.e. routing protocols, or shrink RIBs in any way, but merely limits the number of routing table entries that are loaded into FIBs. This operation is called FIB suppression and its intention is to lower the capacity requirements for fast memory in BGP routers. [50]

In a network where VA is in use, routers are assigned different roles. To distinguish VA capable routers from others, the terms VA and legacy router are used correspondingly. VA routers can furthermore have different roles in respect to particular VPs. A VA router can be simultaneously an Aggregation Point Router (APR) for a certain VP and a non-APR for another. The function of the APR is to operate as a tunnel end-point for a VP so that non-APRs can use this VP in their FIBs instead of real prefixes. The real prefixes that are covered by a VP are called sub-prefixes. In addition to normal and VP prefixes, non-APRs can have Popular Prefixes (PP), i.e. sub-prefixes which are installed normally from RIBs to FIBs even though a VP covering it exists. The idea in this is to minimize the tunneling inherent extra latency and network load for traffic intensive prefixes. [50]

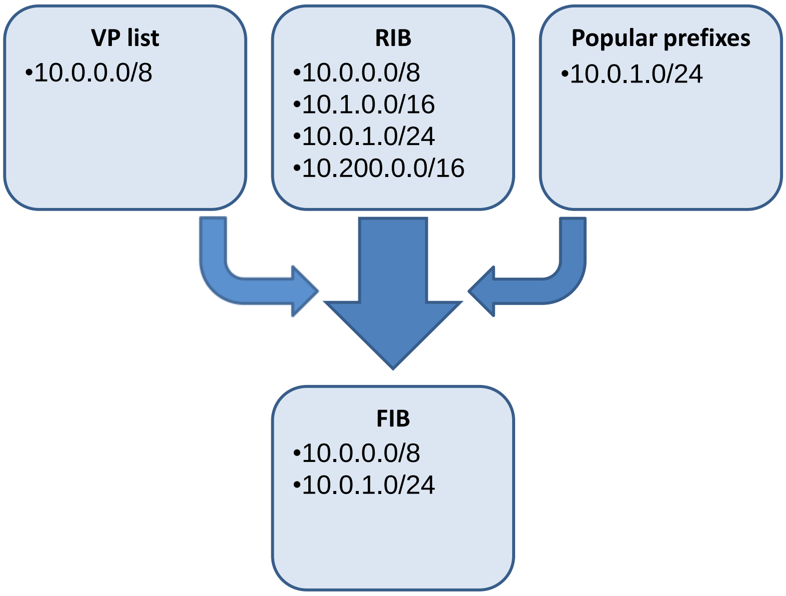


Figure 7: FIB suppression process. The suppression is done from RIB to FIB and is guided by virtual and popular prefix lists.

To visualize the process of selecting prefixes to be installed into the FIB, an example is illustrated in Figure 7. There are four partially overlapping entries in RIB: 10.0.0.0/8 learned from an APR and the rest from adjacent ASs via BGP. Without VA, all the eBGP routes would have to be inserted into the FIB. Configuring 10.0.0.0/8 into the VP-list makes it effectively an active VP and an aggregate route for all the other entries in RIB. Finally, by specifying 10.0.1.0/24 to be a popular prefix it is added into the FIB even though it is covered by the VP (10.0.0.0/8). The actual suppression must happen when routes are inserted from RIB to FIB, not between e.g. Loc-RIB and RIB because Protocol Independent Multicast – Sparse Mode (PIM-SM) requires full routing information on the RIB.

A basic case of VA operation in a network is presented in Figure 8, where red boxes (E-F) represent APRs for VP1 and non-APRs for VP2. Comparably, green boxes (G) are APRs for VP2 and non-APRs for VP1. Blue ones (A-D) are non-APRs for both of the VPs. Similarly, red and green arrows indicate the route to VP1 and VP2 accordingly while blue arrows indicate the non-aggregated popular prefix route. When a data packet comes to the ingress router, it forwards the packet normally according to its FIB while obeying the longest matching rule. If VP for a destination is not specified or a popular prefix overriding any VP is configured, forwarding path will be the normal shortest path (blue). However, if the packet destination belongs to a configured VP (VP1), ingress router forwards it to the closest APR associated with that VP via a tunnel (tunnel 1) which forwards it onwards to the correct external peer via another tunnel (tunnel 2). It should be noted that indeed all of the VA routers can suppress their FIBs if they are non-APRs for at least to one VP. In this example, red and green routers can suppress each other’s sub-prefixes (covered by VP1 & VP2), while blue routers do not have to store neither VP’s sub-prefixes.

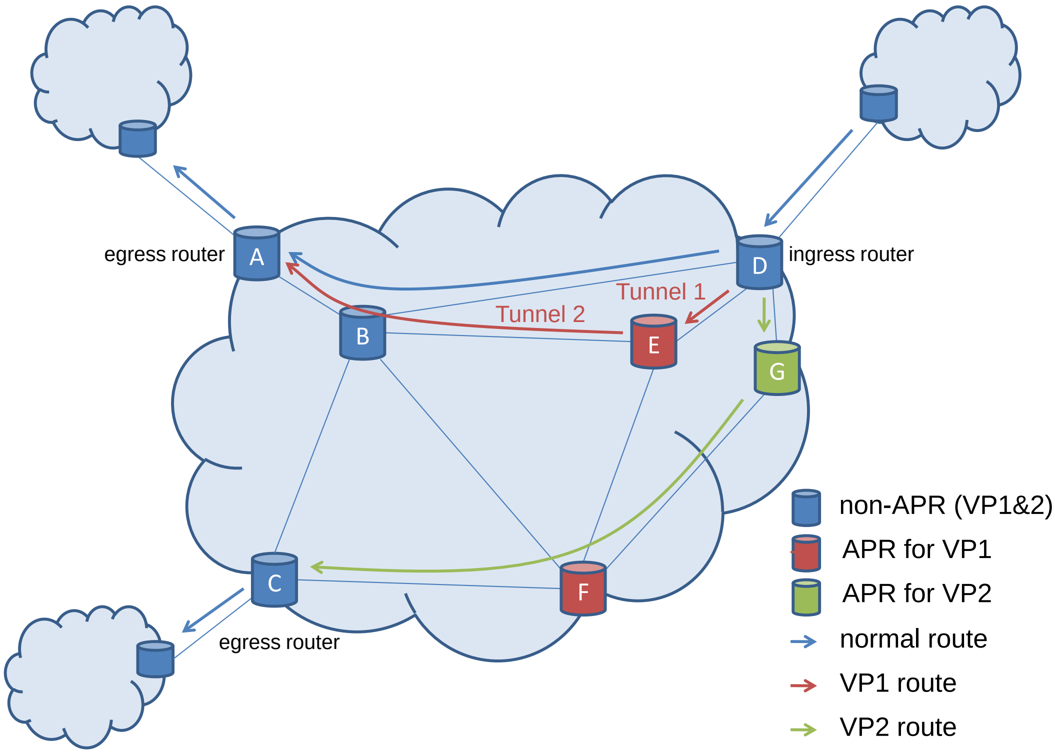


Figure 8: Basic operation of Virtual Aggregation.

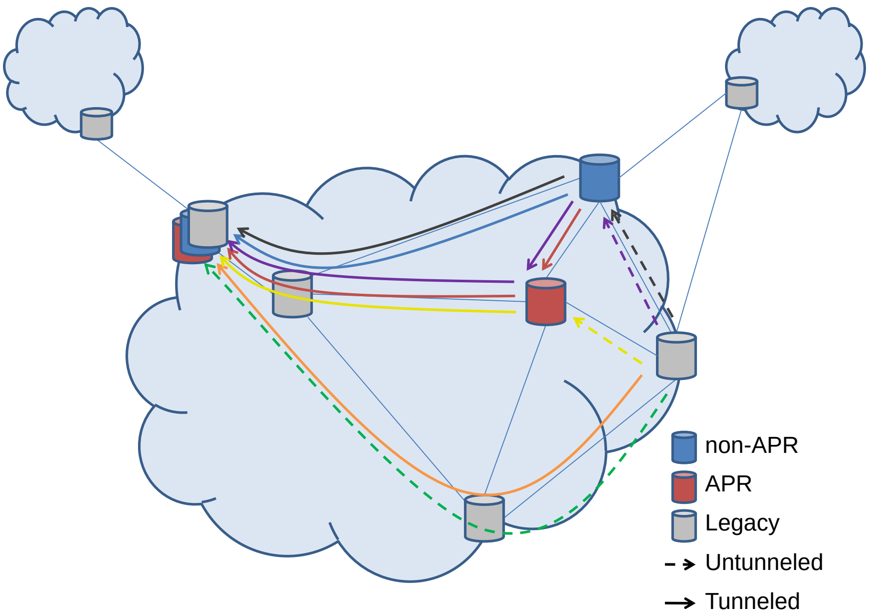


Figure 9: Possible paths (in different colors) a packet can take in a VA enabled network.

Before Virtual Aggregation can be used, a number of things have to be in place. To start with, there must be some sort of plan on how to partition the address space into VPs and a way to determine which routers are associated as APRs for which VPs. Initial set of popular prefixes should also be considered. Setting up the planned configuration involves actions on all VA routers. Non-APRs must configure a VP-list which indicates the VPs it is willing to use in suppressing the FIB. To make all the other routers aware of the VP routes, APRs originate and advertize VPs via iBGP. They must have all the sub-prefixes covered by advertized VPs in their FIBs so that they can forward the traffic ultimately to the correct external peer. In addition, plenty of tunnels have to be configured in the network in order to guarantee correct and loop free packet forwarding. [50]

VPs as well as real prefixes are distributed in an AS using BPG. How exactly VPs are advertized from node to node depends on the network as different techniques (such as route reflectors [8]) may be in use. In general, BGP routers advertize known prefixes to each peer BGP router. The advertisement message, i.e. BGP update message, includes information of the origin of the advertized prefix, the next-hop towards the origin, the prefix itself and several possible attributes influencing the routing decision. To prevent VPs from being advertized outside of the local AS, a community attribute must be specified to prevent it. A community attribute includes a number and a short text, and all BGP routers belonging to a same AS are set to behave similarly for a certain community value.

For routing correctness, every VA router must initiate a tunnel towards every BGP next-hop address. First of all, each non-APR must initiate a tunnel to all of the APRs that are associated with the VPs configured in its VP-list (tunnel 1 in Figure 8). The purpose of this is to allow non-APRs to forward packets directly to correct APRs. Secondly, each APR must have a tunnel towards each external neighbor it has a BGP next-hop route (tunnel 2 in Figure 8) so that APRs can forward packets directly out of the AS and avoid them to be forwarded back via non-APR to APR tunnels. Depending on the Autonomous System Border Router (ASBR), these tunnels can either be terminated on the local ASBR or on the remote ASBR. Legacy ASBRs must terminate the tunnels while non-APRs set the BGP next-hop to remote ASBR, so that they can forward packets without FIB lookup. All currently specified options use Multiprotocol Label Switching (MPLS) to create Label Switched Paths (LSP) (tunnels). In the direct approach LSPs are formed just as described earlier. The use of MPLS inner labels can be beneficial in limiting the amount of tunnels. In that case the inner label identifies the remote ASBR and the outer label or IP tunnel identifies the local ASBR. Importantly, with both mechanisms the remote ASBR will not be aware of the tunnels as all the labels and tunnel headers are stripped away before sending the packets on inter-domain links. With MPLS this kind of operation is called Penultimate Hop Popping (PHP). [50]

Although legacy routers do not have to insert packets into tunnels or create them for that matter, they must be able to forward packets inside tunnels and act as tunnel endpoints. They must also be capable of communicating with other network nodes in creating tunnels that terminate on them. However, as all the above mentioned features are part of IP router's standard operation, most of the routers readily have them. [50]

A conclusion of different paths a packet can take in a VA enabled network is shown in Figure 9. If VA router is on ingress point, the packet can either be tunneled towards the external neighbor directly (blue) or via the closest VP matching APR (red). The case where the ingress point is an APR is not shown in the figure, but anyhow APRs always tunnel packets straight towards the external neighbor. If the ingress point is a legacy router which is capable of tunneling packets, packets will be tunneled directly to the correct external neighbor (orange). If the legacy router cannot initiate tunnels, it forwards packets according to IGP routes. In that case packets can go all the way to the egress router without passing any VA routers on its way (green) or find its way on some VA router (black, purple and yellow) and end up tunneled to the external neighbor. VA routers do not make a difference on where a packet came from and therefore there is no difference whether the packet came directly from inter-domain neighbor or via some legacy router(s). Also, where the tunnels actually terminate depends on whether the egress router is VA capable or a legacy one. In the former case tunnels must go all the way to the external neighbors while the tunnel headers and labels are stripped in the egress point. In the latter case, normal routing is enough to ensure that loops are not formed. [50]

5.3 Simple VA

In many cases it is not necessary to suppress FIBs on all routers and for that purpose a simpler setup of VA could be more ideal. By allowing only the existence of one VP (namely 0/0) covering all the sub-prefixes, a great deal of planning, configuration and management can be avoided. One must only tell the routers whether they are core or edge routers, i.e. whether they install full FIBs or not. This kind of default route usage is naturally beneficial only when the ISP has some kind of a core-edge topology and some older routers with limited FIBs on the edge. Still, in comparison to traditional use of default routes (as explained in Section 2.2) FIB can be saved on more routers. Unlike with traditional default routes, there are no limits in using simple VA on transit or on any other edge router, even though an external neighbor would require full DFZ routing table. [54]

Despite the fact that configuration could be made extremely easy, similar tunneling arrangements than with full VA are in order. A tunnel must exist from FIB-suppressing routers to all FIB-installing routers and from the latter ones to external neighbors (indicated by BGP next-hop). In case some legacy routers are present, they too must perform all the tunneling operations mentioned in Section 5.2. All in all, simple VA is a technique somewhere in between of the conventional network planning strategies and the full VA both in its capability to shrink FIBs and in required changes to typical configurations. [54]

5.4 Limitations and Overhead

The use of VA does not come without downsides as can be seen by looking at Figure 8 and the possible path stretch VA induces by causing paths to go through APRs which are not necessary on the shortest paths. To be formal, path stretch tells the amount of hops in comparison to the shortest path and is calculated with equation 1. Path stretch and otherwise non-optimal routes cause packets to experience increased latency and network to suffer from increased load as packets may visit more routers and take longer on their way to network border. Other overheads come from the tunneling, more complex network management and increased complexity in the network in general. [24], [55]

(1)

As explained in sub-section 3.3.2, traffic engineering requires the insertion of new longer prefixes into routing tables. The goal of Virtual Aggregation is on the contrary to aggregate longer prefixes into shorter ones. Hence, TE makes Virtual Aggregation harder to implement. Popular prefixes or more specific virtual prefixes must be configured to engineer the traffic for the needed parts slightly reducing FIB size savings and increasing network complexity.

An important aspect in planning and operating VA in any network is to control the negative effects. Analyzing TE capabilities, network management and network complexity overheads is hard as they cannot be measured quantitatively without knowing how different networks are operated. Path stretch and FIB size effects are however much easier to consider. Therefore, the focus is put on looking what affects FIB sizes and path stretches and how VA could be used to control them.

FIB space and path stretch are closely tied together and the balance between them can be tuned for example with the number of APRs per VP. By setting a big portion of the routers as APRs, one can eliminate the stretch completely, but the acquired savings on routers FIBs is then also minimal if not even negative due to tunnels. On the other extreme one could assign only one APR per VP causing all the packets to be tunneled via it and allowing most of the routers to hold only the tunnel routes to APRs. By assigning enough routers to VPs APRs so that for each VP there is at least one APR (preferably two for resiliency) close to every ingress router, one can get meaningful savings in FIB size with only a few milliseconds maximum stretch latency. A good strategy is to have an APR for every VP in each POP if possible. And if the POPs are too small for this, the same can be done for small POP groups which cover small geographical areas. [55]

Ballani et al. [55] considered this strategy in AT&Ts network by minimizing the maximum FIB size while constraining the maximum stretch latency to different values. Results indicate that maximum FIB could be as small as 5 percent of the DFRT with 4 millisecond maximum stretch latency. At the same time average stretch latency would be below 0.3 milliseconds.

As APRs are assigned per VP, the number of VPs plays also a role in controlling the path stretch. With only a single VP, the situation transforms to simple VA, which is explained in section 5.3, and only moderate FIB savings can then be expected. Too many VPs naturally increases the minimum FIB size on routers to a too high level, as every router must have routes to all of the VPs (or their sub-prefixes). How many VPs one should use and how they should be allocated depends on the network topology. Ballani et al. [55] considered also different VP allocations in their simulations. They used 128 VPs of prefix length 7 and 1024 VPs allocated in a manner so that each VP has approximately same amount of sub-prefixes. The uniform allocation with 1024 VPs achieved somewhat better results, because the maximum number of sub-prefixes in a VP is smaller.

5.5 Popular Prefixes

The effects of path stretch can be substantially lowered by allowing high traffic volume prefixes to be routed via shortest paths. Direct approach for this is to add such prefixes to Popular Prefixes (PP) in which case they are matched because they are longer prefixes than any covering VP. A number of studies conducted roughly within the past decade (e.g. [56], [57], [58], [59] and [60]) clearly show that a major part of all the traffic is destined to a set of prefixes that take only some percents from the complete routing table. Moreover, according to the same study by Ballani et al. [55] by adding as few as one percent of the POPs most popular routes to each routers FIB would cause about 90 percent of the packets to be routed via the shortest paths. The fact that VA allows prefix popularity to be considered on POP or router basis, allows capturing the effect more efficiently.

Unfortunately, the benefits of popular prefixes are still not quite as good as one might think. The reason for this is that many prefixes cannot simply be added as popular prefixes to the FIB without causing some other prefixes to be incorrectly routed. The longest prefix matching rule that allows one to add a sub-prefix into the FIB even though a shorter VP exists causes also problems in handling popular prefixes. It forces prefixes with longer prefix length to be always preferred over shorter ones, i.e. allowing a popular prefix to be used instead of a VP covering it. Identically, if a popular prefix covers some other prefixes, they will be matched by the PP, instead of a VP. Due to this, traffic towards the PP and the prefixes it covers are forwarded according to the popular (shorter) prefix and forwarded directly out of the AS via tunnels. If the next-hops on a PP and covered prefixes directs to different ASs, packets towards the covered (longer) prefixes may be dropped on the next AS or forwarded via a detour.

To further clarify the problem, an example is gone through. Say there is a VP 10.0.0.0/8, and its sub-prefixes 10.1.0.0/16 and 10.1.1.0/24. If the VP is used to suppress all the prefixes it covers and 10.1.0.0/16 is marked as a popular prefix, FIB would include the VP and the PP but not the 10.1.1.0/24. According to the longest prefix matching rule, traffic towards 10.1.1.0/24 would now be forwarded to the AS indicated by the PP, which may or may not be the same as in 10.1.1.0/24.

If the address space would be perfectly aggregated, such problem would never occur. However, prefix space is highly hierarchical and current trends tend to make the situation worse. The average number of covered prefixes a single prefix has was over 0.68 in 2010 (calculated from a real routing table consisting of over 99 percent of the DFRT at that time). Moreover, 61 percent of the prefixes with prefix length less than 13 covered some other prefixes, and the same for prefix length less than 17 and shorter prefixes was still 37 percent. Respectively, the average number of covered prefixes for the same prefix length limits was 67 and 6.9. How meaningful this is in VA depends on the length of the most popular prefixes, which may vary according to ISP and its POP as well as over time. Traffic engineering and policy based PPs should not be greatly affected as they typically are longest matching prefixes. Nevertheless, parenthood of prefixes should still be checked when installing PPs for any purpose.

Even though the direct PP selection should not be used, popular prefixes can still be useful. The selection could e.g. include all the same prefixes as with the direct approach, but include also all the prefixes they cover. Alternatively PPs could include only those prefixes which do not cover any longer prefixes. Although both of these would offer correct routing in any static situation, they do not take into account the dynamicity of the routing table. If a new prefix would appear in such part of the prefix tree which is covered by a PP, it would again be touched by the same problem with incorrect routing. It should also be remembered that most of the DFRT growth has come from the use of traffic engineering and multihoming introduce these long prefixes. Therefore, the issue is not purely theoretical, but something which would cause problems in VA enabled networks which use PPs e.g. to lower stretch.

To completely overcome the problem would therefore require constant prefix set monitoring and dynamic mechanisms to revise the PPs when erroneous situations are detected. Still, simulations described in Chapter 7 consider both of the introduced PP selection strategies and try to analyze their effectiveness in comparison to the direct most popular prefix selection and the case without popular prefixes. Same simulations analyze also another approach in making use of the unequal popularity of prefixes, which does not have similar problems with changing routing tables as with the use of PPs. The basic idea there is to pick Virtual Prefixes so that most of the traffic inside each one of them is forwarded to the same next-hop AS. This approach is further explained in Chapter 6.

6. Popularity Based Virtual Prefix Allocation

6.1 Overview

The functionality of Virtual Aggregation along with its limitations has been introduced and the Internet traffic has been noted to concentrate on a few prefixes. In this chapter a new VP selection mechanism for VA is formulated. It considers the prefix popularity distribution directly without needing to use additional routes called popular prefixes to forward a big portion of the traffic via shortest paths. As explained in section 5.4, PPs cannot be used without jeopardizing routing correctness or introducing a new task for active network management and there is therefore room for a better solution.

A new solution, called "Popularity Based Virtual Prefix Allocation" tries to make the best use of the notion of unequal popularity distribution by allocating VPs cleverly so that most of the traffic inside any VP is destined to the same next-hop AS. To get the most benefits from that requires also to assign APRs so that there is no path stretch to the popular next-hop ASs. Concretely this means that for each router the closest APR should reside on the shortest path to the closest egress point of VPs popular next-hop AS as visualized in Figure 10. If the traffic volumes from some routers towards the address space covered by a VP are small, it may be enough that APRs are close to the shortest path.

A setup which considers all the above is however hard to build with the additional constraints that should be present. The stretch for unpopular routes should be minimized and the amount of APRs should not be allowed to get too high. There should usually also be an upper limit for all stretch in order to guarantee that latency dependent services will not be disturbed. In general, the closer the APRs are to the ingress points, the less the traffic will be stretched and the more APRs are required.

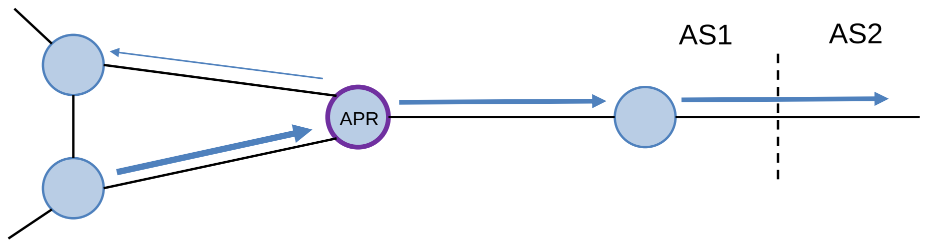


Figure 10: In Popularity Based VP Allocation APRs are selected from those routers that are on the shortest path to VP's popular next-hop AS.

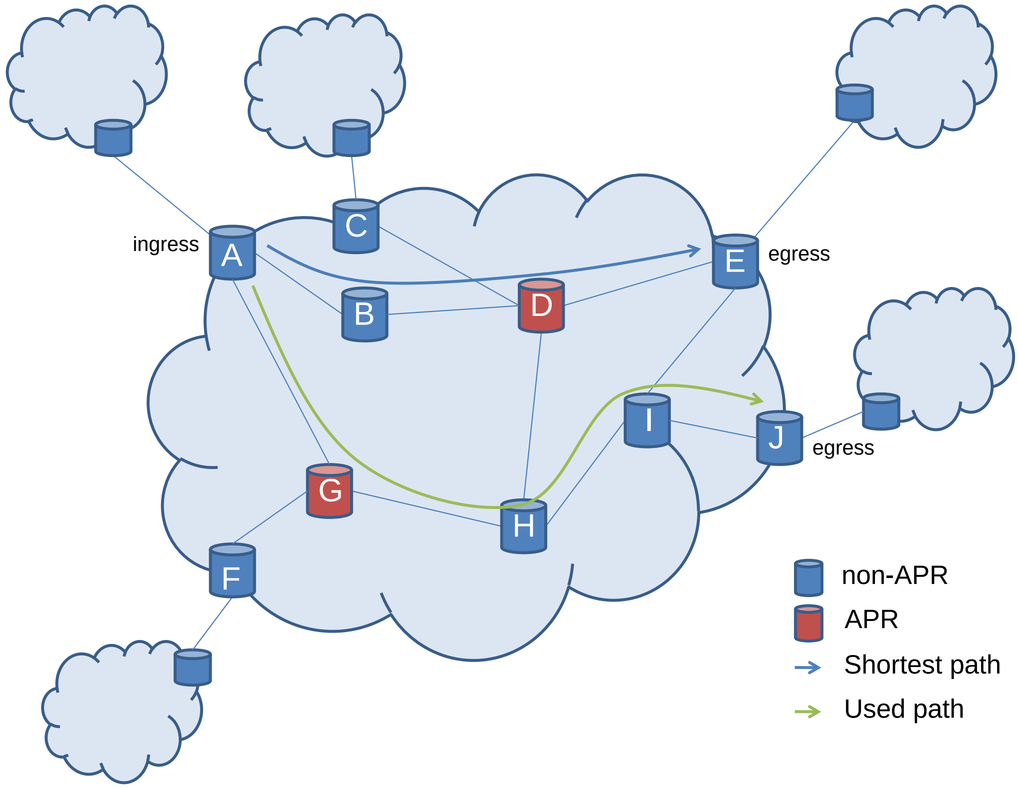


Figure 11: Difficulty of assigning APRs.

Figure 11 illustrates the challenges in assigning APRs with a sample network, where red routers are APRs and blue routers non-APRs for the VP in question. VPs popular next-hop egresses at E and J and additionally the VP covers also prefixes which egress at A. The administrator of the network would like to use only two APRs for the VP at hand in order to save enough FIB space. Let's say he can choose the APRs freely, but must allow traffic to go to popular next-hop AS without any additional delays and limit the delay for other traffic to one hop. If the administrator selects D and G routers to APRs (as is shown in Figure 11) all of the constraints would seem to be satisfied. The only problem is that because G is closer to A than D, A will use G as its APR and the shortest path is no longer used. If H would be used instead of G, such problem would not exist, but then the unpopular traffic from F to A would be stretched too much. Therefore one would need at least three APRs to solve the problem. Then the assignments could for example be A, C, F.

6.2 Selecting Virtual Prefixes

While the idea in Popularity Based Virtual Prefix Allocation dictates to choose VPs so that most of the traffic is forwarded via the shortest path, it does not yet say how the VPs should be picked. One could e.g. pick the most popular prefixes as such and add some short prefixes to allow the forwarding to prefixes which are not covered by any popular prefix. Such a setup would result in a somewhat similar outcome as what Ballani et al. [55] considered in their simulations with the differences that PPs would now be

VPs and paths to covered prefixes would only be stretched, not broken. The freedom of choosing popular prefixes on the router or POP basis would not be possible in this case and the APRs would of course be chosen from the popular prefixes' shortest paths.

Another approach is to try to pick VPs so that they cover as large parts of the address space as possible without breaking the idea that most of the traffic in a VP should go to a single next-hop AS. One way to implement this strategy is to use FIB Aggregation techniques and algorithms (introduced in Sub-section 4.2.1) to formulate a forwarding table with minimal amount of prefixes while considering popular prefixes as the initial set and next-hop ASs instead of IGP next-hops. The use of FIB Aggregation techniques will squeeze the table and allow the unpopular prefixes to be covered by fewer VPs in comparison to using PPs as such. Only such level 4 FIB compression algorithms (see sub-section 4.2.1) which consider all the covered prefixes when determining if a certain node should be an aggregate will guarantee that all prefixes will be covered. It will also squeeze the table most while calculating it requires most processing power and time. Because VPs are selected for a whole AS at a time, support for FIB Aggregation is not required on routers. The pre-calculated set is rather given, e.g. via network management system, to the routers after it has been formed elsewhere.

6.3 Changes to Virtual Aggregation

According to the Virtual Aggregation specifications [50], a VP must not be identical to any real prefix or be covered by one. However, to be able to create VPs flexibly, popularity based VP allocation requires that this requirement is released. The requirement was originally put there to guarantee that packets to real prefixes are never dropped or looped. Packets might get dropped if a VP would be the longest matching prefix for some destinations. By default the VP is put on the FIB, but there should not be any reasons why it could not be suppressed similarly as all the sub-prefixes in non-APRs. Suppression should also include all the VPs which are covered by the locally originated VP.

In case there are legacy routers in the network, this is not enough. If a legacy router would receive a packet to which a VP is the best match, it would be forwarded towards the VPs closest originator APR. Although this might cause path stretch, it should be tolerated if APRs for that VP are scattered all over the topology. Intolerable path stretch should only happen when some VPs are meant to be used only in certain parts of the topology and therefore have APRs only close to that area. In addition, if legacy routers would be used as ASBRs, loops would be possible as they would forward packets back to an APR instead of the remote ASBR. To fix this requires that tunnels from APRs are always directed straight to the remote ASBR and that legacy routers are able to initiate MPLS LSPs.

7. Simulations on Virtual Aggregation

The simulations as part of this thesis try to give an idea how much FIB space can be saved using Virtual Aggregation while also considering the drawbacks. Important part of this is to analyze how large extra latency costs are to be expected and what are the benefits in giving shortest paths towards rather stable most popular prefixes. The basis of these simulations are on an earlier study [55] which considers Virtual Aggregation in AT&T network with two VP allocation schemes and a single approach in selecting directly routed popular prefixes. The simulations of this thesis try to complement and dispute its results by comparing different strategies in selecting popular and Virtual Prefixes. FIB Aggregation (see Section 4.2.1) is also used as a reference, since it is more light weight and straight forward solution for the same problem. It can therefore give minimum target values for FIB sizes.

7.1 Simulation Environment and Setup

Ideally to get a real and accurate snapshot of a ISPs network at a single point in time, the topology (i.e. intra and inter-domain router connections, link latencies, used link weights and policies on all routing protocols), routing table and traffic matrix would be gathered from a target ISP directly. Unfortunately, many of these are considered as business secrets and hence are not widely accessible. The lack of reliable and accurate first hand information made the construction of the simulation environment complex. Absolute correctness had to be sacrificed and information had to be synthesized from several sources and in some cases even randomized in a way believed to produce a somewhat realistic simulation environment. Hence, one may question how well the achieved results on this simulated environment correspond to those that would have been gained using real information. Although this is a valid point, the simulations conducted on VA will further proof the ideas presented in Chapters 5 and 6, and give rough estimates on how well different strategies would work in live networks. The following sub-sections describe the process of constructing the simulated environment using the selected tools and equipment.

7.1.1 Tools and Equipment

The simulations were conducted with Matlab, because it offers a rather easy to use an environment to build the simulations on while allowing a great deal of freedom in implementation. It also offers simple but powerful tools for analyzing the results. Development work was done with a common laptop PC, but the actual simulations were run in a grid computing environment, since the required simulation time would have been months or even years with a single PC. The performance was the only downfall of selecting Matlab, as a single simulation could take days or weeks (depending on the simulation case) even after the code for slowest parts was rewritten to drastically improve the performance.

In Matlab the use of *for* and *if* sentences is very slow in comparison to many common programming languages (e.g. C). On the other hand, Matlab is very fast in handling vectors which take up a consecutive block from the memory. Many of the slowest routines were therefore rewritten from *for* structures to vectorized operations which handle entire vectors instead of going through vector elements one at a time. The use of vertical vectors was also helpful, because Matlab stores matrix elements first in row order and then in column order in the memory.

7.1.2 Topology

A critical part of the whole simulation setup is the topology. To be able to simulate VA requires not only the intra-domain topology, but also the inter-domain connections to different ASs. The starting point in building such a setup was the information available from the Rocketfuel ISP topology mapping engines webpage [61], which includes among others router level ISP topology maps as well as latency and IGP link weight information. Because the information was originally gathered through probing, it is natural that some inaccuracies exist. Furthermore, the usage of Rocketfuel ISP topology maps forced to simulate an outdated situation, since the information was released in years 2002 and 2003. To be able to estimate the feasibility of any topology dependant tool or feature, a set of different ISPs should be considered. However, a decision to use only one topology was made to accommodate for the time limitations. From the ISPs Rocketfuel analyzed, Sprint was chosen because it is one of the largest ISPs in the world.

With the Rocketfuel topology information alone, it is possible to create ISPs internal topology with roughly correct link latencies. A version of the topology map which contains backbone routers and their adjacent nodes was used for getting the router ids, their locations on city accuracy and the interconnections between the routers. A more complete map would also have been available, but it would have included too many routers that actually belong to customers at ISP. For the purpose of simulating concepts like Virtual Aggregation, this would be undesirable because ISPs do not have control over them. As mentioned earlier, Sprint’s topology from Rocketfuel is an abstraction of the real one. A clear indication of this is the fact that the network is not fully connected, having few islands apart from the main network. For the sake of meaningful simulations, only the largest contiguous part is used, which covers over 99% of the routers.

Some estimation and randomization were also necessary to get a reasonably correct set of link latencies, because latency information was available only to backbone links. Fortunately, it was evident that the latency information between routers on different cities was homogenous on city level, i.e. the latency from any city A to any city B did not depend on the selected direct link. IGP weight as a more meaningful value in terms of routing could not be used because it did not share this property. By using the homogenous latency information, latencies for nearly all the links could be specified as the information of routers’ locations was readily available from the Sprint topology map. The only exception was the routers which had unknown location because of the limitations in probing. The latency for links that terminate to such routers is randomly picked from the routers other links. In case there are no links to pick from, a random link from further away is used. While this definitely causes some minor additional inaccuracies, it minimizes the error by keeping the link latency distribution on each router close to what it originally was.

7.1.3 Routing Information and External ASs

BGP routing table, i.e. in our simulations the information about the best external routes, makes it possible to say to which AS each packet should be sent. Route Views [62] has been gathering this kind of information for years by having BGP connections from several locations to many ASs. A RIB dump is taken from a monitoring station which has BGP connection to Sprint and is dated back to the times when the Rocketfuel topologies were mapped (2002). A routing table corresponding to the Rocketfuel Sprint topology could be derived by filtering out all the routes which were not learned directly from Sprint. This table contains about 115 000 routes to nearly two thousand ASs, representing a vast majority of the DFRT in the early 2000's.

Rocketfuel maps contain also links to external ASBRs, i.e. give their IP addresses. Hence, with the internal topology and links to external ASBRs from Rocketfuel and the routing table from Route Views, the only thing still missing in terms of routing was the link between the external ASBRs IP addresses and the next-hop Autonomous System Numbers (ASN) in the routing table. The question was: how to do the mapping between them? In theory, this could have been achieved by simply mapping the router IPs to the ASNs where they belong to. It was also the starting point for the whole mapping and a method based on WHOIS [63] was used for doing this. Unfortunately, the mapping was not complete because ASNs could not be resolved for all of the routers' IP addresses, and moreover, large parts of the acquired ASNs did not match those from the routing table. Therefore only a partial mapping could be done with this approach and for the rest, a mapping between ASNs acquired from the two sources had to be done. Rocketfuel inherent ASNs were replaced by randomly choosing ASNs from the routing table in a round robin fashion. The result was that several ASBRs had more ASNs connected to them, than was originally the case in Rocketfuel topology and thus the visible part of the inter-domain topology is also altered.

In real life, operators use default routes in edge routers to lower the amount of entries in routing tables and to simplify their routing. Again, the information how the default routes are used in Sprint was not available and therefore correct settings could not be specified. A somewhat accurate or at least good setting would be to set the default routes to go from edge routers to most popular (in terms of number of advertized prefixes) core routers. However, default routes were not used as they can be unfavorable in some cases; e.g. when a customer wishes to multihome and needs all the routes advertized separately. Additionally, Virtual Aggregation can in most situations reduce the amount of entries in FIBs sufficiently so that there is no need for default routes anymore.

7.1.4 Traffic Matrix

In Virtual Aggregation, some of the routes are typically stretched creating additional latency to packets. Latency is still to be controlled and therefore it is important to analyze how big percentage of the traffic suffers from the stretch and how significant it is. Traffic matrices are often also business secrets eliminating the direct approach once again. For example, by simply analyzing some packet flow traces one could get some information about which destination and source prefixes are popular and which are not.

However, this does not yet reveal much about how the traffic flows through the Internet, or the Sprint network for that matter. A purely randomization based approach was chosen because no good source of information to build a traffic matrix could be found.

According to e.g. [60] traffic matrix distributions on different locations are typically heavy tailed. A rough estimate is that ten percents of the prefixes carry ninety percents of the traffic and the traffic follows a Zipfian distribution. Locally the ratio may be even higher [55], but here, only prefix wise total traffic volumes across the selected ISPs network are considered. By using a slope of -1.1 on a log-log scale, the ten to ninety ratio is achieved allowing to simulate the popularity distribution on prefixes in a rather realistic way. A research [64] also suggests that ASs follow the Zipfian distribution with -0.9 slope on Europe and -1.1 slope on Asia-Pacific region, which backs up the notion even though North America where Sprint mainly operates was not considered.

In the simulations, each prefix is given a popularity value which represents the amount of traffic destined to it in comparison to other prefixes. Equation 2 is used to get the wanted popularity distribution (shown in Figure 12), i.e. Zipfian distribution with a -1.1 slope on a log-log scale and for that popularity rank for every prefix is randomized. To get more realistic results, the ranks should accurately reflect the actual situation. One could try to model the ranks by weighting prefixes which have rather short prefix length, because they typically cover larger parts of the address space. One might however argue that a good deal of longer prefixes should be very popular as they have been split apart from larger prefixes because so much traffic is destined to them and individualized policing is therefore needed for them. As there seems not to be any clear way to say which prefixes should be more popular than the rest, popularity rank allocation is done at random.

(2)

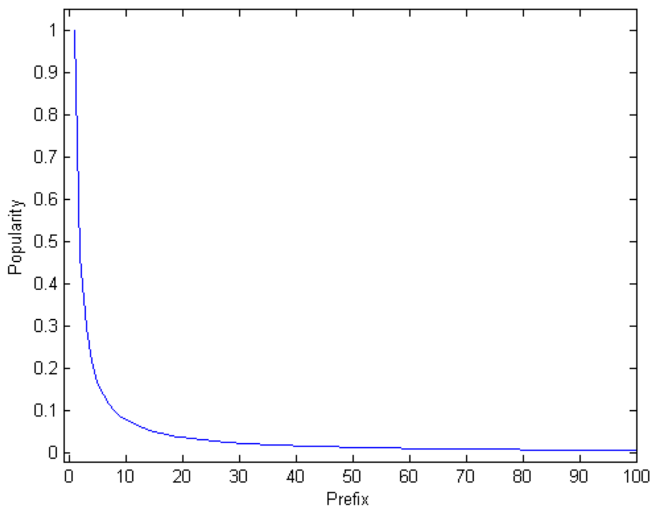


Figure 12: Prefix popularity distribution for the hundred most popular prefixes.

7.2 Overview of the Simulations

There are two main purposes in these simulations: to compare popular prefix selection methods and to validate and analyze the effectiveness of popularity based Virtual Prefix selection method. First a minimum target level is created by calculating FIB sizes after running FIB Aggregation on each router separately. Then a comparison point for ordinary Virtual Aggregation is set by considering Virtual Aggregation with uniform Virtual Prefix allocation and without popular prefixes. Using it, FIB sizes and path stretch inherent latencies are calculated with varying amount of VPs and with different maximum stretch latencies to see how they affect the results.

Only then is the effect of different popular and Virtual Prefix selection methods simulated and compared. Simulation cases were chosen to reveal the differences in achievable FIB sizes and path stretches in the best possible way while trying to also consider the credibility of the simulations. As each simulation takes tens of hours or even several days, simulations could not be repeated in the given time with different random seeds as many times as would have been desirable. Instead, a few runs were done to have at least some kind of confidence for the simulation results. In the following sections, processes and algorithms behind the simulations are first explained and in section 7.6 simulation cases are introduced and the reasons for running them are motivated.

7.3 FIB Aggregation

FIB Aggregation is implemented in these simulations following the idea (same results, different implementation) presented in FIB Aggregation IETF draft [49] level 4B algorithm which considers all the covered prefixes while implementing level 4 aggregation (see sub-section 4.2.1). In our simulations, first a prefix tree is formed. To avoid excessive memory usage, it only holds those prefixes which might be selected during the compression process, i.e. prefixes in a tree where longest real prefixes are its leaves and /0 is its root. During the tree build up, all prefixes in the tree are considered separately. For each prefix the number of covered prefixes towards different next-hops is calculated. In the second phase the tree is traversed in prefix length (level) order, checking on each prefix whether it should be installed or not and whether the prefix already has a next-hop or should it be set to the most popular covered next-hop. Code implementing the tree build up and the level 4B FIB aggregation algorithm are presented in Appendix A.

The result of running the algorithm is a smaller set of prefixes which still does not change routing for the advertized prefixes. FIB Aggregation and the resulting FIB sizes are calculated for each router separately since prefix to adjacent link mappings are usually different on different routers. Figure 13 depicts a simple case of this, showing that several variable forwarding tables are possible even in a very simple topology. In reality, networks may have hundreds or thousands of routers which can handle hundreds of thousands of prefixes, making it very probable that in most of the routers prefixes form unique sets on adjacent links they use.

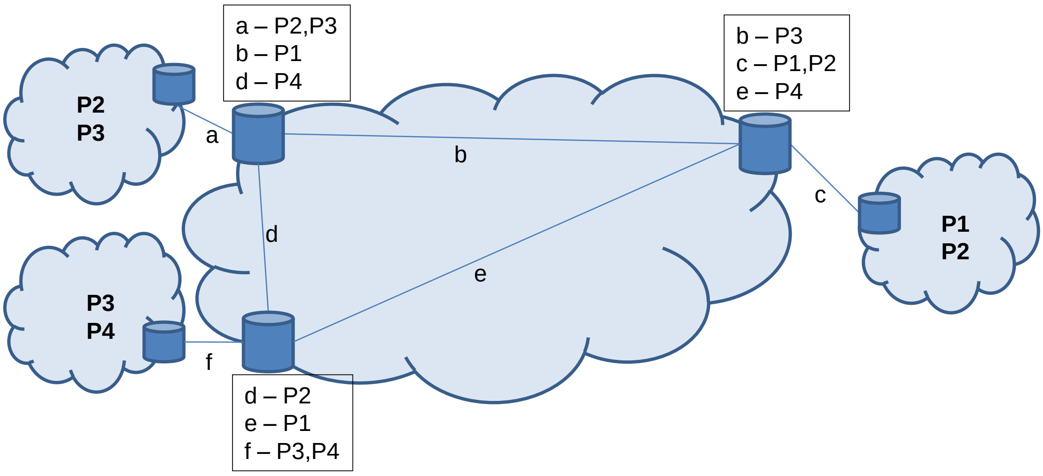


Figure 13: Prefix set forming in different parts of the topology. Px denotes a prefix.

7.4 Virtual Aggregation

Simulating Virtual Aggregation is divided into four main steps: picking a set of popular prefixes which get a differentiated treatment, allocating VPs according to the selected allocation scheme, assigning APRs for each VP and calculating the resulting FIBs and stretches. Results depend heavily on all of the first three while the last step tries to capture the essential statistics which are then later analyzed.

7.4.1 Virtual Prefix Allocation Schemes

The basic idea for uniform allocation algorithm is taken from [55] although the results with the used routing data do not give as uniform distribution as in the previous study. The allocation starts by having all the possible prefixes with prefix length 7 as VPs and continues by splitting those which cover too many sub-prefixes until all are below a specified VP size threshold. If the threshold is small enough, one or more of the sub-prefixes may cover so many other sub-prefixes that the threshold is exceeded from those alone. Because VPs must not be covered by sub-prefixes in any situation, the threshold must simply be exceeded. In fact VP is then set to be the immediate parent of the sub-prefix with shortest prefix-length (i.e. if sub-prefix is of length 8, VP is of length 7). Some of the VPs may still contain very few sub-prefixes either because some /7s contain only few sub-prefixes or because one of the too large VPs children covers only a few sub-prefixes. Appendix A includes code for implementing uniform VP allocation.

Popularity based Virtual Prefix allocation is implemented in these simulations according to the second approach described in section 6.2. Popular prefixes are used as the input for FIB Aggregation level 4b algorithm which forms a set of VPs by considering the adjacent ASs as the next-hops to aggregate on. The implementation of FIB Aggregation algorithm was described in section 7.3.

7.4.2 Strategies for Selecting Popular Prefixes

Section 5.4 gave the reasons for simulating different strategies for selecting popular prefixes. All popular prefixes cannot simply be routed via shortest paths without possibly causing more specific prefixes to be routed to incorrect adjacent ASs due to longest prefix matching. Our simulations have three different strategies, including the malfunctioning direct approach. Because the traffic matrix is based on randomization, it is better to leave the distribution of the traffic's source unconsidered. Therefore, it is possible to only use AS wide prefix popularity information and not to consider different popular prefixes for different POPs or routers.

The selection arguments for the direct approach are very simple: just select the given percentage of most popular prefixes and set them to be tunneled directly to the closest egress points. This method is called the Default Popular Prefix Selection (DPPS). The second considered strategy includes only those most popular prefixes which do not cover any other prefixes; meaning that most of the short prefixes will not be selected even though they were more popular than the longer ones. Because only a part of the most popular prefixes are selected, it is called a Picky Popular Prefix Selection (PPPS). The third strategy sacrifices some FIB for unpopular routes so that most popular prefixes can get the special treatment. More specifically, with this Inclusive Popular Prefix Selection (IPPS) strategy, the most popular prefixes with all their covered prefixes are inserted into the FIBs. With IPPS the amount of PPs cannot be precisely adjusted since if a prefix is added all of its covered prefixes are also added. In our implementation the set of prefixes is added if the amount of accepted PPs was not exceeded beforehand, causing the amount of PPs to be usually a little higher than where the target limit was set on.

7.4.3 APR Assignments

The problem of assigning aggregation points is a complex issue with several things to consider. It determines how much path stretch will accrue and how it is distributed to different routes. APR selection is also the largest factor in deciding how large FIBs routers will have. The purpose of the assignments is to find in some way optimal set of APRs for all the predetermined VPs within the given boundaries. Figure 14 visualizes an example setup in a network after APRs have been assigned showing the relation between the number of APR roles (for different VPs) and FIB size. It also becomes evident that a router may have very small FIB although it serves a great number of routers as an APR. If the goal is to have small FIBs on every router, no router should be an APR for many VPs. Still, there may be topological restrictions which dictate that some routers cannot reduce their FIB size considerably without causing too much stretch on some routes. In Figure 14, the centermost router can serve most of the other routers for one of the VPs while the largest FIB router serves only two other routers, but for many VPs. These two examples clearly show that in terms of FIB size, the latter is inefficient and should be avoided.

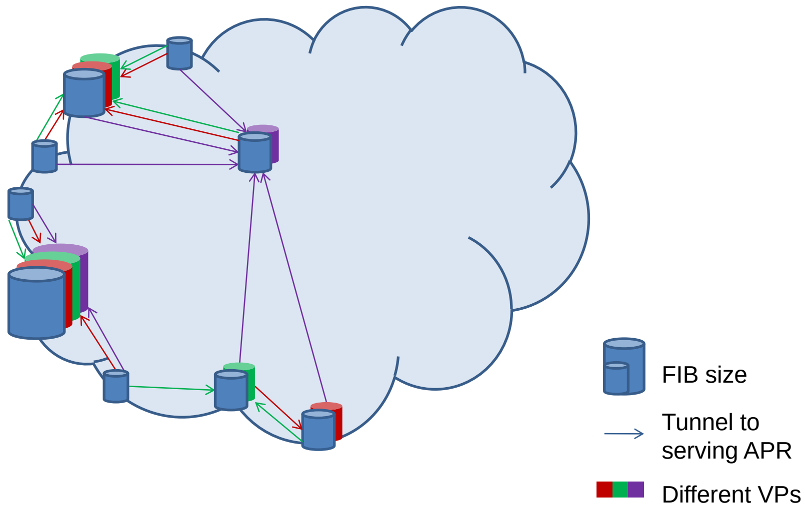


Figure 14: APR assignment algorithm decides which routers are APRs for which VPs.

Based on a previous work [10], Ballani et al. [55] give a reference solution for assigning APRs by specifying a greedy algorithm for minimizing the maximum FIB size. The new algorithm implemented for this thesis (Appendix A) follows the same idea extensively, although tries also to keep the amount of APRs low. A major point in both algorithms is to constrain the maximum path stretch so that the most crucial disadvantages (namely: extra latency and load) can be controlled.

In reality, each non-APR uses the closest APR in terms of IGP metrics. To do that, the whole set of APRs for a single VP would have to be considered simultaneously and it would be hard to find a set of routers that obey both requirements while still roughly minimizing the worst FIB size. Therefore, the used algorithm (in Appendix A) relieves the closeness requirement and allows any APR to serve a router if it simply complies with the stretch limit. This may not even be such a big abstraction of the real situation since close by routers tend to be selected as serving APRs quite often because far away routers will more probably increase the stretch for some destinations beyond the constraint.

Table 1: Notations used in APR assignment algorithm.

Notation	*Meaning*
R	All the ISPs routers
r	Single router in ISPs topology
T	The full routing table consisting of all the external routes
V	The chosen Virtual Prefixes
v	A single Virtual Prefix
P	Popular prefixes
F_r	FIB size in router r
A[r]	The set of Virtual Prefixes for which r is an aggregation point
n_v	Number of sub-prefixes VP v covers

```
Worst FIB Size = 0
for all r in R do
    for all v in V do
        Calculate can_serve_{r,v}
    end for
end for
Sort V in decreasing order of n_v
for all v in V do
    while all routers all served for Virtual Prefix v
        Sort R in decreasing order of |can_serve_new_{r,v}|
        for all r in R do
            if Fr + n_v <= Worst FIB Size && v     A[r] && |can_serve_new_{r,v}| > 0) then
                A[r] = A[r]     v
                F_r = F_r + n_v
                Mark all routers in can_serve_{r,v} as served
            end if
            if All routers are served for v then
                break
            end if
        end for
        if All routers are not served for v and all routers have been checked then
            for all r in R do
                if v     A[r] then
                    A[r] = A[r]     v
                    F_r = F_r + n_v
                    Mark all routers in can_serve_{r,v} as served
                    Worst FIB Size = F_r
                    break
                end if
            end for
        end if
    until while
end for
```

Figure 15: Aggregation point router assignment algorithm to minimize largest FIB size with a constant maximum stretch latency.

To help understand the used APR assignment algorithm in Figure 15 the notations used in it are introduce. In short, r stands for a router, T for a routing table, v for a Virtual Prefix and p for a popular prefix. Capital letter characters indicate the whole set while lower-case characters mean a certain individual in the set. Also, F_r means the FIB size in router r, A[r] the set of Virtual Prefixes for which r is an aggregation point and n_v the number of sub-prefixes covered by VP v. The complete list of notations is summarized in Table 1.

The algorithm splits up to two separate parts of which the first (lines 2-6) depends on the used VP allocation scheme and the second (lines 7-33) is always the same. The first part calculates the set of routers for which each router can be an APR and repeats this for each VP. For uniform allocation, the router set with each router-VP pair is all the routers that do not cause more than threshold amount of extra latency for any packet. For popularity based allocation there is an additional requirement that the APR must be on the shortest path from router r to VPs popular next-hop, so that path stretch can be prevented for popular routes.

The actual APR assignments (lines 7-33) starts by sorting the VPs according to how many sub-prefixes they have (line 7), causing APRs to be selected for largest VPs first. The benefit of this is that FIBs are more of the same size in different routers as smaller VPs fill up the FIBs in routers which are not APRs for larger VPs. If a larger VP would be handled last FIBs would be more evenly loaded before it and some of the FIBs in last VP's APRs would then become larger than any other. While considering a single VP, APRs are assigned in the order of how many not yet served routers a candidate can serve. Additionally the algorithm uses a boundary on FIB size that may not be exceeded while assigning APRs in lines 11-20, so that all the smaller FIB routers that can serve some routers are assigned prior to routers that already have bigger FIBs. The threshold, i.e. worst FIB size, is increased only when smaller FIB routers cannot serve all the routers (lines 21-31). After the threshold is increased, the same routine is repeated until all routers have been served for that VP. At the end, all routers will be served by some APR for each VP, ensuring reachability from any router towards any prefix.

As mentioned earlier in this sub-section, the basis of this algorithm was taken from Ballani's work and was only slightly modified by us. The largest change was the introduction of *can_serve_new* variable which is used to sort the entire router set (R) after each APR assignment. In the original algorithm the router set was sorted only before each VP. This change was done to lower the amount of APRs, since with the original algorithm many routers which could serve a similar but not identical set of routers were all assigned to APRs on the basis of how many routers they can serve even though fewer routers could actually serve the combined set of routers. Minor changes were also done on when a router is selected and marked as an APR.

7.5 Gathering and Analyzing Results

Some key information about how the VPs are chosen, how the APRs are assigned and what kind of FIBs and stretch latencies there exist with each setup, had to be calculated. After a simulation the FIB size of each router was stored along with the results of a calculation which gives several variables related to stretch latencies. Using these variables the behavior in FIB sizes and stretch latencies was analyzed with Matlab and the results introduced in Chapter 8 were constructed.

7.6 Simulation Cases

In total, nine sets of simulations were performed, each trying to capture one behavior with the selected aggregation strategy. Two of these are about FIB Aggregation and the rest about Virtual Aggregation. With Virtual Aggregation, simulations are run to show

how the amount of VPs and the maximum stretch with both uniform and popularity based VP allocation schemes influences the outcome, and what kind of effects different popular prefix selection strategies gives with different amount of PPs. For better credibility a set of five different random seeds is used with each configuration. The purpose of this is to show the effects of randomization between the simulation sets while still being able to show that the results do not depend greatly on the random values.

The first FIB Aggregation simulation set gives an overall view how much FIB Aggregation can shrink FIBs by running the algorithm for all the routers. Because compressing every routers' FIB with FIB Aggregation level 4B algorithm and calculating their FIB sizes takes very long to compute with our implementation, a smaller set of routers is useful so that multiple random seeds can be used to evaluate simulations on a few weeks time frame. The second set considers only ASBRs and 30 core routers allowing it to capture the effect of randomization on the selected sample set. By comparing the results of these two sets it is possible to get better confidence intervals for the acquired results than would have been with the first set alone.

Out of total seven Virtual Aggregation simulation sets, five will capture the effects with uniform allocation. The first reveals how FIB sizes and path stretches vary in relation to maximum stretch constraint while the second does the same with VP size and amount of VPs. The last three will show how much lower stretch latencies can be expected by using different amount of popular prefixes and how the use of different popular prefix selection strategies influences the outcome. Popularity based VP allocation scheme is run with different amount of popular prefixes and with different maximum stretch latency constraints. Similarly as with FIB Aggregation, Virtual Aggregation simulations are repeated with different random seeds.

8. Simulation Results

This Chapter represents results of the simulations described in the previous Chapter. Results show how FIB Aggregation was able to shrink FIBs and how different settings with Virtual Aggregation influenced the achieved FIB sizes and path stretch.

8.1 FIB Aggregation

Running FIB Aggregation on all the 7303 routers in the simulation topology resulted in quite high savings in terms of FIB size on most of the routers. On average the remaining FIB size on a router was 3017 prefixes, i.e. 2.6 percents of the full DFZ FIB. Even the largest FIB was 58 percents smaller than the full DFRT and had 48473 prefixes. As depicted in the FIB size distribution in Figure 16, most of the routers can actually manage all the eBGP routes with very few FIB entries. Concretely, 90 percents of the routers can manage with a FIB that can hold only 12 percents of the DFRT. Many of the routers have only a single entry (indicated by the leftmost bar in the figure), illustrating that they have only one link over which the inter-domain routes are learned. However, the situation is somewhat worse in reality when internal routes are also considered.

It was also considered how the location of the router affects its FIB size by defining a backbone (BB) router set and an AS border router (ASBR) set. Routers are considered as ASBRs if they are an egress point towards any prefix while backbone routers are readily indicated in the Rocketfuel topology map. The result was that FIB size correlates heavily with the location. As Figure 16 shows, largest FIBs are typically found in backbone (green bars). Non-backbone AS border routers (red bars) and most of the non-ASBR and non-BB routers (blue bars) have very small FIBs. BB routers can also be ASBRs and in fact in the simulation topology a large majority of the backbone routers are also border routers, while the amount of ASBRs is still over twice the amount of BB routers. In total, 72 percents of the cumulative amount of FIB entries is found on ASBRs and 64 percents on BB routers. Together they still only constitute roughly three quarters of the total FIB load as there are quite few non-ASBR BB routers.

The presented results above were gathered using only one random seed value. Ten simulations which considered only ASBRs and 30 core routers were run to show that the randomization does not affect the trend but merely change the absolute values. Among these simulations average FIB, largest FIB and FIB variance were calculated and indeed, the results did not vary too much. The difference of the largest value from the smallest value in each set is presented in Table 2.

Table 2: Difference of the largest value from the smallest value within each identifier set (using FIB Aggregation).

Average FIB	Largest FIB	FIB variance
3.0 %	6.7 %	7.9 %

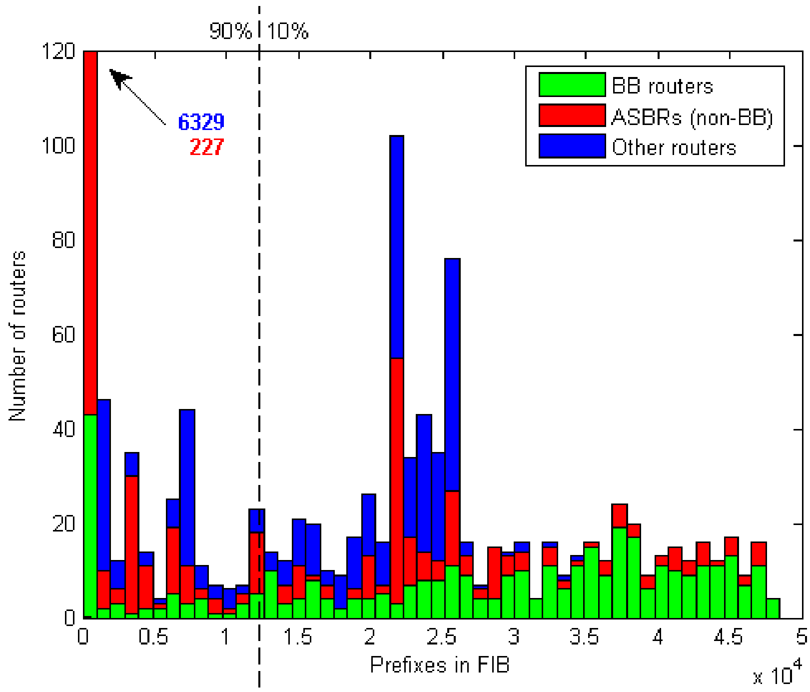


Figure 16: Routers' FIB size distribution with FIB Aggregation. Dashed line shows the 90th percentile.

8.2 Virtual Aggregation with Uniform Virtual Prefix Allocation

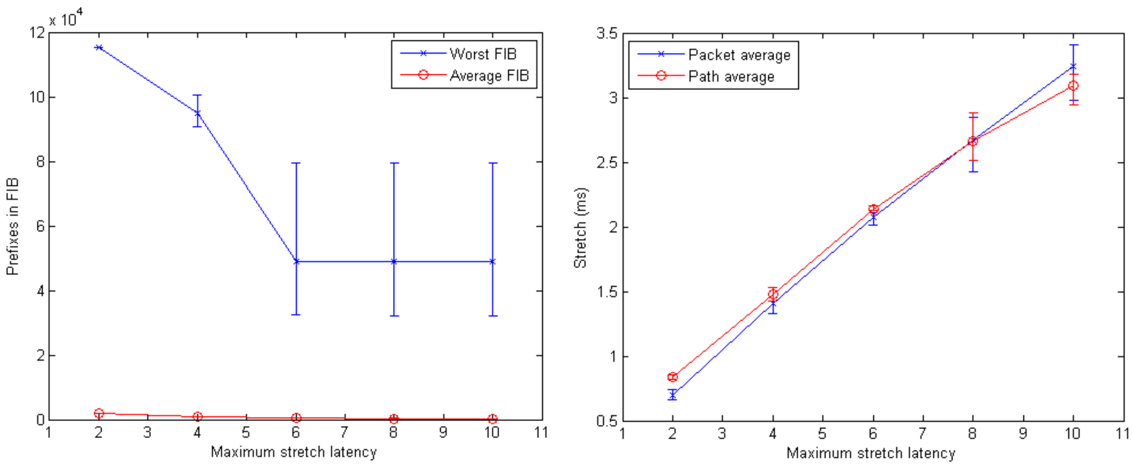
The effectiveness of Virtual Aggregation in shrinking FIBs should be better than of FIB Aggregation as it has more severe drawbacks like increased latency and router load. Our simulations show that this is indeed true if the configuration values for the APR assignment algorithm (presented in section 7.4.3.) are chosen wisely. What kind of FIB savings are to be expected in general as well as how the number of VPs and the maximum stretch latency constraint affects FIBs and stretch latencies is illustrated in this section.

As said in section 7.1.4, there is no real or simulated view on how the traffic actually flows in the network. In these simulations two measures are used to describe the stretch and both assume that traffic is injected from each ASBR with equal capacities and with a pre-determined distribution on prefixes. The first measure is about the path wise stretch latency and the second about packet wise stretch latency. Path stretch latency is used to show how routes are affected with each configuration. Increased packet stretch latency on the other hand illustrates how latencies are altered on packet level. They are both included in the results to show how well the unequal popularity among the prefixes can be taken advantage of with different mechanisms and setups. More precisely, it is their difference that shows it.

To start with, results first show what happens when the maximum stretch latency is gradually changed from two to ten milliseconds. Figure 17a depicts the maximum and average FIB sizes and Figure 17b the average path and packet stretch latencies with VP size limited to 0.1 percents of the full routing table (justified later), giving 1444 prefixes. Maximum FIB size could not be reduced when stretch latency was constrained to two milliseconds and substantial reduction to it could only be seen when the stretch constraint was set to at least six milliseconds; although the error margin was also significantly increased at that point. Interestingly, no further maximum FIB savings could be achieved with larger latency constraints. Because average stretch latency seems to increase constantly (with a slope of ~ 0.3) and average FIB size is lowered only slightly when more stretch is allowed, maximum stretch latency is constrained to six milliseconds in the rest of our simulation cases.

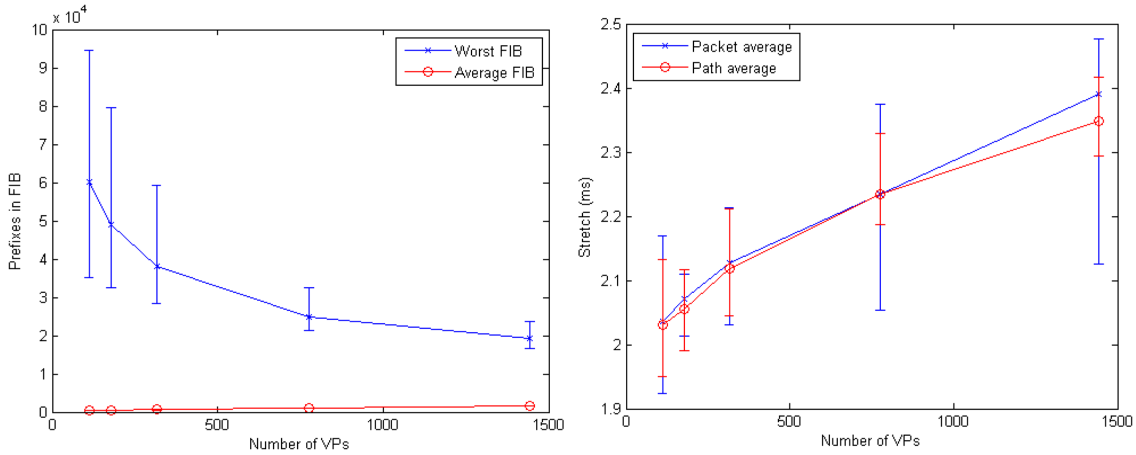
The second simulation case is to reveal how the maximum VP size and the number of VPs affects FIB sizes and path stretch latencies. In forming VPs, the VP size was used as the main parameter. However, because the number of VPs is more intuitive and with uniform VP allocation the number of VPs changes with the maximum allowed VP size, the latter is used in results. Figure 18 depicts what kind of an effect the number of VPs gives with constant six millisecond maximum stretch latency. Figure 18a shows how both maximum and average FIB size varies and Figure 18b illustrates the relation between the number of VPs and average stretch latencies.

From Figure 18a it is visible that a more fine grained VP allocation has a positive effect on the worst FIB size while at the same time average FIB size increases only mildly as aggregators are spread more evenly on the network. Figure 18b shows a rather constant and small increase to the average path and packet stretch latencies as the number of VPs is increased. The packet wise average stretch latency increases slightly faster than the path wise average stretch latency because heavy traffic prefixes are by chance stretched more than prefixes on average. The maximum VP size of 0.1 percents in the previous simulation case was chosen because it gives smallest worst FIB size.



(a) (b)

Figure 17: Average and maximum FIB size and average stretch latency with different maximum stretch latencies (uniform VP allocation).



(a) (b)

Figure 18: Average and maximum FIB size and average stretch latency with different number of VPs (uniform VP allocation).

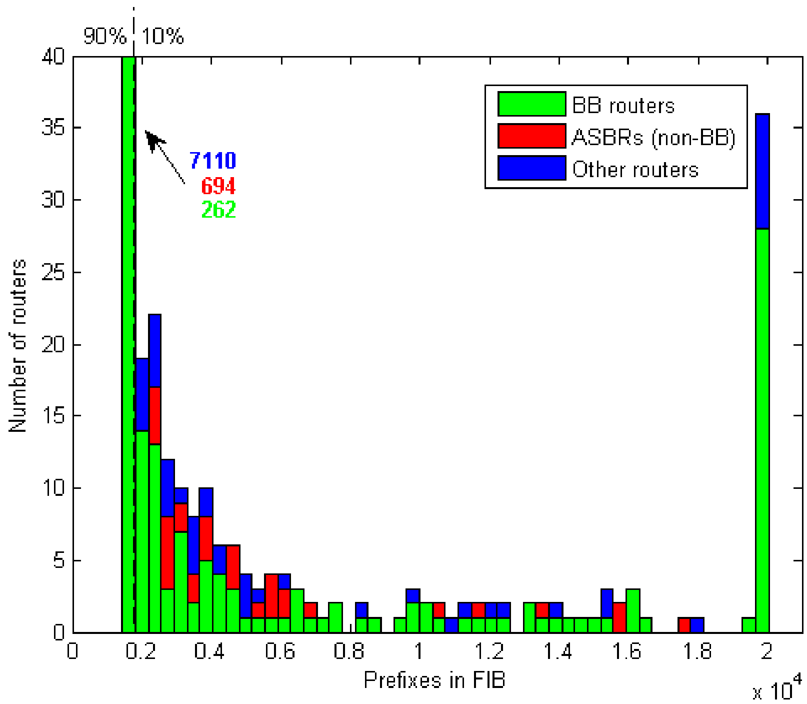


Figure 19: Routers' FIB size distribution with Virtual Aggregation 6 ms maximum stretch and 1444 uniformly allocated VPs.

Table 3: Difference of the largest value from the smallest value within each identifier set using Virtual Aggregation with 1444 uniformly allocated VPs and maximum stretch latency constrained to six milliseconds.

Average FIB	Largest FIB	FIB variance
1.5 %	42 %	70 %

In order to better see the FIB savings on the topology with uniform VP allocation and without handling popular prefixes in any special way, similar analysis as with FIB Aggregation in section 8.1 is conducted here. For that, the simulation settings which gave the best results in terms of lowest worst FIB size are used. That is, maximum stretch latency is set to six milliseconds and the number of VPs to the largest simulated (1444 VPs). Results for this in Figure 19 show that, again, most of the FIB load is concentrated on backbone routers and less than one percent of all routers have more than ten percent of the full DFRT. Differences among simulations with different random seeds are now somewhat higher than with FIB Aggregation, which can also be seen from the error bars in Figure 18a. Still the effect is limited to how FIB load is distributed in the network and does not change the actual cumulative load indicated by the average FIB size. Table 3 gives the exact numbers on how average FIB, largest FIB and FIB variance changed due to different random seeds.

8.3 Virtual Aggregation with Popular Prefixes

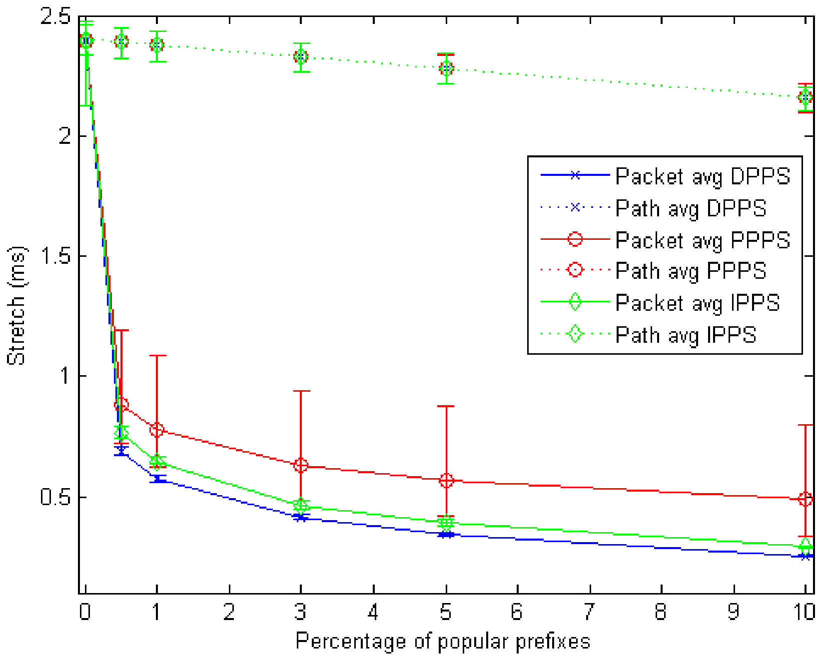


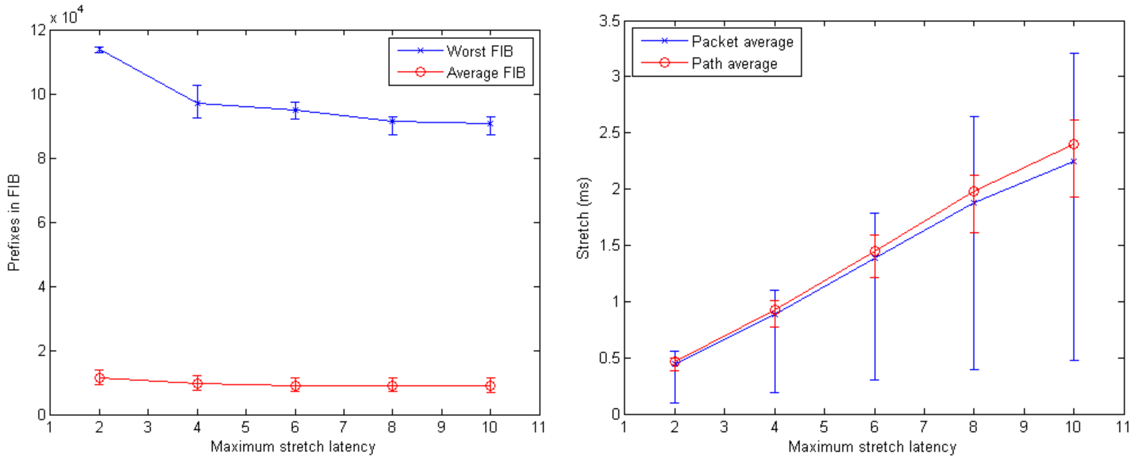
Figure 20: Effect of using popular prefixes with three different styles. DPPS: pick most popular to PPs. PPPS: pick only non-covering most popular to PPs. IPPS: pick most popular and the prefixes they cover to PPs.

Popular prefixes (PP) are used to forward a big part of the traffic directly out of the AS via shortest paths, eliminating path stretch completely for the affected prefixes from the routers in which they are used. Because in the simulations PPs are set identically on all of the routers, a prefix is chosen to be either popular or not for the whole network. In case of the default PP selection (DPPS) and picky PP selection (PPPS), FIBs are increased directly according to the number of prefixes considered popular. As explained in sub-section 7.4.2, the FIB size increase in inclusive PP selection (IPPS) is typically slightly larger than the amount of popular prefixes and is affected by the number of prefixes popular prefixes cover. In our simulations this had a negligible effect and FIB sizes were practically identical with all three methods.

Figure 20 illustrates the effect of each PP selection strategy on average path and packet stretch latencies. With all three strategies the improvements are small and linear for average path stretch latencies – as was expected. However, for average packet stretch latencies the results are more dramatic and there are some differences between the PP selection methods. As anticipated, DPPS gives the best results because it selects the most popular prefixes and is able to direct most traffic via PPs. With it, it's possible to get as low as 0.5 millisecond average stretch latency for all of the traffic by increasing FIBs with a reasonable one percents of the full DFRT. But, because DPPS does not function properly with the covered prefixes, it can only be used as an upper bound on the improvements achievable with methods that do not cause packets to get forwarded erroneously. In fact, IPPS delivers only slightly worse results with a very small error margin. For PPPS the results are worse even though in the most favorable simulation run, it is as good as IPPS.

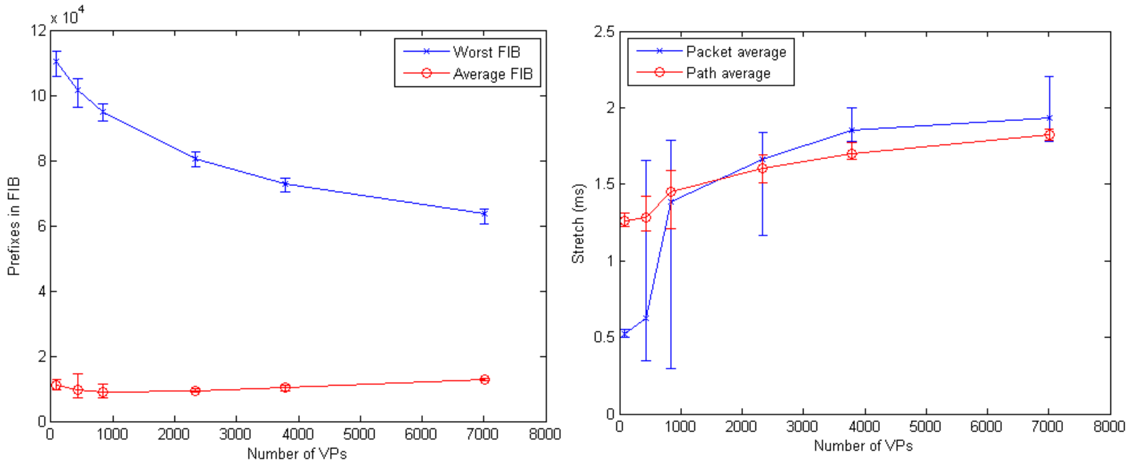
8.4 Virtual Aggregation with Popularity Based Virtual Prefix Allocation

The popularity based Virtual Prefix allocation scheme is designed to reduce stretch in comparison to Uniform Allocation scheme. Our simulations prove this, although, the overall results are less impressive. Similarly as with Uniform Allocation, graphs are plotted on FIB size and average stretch behavior while letting the maximum stretch latency (Figure 21) and the number of VPs (Figure 22) to change in turn. Figure 21b illustrates that average stretch increases linearly with the maximum stretch constraint, i.e. quite similarly than with uniform VP allocation. Figure 22b shows stretch to be very small when there are only a couple hundred VPs but to become significantly worse when the number of VPs is increased. By comparing the average path and packet stretches, it can actually be seen that the method was unable to take advantage of the prefix popularity distribution when there were close to one thousand VPs or more. The sharp increase in the average packet stretch is due to the Zipfian distribution which is used to simulate the prefix popularity. At a certain point (~800 VPs) the average of the most popular prefixes becomes almost the same as the average of all the prefixes; after which the stretch improvements are small. The large error bars here indicate that with different random seeds the point when the stretch sharply increases is different.



(a) (b)

Figure 21: FIB size and stretch latency with different maximum stretch latencies (popularity based allocation).



(a) (b)

Figure 22: FIB size and stretch latency with different number of VPs (popularity based allocation).

With popularity based VP allocation FIB savings are quite modest in comparison to uniform VP allocation. Worst FIB sizes are especially poor (shown in Figures 21a and 22a) while average FIB size varies between 10 and 15 percents of the full DFRT. Again by looking at the FIB distribution among the routers in Figure 23 it can be seen that the largest FIBs are on backbone routers. However, a notable amount of non-ASBR and non-BB routers (blue bars) have semi-large FIBs. This is the main source for the relatively high average FIB size and reflects the behavior of our method and APR assignment algorithm in general. Because over half of the routers are an APR for at least one VP and because the maximum VP size is nearly 20 percents of the full DFRT in the worst case (depending on the number of VPs and the random seed), a single VP may be responsible for a great number (at most ~21 000) of entries in many routers. 90^{th} percentile (dashed line in Figure 23) further describes that most of the routers will have to do with an average sized FIB, having about 19 percents of the DFRT. These results seem also to be sufficiently stable with different random values and in comparison to simulations on uniform allocation. FIB distribution and largest FIBs vary only mildly while average FIB experiences more variance. Table 4 gives the exact number on how much average FIB, maximum FIB and FIB variance vary within the set of simulations.

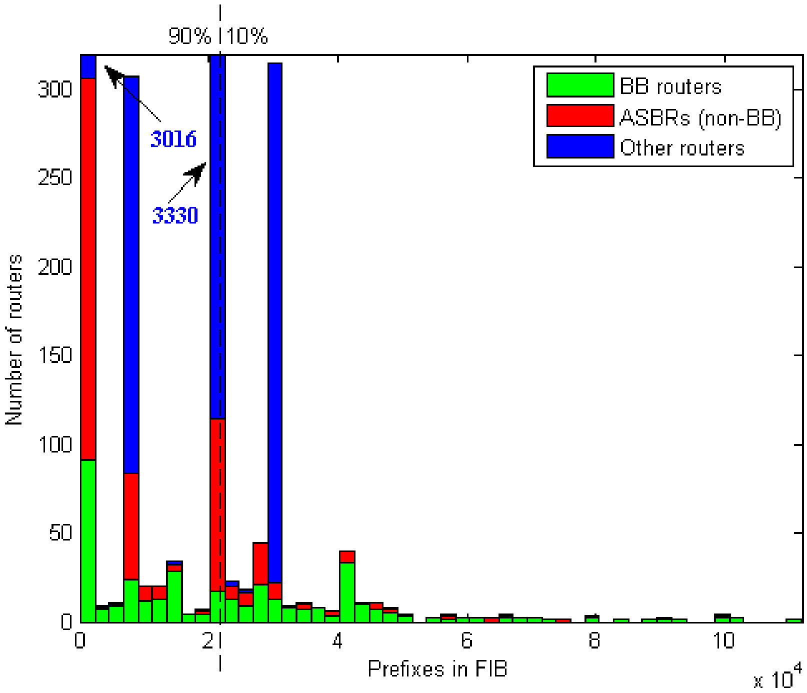


Figure 23: Routers' FIB size distribution with Virtual Aggregation using popularity based VP allocation. Dashed line shows the 90th percentile.

Table 4: Difference of the largest value from the smallest value within each identifier set using Virtual Aggregation with popularity based VP allocation and a 6 millisecond maximum stretch latency.

Average FIB	Largest FIB	FIB variance
34 %	7.3 %	59 %

9. Analysis of the Simulation Results

Chapter 7 introduced the setup on which the simulations were conducted, the methods used to conduct them and the simulation cases which were chosen to run. Chapter 8 presented the simulation results and in this chapter we finally analyze the results. Because randomization was the only option to simulate several details in building our simulation environment and in measuring path stretches, the correctness of the results should be further analyzed. Therefore, it is necessary to consider how our results using uniform VP allocation with and without popular prefixes (DPPS) correlate with those Ballani et al. [55] achieved in their simulations on AT&T network. However, before that, the pros and cons in using PPs to lower stretches are analyzed and the popularity based VP allocation scheme is compared to other mechanisms. The reliability check finishes this Chapter.

9.1 Benefits of Popular Prefixes

Comparing the results from uniform VP allocation VA with and without directly routed popular prefixes shows clearly the benefits in using PPs. Several times smaller path stretch can be anticipated with all three simulated selection methods than completely without PPs (see Figure 20). Importantly, PPs can be chosen so that they give significant reduction to stretch and do not violate routing correctness in any static situation. According to our simulations, the best way to do this is to include the most popular prefixes and all their covered prefixes to PPs (IPPS). To further analyze the goodness of IPPS, Figure 24 was plotted to show how the number of PPs influences routes and traffic in the network. With only one percent of prefixes as PPs the amount of traffic impacted can be lowered from 86 percents to only 24 percents, indicating that over 75 percents of the traffic would be forwarded via the same route as when VA is not deployed. Traffic stretched, i.e. the relative stretch a packet will experience on average, can also be lowered significantly and while considering the latency increase with one percent of PPs, traffic is stretched only 16 percents. If popular prefixes would be considered on each POP separately, the results would be even better.

However, as mentioned in Chapter 5, using PPs might still be risky and at least require more from the network management because the set of prefixes usually changes over time. Again, this can cause some of the prefixes to be routed incorrectly if the set of PPs is not revised after the introduction of new prefixes which are covered by a PP. Whether good and fast revision mechanisms can be built dictates how feasible scenario it actually is to use PPs.

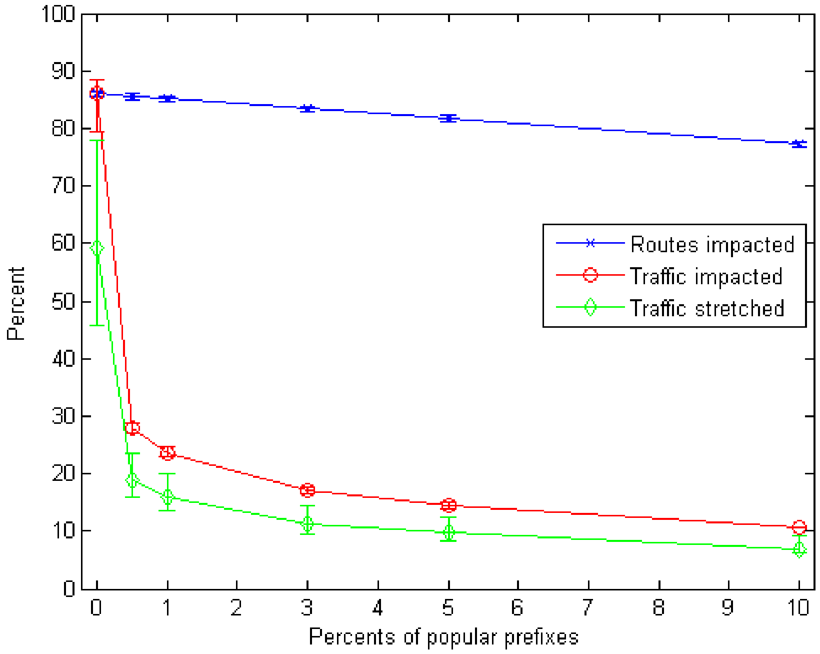


Figure 24: Percents of routes & traffic impacted and percents of traffic stretched with uniformly allocated VPs, IPPS and 6ms maximum stretch.

9.2 Using Popularity Based VP Allocation

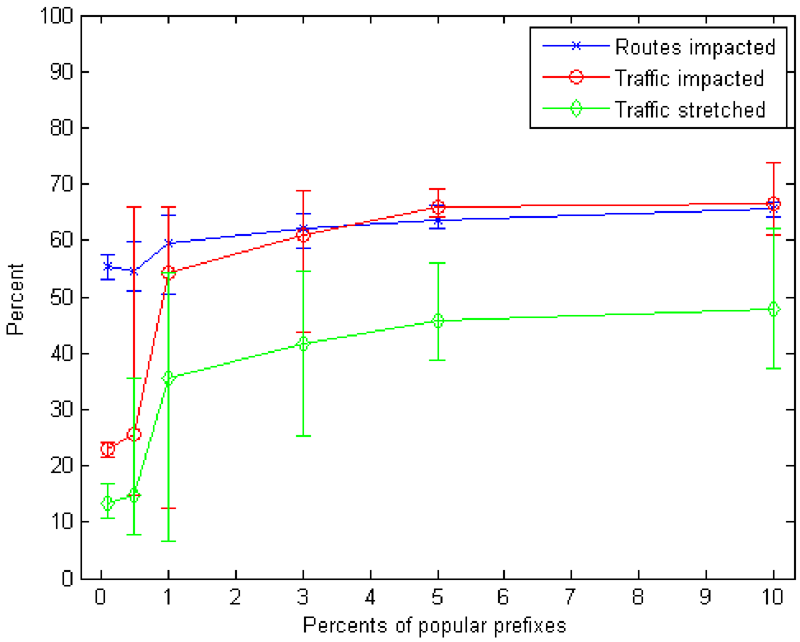


Figure 25: Percents of routes & traffic impacted and percents of traffic stretched with popularity based VP allocation and 6ms maximum stretch.

Figure 25 summarizes how big part of the traffic is stretched and how large latency increase is accrued when using popularity based VP allocation. With 0.1 percents of prefixes considered as popular the average stretch latency (traffic stretched) is increased only 13 percents which is distributed on 23 percents of the traffic (traffic impacted). By comparing this with the graphs on Figure 24, it becomes evident that the results are quite similar with uniform VP allocation and PPs. With uniform VP allocation and without PPs traffic stretched and traffic impacted increases constantly as the amount of VPs is increased. The former stays always above 50 percents and the latter above 80 percents, but in comparison to what popularity based VP allocation achieves, this is much worse.

So, if the stretch is so much lower with popularity based VP allocation, how much worse FIB savings could be tolerated to still make it an appealing option? Ultimately it is up to the network administrator who has to consider the magnitude of required FIB savings. But, if most of the ISP’s routers are capable of handling at least average sized FIBs (~30 % of the full DFRT) and there are no tools to update PP-list sufficiently fast after a change in routing table, using popularity based VP allocation might be beneficial. By looking into how DFRT has grown in the past (see Figure 3 in Chapter 3) it can be seen that routers would get several years more operational time (in terms of sufficient memory space for FIBs) if FIBs would be two thirds smaller than they are. That is, a router running out of memory in 2002 would have had enough memory still in year 2010 and although DFRT growth speed has constantly accelerated, the prolongation would be almost as long today and certainly long enough to cause the router to be replaced for other reasons.

On the other hand, if FIB sizes are to be pushed low in every router, on the basis of these simulations, the use of popularity based VPs is not advisable. A need to do so might emerge for example if DFRT would be suddenly increased due to e.g. deployment of IPv6 and routers with large enough FIBs would not be available. To sum up, there are situations where popularity based VPs would seem better than the alternatives, but overall it may not give small enough FIBs, considering that shrinking FIBs is the reason VA could be deployed in the first place.

9.3 Reliability of the Simulations

As properly explained in Chapter 7, exact information about the used Sprint topology, its routing table or its traffic matrix was not available. Instead, incomplete measurements of the given network along with network independent statistics were used to simulate the environment. This puts the acquired absolute results in doubt as there are uncertainties in the simulation environment. However, by comparing the results shown in Chapter 8 with results gained from similar simulations elsewhere on a real data, the reliability of the former results can hopefully be increased.

The comparison is started from uniform VP allocation without PPs. In Ballani’s et al. paper [55] the maximum stretch value after which no further reduction in maximum FIB size could be achieved was 4 milliseconds. In our simulation the effect was similar, but the crucial point was on 6 milliseconds. This behavior is topology dependant and can well be explained with the different used topologies. With about one thousand VPs and six milliseconds maximum stretch latency, our simulations give about 80 percent maximum FIB size reduction, about 99 percent average FIB size reduction (in comparison to DFRT) and about 2.3 millisecond average packet stretch latency. Ballani et al. got four times larger maximum FIB, four times smaller average FIB and three times smaller stretch with 4 millisecond maximum stretch and 1024 VPs. Although the differences seem large and their results do not fit into the error margins of our study, there are many reasons that could possibly explain them (e.g. different topology, different applied maximum stretch limit, different APR allocation algorithm, usage of default routes in Ballani's et al. work) and most importantly the FIB size and stretch changes similarly, but with different absolute values. According to simulations on [55], the different topologies of AT&T and Sprint alone could explain half of the difference in maximum FIB sizes and the difference in average FIB sizes completely.

Similar behavior can also be seen with the use of popular prefixes. In Ballani's et al. simulations having only most popular prefixes as PPs (i.e. DPPS), the amount of traffic impacted by VA was reduced from 100 percents to almost ten percents with one percent of prefixes as PPs. In our case (depicted in Figure 24), only 82 percents of the traffic were impacted without PPs and 20 percents when one percent of PPs were allocated using IPPS. The most significant influence to the worse results is how PPs are used. Because of the lack in detailed information same PPs were used on whole ISP, while Ballani et al. could pick different PPs for different POPs. Additionally, topology, prefix set and VP allocations change the outcome of the simulations.

Shown that the results change similarly in both considered simulations when setup values are altered and that the differences could well be explained with a number of factors, it should be safe to say that although randomization had to be used to simulate many details, the results can be considered valid.

10. Conclusions

The foremost purpose of this thesis was to make the reader familiar with the challenges in Internet inter-domain routing and the remedy approaches Internet community has considered. This thesis started by finding out the main challenges and by considering what has so far been done to overcome them. In general, scalability and the difficulty to introduce changes were found to be the main themes around many separate problems; envisioning Internet a difficult future. When studying the most acute problem, i.e. routers running out of fast memory in the face of growing FIBs, it became evident that it is not affected only by the amount of Internet routes, but also by the operators' VPN customer routes. The criticality of the problem was also found to be less severe than many might think, because router vendors seem to be able to provide routers with large enough memories if their customers are willing to pay for it. The problem is really about how long ISPs should be able to use their routers and how quickly the memory size becomes the prohibiting factor. Several approaches to solve issues in Internet inter-domain routing were studied and three were introduced as well as briefly analyzed. None of the three were sawn to be significantly better than the others while they vary in effectiveness to solve problems and in deployment easiness.

As also promised, Virtual Aggregation as part of the easiest to deploy solution (Evolution) was examined rather deeply by elaborating its functionality, considering its pros and cons, formulating possible improvements, simulating its operation on a real-like environment and analyzing its ability to shrink FIBs while considering the most severe drawback: path stretch. The lack of available information about the details in ISPs network made the construction of the simulation environment challenging. However, by combining information from several sources, a real like environment could be constructed and meaningful results which are also comparable to a previous study could be achieved. After more or less repeating some of the simulations conducted in [55], it was feasible to evaluate how path stretch can be lowered by using directly routed popular prefixes and how the popularity based Virtual Prefix allocation scheme compares to other simulated operation modes with Virtual Aggregation.

Appendix A represents several key functions used to implement the simulations on Matlab. The example code shows how prefix tree was built, how the used FIB aggregation, uniform VP allocation and APR allocation algorithms were implemented. In total over four thousand lines on Matlab code were written to construct the simulation environment, to build the functionality and simulation cases and to capture and present the results.

Simulation results in Chapter 8 showed clear benefits in shrinking FIBs and thus proved Virtual Aggregation to be an efficient mechanism to lower routers' memory load. Results also showed that if a large part of the traffic is not routed via shortest paths (e.g. by giving traffic intensive prefixes special treatment), path stretch may become too severe; diminishing networks usefulness and increasing operators expenses. To actually lower operators' costs, VA needs to be configured so that it gives small enough FIBs on most of the routers while limiting stretch to a very low level, i.e. allowing an ISP to operate its network longer with existing equipment. The new popularity based Virtual Prefix allocation scheme described and considered in this thesis delivers just that, although, not as efficiently as when Virtual Prefixes are allocated uniformly and popular prefixes are routed via separate tunnels.

However, directly routed popular prefixes cannot be allocated as such without causing some packets to get dropped on AS border. This thesis illustrated that this can best be avoided by also including all the prefixes which popular prefixes cover in to the FIBs. The acquired path stretch reduction would then be only slightly lower than if FIBs would only include virtual and popular prefixes. Still, constant routing table monitoring and popular prefix set adjustments are required if the set of covered prefixes changes. Exactly how much overhead this would incur is unknown, but the effort of revising FIBs and rebuilding tunnels is not expected to be overwhelming if the existing network management tools are harnessed to do the job in a centralized fashion. Of course the overhead depends on how often such crucial changes happen.

With the knowledge gained during working with this thesis the previously described use of Virtual Aggregation which routes popular prefixes outside of the Virtual Prefixes is recommended if substantial reduction to most of the routers' FIB size is required. If the FIB shrinking need is mild, it is better to turn to FIB Aggregation which does not induce path stretch at all and is an easy to deploy router local solution. In fact, according to our simulations and the data available from the literature [49], FIB Aggregation can shrink FIBs considerably on many routers leaving only part of the routers to handle significantly large FIBs. Therefore, it is a little controversial whether the router memory issue will ever reach a point when FIB Aggregation would not be enough. Then yet again, it can be a cost reduction mechanism for operators if they can also buy new routers with less memory and have only few routers which can hold the entire DFRT and more.

To better respond to a situation where the maximum FIB should not be pushed as small as possible and the FIB load is rather concentrated on a few routers than distributed all over the topology, the way aggregation points are allocated should be revised. By looking at Figure 14 the situation would change so that there would be few routers rather close to the center of the topology with large FIBs, most of the routers with very small FIBs in the edge and some routers with medium sized FIBs only where needed to further limit the path stretch. Since the aggregation points would then be pushed quite far away from routers on average, Virtual Prefix allocations should also make it possible without high path stretch; that is, allocating Virtual Prefixes based on prefix popularity.

Further research should be done to understand exactly what kind of an aggregation point allocation algorithm would give the most favorable results. In retrospective, this is also something this thesis could have focused on since it could yield better results in terms of FIB load and path stretch and therefore increase the feasibility of VA. In fact, it is the likely path we are going to pursue in the near future. In this context the way VPs are formed becomes increasingly important motivating one also to think of better ways to allocate them. We are also considering how the ongoing IPv6 transition affects the feasibility of different aggregation techniques.

As shrinking FIBs is only a partial solution in truly improving Internet inter-domain routing scalability and issues in Internet routing in general, Virtual Aggregation and FIB Aggregation should not be used for an argument to delay larger modifications. Instead, more focus should be put on how the entire Internet architecture could be revised to something better which lacks many of the current issues. The optimality of the actual outcome is less important and may have to be sacrificed to some extent in hope for any major improvements to take place. The Evolution towards global routing scalability is a good proposal to start with, but further work is still required to have a highly convincing outline of the next big change in Internet architecture.

All in all, the evolution of Internet is a highly current topic in general and in research. Internet has established its irreplaceable position in people's everyday lives, played a big role in increasing globalization and become an infrastructure without which the society could not operate anymore. With popularity come also the responsibility and the need to be reliable. In that light, even though Internet is working reasonably well, several issues threaten to make the situation worse and prohibit further growth as well as the increase in reliability and feature richness. As remedy actions in Internet scale take several years at best and probably even longer, the work and decisions have to be made well in advance before any results are to be expected. The studies conducted for this thesis have been a part of that great endeavor by concentrating on both understanding the problems in detail and trying to develop mechanisms for short term improvements. A new algorithm for selecting aggregation points and a few notions about the Virtual Aggregation were the main contributions of this thesis.

References

[1] Miniwatts Marketing Group, Internet World Stats: Internet Usage Statistics. Online document. Updated on 30.6.2010. Cited on 28.1.2011. Available: wwww.internetworldstats.com/stats.htm.

[2] Postel, J. IETF RFC 791 - Internet Protocol. Online document. Updated in September 1981. Cited on 8.7.2011. Available: http://tools.ietf.org/html/rfc791.

[3] Fuller, V. and Li, T. IETF RFC 4632 - Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. Online document. Updated in August 2006. Cited on 7.8.2010. Available: http://tools.ietf.org/html/rfc4632.

[4] Deering, S. and Hinden, R. IETF RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification. Online document. Updated in December 1998. Cited on 8.7.2010. Available: http://tools.ietf.org/html/rfc2460.

[5] Srisuresh, P. and Egevang, K. IETF RFC 3022 - Traditional IP Network Address Translator (Traditional NAT). Online document. Updated in January 2001. Cited on 7.8.2010. Available: http://tools.ietf.org/html/rfc3022.

[6] Huston, G. The 32-bit AS number report. Online document. Updated on 28.1.2011. Cited on 28.1.2011. Available: http://www.potaroo.net/tools/asn32/.

[7] Rekhter, Y., Li, T. and Hares, S. IETF RFC 4271 - A Border Gateway Protocol 4 (BGP-4). Online document. Updated in January 2006. Cited on 8.7.2010. Available: http://tools.ietf.org/html/rfc4271.

[8] Bates, T., Chen, E. and Chandra, R. IETF RFC 4456 - BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). Online document. Updated in April 2006. Cited on 8.7.2010. Available: http://tools.ietf.org/html/rfc4456.

[9] Traina, P., McPherson, D. and Scudder, J. IETF RFC 5065 - Autonomous System Confederations for BGP. Online document. Updated in August 2007. Cited on 8.7.2010. Available: http://tools.ietf.org/html/rfc5065.

[10] Kim, C., Gerber, A., Lund, C., Pei, D. and Sen, S. Scalable VPN Routing via Relaying. *Proceedings of the ACM SIGMETRICS*, 2008.

[11] Meyer, D., Zhang, L. and Fall, K. IETF RFC 4984 - Report from the IAB Workshop on Routing and Addressing. Online document. Updated in September 2007. Cited on 8.7.2010. Available: http://tools.ietf.org/html/rfc4984.

[12] Narten, T. IETF Draft - On the Scalability of Internet Routing. Online document. Updated on 17.2.2010. Cited on 8.7.2010. Available: http://tools.ietf.org/html/draft-narten-radir-problem-statement-05.

[13] Scudder, J. Router Scaling Trends. APRICOT Future of Routing workshop. Updated on 27.2.2007. Cited in 8.7.2010. Available: http://submission.apricot.net/chatter07/slides/future_of_routing/apia-future-routing-john-scudder.pdf.

[14] Arkko, J. Solving the Routing Scalability Problem --The Hard Parts. *APRICOT Future of Routing workshop*. Updated on 27.2.2007. Cited on 8.7.2010. Available: http://submission.apricot.net/chatter07/slides/future_of_routing/apia-future-routing-jari-arkko.pdf.

[15] Hardin, G. The Tragedy of the Commons. *Journal of Natural Resources Policy Research*, 2009, vol. 1, issue 3, pages 243-253.

[16] Huston, G. IPv4 Address Report. Online document. Updated on 28.1.2011. Cited on 28.1.2010. Available: http://www.potaroo.net/tools/ipv4/.

[17] Huston, G. BGP in 2009. *Asia Pacific Regional Internet Conference on Operational Technologies*. 2010. Cited on 2.2.2011. Available: http://www.potaroo.net/presentations/2010-03-04-bgp2009.pdf.

[18] Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J. and Jahanian, F. Internet inter-domain traffic. *ACM SIGCOMM computer communications review*, 2010, vol. 40, issue 4, pages 75-86.

[19] Abley, J., Lindqvist, K., Davies, E., Black, B and Gill, V. IETF RFC 4166 - IPv4 Multihoming Practices and Limitations. Online document. Updated in July 2005. Cited on 2.2.2011. Available: http://tools.ietf.org/html/rfc4116.

[20] Huston, G. AS6447 - BGP Table Analysis Report. Online document. Updated on 2.2.2011. Cited on 2.2.2011. Available: http://bgp.potaroo.net/as6447/.

[21] Carpenter, B., Crowcroft, J. and Rekhter, Y. IETF RFC 2101 - IPv4 Address Behaviour Today. Online document. Updated in February 1997. Cited on 2.2.2011. Available: http://tools.ietf.org/html/rfc2101.

[22] Francis, P. and Xu, X. Extending Router Lifetime with Virtual Agregation. *The Internet Protocol Forum*. Online document. Updated in April 2010. Cited on 2.2.2011. Available: http://www.ipjforum.org/?p=255.

[23] Meng, X., Xu, Z., Zhang, B., Huston, G., Lu, S. and Zhang, L. IPv4 Address Allocation and the BGP Routing Table Evolution. *ACM computer communications review*, 2005, vol. 35, issue 1, pages 71-80.

[24] Ballani, H. Harnessing tunnels for dirty-slate network solutions. Doctoral dissertation, Cornell University, 2009.

[25] Moore, G., E. Cramming more components onto integrated circuits. *Electronics*, 1965, vol. 38, issue 8.

[26] Li, T. Router scalability and Moore's law. *IAB workshop on the future of routing and addressing in the Internet*, 2006. Cited 2.2.2011. Available: http://www.iab.org/about/workshops/routingandaddressing/Router_Scalability.pdf .

[27] Fall, K., Iannaccone, G., Ratnasamy, S. and Godfrey, P. B. Routing Tables: Is Smaller Really Much Better? *Eighth ACM workshop on Hot Topics in Networks*, 2009. Cited on 2.2.2011. Available: http://berkeley.intel-research.net/sylvia/hotnets2009-final156.pdf.

[28] Huston, G. The BGP Instability Report. Online Document. Updated on 18.2.2011. Cited on 18.2.2011. Available: http://bgpupdates.potaroo.net/instability/bgpupd.html.

[29] Varadhan, K., Govindan, R. and Estrin, D. Persistent route oscillations in interdomain routing. *Computer Networks*, 2000, Vol. 32, issue 1, pages 1-16.

[30] Labovits, C., Ahuja, A., Bose, A. and Jahanian, F. Delayed Internet Routing Convergence. *IEEE/ACM Transactions on Networking*, 2001, vol. 9, issue 3, pages 293-306.

[31] Mao, Z. M., Govindan, R., Varghese, G. and Katz, R. H. Route flap damping exacerbates internet routing convergence. *ACM SIGCOMM computer communications review*, 2002, vol. 32, issue 4, pages 221-233.

[32] Jakma, P. IETF draft - Revised Default Values for the BGP 'Minimum Route Advertisement Interval'. Online document. Updated in November 2008. Cited on 4.8.2010. Available: http://tools.ietf.org/html/draft-jakma-mrai-00.

[33] Villamizar, C., Chandra, R. and Govindan, R. IETF RFC 2439 - BGP Route Flap Damping. Online document. Updated in November 1998. Cited on 2.2.2011. Available: http://www.ietf.org/rfc/rfc2439.txt.

[34] Smith, P. and Panigl, C. RIPE Routing Working Group Recommendations on Route-flap Damping. Online document. Updated in May 2006. Cited on 4.8.2010. Available: http://www.ripe.net/ripe/docs/ripe-378.html.

[35] Labovits, C., Ahuja, A., Wattenhofer, R. ja Venkatachary, S. The impact of Internet Policy and Topology on Delayed Routing Convergence. *Proceedings of the IEEE INFOCOM*, 2001.

[36] Kent, S. and Seo, K. IETF RFC 4301 - Security Architecture for the Internet Protocol. Online document. Updated in December 2005. Cited on 18.2.2011. Available: http://tools.ietf.org/html/rfc4301.

[37] Toonk, A. Chinese BGP hijack, putting things into perspective. Online document. Updated 21.11.2010. Cited 3.12.2010. Available: http://bgpmon.net/blog/?p=323.

[38] Perkins, C. IETF RFC 5944 - IP Mobility Support for IPv4, Revised. Online document. Updated in November 2010. Cited on 2.2.2011. Available: http://tools.ietf.org/html/rfc5944.

[39] Johnson, D., Perkins, C. and Arkko, J. IETF RFC 3775 - Mobility Support in IPv6. Online document. Updated in June 2004. Cited on 2.2.2011. Available: http://tools.ietf.org/html/rfc3775.

[40] Devarapalli, D., Wakikawa, R., Petrescu, A. and Thubert, P. IETF RFC 3963 - Network Mobility (NEMO) Basic Support Protocol. Online document. Updated in January 2005. Cited on 2.2.2011. Available: http://tools.ietf.org/html/rfc3963.

[41] Baliga, J., Hinton, K. and Tucker, R.S. Energy Consumption of the Internet. *Proceedings of the COIN-ACOFT*, 2007.

[42] Namiki, S., Hasama, T., Mori, M., Watanabe, M. and Ishikawa, H. Dynamic Optical Path Switching for Ultra-Low Energy Consumption and Its Enabling Device Technologies. *Proceedings of the SAINT*, 2008.

[43] Li, T. Recommendation for a Routing Architecture. Online document. Updated on 29.11. 2010. Cited on 1.12.2010. Available: http://datatracker.ietf.org/doc/draft-irtf-rrg-recommendation/?include_text=1.

[44] Farinacci, D., Fuller, V., Meyer, D. and Lewis, D. IETF draft - Locator/ID Separation Protocol (LISP). Online document. Updated in October 2010. Cited in 1.12.2010. Available: http://tools.ietf.org/html/draft-ietf-lisp-09.

[45] Fuller, V., Farinacci, D., Meyer, D. and Lewis, D. IETF draft - LISP Alternative Topology (LISP+ALT). Online document. Updated in April 2010. Cited on 1.12.2010. Available: http://tools.ietf.org/html/draft-ietf-lisp-alt-04.

[46] Flinck, H. Compact routing in locator identifier mapping system. Online document. Cited on 2.2.2011. Available: http://www.tschofenig.priv.at/rrg/CR_mapping_system_0.1.pdf.

[47] Letong, S., YinXia, ZhiLiang, W. and Jianping, W. A Layered Mapping System For Scalable Routing. Online document. Cited on 2.2.2011. Available: https://docs.google.com/fileview?id=0BwsJc7A4NTgeOTYzMjFlOGEtYzA4OC00NTM0LTg5ZjktNmFkYzBhNWJhMWEy&hl=en.

[48] Zhang, B., Zhang, L. and Wang, L. IETF draft - Evolution Towards Global Routing Scalability. Online document. Updated in October 2009. Cited on 2.2.2011. Available: http://tools.ietf.org/html/draft-zhang-evolution-02.

[49] Zhang, B., Wang, L., Zhao, X., Liu, U. and Zhang, L. IETF draft - FIB Aggregation. Online document. Updated in October 2009. Cited on 2.2.2011. Available: http://tools.ietf.org/html/draft-zhang-fibaggregation-02.

[50] Francis, P., Xu, X., Ballani, H., Jen, D., Raszuk, R and Zhang, L. IETF draft - FIB Suppression with Virtual Aggregation. Online document. Updated on 31.8.2010. Cited on 2.2.2011. Available: http://tools.ietf.org/search/draft-ietf-grow-va-03.

[51] Meyer, D., Lewis, D. and Farinacci, D. IETF draft - LISP Mobile Node. Online document. Updated on 25.10.2010. Cited on 2.2.2011. Available: http://tools.ietf.org/html/draft-meyer-lisp-mn-04.

[52] Arends, R., Austein, R., Larson, M., Massey, D. and Rose, S. IETF RFC 4033 - DNS Security Introduction and Requirements. Online document. Updated in March 2005. Cited on 18.2.2011. Available: http://tools.ietf.org/html/rfc4033.

[53] Moskowitz, R. and Nikander, P. IETF RFC 4423 - Host Identity Protocol (HIP) Architecture. Online document. Update in May 2006. Cited on 18.2.2011. Available: http://tools.ietf.org/html/rfc4423.

[54] Francis, P., Xu, X., Ballani, H., Raszuc, R. and Chang, L. IETF Draft - Simple Virtual Aggregation (S-VA). Online document. Updated on 31.8.2010. Cited on 2.2.2011. Available: http://tools.ietf.org/html/draft-ietf-grow-simple-va-01.

[55] Ballani, H., Francis, P., Cao, T. and Wang, J. Making Routers Last Longer with ViAggre. *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, 2009.

[56] Fang, W., Peterson, L. Inter-AS traffic patterns and their implications. *Proceedings of the GLOBECOM*, 1999.

[57] Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J. and True, F. Deriving traffic demands for operational IP networks: methodology and experience. *IEEE/ACM Transactions on Networking*, 2001, vol. 9, issue 3, pages 265-280.

[58] Rexford, J., Wang, J., Xiao, Z. and Zhang, Y. BGP routing stability of popular destinations. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, 2002.

[59] Taft, N., Bhattacharyya, S., Jetcheva, J. and Diot, C. Understanding traffic dynamics at a backbone POP. *Proceedings of the SPIE* 4526, 150, 2001.

[60] Broido, A., Hyun, Y., Gao, R. and Claffy, kc. Their Share: Diversity and Disparity in IP Traffic. Lecture Notes in Computer Science, 2004, vol. 3015, pages 113-125.

[61] Anderson, T., Mahajan, R., Spring, N. and Wetherall, D. Rocketfuel: An ISP Topology Mapping Engine. Online document. Cited on 2.2.2011. Available: http://www.cs.washington.edu/research/networking/rocketfuel/.

[62] Advanced Network Technology Center, University of Oregon. University of Oregon Route Views Project . Online document. Cited on 2.2.2011. Available: http://www.routeviews.org/.

[63] Team cymru. IP to ASN Mapping. Online ducument. Cited on 2.2.2011. Available: http://www.team-cymru.org/Services/ip-to-asn.html.

[64] Chang, H., Jamin, S., Mao, Z. M. and Willinger, W. An Empirical Approach to Modeling Inter-AS Traffic Matrices. *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 2005.

Appendix A

Simulations described in this thesis were conducted using Matlab. This appendix presents Matlab code that implements some main sub-tasks used in the simulations.

Prefix tree build up

```
function create_prefix_tree

% Creates a prefix tree following the set of popularPrefixes. The tree
% consist of every node from those prefixes to the root, i.e. longer prefixes than
% the corresponding popular prefixes are not inserted in to the tree in order to save
% memory and processing.

global popularPrefixes
global prefixCells
global CoveredNHs

% Slice the popularPrefixes matrix into 24 cell columns where rows are prefixes
% with the specific slash value (e.g. /16). Start from /24 and go through
% the lists in order. For each prefix, see if it already has a parent
% (binary search). If no, add it. Then update the popularity value
% (cumulative) and set this node to be its child (determine if it is left
% or right). When all matrices have been traversed, traverse the three from
% root to leafs (/n matrix at a time) and mark the parents (cannot be done
% sooner, because parent matrices are sorted and indexes don't therefore
% hold).

% - Format in /n matrices is the following:
% column 1=ASN 2=prefix_int 3=prefix_len 4=popularity 5=parent 6=left_child
% 7=right_child
% - Encoding in the child and parent info is the following:
% (cell,row) = cell + 100*row (e.g. cell 18, row 7 = 718)

% CoveredNHs is a cell array of cell arrays which holds information about
% the nodes (similarly as in prefixCells, i.e. (2,3) = CoveredNHs{2}{3}. An
% integer matrix is in each inner cell and has the format: column 1:NH_ASN
% 2:number of covered hits 3:NHs cumulative popularity

% Create the cell array (correct to be able to support longer than /24s)
prefixCells = cell(1,25);

% Create the cell structure for inner NH cells
CoveredNHs = cell(1,25);

% Count how many lines in each matrix
count = zeros(1,25);
for i = 1:size(popularPrefixes,1)
    count(popularPrefixes(i,3) + 1) = count(popularPrefixes(i,3) + 1) + 1;
end

% Initialize the /n matrices and inner NH cells
for i = 1:25
    prefixCells{i} = zeros(count(i),7,'uint32');
end

% Copy the lines to correct cell matrix
lineCounter = ones(1,25);
for i = 1:size(popularPrefixes,1)
    prefixCells{popularPrefixes(i,3) + 1}(lineCounter(popularPrefixes(i,3) + 1),1:4)
= popularPrefixes(i,:);
    lineCounter(popularPrefixes(i,3) + 1) = lineCounter(popularPrefixes(i,3) + 1) +
1;
end
```

```
% Create the prefixTree
% Go through the matrices in order (/24 -> /0)
for i = 1:length(prefixCells)
    currentCellNro = length(prefixCells) - i + 1;

    % Sort the current matrix according to prefix_int
    [l, k] = sort(prefixCells{currentCellNro}(:,2));
    prefixCells{currentCellNro} = prefixCells{currentCellNro}(k,:);

    % Go through the current matrix
    [rows, h] = size(prefixCells{currentCellNro});
    CoveredNHs{currentCellNro} = cell(rows,1);
    for j = 1:rows

        % if not at root, do the following
        if(currentCellNro ~= 1)
            % Determine the parent
            parent_int = bitshift(prefixCells{currentCellNro}(j,2), -int8(32 - (cur-
rentCellNro - 2)));
            parent_int = bitshift(parent_int, int8(32 - (currentCellNro - 2)));

            % Sort the next matrix, i.e. where parents lie
            [l, k] = sort(prefixCells{currentCellNro - 1}(:,2));
            prefixCells{currentCellNro - 1} = prefixCells{currentCellNro - 1}(k,:);

            % See if parent prefix already exists
            parent_index = binarySearch(prefixCells{currentCellNro -
1}(:,2),parent_int);
            if(parent_index == 0)

                % If not, create it and put it to the list
                parent = zeros(1,7,'uint32');
                parent(2) = parent_int;
                parent(3) = prefixCells{currentCellNro}(j,3) - 1;
                prefixCells{currentCellNro - 1}(end + 1,:) = parent; % SLOW!!! Copies
all nodes to a new matrix! Too slow?
                [parent_index, l] = size(prefixCells{currentCellNro - 1});
            end

            % Determine wether this prefix is parents left or right child
            % and update parents child info accordingly
            if(prefixCells{currentCellNro}(j,2) == parent_int) % left child
                prefixCells{currentCellNro - 1}(parent_index,6) = currentCellNro +
100 * j;
            else % right child
                prefixCells{currentCellNro - 1}(parent_index,7) = currentCellNro +
100 * j;
            end
        end

%Part of the FIB aggregation 4B algorithm
        % Update covered Next-hop ASN count & ASN wise covered
        % cumulative popularity and store it in a different structure.
        % The format in this structure is the following: 1=ASN
        % 2=ANS_count 3=ASN_cumulative_popularity.
        % For each prefix, copy left childs matrix if not empty and in
        % case it is, copy right childs matrix. If both are empty,
        % create new from current prefix's information. To make the
        % matrix complete, merge information from right childs matrix
        % (if neither childs matrix is empty) and from the current
```

```
        % left child exists
        if(prefixCells{currentCellNro}(j,6) ~= 0)
            leftChildCell = rem(prefixCells{currentCellNro}(j,6),100);
            leftChildRow = fix(prefixCells{currentCellNro}(j,6)/100);

            % Left child as the basis
            CoveredNHs{currentCellNro}{j} = CoveredNHs{leftChildCell}{leftChildRow};

            % Add info from right child if it exists
            if(prefixCells{currentCellNro}(j,7) ~= 0)
                rightChildCell = rem(prefixCells{currentCellNro}(j,7),100);
                rightChildRow = fix(prefixCells{currentCellNro}(j,7)/100);
                [a,b] = size(CoveredNHs{rightChildCell}{rightChildRow});
                for c = 1:a
                    % Should binary search be used instead of find
                    % (would have to sort after each add)
                    index = find(CoveredNHs{currentCellNro}{j}(:,1) == Cove-
redNHs{rightChildCell}{rightChildRow}(c,1));
                    if(index > 0)
                        CoveredNHs{currentCellNro}{j}(index,2) = Cove-
redNHs{currentCellNro}{j}(index,2) + CoveredNHs{rightChildCell}{rightChildRow}(c,2);
                        CoveredNHs{currentCellNro}{j}(index,3) = Cove-
redNHs{currentCellNro}{j}(index,3) + CoveredNHs{rightChildCell}{rightChildRow}(c,3);
                    else
                        CoveredNHs{currentCellNro}{j}(end + 1,1) = Cove-
redNHs{rightChildCell}{rightChildRow}(c,1);
                        CoveredNHs{currentCellNro}{j}(end,2) = Cove-
redNHs{rightChildCell}{rightChildRow}(c,2);
                        CoveredNHs{currentCellNro}{j}(end,3) = Cove-
redNHs{rightChildCell}{rightChildRow}(c,3);
                    end
                end
            end

        % No left child. Right child exists
        elseif(prefixCells{currentCellNro}(j,7) ~= 0)
            rightChildCell = rem(prefixCells{currentCellNro}(j,7),100);
            rightChildRow = fix(prefixCells{currentCellNro}(j,7)/100);

            % Right child as the basis
            CoveredNHs{currentCellNro}{j} = Cove-
redNHs{rightChildCell}{rightChildRow};
        end

        % If real prefix, add its ASN hit and per ASN popularity
        if(prefixCells{currentCellNro}(j,1) ~= 0)
            if(isempty(CoveredNHs{currentCellNro}{j}) ~= 1)
                index = find(CoveredNHs{currentCellNro}{j}(:,1) == prefix-
Cells{currentCellNro}(j,1));
                if(index > 0)
                    CoveredNHs{currentCellNro}{j}(index,2) = Cove-
redNHs{currentCellNro}{j}(index,2) + 1;
                    CoveredNHs{currentCellNro}{j}(index,3) = Cove-
redNHs{currentCellNro}{j}(index,3) + prefixCells{currentCellNro}(j,4);
                end
            else
                CoveredNHs{currentCellNro}{j}(end + 1,1) = prefix-
Cells{currentCellNro}(j,1);
                CoveredNHs{currentCellNro}{j}(end,2) = 1;
                CoveredNHs{currentCellNro}{j}(end,3) = prefix-
Cells{currentCellNro}(j,4);
            end
        end
```

```
        if(currentCellNro ~= 1)
            %Sort the matrix where parent lies so that next parent search
            %can be done (not in root)
            [l, k] = sort(prefixCells{currentCellNro - 1}(:,2));
            prefixCells{currentCellNro - 1} = prefixCells{currentCellNro - 1}(k,:);
        end
    end
end

% Traverse the tree level by level from root to parents of leafs
for i = 1:length(prefixCells) - 1

    % Go through every prefix in the current level
    [rows, h] = size(prefixCells{i});
    for j = 1:rows

        % Mark the parent according to child info
        child1Cell = rem(prefixCells{i}(j,6),100);
        child2Cell = rem(prefixCells{i}(j,7),100);

        % Left child
        if(child1Cell ~= 0)
            child1Row = fix(prefixCells{i}(j,6) / 100);
            prefixCells{child1Cell}(child1Row,5) = i + (100 * j);
        end

        % Right child
        if(child2Cell ~= 0)
            child2Row = fix(prefixCells{i}(j,7) / 100);
            prefixCells{child2Cell}(child2Row,5) = i + (100 * j);
        end
    end
end
end
```

FIB Aggregation

```
function FIB_Aggregation_4B ()

global prefixCells
global CoveredNHs

% First half of the algorithm is integrated with prefix tree build up
% All variables and explanations are for NH-ASN based solution, but it
% works also for regular FIB aggregation where each router calculated its
% FIB locally and NH router id is there used in stead of NH-ASN.
% For all prefixes in the tree: in case non-real prefix, mark ASN according
% to the one with most covered Next-hop ASN hits. Although the algorith
% desciption in draft-zhang-fibaggregation-02 tells to go through the tree
% recursively in postorder, CoveredNHs already has all the neccessary
% information and all nodes can thus be traversed iteratively prefix length
% level at a time.

for i = 1:length(prefixCells)
    for j = 1:size(prefixCells{i},1)

        % Parent ASN
        parentASN = 0;
        temp = get_closest_marked_parent(i + (100 * j));
        if(temp ~= 0)
            parentASN = prefixCells{rem(temp, 100)}(fix(temp / 100),1);
        end

        % If not a real prefix, calculate new ASN
        if(prefixCells{i}(j,1) == 0)

            % Set prefixes NH-ASN to be Next-hop ASN with most hits. If
            % more than one with the same #hits, randomise the selection.
            [l, k] = sort(CoveredNHs{i}{j}(:,2), 'descend');
            last = 1;
            while(last < size(CoveredNHs{i}{j},1) && CoveredNHs{i}{j}(k(last),2) ==
CoveredNHs{i}{j}(k(last + 1),2))
                last = last + 1;
            end

            % Pick the candidate new ASN
            newASN = CoveredNHs{i}{j}(k(ceil(rand * last)),1);

            % If none of the most popular ASNs equals to the parent ASN,
            % new ASN should be inserted
            for m = 1:last
               if(CoveredNHs{i}{j}(k(m),1) == parentASN)
                   newASN = 0;
               end
            end

            % If real prefix, ANS is old ASN or 0
        else
            if(prefixCells{i}(j,1) == parentASN)
                newASN = 0;
            else
                newASN = prefixCells{i}(j,1);
            end
        end

        % Set the new ASN
        prefixCells{i}(j,1) = newASN;
    end
end
```

Uniform VP Allocation

```
function select_VPs()

% Uniform Allocation: Partitions the address space into /7
% prefixes (Virtual Prefixes) and slices them up so that no single VP covers
% more than 2 percentage of all the real prefixes. VPCovers info is
% calculated simultaneously as it is used to indicate the prefixes that
% belong to yet too large VPs in each iteration. Such VPs are split untill
% all VPs are below the allowed size or a real prefix would be used (only
% possible when allowed size <= 1).

global ASPrefixes
global VPList
global prefixCells
global simulationMode
global VPCovers
global maxNroOfPrefixesInVP
global VPPopASNs

% Uniform Allocation
if(simulationMode == 2)

    % Initiate variables
    VPList = zeros(128,2,'uint32');
    VPCovers = cell(128,1);

    % Put /7s to the VPList
    for i = 1:128
       VPList(i,1) = 2 * (i - 1) * (256 ^ 3);
       VPList(i,2) = 7;
    end

    % Split VPs untill no VP countains more than maxNroOfPrefixesInVP of
    % the BGP routing table with the exception that all VPs must be shorter
    % than real prefixes.

    % Maximum size for a VP
    maxVPSize = ceil(length(ASPrefixes) * maxNroOfPrefixesInVP);

    % Update the VPCovers
    for i = 1:length(ASPrefixes)
        found = 0;
        j = 1;
        while(found == 0 && j <= length(VPList))
            if(prefixCover(VPList(j,:),[ASPrefixes(i,2),ASPrefixes(i,3)]) > 0)

                % A cell array of pointers (indexes) to prefixes in
                % ASPrefixes
                VPCovers{j}(end + 1,1) = uint32(i);
                found = 1;
            end
            j = j + 1;
        end
    end
```

```
    % Split VPs if < maxNroOfPrefixesInVP && < preflen(real_pref_in_VP)
    modifiedIndexes = 1:length(VPList);
    while(~isempty(modifiedIndexes))
        tempIndexes = [];
        for i = 1:length(modifiedIndexes)
            if(length(VPCovers{modifiedIndexes(i)}) > maxVPSize && VPL-
ist(modifiedIndexes(i),2) < min(ASPrefixes(VPCovers{modifiedIndexes(i)},3)) - 1)

                % Correct old VP to be the first half
                VPList(modifiedIndexes(i),2) = VPList(modifiedIndexes(i),2) + 1;
                tempCovers = cell(2,1);
                for j = 1:length(VPCovers{modifiedIndexes(i)})
                    if(prefixCover(VPList(modifiedIndexes(i),:), [ASPrefix-
es(VPCovers{modifiedIndexes(i)}(j),2), ASPrefix-
es(VPCovers{modifiedIndexes(i)}(j),3)]) > 0)
                        tempCovers{1}(end + 1,1) = VPCovers{modifiedIndexes(i)}(j);
                    end
                end
                tempIndexes(end + 1) = modifiedIndexes(i);

                % Create new VP (second half)
                VPList(end + 1,1) = VPList(modifiedIndexes(i),1) + 2 ^(32 - VPL-
ist(modifiedIndexes(i),2));
                VPList(end,2) = VPList(modifiedIndexes(i),2);
                for j = 1:length(VPCovers{modifiedIndexes(i)})
                    if(prefixCover(VPList(end,:), [ASPrefix-
es(VPCovers{modifiedIndexes(i)}(j),2), ASPrefix-
es(VPCovers{modifiedIndexes(i)}(j),3)]) > 0)
                        tempCovers{2}(end + 1,1) = VPCovers{modifiedIndexes(i)}(j);
                    end
                end

                VPCovers{modifiedIndexes(i)} = tempCovers{1};
                VPCovers{end + 1,1} = tempCovers{2};
                tempIndexes(end + 1) = length(VPCovers);
            end

        end
        modifiedIndexes = tempIndexes;
        %fprintf('made a round\n');
    end

    %Remove empty VPs
    i = 1;
    while(i <= size(VPList,1))
        if(isempty(VPCovers{i}))
            VPList(i,:) = [];
            VPCovers(i,:) = [];
        else
            i = i + 1;
        end
    end
end
end
```

APR Allocation

```
% Sort VPList, VPCovers and nroCovered in decending order according to the
% number of covered real prefixes.
nroCovered = zeros(nroOfVPs,1,'uint32');
for i = 1:nroOfVPs
    nroCovered(i,1) = length(VPCovers{i});
end
[l, k] = sort(nroCovered,'descend');
VPList = VPList(k,:);
VPCovers = VPCovers(k,:);
if(simulationMode == 3)
    VPPopASNs = VPPopASNs(k,:);
end
nroCovered = nroCovered(k,:);

% Create VP, ASN, id structure
create_VP_ASN_id_structure;

% Undirect external routes per VP
undirectExtRoutes = zeros(nroOfVPs, nroOfIds,'uint32');
for vp = 1:nroOfVPs
    for id = 1:nroOfIds
        undirectExtRoutes(vp,id) = length(union(VPCovers{vp}, directExtPrefix-
es{id}));
    end
end
fprintf('undirectExtRoutes formed at ');
fprintf('%i ',fix(clock));
fprintf('\n');

% Splitting the VAP case in smaller parts to lower the memory need (>20GB
% otherwise with 15% VPs)
divisor = nroOfVPs;
VPSetRound = 1;
firstVP = 1;
lastVP = 0;
served = cell(0);
servable = cell(0);
servingAPRs = zeros(nroOfIds, nroOfVPs,'uint16');
while(lastVP < nroOfVPs)
% ---------- Can serve ----------

    if(VPSetRound == 1)
        lastVP = fix(nroOfVPs / divisor);
    elseif(VPSetRound == divisor)
        firstVP = fix(nroOfVPs / divisor) * (VPSetRound - 1) + 1;
        lastVP = nroOfVPs;
    else
        firstVP = fix(nroOfVPs / divisor) * (VPSetRound - 1) + 1;
        lastVP = fix(nroOfVPs / divisor) * VPSetRound;
    end

    % Create a datastructure for can serve information
    servable = cell(nroOfIds,(lastVP - firstVP + 1));

    % Create a datastructure for served routers (ids per vp)
    served = cell(nroOfIds,(lastVP - firstVP + 1));
```

```
% VAD can serve
if(simulationMode == 2)

    % Try out all the router ids as APRs
    for id = 1:nroOfIds

        % Go through every VP
        for vp = firstVP:lastVP%1:nroOfVPs

            % Calculate a list of router ids that can be served with id(id) and
            % VP(vp).
            servable{id,vp - firstVP + 1} = can_serve_VAD(id,vp);
        end
    end

% VAP can serve
elseif(simulationMode == 3)

    % Initialize temporary variable to store each ids possible APRs in each
    % VP
    possibleAPRs = cell(nroOfIds,(lastVP - firstVP + 1));

    % Go through every VP
    for vp = firstVP:lastVP

        % Try out all the routers and find out which ids can serve it as an APR
        for id = 1:nroOfIds
            possibleAPRs{id,vp - firstVP + 1} = can_be_served_by_VAP(id,vp);
        end
    end

    % Create a reverse structure of possible APRs (== servable)
    for id = 1:nroOfIds
        for vp = firstVP:lastVP
            for i = 1:length(possibleAPRs{id,vp - firstVP + 1})
                servable{possibleAPRs{id,vp - firstVP + 1}(i),vp - firstVP + 1}(end +
1) = id;
            end
        end
    end
end

% --------- //Can serve ---------

% Go through every VP (in order)
for vp = firstVP:lastVP

    servedIds = [];
    servingRounds = 0;

    % Repeat all the rest untill all routers are served for the VP(vp)
    allServed = 0;
    while(allServed == 0)

        allChecked = 0;
        servingRounds = servingRounds + 1;
```

```
        % Sort router ids in decreasing order of nro of unserved entries in
        % servable
        nroOfEntries = zeros(nroOfIds,1,'uint16');
        for id = 1:nroOfIds
            nroOfEntries(id,1) = sum(ismember(servable{id,vp - firstVP + 1}, serve-
dIds) == 0);
        end
        [l, k] = sort(nroOfEntries,'descend');
        orderedIds = ids(k,:);

        % Go through every id following the above sorting and try to insert
        % them as APRs for the VP(vp)
        for i = 1:nroOfIds
            id = orderedIds(i);

            if(FIBSizes(id) + undirectExtRoutes(vp,id) <= worstFIBSize && ismemb-
er(vp,APRs{id}) == 0 && nroOfEntries(id) > 0)

                % Add id as APR for vp
                APRs{id}(end + 1) = vp;

                % Mark all routers in servable{id,vp} as served
                served{id,vp - firstVP + 1} = setdiff(servable{id,vp - firstVP + 1},
servedIds);
                servedIds = union(servedIds, servable{id,vp - firstVP + 1});

                % Increase ids FIB size
                FIBSizes(id) = FIBSizes(id) + undirectExtRoutes(vp,id);
                break;
            end

            if(i == nroOfIds)
                allChecked = 1;
            end
            if(length(servedIds) == nroOfIds)
                allServed = 1;
                break;
            end
        end

        % All routers are still not served with current worstFIBSize
        if(allServed == 0 && allChecked == 1)

        % With id as new APR, will it serve any new ids?
            for i = 1:nroOfIds
                id = orderedIds(i);
                if(ismember(vp,APRs{id}) == 0)
                    APRs{id}(end + 1) = vp;
                    served{id,vp - firstVP + 1} = setdiff(servable{id,vp - firstVP +
1}, servedIds);
                    servedIds = union(servedIds, servable{id,vp - firstVP + 1});
                    %newFIBRoutes = union(VPCovers{vp}, directExtPrefixes{id});
                    FIBSizes(id) = FIBSizes(id) + undirectExtRoutes(vp,id);
                    %temp = sort(FIBSizes);
                    %worstFIBSize = temp(round(0.99 * length(FIBSizes)));
                    worstFIBSize = FIBSizes(id);
                    break;
                end
            end
        end

        if(length(servedIds) == nroOfIds)
            allServed = 1;
            break;
        end
    end
end
```

```matlab
function [servingIds] = can_be_served_by_VAP (id, VP)

% Gives the set of servable ids with the given id as the APR for the given
% VP. Whether some node can be served, is determined by the topology and
% the stretch limit maxStretchLatency.

global sPaths
global SPCosts
global maxStretchLatency
global VPASNs
global nearestASBRs
global SPsA2ASBR
global VPPopASNs
global ASNList
global ASBRList

% For given node(id) determine which nodes can serve it as an APR in the
% given VP(vp). I.e. Which APRs are on the path from the node to ASN-NHs
% ASBR and so that stretch for other ASNs ASBR does not violate the stretch
% constraint. Returns a list of ids that can serve id as an APR.

servingIds = zeros(0,'uint16');

% Get the set of ids that are on the path
ASBR = nearestASBRs(id,ASNList == VPPopASNs(VP),2);
onPathIds = sPaths{id,ASBR}; %SPsA2ASBR{id,ASBRList == ASBR};

% From on-the-path ids, take only those that don't end up violating the
% stretch constraint to other ASNs closest ASBRs.

% To APR latencies
toAPRLatencies = SPCosts(id, onPathIds);

% Direct latencies
directLatencies = nearestASBRs(id, VPASNs{VP});

for i = 1:length(onPathIds)
    apr = onPathIds(i);

    % From APR latencies with apr
    fromAPRLatencies = nearestASBRs(apr, VPASNs{VP});

    % If too much stretch to one of the ASs, can't be an APR, otherwise can
    if(toAPRLatencies(i) + fromAPRLatencies - directLatencies <= maxStretchLatency)
        servingIds(end + 1) = apr;
    end
end
end
```

```
function [servableIds] = can_serve_VAD (id, VP)

% Gives the set of servable ids with the given id as the APR for the given
% VP. Whether some node can be served, is determined by the topology and
% the stretch limit maxStretchLatency.

global ids
global VPList
global adjMtrx
global sPaths
global SPCosts
global maxStretchLatency
global VPASNs
global nearestASBRs

% For each node determine if given node(id) can serve as an APR for given
% VP (does not violate stretch constraint). Return a list of ids that can
% be server.

servableIds = zeros(0,'uint16');

% Determine the shortest latencies from apr(id) to nearest ASBRs of each AS
fromAPRLatencies = nearestASBRs(id,VPASNs{VP});

for i = 1:length(ids)

    % Determine the direct latencies from i to nearest ASBRs of each AS
    directLatencies = nearestASBRs(i,VPASNs{VP});

    if(SPCosts(i,id) + fromAPRLatencies - directLatencies <= maxStretchLatency)
        servableIds(end + 1,1) = i;
    end
end
end
```