PSP: Path State Protocol for Inter-Domain Routing

M.Sc. Thesis Proposal

Submitted by: **Mr. Dotan Guy**

Advisors: **Dr. Chen Avin, Prof. Ran Giladi**

Department of Communication Systems Engineering

Faculty of Engineering

Ben-Gurion University of the Negev

December 29, 2010

"If we knew what it was we were doing, it would not be called research, would it?"

Albert Einstein

Abstract

Link-state routing protocols are known to be robust and to support shortest path routing, but they do not support policy-based routing and suffer from scalability problems. Therefore, traditionally, link state protocols are used for intra-domain routing while path (or distance) vector protocols, such as BGP, are used for inter-domain routing where policies and scale are dominant. In this thesis we present PSP, a path state protocol for policy-based inter-domain routing for a network of Autonomous Systems (ASes) such as the Internet. A path state protocol is an extended link-state routing protocol that enables both link-state and path-state advertisements (of path costs). PSP supports policy-based routing similar to BGP and takes into account the Internet-like policies that are based on business relations between ASes. PSP can run on arbitrary multi-region (i.e., hierarchies, areas) networks that reduce global traffic and improve scalability. We provide two versions of the protocol: SLP (Shortest Legal Paths), which guarantees forwarding of messages along the policy-based shortest path, and SCLP (Shortest Costumer-preferred Legal Paths), which is easier to implement in a network. We prove the correctness of both algorithms and provide various performance measurements on a real ASes network.

Contents

List of Figures

Abbreviations

AS - **A**utonomous **S**ystem

BGP - **B**order **G**ateway **P**rotocol

EGP - **E**xterior **G**ateway **P**rotocol

HLP - **H**ybrid **L**ink-state and **P**ath-vector

IGP - **I**nterior **G**ateway **P**rotocol

IS-IS - **I**ntermediate **S**yatem to **I**ntermediate **S**ystem

LSA - **L**ink-**S**tate **A**dvertisement

LSP - **L**ink-**S**tate **P**acket

PSNP - **P**artial **S**equence **N**umber **P**acket

PSP - **S**equence **N**umber **P**acket

SCLP - **S**hortest **C**ustomer preferred **L**egal **P**ath

SLP - **S**hortest **L**egal **P**ath

Chapter 1

Introduction

In the early days of the Internet, then known as ARPANET, in the 70's and 80's, the concept of *routing domains* did not exist. The network was small, simple (tree structured), and owned solely by the United States Department of Defens, so simple routing protocols such as RIP and DECnet PHASE IV routing algorithm were sufficient to handle the network's routing. Towards the late 80's, when the number of hosts in the Internet reached a few thousands, the simple routing protocols in use became inefficient and couldn't scale to the growing network. In 1987 the concept of *routing domains* (also known as AS - Autonomous System) was introduced by P. Tsuchiya [4]. The idea was to divide the network into interconnected autonomic entities, each managed by a single authority, called Domains. The emergence of the new concept led to the separation of routing protocols into 2 families:

1. IGP (Interior Gateway Protocol) - intra-domain routing protocols that are responsible for maintaining reachability from any location within a given AS to any other location within the AS by calculating and advertising routes within an AS.

2. EGP (Exterior Gateway Protocol) - inter-domain routing protocols that guarantee global reachability across the entire network by computing routes between ASes while conforming to the commercial relationships between them.

The first inter-domain routing protocol in the Internet was EGP-2 [5]. EGP-2 had significant limitations on the network topology (all AS's had to be attached to a single core-AS) and could not cope with the rapidly growing Internet. In the early 90's a new protocol called BGP-1 replaced EGP-2 as the inter-domain routing protocol in the Internet. In the first few years the new protocol had to be changed to support new technologies (CIDR for example) and to remove useless features no so new versions of BGP were released until in 1992 the BGP-4 [6] was introduced and remained the de-facto inter-domain routing protocol in the Internet until today.

Most neighboring ASes in the Internet have one of two business *link relations*: I) **Provider-Customer** relationship: the customer pays its provider for access to the rest of the Internet. The provider may, in turn, be a customer of another AS. II) **Peer-to-Peer** relationship: two peers find it mutually advantageous to exchange traffic between their respective customers. A peer-to-peer relation is between two ASes with networks of similar size or scope that exchange a comparable volume of traffic. [1] Depending on the link type an AS can take one of the three *roles*: customer, provider, or peer.

These roles and business relations translate, in turn, to three rules that govern the policies taken by ASes in the Internet [7]

R1) AS Policy for its customers - an AS gives its customers transit services toward *all* of its neighboring ASes.

R2) AS Policy for its providers - an AS gives its providers transit services only toward its customers.

R3) AS Policy for its peers - an AS gives its peers transit services only toward its customers.

Equivalently, these three rules can be described by the following condition (policy): an AS x will allow to transit data between two of its neighbors, through itself, if and only if at

[1] Two ASes with a *sibling* relationship are ASes with different AS numbers, but are under the same administrator and we consider them as a single AS.

least one of these neighbors is a customer of x. The result of this condition is that not every possible path between ASes is a policy compliant and *legal* path (i.e., a path that obeys rules R1-R3).

Any inter-domain routing protocol needs to take these policies into account. Current proactive routing protocols can be divided into two main families: *link-state* and *distance vector* (or *path vector*) protocols. The main difference between these two concepts can be described as follows: in link-state every node advertises its **cost to its neighbors** to **all** the other nodes in the network, while in distance vector every node advertises its **cost to all the other nodes** in the network to its **neighbors**. The focus of link-state protocols is to find shortest paths and they are known to be more stable, robust, and converge faster after topology changes than distance vector protocols. On the downside, link-state protocols do not scale well since every cost change of an edge needs to be broadcast to the entire network. In addition, current link-state protocols do not support policy-based routing. Therefore intra-domain protocols traditionally use link-state protocols such as Open Shortest Path First (OSPF), and Intermediate system to intermediate system (IS-IS) [8].

Distance and path vector protocols (path-vector protocols are similar to distance vector but advertise paths instead of costs) usually use less traffic and it is easier to control their advertisements. Implementing policies in distance-vector protocols is easier since a node can simply decide whether to advertise a destination to its neighbor (depending on its policy) without causing any contradictions or complications in other nodes. This is part of the reason that the currently used inter-domain routing protocol in the Internet, BGP, is a path-vector protocol and not a link-state protocol. On the downside, path vector protocols are less stable since they react much more slowly to topology changes and are much more vulnerable to attacks (advertising a misleading *path* can cause more damage than advertising a misleading edge cost).

1.1 Our Contribution

Nowadays, BGP has many deficiencies. A lot of work was done in recent years to analyze the performance and problems of BGP [9, 10, 11, 12, 13], and the need for a solution to these problems led to a series of improvements and changes to the BGP protocol that eventually increased its complexity to a very high level. Furthermore, many of the problems remain unsolved because the "point fixes" given alleviated one problem but many times at the cost of making other problems worst.

In this thesis we propose a new protocol that combines link-state and path vector into a Path-State Protocol (PSP). The main features of the new protocol are that it supports policy-based routes like in path vector and BGP protocols, and it advertises information (i.e., path states) globally like link-state and IS-IS protocols.

In PSP the network is partitioned into arbitrary regions (groups of connected ASes that agree to share more information) where every edge in the network must belong to exactly one region. The members of each region use, between themselves, a policy-based link-state protocol to find best paths within the region. In turn, ASes that belong to more than one region advertise to each of their regions (in a link-state fashion) the *path states* they have learned from all other regions they belong to. The outcome of this approach is that PSP can find shortest *legal* paths (i.e., paths that obey the policies) while reacting faster to topology changes and avoid oscillations or count-to-infinity type problems. If each edge is a region, then PSP has similar behavior to that of path-vector protocol and BGP. On the other extreme, where the whole network is a single region, PSP behaves like a link-state protocol (but with the addition of policy support).

We present two versions of PSP: SLP (Shortest Legal Paths) that guarantees forwarding of messages along the shortest policy-based paths and SCLP (Shortest Customer-preferred Legal Paths), which is easier to implement in a network and finds paths that are similar to the paths found by BGP. SLP requires the use of *extended* forwarding tables, which need to

know the role of the previous hop (or equally the *incoming* port of a message) for forwarding, while SCLP uses standard forwarding tables. We prove the correctness of both algorithms and provide various performance measurements on a real ASes network. In particular, we show that for about 42% of the source-destination pairs in the graph SLP finds shorter paths than SCLP.

The rest of this thesis is organized as follows: Chapter 2 compares PSP features to related protocols. The main chapter in this thesis is Chapter 3, which presents PSP (Path-State Protocol) including a detailed explanation of its operation and proofs of its properties. In Chapter 4 we present a simulation we built for evaluating PSP's performance and its results. Finally Chapter 5 discusses the conclusions of our work and possible directions in future research.

Chapter 2

Previous Work: PSP vs. Related Protocols

This chapter discusses the details of some major existing routing protocols and then compares their properties to PSP. In Section 2.1 we discuss IS-IS [8], which is a commonly used intra-domain routing protocol in the Internet. As PSP, IS-IS is a link state protocol and also divides the network into sub-hierarchies called areas; therefore it is suitable for comparison with PSP. Then in Section 2.2 we discuss BGP [6] the only inter-domain routing protocol in the Internet since 1994 and until today. BGP belongs to the path-vector family and it is flat (it doesn't define any partitioning of the network into smaller routing domains). Finally, in Section 2.3 we compare the properties of PSP to the above protocols and to a new suggested protocol - HLP (Hybrid Link-state and Path vector). HLP was suggested by Subramanian et al. [14] and it is the protocol closer to our approach. HLP uses the hierarchical nature of the relations between ASes in the Internet (peer-peer, customer-provider) to divide the network into hierarchies. A link-state protocol operates inside each hierarchy, while information between hierarchies (peer-peer links) is transferred in path vector, much like BGP, by FPV (Fragmented Path Vector) packets that include only the high level path (a path of hierarchies)

and a path cost for each destination.

2.1 IS-IS

2.1.1 Overview

The IS-IS (intermediate system to intermediate system) protocol was developed by Digital Equipment Corporation and standardized by the ISO in 1992 as ISO 10589 for communication between network devices that are termed Intermediate Systems. It is a link state intra-domain dynamic protocol designed originally to operate in OSI Connectionless Network Service (CLNS) and for routing of OSI traffic only, but since IP routing became much more popular during the 1990's the IETF organization published an extension to the protocol (RFC 1195) called integrated IS-IS that defines the operation of IS-IS in pure IP environments, pure OSI environments, and dual environments. Nowadays integrated IS-IS is deployed extensively in the top-tier Internet service provider (ISP) networks and about 25 extensions for IS-IS were published by the IETF for different purposes such as MPLS, traffic engineering, IPv6.

IS-IS enhances protocol's scalability by dividing the routing domain into areas and defining two routing levels: level 1 routing (intra-area) and level 2 routing (inter-area). Each IS resides in exactly one area. There must be connectivity between all level 2 bridges in the domain (backbone). For a packet destined for another area, a Level 1 IS sends the packet to the nearest Level 2 IS in its own area, regardless of the destination area. Then the packet travels via Level 2 routing to the destination area, where it may travel via Level 1 routing to the destination. It should be noted that selecting an exit from an area based on Level 1 routing to the closest Level 2 IS might result in suboptimal routing [15].

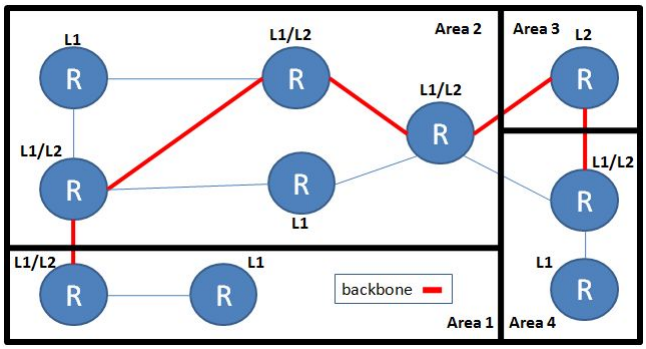


Figure 2.1: Example of a hierarchical IS-IS network divided to areas. The backbone comprises the red links through which inter-area routing information traverses.

2.1.2 Protocol Structure

Address Structure

The OSI framework is hierarchical, separating routing domains into areas composed of one or many local networks.

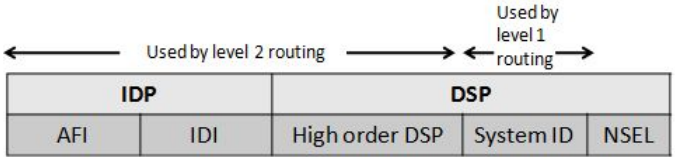


Figure 2.2: The routing domain part is composed of the AFI, IDI, and possibly the first bytes of the DSP. It is followed by a 16 bit area address; the next 48 bits are the host identifier. The 8 bit NSEL identifies the transport protocol and therefore is not used for routing[1].

IS Types

An IS can be one of three types:

Level 1 IS – can have only level 1 and level 1-2 neighbors from its own area and knows all the level 1 and level 1-2 ISs in its own area. It handles and propagates only level 1 LSPs

and includes level 1 databases only. When receiving a message destined for a different area it forwards it to the closest level 1-2 IS.

Level 1-2 IS – can have neighbors of all types and from different areas. Knows all the level 1 ISs in its area and all the level 1-2 and level 2 ISs in the routing domain. It handles and propagates level 1 and level 2 LSPs and includes level 1 and level 2 databases.

Level 2 IS - can have only level 2 and level 1-2 neighbors. Can be attached to different areas. It handles and propagates only level 2 LSPs and includes level 2 databases only.

Message Types

IS-IS defines three main groups of messages:

Hello messages – are used for the adjacency discovery phase of the protocol. Each IS sends hello messages in a predefined interval. When an IS receives hello messages it initiates a short handshake mechanism (described in the following sections) that produces (if successful) a new adjacency entry in the adjacency database.

LSP messages – LSP (link-state packets) contain the data from which the routing tables are calculated. each LSP includes a sequence number and the LSP generator system ID and area. A Level 1 LSP created by a specific IS includes the level 1 capable neighbors of that IS and the costs of the links. A Level 2 LSP created by a specific IS includes the level 2 capable neighbors of that IS and the costs of the links, and also the area addresses this IS is attached to and the costs of the links.

SNP messages – SNPs (sequence number packets) are used mainly for database synchronization. A CSNP (complete SNP) is sent both ways immediately after a new adjacency is formed and it consists of the headers of all the LSP entries in the originating IS LSP database. When an IS receives a CSNP it compares its LSP database to the CSNP sender's LSP database. A PSNP (partial SNP) consists of the header of a single LSP and is used for acknowledgments and requests for routing data, as will be explained in the following

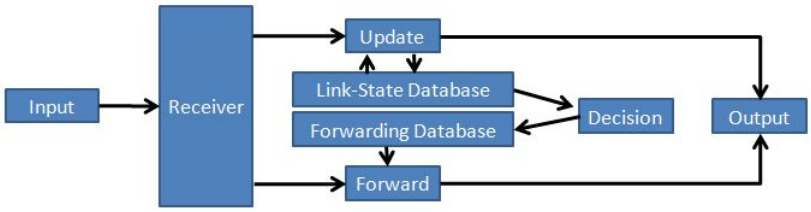


Figure 2.3: IS-IS protocol schematic structure.

sections.

Protocol Components

Link-State database - Contains the LSPs of all the nodes in the appropriate level (in a level 1-2 IS there are 2 Link-State databases). Every LSP contains the links state and cost from the originator of the LSP to all of its neighbors. With the LSPs of all the nodes of a network, a complete graph of the network can be built.

Forwarding database - The forwarding table of the node. When a data packet is received, the entry in the forwarding table with the same destination address as the destination address in the packet tells the node towards which port the data packet should be forwarded.

Receiver process - Receives packets from the external ports. Forwards IS-IS protocol packets to the update process and all other packets to the forwarding process.

Update process – It is the main process that consists of most of the IS-IS protocol logic. Builds the LSP databases (one for each level) from received LSPs. Handles handshake mechanism with neighboring ISs and builds the adjacencies databases (one for each level). Purges old entries in LSP databases and adjacency databases. Initiates periodic sending of LSP and hello messages. Propagates received LSPs from other ISs.

Decision process – Configures the bridges routing table by performing the Dijkstra algorithm on the LSP databases as described in the following sections.

Forwarding process – Forwards packets according to the forwarding database.

2.1.3 Protocol Operation

Neighbors Discovery

The first phase of the protocol is neighbors discovery. Every IS sends a hello message periodically. Every entry in the adjacency database has an age. Old entries are purged. A hello message includes the following data: originator system ID, originator area address, originator IS level, other side system ID (if known; used for reliability). When an IS receives a hello message it searches its adjacency database. If the originator is known (there is an existing "up" state entry with its ID) the appropriate entry's age is set to 0. If the originator is unknown a new adjacency entry is formed with an "initializing" state and a reply is sent that includes the other side's ID. If there is an existing "initializing" state entry with its ID and the message contains the local ISs ID, then the entry's state is changed to "up".

Figure 2.4: IS-IS handshake protocol for neighbors discovery.

Database Synchronization

Database synchronization occurs by flooding LSPs throughout the area/backbone (level 1/level 2). Each IS generates an LSP periodically and also every time there is a change in its adjacency database.

Shortest-Paths Computation

Shortest-paths computation is carried out differently in each level:

Level 1 shortest paths computation: Dijkstra algorithm on level 1 LSP database gives shortest

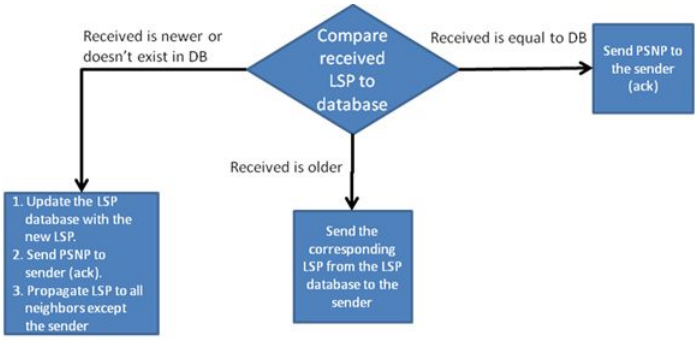


Figure 2.5: Flow chart for IS actions when receiving an LSP.

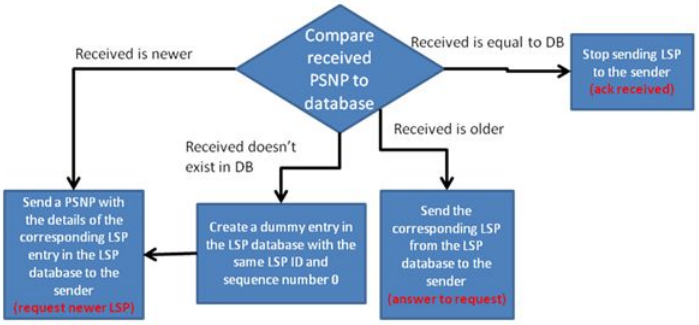


Figure 2.6: Flow chart for IS actions when receiving a PSNP.

paths from the source IS to every other IS in the area. Forwarding ports are deducted from the computed shortest paths for each destination. The routing table is then configured with the (key , value) pair (destination system ID, port).

Level 2 shortest paths computation: As mentioned, level 2 LSPs contain a list of neighboring ISs and a list of neighboring areas. Dijkstra algorithm is performed on the IS neighbors lists and then the closest IS for each area can be found. The routing table is then configured with the (key, value) pair (destination area, port).

2.2 BGP

2.2.1 Overview

The Border Gateway Protocol (BGP) is a path-vector protocol. Path-vector protocol by definition is same as distance vector protocol with additional path information. Routing information in BGP contains the length of the path to the destination, as well as the actual full path used to reach the destination. This eliminates the "count to infinity" problem, a well-known problem associated with distance vector protocols due to lack of loop detection when links fail. Loop detection in BGP is possible because the AS path information carried in advertisements stores the list of ASes the route traverses.

Internet routing is destination-based and is done on a hop-by-hop basis. The destinations are expressed as network prefixes supporting Classless Inter-domain Routing. BGP has two distinct modes of operation: eBGP (External BGP) exchanges reachability information between Autonomous Systems, while iBGP (Internal BGP) exchanges external reachability information within an Autonomous System. BGP messages are sent using the TCP protocol to ensure reliable delivery. Since TCP itself can operate without exchanging any messages for a long time, BGP uses periodic keep-alive messages between two peers in a BGP session to ensure their aliveness.

An AS uses BGP to advertise routes to its neighboring ASes, indicating its routing decisions and its willingness to carry traffic on behalf of others. The essential BGP protocol is based on two types of BGP messages: *announcements* and *withdrawals*. An *announcement* message sent from a router R_a in AS A to a router R_b in AS B to the destination prefix P with routing information R indicates that R_a is willing to carry traffic from R_b destined to P following the route R. The routing information advertised also indicates the routing decisions of the advertiser. In this case, R_a uses the route R to reach destination P. Given all the available routes received from ones neighbors stored in the RIB (Routing Information Base),

the router uses its routing policies to select the best route. In this example, R_b may have received multiple routes for destination P, one from each of its providers. It will choose the best one based on its own routing policies and may in turn announce the route to its customers. Routing policies also determine the available routes between a pair of Internet hosts. R_a may decide not to advertise the route to R_b due to its specific policies. Because of the way in which BGP policies filter and propagate routes, physical connectivity on the Internet does not imply logical reachability. Thus, BGP does not ensure global reachability, which is important to assure the ability of any to any communication. The way each AS implements its routing policies is explained in detail in Section 2.2.2.

Withdrawals, as expected, indicate that the sender no longer has a route to reach the given destination. They invalidate the last announcement sent regarding the route. This can be caused by things such as transient failures caused by a flapping link, congestion that can cause session time out, and software upgrade on routers that may require reboot. *Announcements* can also serve as implicit *withdrawals*, modifying the attributes of the route that was last sent.

2.2.2 Routing Policies

In this section we explain the process each *announcement* message goes through once it arrives to an AS. This process is comprised of 3 main parts explained in the following sections:

Applying Import Policy

Each BGP router gets *announcements* from its eBGP sessions (from different ASes) and from its iBGP sessions (other BGP speakers in the same AS). *Announcements* that arrive through an eBGP session are processed according to the import policies of the AS, which are divided into 2 types:

a. Implicit import policies (defined by the protocol) - discard any announcement that contains the receiving AS in the AS path parameter (for loop prevention).

b. Explicit import policies (defined by the AS administrator) - assign a local preference value according to administrator's decision.

The AS administrator applies its import policy by setting the value of the "local preference" parameter for each path it gets. As explained in the next section, the path selection process considers the "local preference" parameter as the first criterion for comparing paths; therefore the AS administrator can prioritize the paths advertised to it, thus applying its explicit import policies.

Path Selection

After setting the import policies of the received announcement the BGP router sends it to all other BGP routers in the same AS using iBGP sessions. Each BGP router then selects its best path to the advertised prefix out of the set of received announcements according to the value of the "local preference" value. In case of a tie, BGP considers the following parameters as tie breakers (ordered from highest to lowest priority):

1. AS path length.

2. MED (Multiple Exit Discriminator) value.

3. Smallest distance to the BGP router in the receiving AS that is connected to the "next hop" AS according to the AS path parameter.

Applying Export Policy

After each BGP router selects its best path to a prefix it applies its export policies. This is done by deciding to which of its neighbors the specific prefix will advertised. If, for example,

AS A's policy is that it doesn't want to provide transit services to its neighbor AS B, then AS A will not send any announcements to AS B other than announcements that include prefixes of A itself. That way only data that is destined to A will be sent from B to A.

2.2.3 BGP Problems

Scalability

The Internet is growing rapidly and with it the number of ASes and the BGP routing table size. According to [2], in 1999 there were approx. 5000 ASes in the Internet and approx. 50000 active entries in the BGP routing table. In 2010 there are approx. 31000 ASes and approx. 300000 active entries in the BGP routing table (Figure 2.7). Another reason for the BGP scalabilty problem, other than the natural growth of the Internet, is that in recent years stub ASes tend to create more links to the Internet (multihoming) for resilience and load balancing reasons, causing the same IP prefixes to reach certain ASes from multiple paths and increase their routing table size.

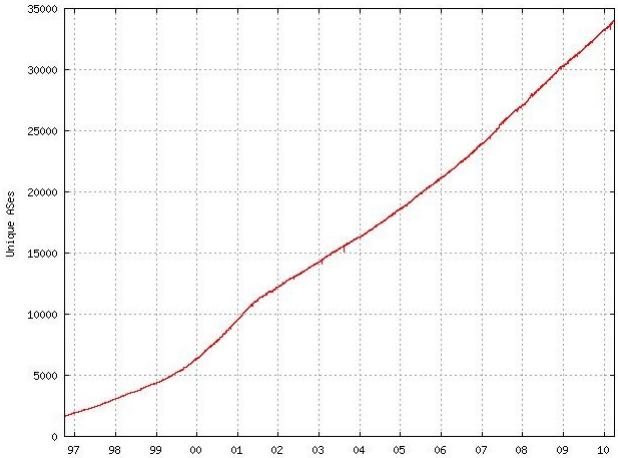


Figure 2.7: Number of ASes in the Internet [2].

Convergence

A router that regularly advertises and shortly afterwards withdraws its route is called a flapping router. A mechanism called BGP route flap damping [16], which ignores routes that change too often, is often used to prevent storms of advertisements due to flapping routers, but the downside of it is that it significantly increases the convergence time. The average convergence time of most of the inter-domain routes in the Internet is between 2 and 5 minutes, where in routes that are not stable and tend to flap the convergence time can be on the order of hours [14].

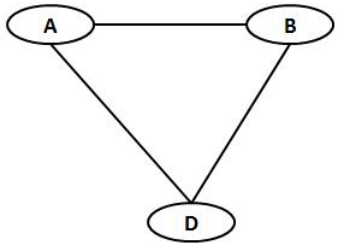


Figure 2.8: An example network. AS A, AS B and AS D run BGP between them.

Route Oscillations

BGP allows each AS to choose its routing policies without any limitations. This approach may cause a situation in which some AS's policies contradict each other and prevent BGPs from converging. This kind of instability is shown in the following example taken from [17]: In Figure 2.8 AS D is the only AS that advertises a destination network 10.1.3.0/24. AS As policy is to get to the destination network via AS B if possible, and AS Bs policy is to get to the destination network via AS A if possible. When AS D advertises the destination network to A and B, they both choose the direct path towards D. In the next stage both A and B advertise to each other their known path to 10.1.3.0/24 - A advertises the path (A,D) and B advertises the path (B,D). Now A chooses its favorite path to the destination network

Protocol	IS-IS	BGP	HLP	PSP-SLP	PSP-SCLP
Type	link-state	path vector	combined	path state	path state
Supports policy	-	+	+/-	+	+
Shortest paths	-	-	-	+	-
Hierarchies	flexible	none	strict	flexible	flexible
Standard forwarding table	+	+	+	-	+
Privacy	none	strong	partial	partial	partial
Stability	+	-	+	+	+

Table 2.1: Comparison between routing protocols

(A,B,D) and advertises it to B. Simultaneously, B chooses the path (B,A,D) and advertises it to A. A gets the path (B,A,D) from B, disqualifies it since it appears in the AS-PATH and chooses again the direct path (A,D). The same process happens in B and the whole scenario starts over, thus causing route oscillations.

2.3 Protocol Comparison

In Table 2.1 we display an overview of the main features of PSP (with its two versions - SLP and SCLP) compared to IS-IS, BGP, and HLP [14], which is a protocol that also combines both link-state and path vector protocols.

IS-IS [8] is a link-state routing protocol developed by ISO for intra-domain routing. As in PSP, IS-IS divides the network into regions (the regions are called areas in IS-IS) with the flexibility of setting any partitioning of the network to areas as long as each area contains a set of connected nodes. As an intra-domain routing protocol IS-IS does not support nodes

policies and therefore it is not suitable for inter-domain routing. PSP, on the other hand, does support nodes policies by calculating shortest paths on a special graph that includes only paths that are *legal*, i.e., they are compliant with the nodes policies.

In IS-IS each node has routing levels 1 and/or 2. A level 1 node knows only its own area (intra-area routing) and a level 2 node knows all other level 2 nodes, including other areas (inter-area routing). There are also nodes with level 1/2, which have both level 1 and level 2 capabilities. The result of this concept is that a level 1 node that needs to send data to a node in a different area has no knowledge of the remote area location and therefore sends the data to the nearest level 2 node ("hot potato" routing), and therefore IS-IS paths are sub-optimal. In contrast, PSP with SLP chooses only the shortest legal paths. Another difference is that in IS-IS there is no privacy in the sense that level 2 nodes know the global topology of all level 2 nodes even in other areas. PSP nodes are ignorant about the topology of regions they do not belong to.

Nowadays, BGP has many deficiencies. A lot of work was done in recent years to analyze the performance and problems of BGP [9, 10, 11, 12, 13] and the need for a solution to these problems led to a series of improvements and changes to the BGP protocol that eventually increased its complexity to a very high level.

Many of the problems BGP suffers from relate to the fact that it is a path vector protocol. For example, the average convergence time for most of the inter-domain routes in the Internet is between 2 and 5 minutes, where in routes that are not stable and tend to flap the convergence time can be on the order of hours [18]. PSP, on the other hand, runs a link-state protocol inside a region and therefore any event (new edge, change of cost, failed edge, etc.) will be immediately flooded to all the nodes in the region. Only in case a path-cost is changed will it be broadcasted to other regions (similar to BGP that advertises only path changes). PSP's property of global view of the network and finding only shortest (customer preferred) legal paths also prevents the long convergence time that may happen in BGP as described in Example 4.1 in [19].

The fact that in BGP a node does not have the whole network graph means that a certain amount of trust is required between nodes, and therefore attacks in the form of false advertisements are fairly easy to accomplish and special mechanisms need to be developed in order to cope with this problem. In PSP this kind of attack is much harder to accomplish as every node has a full graph of its region, so false advertisements are easier to identify and ignore. BGP keeps the node's policies private so sometimes contradiction between policies causes unstable behavior [20, 17]. In PSP policies of the nodes are known to everyone so such contradictions are again easier to identify.

HLP (Hybrid Link-state and Path vector) suggested by Subramanian et al. [14] is the protocol closest to our approach. HLP uses the hierarchical nature of the relations between ASes in the Internet (peer-peer, customer-provider) to divide the network into hierarchies. A link-state protocol operates inside each hierarchy, while information between hierarchies (peer-peer links) is transferred in path vector, much like BGP, by FPV (Fragmented Path Vector) packets that include only the high level path (a path of hierarchies) and a path cost for each destination. The hierarchies in HLP are strict and inferred only from the Internet structure – each hierarchy is built from a tier 1 node (a node with no providers) and all of its direct and indirect customers (customers, customer's customers, etc.); therefore they are forced to be of very large scale. On the contrary, PSP gives the flexibility of partitioning the network into almost arbitrary regions (the equivalent term for hierarchies in PSP). In addition, HLP does not describe any method by which policy is supported inside the hierarchies; such a method is one of the major contributions of PSP.

Chapter 3

PSP - Path State Protocol

3.1 Protocol Overview

This section describes PSP (Path State Protocol) operation in a network. PSP is an inter-domain routing protocol that combines two families of routing protocols - link-state and path vector. It divides the network into regions and each region runs a policy based link-state protocol that relates to the nodes routing policies and computes paths by performing the Dijkstra Algorithm on a special graph called *policy graph*.
Since PSP relates to Internet as the network graph G, then the graph describes the connections between ASes when the connections are divided into 2 types based on the business relationships in the Internet:

1. *Provider-Customer relationship:* the customer pays its provider for access to the rest of the Internet. The provider may, in turn, be a customer of another AS.

2. *Peer-to-Peer relationship:* In a peer-to-peer relationship, the two peers find it mutually advantageous to exchange traffic between their respective customers. A peer-to-peer relationship is between two ASes with networks of similar size or scope that exchange a comparable volume of traffic.

According to Gao and Rexford ([7]) AS policies in the Internet follow these rules:

R1) Policy for customers - an AS gives its customers transit services towards all of its links.

R2) Policy for providers - an AS gives its providers transit services only towards its customers.

R3) Policy for peers - an AS gives its peers transit services only towards its customers.

Figure 3.1 shows examples of valid and invalid paths in the Internet according to the above rules. Path 3 is invalid since AS B transfers data between two providers (A and Z) which contradicts rule R2. Path 4 is invalid since AS B transfers data between its peer C and its provider Z which contradicts rules R2 and R3.

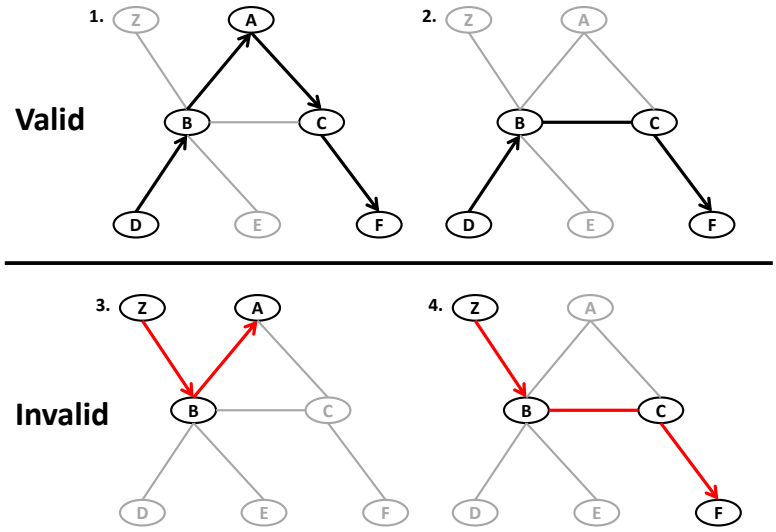


Figure 3.1: Examples of 2 valid and 2 invalid paths between ASes in the Internet ([3]). A directed edge represents a customer-to-provider relationship, while an undirected edge represents a peer-to-peer relationship.

We start with some notations in Section 3.2, then we describe the operation of PSP in the intra-region perspective (Section 3.3) - we explain the construction (Section 3.3.1) and the

properties (Section 3.3.2) of the policy graph. Then we describe the two intra-region routing protocols - SLP and SCLP (Section 3.3.3), and finally we show the operation of PSP inside a region using an example network (Section 3.3.4).

The concept of regions and the operation of PSP in the inter-region perspective is then explained in Section 3.4.

3.2 Network Model and Notations

We model the Internet as a graph of ASes called the *Internet Graph* $G = (V, E, w, T)$, where V is the vertices set (ASes), E is the (undirected) edges set (links), w is an edge weights function (link costs), and T is an edge type function that defines the `Link Type` of each edge (Provider-Customer or Peer-Peer) and the `Role` of each AS in the link. Let N(u) denote the set of neighbors of $u \in$ V. Let (x, y) and $\langle x, y \rangle$ denote undirected and directed edges between nodes x and y, and $w(x, y)$ be the weight of the edge between x and y. The `Role` of each node in each link is denoted by $L_x(x, y)$, which indicates the `Role` of node x in the link between x and y.

Each node $x \in V$ advertises its links (or paths) in *Link State Advertisements* (LSA) to all the nodes in the network (or region) by some given broadcast mechanism (e.g., a type of reverse path forwarding). Each LSA advertised by x contains the cost of the link from x to a neighbor y and the `Role` of x on the link (i.e., "provider", "customer", or "peer"). If, for example, x is a customer of y, then x will advertise the link (x, y) with $L_x(x, y) =$ "customer" while y will advertise the link (y, x) with $L_y(x, y) =$ "provider". In case (x, y) is a peer-to-peer link then both x and y will advertise it as peers.

As mentioned, PSP is a link-state based, inter-domain routing protocol that also supports advertisements of path states information. Advertisements of paths are carried out by (limited) broadcasts instead of sending it locally to neighbors, as in path vector protocols (e.g., BGP). Another major difference from standard link-state protocols is that PSP supports policy-based routing according to rules R1-R3.

In PSP the network is divided into regions (groups of ASes) where each region runs a policy-based link-state protocol that finds what we call *legal* routes between nodes by performing the Dijkstra Algorithm on a special graph named the *policy graph*. For simplicity of presentation we start our explanation by considering a network that is a single region; we extend the discussion to a network with multiple regions in Section 3.4.

3.3 Intra-Region

3.3.1 Building G^* - Policy Graph

As a result of the link state advertisements by all the nodes on all the links, each node in the network knows the network graph $G = (V, E, w, T)$. From G, each node builds the *policy graph*, a directed graph $G^* = (V^*, E^*, w*)$ that reflects the link relations between ASes in the network. Since G^* is built from G, all the nodes in the network share the same policy graph. In turn, G^* is used as the input to the Dijkstra algorithm to find shortest legal paths. We now describe the construction of G^* from an Internet graph G; in particular we define V^*, E^*, and w^*, the sets of vertices, directed edges, and edges weights of G^*, respectively.

Creating the vertex set V* - Each node $x \in$ V is represented by the following nodes in V*:

x_{dst} A node for data whose destination is x.

x_p^+ A node for data flow **from** x's providers or peers.

x_c^+ A node for data flow **from** x's customers and data flow that originates in x.

x_p^- A node for data flow **to** x's providers or peers.

x_c^- A node for data flow **to** x's customers.

Creating the directed edge set E* - for each node $x \in$ V:

1. According to rules R1, R2, and R3, for each $x \in V$, insert the following edges with weight 0 into E^* (see Fig. 3.2 (a)):

   (a) $\langle x_c^+, x_{dst} \rangle$ and $\langle x_p^+, x_{dst} \rangle$ - Any data that arrive to x can terminate at x (provided that x is the destination).

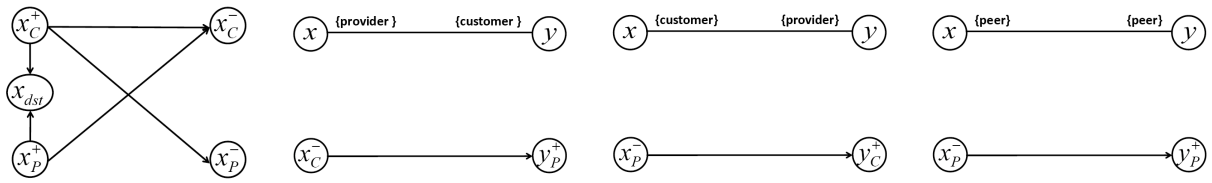


Figure 3.2: (a) The nodes that represent node $x \in G$ in G^* and the links between them. (b)-(d) The upper links are links between node x and node y in the network graph G. The lower links are the same links as they appear in the policy graph G^*.

(b) $\langle x_c^+, x_c^- \rangle$ and $\langle x_c^+, x_p^- \rangle$ - Data flow is permitted from x's customers to any of x's neighbors, providers, customers, or peers (rule R1). Data that originate in x are also permitted to flow towards any of x's neighbors.

(c) $\langle x_p^+, x_c^- \rangle$ - Data flow from x's peers or providers is permitted only to x's customers (rules R2 and R3).

2. For each node y that is a *customer* of x insert the edge $\langle x_c^-, y_p^+ \rangle$ into E^* with weight $w(x, y)$ (see Fig. 3.2 (b)).

3. For each node y that is a *provider* of x insert the edge $\langle x_p^-, y_c^+ \rangle$ into E^* with weight $w(x, y)$ (see Fig. 3.2 (c)).

4. For each node y that is a *peer* of x insert the edge $\langle x_p^-, y_p^+ \rangle$ into E^* with weight $w(x, y)$ (see Fig. 3.2 (d)).

Algorithm 1 Building policy graph for node s

Require: G = (V,E,w,T) network graph, built from the link-state advertisements of all the nodes in the network.

Ensure: G^* = (V*,E*,w) policy graph.

1: **for all** node $u \in V$ **do**
2: insert nodes $u_{dst}, u_p^+, u_c^+, u_p^-, u_c^-$ into V*
3: insert edges $< u_c^+, u_{dst} >$, $< u_p^+, u_{dst} >$, $< u_c^+, u_c^- >$, $< u_c^+, u_p^- >$, $< u_p^+, u_c^- >$ into E*
4: **end for**
5: **for all** edge $(u, v) \in E$ **do**
6: **if** $L_u(u, v) = customer$ and $L_v(u, v) = provider$ **then**
7: insert edges $< u_p^-, v_c^+ >$ and $< v_c^-, u_p^+ >$ into E*
8: **end if**
9: **if** $L_u(u, v) = provider$ and $L_v(u, v) = customer$ **then**
10: insert edges $< u_c^-, v_p^+ >$ and $< v_p^-, u_c^+ >$ into E*
11: **end if**
12: **if** $L_u(u, v) = peer$ and $L_v(u, v) = peer$ **then**
13: insert edges $< u_p^-, v_p^+ >$ and $< v_p^-, u_p^+ >$ into E*
14: **end if**
15: **end for**

3.3.2 Properties of the Policy Graph

In this section we first define a legal path by conditions inferred from the rules R1, R2, R3 presented in Section 3.1 and then present and prove the properties of the policy graph G^*.

Definition 3.1 ($walk(x, y, z)$) *A $walk(x, y, z)$ is a directed path $x \rightarrow y \rightarrow z$.*

Definition 3.2 (legal *walk*) *A $walk(x, y, z)$ is legal iff it complies with R1 or R2 or R3. Alternatively, a $walk(x, y, z)$ is legal iff one of these conditions holds:*

1. *x is a customer of y and z is a customer of y according to R1.*
2. *x is a customer of y and z is a peer of y according to R1.*
3. *x is a customer of y and z is a provider of y according to R1.*
4. *x is a provider of y and z is a customer of y according to R2.*
5. *x is a peer of y and z is a customer of y according to R3.*

Note that these 5 conditions are equal to the following rule: A $walk(x, y, z)$ is legal iff at least one of y's neighbors (x, z) is a customer of y.

Definition 3.3 (policy path) *A path $\widehat{P}$=$(a_0, a_1, \ldots, a_{2n}, a_{2n+1})$ in G^* where $a_i \in V^*$ from node s to node d is called a* policy path *iff:*

1. $a_0 = s_c^+$.
2. $a_{2n+1} = d_{dst}$.
3. $\langle a_i, a_{i+1} \rangle \in E^*$, $0 \leq i \leq 2n$.
4. $a_{2k+1} \in u^-$ *for some* $u \in V$, $0 \leq k < n$.
5. $a_{2k} \in u^+$ *for some* $u \in V$, $1 \leq k \leq n$.

For a policy path $\widehat{P}$ from s to d let $F(\widehat{P})$ be the corresponding path in G. Namely, $F(\widehat{P}) = (s, b_1, \ldots, b_{n-1}, d)$, where $\{b_i, s, d\} \in V$ and for each $0 < i < 2n$, a_{2i} represents b_i in G^*. Note that every policy path $\widehat{P}$ in G^* represents a path in G (i.e., $F(\widehat{P})$) but a path P in G is not necessarily represented by a policy path in G^*.

Definition 3.4 (legal path) *A path $(a_1 \rightarrow a_2 \rightarrow ... \rightarrow a_n)$ $a_i \in V$ in G is legal iff one of the following conditions holds:*

1. $n = 2$.
2. $n > 2$ *and for every* k, $1 \leq k \leq n-2$ *the* $walk(a_k, a_{k+1}, a_{k+2})$ *is a legal walk.*

A policy path $\widehat{P}$ in G^* is a legal policy path iff there exists a legal path P in G such that $F(\widehat{P})$=P.

Theorem 3.1 *A path P in G is a legal path iff there exists a policy path $\widehat{P}$ in the policy graph G^* such that $F(\widehat{P}) = P$.*

Proof. For the proof of this Theorem we first show that all the policy paths in G^* are legal (Lemma 3.1) and then we show that all the legal paths in the network graph G exist in the G^* (Lemma 3.2).

Lemma 3.1 *All the policy paths in the policy graph G^* are legal policy paths.*

Proof. We prove the lemma by showing that illegal *walks* don't exist in G^*:
According to Definition 3.2 a $walk(x, y, z)$ is illegal iff both x and z are not customers of y. Therefore a $walk(x, y, z)$ is illegal iff x is a provider or peer of y and z is a provider or peer of y.
We can clearly see that for data to flow in an illegal path $x \rightarrow y \rightarrow z$, G^* must contain the edge $< y_p^+, y_p^- >$ but according to Section 3.3.1 item 1, the edge $< y_p^+, y_p^- >$ doesn't exist in G^* and therefore none of the *walks* described above exists in G^*.

We have shown that illegal *walks* don't exist in G^* $\Rightarrow$ illegal policy paths don't exist in G^* $\Rightarrow$ all the policy paths in the policy graph G^* are legal paths. ■

Lemma 3.2 *All the legal paths in the network graph G exist in the policy graph G^*.*

Proof. We will go over all the possible legal *walks* in G according to the legal *walk* definition and show that each of these *walks* exists in G^*:

1. $walk(x, y, z)$ when x is a customer of y and y is a provider of z. In this case for data to flow in the path $x \rightarrow y \rightarrow z$ G^* must contain the edge $< y_c^+, y_c^- >$, which indeed exists in G^* according to Section 3.3.1 item 2.a, and therefore $walk(x, y, z)$ exists in G^*.

2. $walk(x, y, z)$ when x is a customer of y and y is a customer of z. In this case for data to flow in the path $x \rightarrow y \rightarrow z$ G^* must contain the edge $< y_c^+, y_p^- >$, which indeed exists in G^* according to Section 3.3.1 item 2.a, and therefore $walk(x, y, z)$ exists in G^*.

3. $walk(x, y, z)$ when x is a customer of y and y is a peer of z. In this case for data to flow in the path $x \rightarrow y \rightarrow z$ G^* must contain the edge $< y_c^+, y_p^- >$, which indeed exists in G^* according to Section 3.3.1 item 2.a, and therefore $walk(x, y, z)$ exists in G^*.

4. $walk(x, y, z)$ when x is a provider of y and z is a customer of y. In this case for data to flow in the path $x \rightarrow y \rightarrow z$ G^* must contain the edge $< y_p^+, y_c^- >$, which indeed exists in G^* according to Section 3.3.1 item 2.a, and therefore $walk(x, y, z)$ exists in G^*.

5. $walk(x, y, z)$ when x is a peer of y and z is a customer of y. In this case for data to flow in the path $x \rightarrow y \rightarrow z$ G^* must contain the edge $< y_p^+, y_c^- >$, which indeed exists in G^* according to Section 3.3.1 item 2.a, and therefore $walk(x, y, z)$ exists in G^*.

We have shown that all the legal *walks* in G exist in G^*; therefore all the legal paths in G exist in G^*. ■

Lemma 3.1 and Lemma 3.2 show that a path in G is legal iff the corresponding policy path exists in G^*.

■

Theorem 3.1 enables us to use G^* to find legal paths in G in general and in particular to find shortest legal paths in G.

3.3.3 Legal Paths Routing Algorithms

Every node $s \in V$ that receives the LSAs from all the other nodes in the network, builds the policy graph G^* according to the rules presented in Section 3.3.1. In the following subsections we define the steps that each node $s \in V$ takes after constructing the policy graph in order to calculate legal paths and update its forwarding table. But, first we define two types of forwarding tables:

I. Standard forwarding table: For a node s, a standard forwarding table $\mathcal{F}_s$ maps a destination node w to a neighbor of s, with the semantics that a packet arriving at s and destined to w will be sent to the neighbor of s given by $\mathcal{F}_s$. In short, standard forwarding table mapping can be described as (destination address)⇒(port). The structure of standard forwarding table is shown in Figure 3.3 (a).

II. Extended forwarding table: For a node s, an extended forwarding table $\bar{\mathcal{F}}_s$ maps a destination node w and a Boolean indicator called `Internal` to a neighbor of s. The value of `Internal` indicates whether the data come from within s or from customers of s (`Internal` = True) or data that come from providers or peers of s (`Internal` = False). The semantics of $\bar{\mathcal{F}}_s$ is that for a packet arriving to s and destined to w, if the packet came from a peer or a provider of s it will be sent to the neighbor of s given by the forwarding table entry with `Internal` = False, and if the packet originated in s or arriving from a customer of s it will be sent to the neighbor of s given by the forwarding table entry with `Internal` = True. In short, extended forwarding table mapping can be described as (destination address, previous node type)⇒(port). The structure of extended forwarding table is shown in Figure 3.3 (b).

We next present two routing algorithms for legal path calculation and forwarding tables updating: *SLP* for calculating shortest legal paths with an *extended forwarding table* and *SCLP* for calculating suboptimal legal paths but with *standard forwarding table*. Both algorithms use the Dijkstra Algorithm (Algorithm 2) to find shortest paths on the policy graph but the updating of the forwarding table is performed according to the specific algorithm.

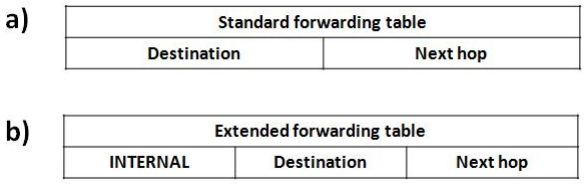


Figure 3.3: The structure of standard and extended forwarding tables.

Algorithm 2 Dijkstra

Require: G = (V,E,w), source node s.

Ensure: T = (V,E*) shortest path tree from source node s.

1: **for all** $u \in V^*$ **do**
2: $dist[u] = \infty$
3: $previous[u]$ = undefined
4: **end for**
5: $dist[s]= 0$
6: Q = all nodes in the graph
7: **while** Q is not empty **do**
8: u = the node with the smallest $dist[]$ in Q
9: remove u from Q
10: **for all** $v \in N(u)$ **do**
11: alt = $dist[u]$ + w(u,v)
12: **if** alt< $dist[v]$ **then**
13: $dist[v]=$ alt
14: $previous[v]= u$
15: **end if**
16: **end for**
17: **end while**

SLP (Shortest Legal Paths)

The SLP algorithm ensures that each packet from a source node s to a destination node d ($s, d \in V$) will traverse the network in a shortest legal path. Consider a node w on the path to d and a packet arriving to w destined to d. In general, packets that arrive to w from neighbors with different roles may be forwarded to different routes. For example, according to rules R1, R2, and R3, data that arrive to w from a customer is treated differently than data from a provider or a peer of w. This demand requires the use of an *extended forwarding table*.

To update its forwarding table any node $s \in V$ runs Dijkstra twice on G^*:

1. Starting from s_p^+. This case is for calculating shortest legal paths for data arriving to s from providers or peers. This phase updates $\bar{\mathcal{F}}_s$'s entries with `Internal` = False.

2. Starting from s_c^+. This case is for calculating shortest paths for data arriving to s from customers and for data that originate in s. This phase updates $\bar{\mathcal{F}}_s$'s entries with `Internal` = True.

Two runs of Dijkstra are required because according to R1, R2, and R3, data that arrive to some node x from its customer is treated differently than data from a provider or a peer of x; therefore path calculation needs to be done separately for these two types of data. The SLP algorithm is formally described in Algorithm 3.

The main theorem of this section is the formal claim that SLP indeed finds shortest legal paths.

Theorem 3.2 *Let G be an Internet graph. If all the nodes in G calculate their forwarding table according to the SLP Algorithm, then every packet will traverse the network in the shortest legal path.*

Proof. Let us consider the path a packet traverses starting from node $s = a_0$ and destined to node $d = a_n$: a_0 calculates the path $(a_0, a_1, \ldots, a_{n-1}, a_n)$ in G that is a shortest

Algorithm 3 SLP - Shortest Legal Paths

Require: $G = (V, E, w, T)$ network graph, built from the link-state advertisements of all the nodes in the network.

Ensure: An extended forwarding table $\bar{\mathcal{F}}_s$.

1: build $G^* = (V^*, E^*, w^*)$ from G according to Section 3.3.1
2: run Dijkstra on G^* with node s_p^+ as the source and get the shortest-paths tree T
3: **for all** destination node $u_{dst} \in V^*$ **do**
4: insert the following entry to $\bar{\mathcal{F}}_s$: `Internal` = False, destination = u, next-hop = from T
5: **end for**
6: run Dijkstra on G^* with node s_c^+ as the source and get the shortest-paths tree T
7: **for all** destination node $u_{dst} \in V^*$ **do**
8: insert an entry to $\bar{\mathcal{F}}_s$: `Internal` = True, destination = u, next-hop = from T
9: **end for**

legal path. It is legal because all the paths in G^* are legal (proved by Theorem 3.1) and shortest since we used the Dijkstra algorithm to find it. Now we need to prove that the path calculation is consistent.

Definition 3.5 (consistent path) *A path $(a_0, a_1, \ldots, a_{n-1}, a_n)$ in G from s to d is consistent iff for all k, $0 < k < n$, the path calculated in a_k for data that come from a_{k-1} and destined to a_n is $(a_k, a_{k+1}...a_{n-1}, a_n)$.*

For any $0 < k \leq n$ let $u = a_k$, there are 2 scenarios:

1. *a_{k-1} is a customer of u*: in this case s's (a_0) shortest path to d (a_n) in G^* will include u_c^+ because a_{k-1} is a customer of u as shown in Fig. 3.2 (c). Now, u will calculate the next hop for data that come from a^{k-1} by running Dijkstra on G^* starting from u_c^+ (`Internal` = True). The path $(u_c^+, \ldots, d_{dst})$ in G^* that u finds is the shortest path (by Dijkstra). This path is identical to the sub-path $(u_c^+, \ldots, d_{dst})$ found by s otherwise contradicting the optimally of

the path $(s, \ldots, u_c^+, \ldots, d_{dst})$ found by s.

2. *a_{k-1} is a provider or peer of u*: we use a similar argument but here the shortest path that starts from s to d will include u_p^+.

We proved that any node will find the shortest legal path for any destination and that the path is consistent, which concludes the proof of Theorem 3.2. ■

SCLP (Shortest Customer-Preferred Legal Paths)

SLP uses extended forwarding tables since it requires information about the previous hop of a packet. Next, we present an alternative algorithm, SCLP, that eliminates the need for information about the packet's previous hop, and thus enables the use of *standard forwarding tables*. The downside of SCLP is that it finds *shortest customer preferred legal paths* instead of shortest legal paths. A customer preferred path is a path that always goes via a customer if possible, formally:

Definition 3.6 (Customer Preferred Legal Path) *A legal path $P = (a_0, a_1, \ldots, a_n)$ from node $s = a_0$ to node $d = a_n$ is a legal* customer preferred *path iff for all $0 \leq i < n$, a_{i+1} is* not *a customer of a_i iff there is no legal path from a_i to d that starts with one of $a_i's$ costumers.*

Customer preferred legal paths are similar to the preference rule given by Gao and Rexford in [7] that instructs BGP speakers to prefer customer routes over peer or provider routes to make the protocol more stable. Essentially, the difference between SLP and SCLP is that for every destination SLP can hold up to two entries in its forwarding table while SCLP reduces this to exactly one entry by either removing the non-customer entry when two entries exist or updating the non-customer when it is the only entry. When the only entry is the customer entry SCLP will take it as it is. In SCLP any node $s \in V$ first modifies G^* by changing the weight of every edge $\langle x_c^+, x_p^- \rangle$ from 0 to `MaxPath` (which is a value larger than the cost of any path in the network). Then s runs Dijkstra once on the modified G^* and updates its forwarding table. The Dijkstra starts from s_c^+; note that shortest paths found

by Dijkstra will use non-provider-customer edges on paths only at nodes that cannot reach the destination via customers, exactly as required by the customer preferred path definition. The SCLP Algorithm is described in Algorithm 4 and we can prove the following theorem about its correctness.

The operation of SCLP in comparison to SLP can be described in the following way: For each destination SLP creates 2 entries in the forwarding table (one with `Internal` =false, which represents paths through customers only and one with `Internal` =true, which represents paths through all of the links). SCLP picks one of these entries and drops the `Internal` parameter in the following way:

1. If the entry with `Internal` =false contains the value NULL (meaning no possible path through customer) then SCLP picks the entry with `Internal` =true and removes the entry with `Internal` =false.

2. If the entry with `Internal` =false contains a value different from NULL (meaning there is a path through customer) then SCLP picks the entry with `Internal` =false and removes the entry with `Internal` =true.

Algorithm 4 SCLP - Shortest Customer-preferred Legal Paths

Require: $G = (V, E, w, T)$ network graph, built from the link-state advertisements of all the nodes in the network.

Ensure: The forwarding table $\mathcal{F}_s$.

1: build $G^* = (V^*, E^*, w^*)$ from G according to Section 3.3.1.
2: for each edge $\langle x_c^+, x_p^- \rangle$ in G^* change its cost to `MaxPath` .
3: run Dijkstra on G^* with node s_c^+ as the source and get the shortest-paths tree T.
4: **for all** destination node $u_{dst} \in V^*$ such that $dist[u_{dst}] < \infty$ **do**
5: insert an entry to $\mathcal{F}_s$: for destination u the next hop is according to T
6: **end for**

We now prove the properties of SCLP:

Theorem 3.3 *Let G be an Internet graph. If all the nodes in G calculate their forwarding table according to SCLP, then standard forwarding tables can be used and every packet will traverse the network in a legal path and reach its destination.*

Proof. We start by proving the guaranteed delivery part of the Theorem in Lemma 3.5 by induction on the path a packet traverses and with the help of Lemma 3.3 and Lemma 3.4. We then prove in Lemma 3.6 that a packet will traverse the network only in a legal path and that nodes don't need any information on the previous node of a packet, which means *standard forwarding tables* can be used.

Lemma 3.3 *If P=($a_1, a_2 ... a_{n-1}, a_n$) is a legal path and a_2 is a customer of a_1 then $\forall i$ $1 \leq i < n$, a_{i+1} is a customer of a_i.*

Proof. We will prove this by induction:
Base case - a_2 is a customer of a_1.
Induction hypothesis - $\forall i$ $1 \leq i \leq k < n$, a_{i+1} is a customer of a_i.
Inductive step - We now show that if the inductive hypothesis is true, then $\forall i$ $1 \leq i \leq k+1$, a_{i+1} is a customer of a_i: a_{k-1} is provider of a_k (induction hypothesis) and a_k's policy is to give its providers transit only towards its customers (R2); therefore a_{k+1} must be a customer of a_k. ■

Lemma 3.4 *Let all the nodes in a network G calculate their forwarding table according to SCLP and node a_1 calculates a legal path P=($a_1, a_2 ... a_{n-1}, a_n$). Then if every node a_i, $\forall i$ $1 \leq i < n$ picks the entry with the* ***same*** `Internal` *value (either true or false) then P is a consistent path.*

Proof. Let us consider the path a packet traverses starting from node a_1 and destined to node a_n: a_1 calculates the path $(a_1, a_2 ... a_{n-1}, a_n)$. Now let us consider these two scenarios:

1. Every node a_i, $\forall i\ 1 \leq i < n$ picks the entry with `Internal` =true. For each node $a \in V$ the `Internal` =true entry is updated by the Dijkstra run on G^* starting from a_c^+, which is connected to a_c^- and a_p^-, meaning it can use any of the outgoing links from a to its neighbors. Therefore, in this case, every node a_i, $\forall i\ 1 \leq i < n$ chooses the absolute shortest legal path to a_n and since they all calculate it with the same Algorithm (Dijkstra) on the same graph (G^*) it is consistent.

2. Every node a_i, $\forall i\ 1 \leq i < n$ picks the entry with `Internal` =false. In this case a_2 must be a customer of a_1 and therefore $\forall i\ 1 \leq i < n$, a_{i+1} is a customer of a_i (Lemma 3.3). That means that every node a_i, $\forall i\ 1 \leq i < n$ chooses the *shortest legal customer preferred path* to a_n and since they all calculate it with the same Algorithm (Dijkstra) on the same graph (G^*) it is consistent.

■

Lemma 3.5 *If all the nodes in a network calculate their forwarding table according to SCLP, then a packet from any source node to any destination node will get to the destination iff a legal path between the source and the destination exists.*

Proof. Let us consider the path a packet traverses starting from node a_1 and destined to node a_n. a_1 calculates the path $(a_1, a_2...a_{n-1}, a_n)$. Let a_c be the first node in the path (the node with the smallest index), which has a legal path to a_n through one or more of its customers. We now examine 2 possible scenarios and relate them to the comparison to SLP (as explained above):

1. a_c doesn't exist. In this case all the nodes in the path $(a_1, a_2...a_{n-1}, a_n)$ pick the entry with `Internal` =true and the path is a *consistent path* as proven in Lemma 3.4 and a packet is guaranteed to arrive to a_n.

2. a_c exists. In this case all the nodes in the path $(a_1, a_2...a_c)$ pick the entry with `Internal` =true and the resulting path for a packet from a_1 to a_c is a *consistent path* as proven

in Lemma 3.4; therefore a packet is guaranteed to arrive to a_c. a_c will calculate the path $(a_c, b_1...b_k, a_n)$ to a_n in which all the nodes pick the entry with `Internal` =false (Lemma 3.3) and the resulting path for a packet from a_c to a_n is a *consistent path* as proven in Lemma 3.4; therefore a packet is guaranteed to arrive to a_n.

■

Lemma 3.6 *If all the nodes in a network calculate their forwarding table according to SCLP, then every packet will traverse the network in a legal path and standard forwarding tables can be used.*

Proof. Let us consider the path a packet traverses starting from node a_1 and destined to node d while a_i denotes the i'th node the packet visited. We will show by induction that every hop the packet traverses is a legal move and that *standard forwarding tables* can be used:

Base case - The first move is from a_1 to a_2, which is legal since a_1 is the origin and therefore any move from it is legal. Trivially, no information is needed on the previous node since a_1 is the origin node.

Induction hypothesis - All the moves from a_1 to a_k are legal and do not require any information on the previous node.

Inductive step - We now show that if the inductive hypothesis is true then all the moves from a_1 to a_{k+1} are legal and do not require any information on the previous node. For that we just need to show that the move from a_k to a_{k+1} is legal and does not require information on a_{k-1}. Let us consider these 2 possible scenarios for the move from a_k to a_{k+1}:

1. d is reachable via one or more of a_k's customers. In this case a_{k+1} is a customer of a_k because SCLP gives preference to paths through customers and since all of a_k's neighbors get transit access to all of a_k's customers (according to rules R1, R2, R3

presented in Section 3.1) any arriving packet can be legally forwarded to a_{k+1} regardless of the type of its previous node (a_{k-1}).

2. d is not reachable via any of a_k's customers. In this case a_{k+1} is a provider or a peer of a_k, but it is guaranteed that any arriving packet destined to d arrived from a customer of a_k because any other option yields an illegal path that cannot be found on G^* (proved by Theorem 3.1); therefore any arriving packet can be legally forwarded to a_{k+1} regardless of the type of its previous node (a_{k-1}).

We have proved that every packet will traverse the network in a legal path and there is no need for any information on the previous node of a packet in the forwarding table and therefore *standard forwarding tables* can be used. ■

This concludes the proof of Theorem 3.3 which describes the properties of SCLP. ■

3.3.4 Example

Overview

In this section we show the operation of PSP inside the hierarchy presented in Figure 3.4 and we consider two cases:

1. PSP with SLP.

2. PSP with SCLP.

Both SLP and SCLP Algorithms are presented in Section 3.3.3.

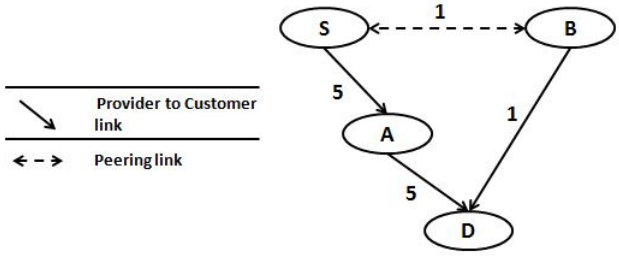


Figure 3.4: An example of network graph G.

Policy Graph Construction

First, each node sends its LSA, which is then propagated throughout the network.
Each node then constructs the policy graph shown in Figure 3.5 according to Section 3.3.1 and performs a path selection algorithm on it. In this example we will focus on the selected path for data from node S to node D. For the simplicity of presentation we will refer to an induced graph of G^* shown in Figure 3.6, which consists of only the nodes and edges that are relevant for data from S to D (we removed all incoming edges of node S and all outgoing edges of node D and also all the irrelevant nodes).

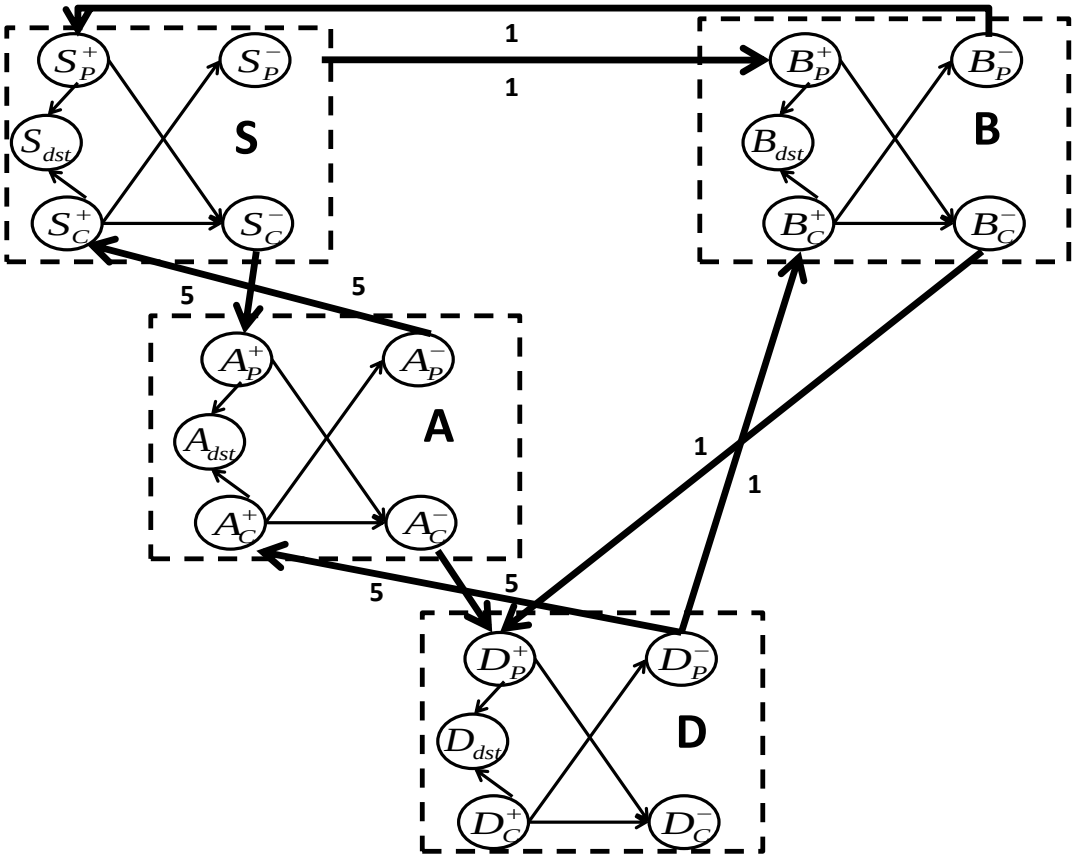


Figure 3.5: The policy graph G^*.

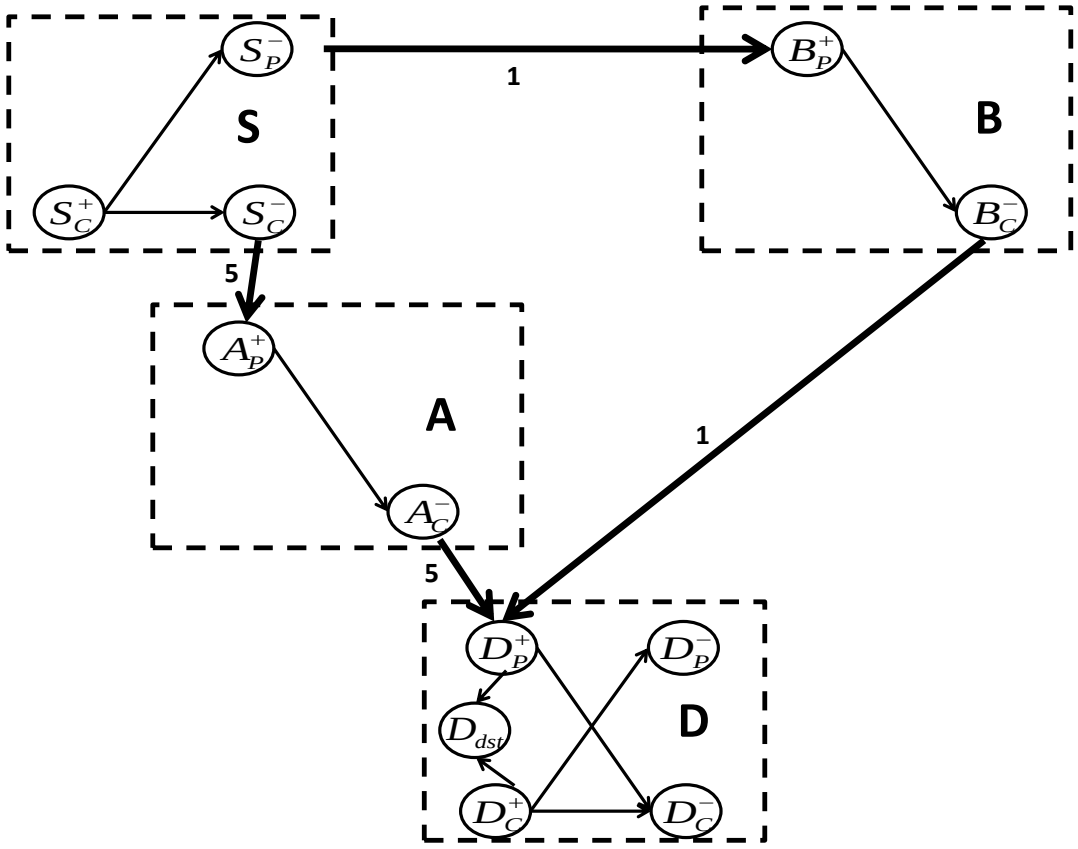


Figure 3.6: An induced graph of the policy graph G^*. PSP with SLP will result in the path $S \rightarrow B \rightarrow D$ and PSP with SCLP will result in the path $S \rightarrow A \rightarrow D$ for data from node S to node D.

PSP with SLP

In this case the path calculation process regarding data that originates in node S and destined to node D will be as follows:

1. Node S will run Algorithm 2 (Dijkstra) on G^* starting from node s_c^+ since it is the origin of the data (`Internal` =true). The selected path will be $S \rightarrow B \rightarrow D$ and therefore the data will be forwarded to node B.

2. Node B will run Algorithm 2 (Dijkstra) on G^* starting from node B_p^+ since node S is a peer of node B (`Internal` =false). The selected path will be $B \rightarrow D$ and therefore the data will be forwarded to node D.

As we can see, in the example network presented in Figure 3.4, if all the nodes operate according to PSP with SLP, then data from node S to node D will traverse the network in the path $S \rightarrow B \rightarrow D$ which is the *shortest legal path*.

PSP with SCLP

In this case the path calculation process regarding data destined to node D will be as follows:

1. Node S will modify the weight of edges $< S_p^-, B_p^+ >$ and $< B_c^-, D_p^+ >$ to `MaxPath` and run Algorithm 2 (Dijkstra) on the modified G^* starting from node S_c^+. Since node D is reachable via a customer of S the selected path will be $S \rightarrow A \rightarrow D$ and therefore the data will be forwarded to node A.

2. Node A will modify the weight of edges $< S_p^-, B_p^+ >$ and $< B_c^-, D_p^+ >$ to `MaxPath` and run Algorithm 2 (Dijkstra) on the modified G^* starting from node A_c^+. Since node D is reachable via a customer of A the selected path will be $A \rightarrow D$ and therefore the data will be forwarded to node D.

As we can see, in the example network presented in Figure 3.4, if all the nodes operate according to PSP with SCLP, then data from node S to node D will traverse the network in the path $S \rightarrow A \rightarrow D$ which is the *shortest legal customer preferred path*.

3.4 Inter-Region

3.4.1 Overview

PSP supports the partitioning of the network into arbitrary regions (groups of ASes), which in turn reduces the broadcasting of advertisements in the network and makes the protocol more scalable. In each region nodes run one of the previously mentioned routing algorithms (SLP or SCLP) on the region graph, which is composed of two parts: an internal part (the region's nodes and edges) and an external part (edges to nodes that do not belong to the region). Aggregation of advertisements for external paths allows the division into regions to significantly reduce the number of advertisements and the sizes of the forwarding tables.[1] Fig. 3.7 shows an example of a multi-region network, a region graph, and a region graph with advertisements aggregation.

In this section we first introduce some new definitions (Section 3.4.2), explain what each node advertises to its neighbors (Section 3.4.3) and how it builds the network graph of its region (Section 3.4.4), and finally we prove the properties of a partitioned network that works with PSP (Section 3.4.5).

3.4.2 Definitions

Definition 3.7 (region) *A region R is a set of connected nodes in the network graph G. Nodes inside a region run between them an inter-domain link-state protocol such as SLP or SCLP (Section 3.3.3).*

Definition 3.8 (network partitioning) *A network partitioning $\Re(G)$ of a network graph G=(V,E,w,T) is a set of regions $(R_1, R_2...R_n)$ with edge sets $(E_1, E_2...E_n)$, when E_i is the edge set of region R_i and vertex sets $(V_1, V_2...V_n)$, when V_i is the vertex set of region R_i that follows these conditions:*

[1] The discussion about the aggregation method is out of the scope of this thesis.

1. $E_1 \cup E_2 \cup ... \cup E_n = E$.

2. *All the edge sets of the regions are disjoint to each other* ($\forall i, j \in [1...n], i \neq j$, $E_i \cap E_j = \phi$).

3. $V_1 \cup V_2 \cup ... \cup V_n = V$.

Note that the above conditions mean that a node can belong to several regions but each edge belongs to exactly one region and that neighboring nodes belong to a common region.

3.4.3 Advertisements

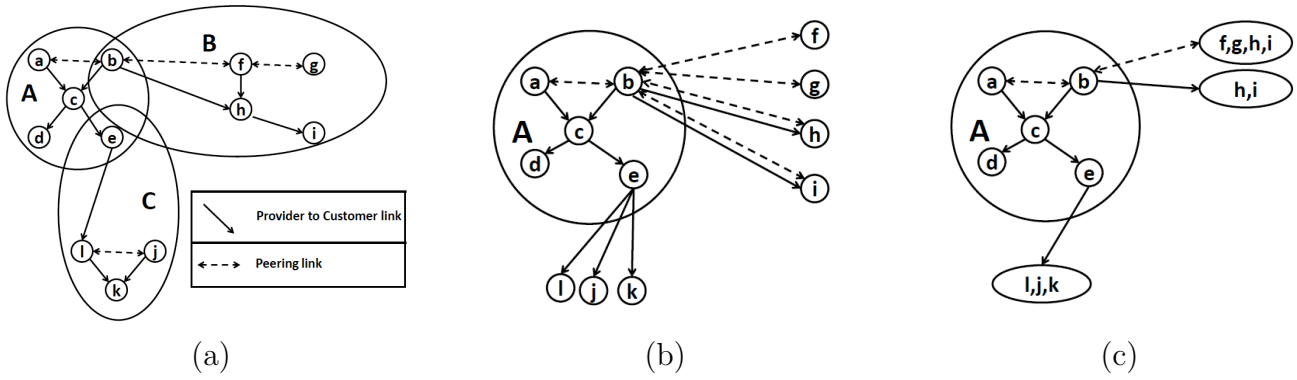
A node $u \in G$ in the network will operate as follows: Let $R^u = \{R^u_1, R^u_2 ... R^u_j\}$ be the set of region node u belongs to. For every region R^u_k, $1 \leq k \leq j$, node u will advertise LSAs to all the nodes in region R^u_k. Each LSA can be one of the following two types:

1. Internal entries - For each neighbor v of node u s.t. $v \in R^u_k$, the LSA will contain: v, the `Role` of u in the link between (u, v), and the link cost (i.e., $w(u, v)$).

2. External entries - For each entry in u's forwarding table for a destination node w and the next hop v toward the destination is not in R^u_k, the LSA will contain: w, the `Role` of node u in the link between u and v, the total cost of the path from node u to w. Note that in case PSP is operating with SLP, then u may have two entries in its forwarding table for each destination (for different roles). This means that u will include two entries for each destination in its advertised LSA.

3.4.4 Building Region Network Graph

The region graph $G_i = (V'_i, E'_i)$ of a region R_i is the graph each node in region R_i builds upon receiving the LSAs of its region. It includes two parts:



(a) (b) (c)

Figure 3.7: (a) A possible partitioning of a network to regions. (b) The region network graph of region A without aggregation. (c) The region network graph of region A with aggregation.

1. Internal part - The internal vertex set V_i is the set of nodes of the region and the internal edge set E_i is a subset of E (the edge set of the network graph G). Note that the internal part must be connected (Definition 3.7). In the example shown in Fig. 3.7 (b), the internal part is surrounded by a circle.

2. External part - The external vertex set V_i^e has all the external nodes (nodes where the next hop does not belong to region R_i) known to the nodes of region R_i by advertisements they got from other regions they belong to. Each node in V_i^e is attached by one or more edges to nodes in V_i where each such edge is a virtual edge that represents the full path from a region's R_i node that got the advertisement for the external node. These edges form the external edge set E_i^e. In the example shown in Fig. 3.7 (b), the external part is composed of all the edges and nodes that are outside the circle.

The nodes and edges of G_i are now: $V_i' = V_i \cup V_i^e$ and $E_i' = E_i \cup E_i^e$.

To update its forwarding table, each node v that belongs to a region R_i builds its region network graph $G_i = (V_i', E_i')$. From this graph v builds its policy graph G_i^* as explained in Section 3.3.1, performs the Dijkstra runs on it according to the protocols defined to work in region R_i - SLP or SCLP - and updates its forwarding table accordingly. Note that since node

v has a single forwarding table but can belong to several regions, all the regions v belongs to must run the same protocol - SLP or SCLP - because each of them requires a different structure of a forwarding table. This limitation effectively means that all the regions in the network must run the same protocol since by the network partitioning definition (Definition 3.7) each two neighboring nodes belong to a common region.

3.4.5 Properties of a Partitioned Network

In this section we prove that in a partitioned network all packets will get to their destination in the shortest legal customer preferred path (if all the regions work with PSP ith SCLP) or in the shortest legasl path (if all the regions work with PSP ith SLP).

Theorem 3.4 *Let G be an Internet graph and $\Re(G)$ be a network partitioning of G into regions. If all the nodes in G use PSP with SLP (SCLP) protocol to calculate their forwarding tables, then a packet from any source node to any destination node will get to the destination in the shortest legal path (shortest customer preferred legal path).*

Proof. Since the proofs of the SCLP case and the SLP case are very similar, we will start with the proof of the SCLP case and then explain the difference with the SLP case.
Let a_1 and a_n be source and destination nodes (respectively) in G, and P=$(a_1, a_2..., a_n)$ be the shortest legal customer preferred path from a_1 to a_n.
We will now prove by induction that each node $a_i \in P$ finds the shortest legal customer preferred path from it to a_n, meaning that the actual path a packet from a_1 to a_n will traverse is exactly P.
Base case - Since a_{n-1} and a_n are neighbors they belong to a common region (Definition 3.8) and since SCLP runs inside a region and finds shortest legal customer preferred paths (proven in Section 3.3.3), node a_{n-1} will find the shortest legal customer preferred path from it to a_n, which in this case is the edge (a_{n-1}, a_n).

Induction hypothesis - The induction hypothesis is that every node a_j, $1 < k \leq j < n$ finds the shortest legal customer preferred path from it to a_n.

Inductive step - We now show that node a_{k-1} also finds the shortest legal customer preferred path from it to a_n. We consider 2 cases:

1. All the nodes a_j, $1 \leq k-1 \leq j \leq n$ belong to the same region R. In this case all the nodes in P belong to R and therefore all the edges of P belong to R, and since SCLP runs inside a region and finds shortest legal customer preferred paths (proven in Section 3.3.3), node a_{k-1} will find the shortest legal customer preferred path from it to a_n.

2. Nodes a_j, $1 \leq k-1 \leq j \leq n$ do not belong to a common region. In this case we consider node a_l, which is the node with the highest index, such that all nodes a_m, $k-1 \leq m < l$ belong to the same region (l must be at least k since neighboring nodes belong to a common region). Since a_l knows the shortest legal customer preferred path to node a_n (induction hypothesis) it will advertise a_n to a_{k-1} as an external node with a virtual link between a_l and a_n that represents the whole path from a_l to a_n. Node a_{k-1} will then find the shortest legal customer preferred path to a_n by running SCLP on its policy graph in which a_n appears as an external node.

This concludes the proof for the SCLP case. in the SLP case, we use the same proof but we must also state that since each node has two entries in its forwarding table for each destination, it advertises these two entries and therefore each node that gets these advertisements can calculate shortest legal paths for each destination and for the different types of previous hops in SLP (data from peers/providers or data from customers/data that originates in the node).

We have shown that each node $a_i \in P$ finds the shortest legal customer preferred path from it to a_n, meaning that the actual path a packet from a_1 to a_n will traverse is exactly P. ■

Chapter 4

Simulation

To study the performance of PSP we used a real life dataset of connections between the ASes in the Internet taken from CADIA [3] and a Java program we wrote using the JGraphT package [21]. The results presented in the next sub-section deal with the differences in path lengths between SLP and SCLP. The second set of results is presented in Section 4.2 and deals with the partitioning of the graph into regions and its influence on the number of protocol messages.

The dataset contains all the connections between ASes in the Internet with their types: provider-customer, peer-peer, or siblings. We considered sibling ASes as a single AS so the set includes 33382 different ASes and 80250 links. Both simulations are based on an Internet graph that was built from the mentioned dataset.

4.1 Path Calculations

In this simulation we compared path lengths between our two proposed routing algorithms - SLP and SCLP. Since we have already proved that both algorithms keep their shortest paths properties in a partitioned network (Theorem 3.4), we considered the network graph in this

simulation as a single region.

The simulation was constructed in the following way - first we built the global policy graph from the dataset (in case the network is not partitioned into regions the policy graph is identical in all the nodes, hence it is global), then we selected a node u (the selection method will be explained further in this section), and calculated the paths from u to any other node in the network twice - once with SLP and once with SCLP. The calculation of paths was done by running Dijkstra on the policy graph (SLP) or modified policy graph (SCLP) starting from u_c^+ as this is the starting node in the path calculation for data from u to any other node in the network. The meaning of modified policy graph is that the weight of every edge $\langle x_c^+, x_p^- \rangle$ was changed to `MaxPath` in order to ensure the selection of shortest customer preferred paths (as described in Section 3.3.3). After the execution of Dijkstra, path lengths were calculated by hop-count.

The node selection process chose nodes for path calculation by the number of customers the node has in descending order. The reason is that according to rules R1-R3 nodes with no customers can not transmit any data and can only be the data origin or destination; therefore they are less interesting to measure. In addition we considered nodes by their *tier*. A node has tier-x according to the following definition:

Definition 4.1 (tier) *Node u's tier is determined according to the following rules:*

1. *If u has no providers, then u is a tier 1 node*[1]*.*
2. *If u has a set of providers and its provider with the lowest tier is a tier x node, then u is a tier x+1 node.*

Overall we measured path lengths from 11540 nodes (to all nodes), which are 34.5% of the whole network, and the results were normalized to the whole network by weighting the result for each tier by its size.

[1]Nodes with no providers and no customers are considered tier 2 nodes.

	Nodes	SLP shorter paths	Difference in hop count	Difference in percentage
Tier 1	14	38%	0.61	17%
Tier 2	9496	23%	0.33	9%
Tier 3	17099	48%	1.03	21%
Tier 4	6002	50%	1.01	18%
Tier 5	733	61%	1.37	21%
Tier 6	38	50%	0.75	11%
Network	33382	**41.9%**	**0.84**	**18.1%**

Table 4.1: Path length comparison between SLP and SCLP

Our major finding is that while the average path length of SLP is 3.78 hops, SCLP average path length is 4.62. This is an improvement of about 18% or 0.84 hops per node. If a node sends one message to each destination, SLP reduces the number of hops traversed in the network by $28,000$ on average. Moreover SLP finds shorter paths for about 42% of all source-destination paths in the network. These results and the results for each tier are presented in Table 4.1 and Fig. 4.1.

Each row in Table 4.1 represents a tier x and has the following information: The column **Nodes** represents the number of nodes in the tier. The column **SLP shorter paths** represents the percentage of source-destination pairs in which SLP's path is shorter than SCLP's path out of all source-destination pairs whose source is a tier x node. The columns **Difference in hop count** and **Difference in percentage** represent the average improvement of path length in SLP compared to SCLP in hop count and in percentage, respectively. The last row shows the interpolated result for the whole network. Note that tiers $2, 3, 4$ have almost 98% of the nodes, and among them the improvement in tiers $3, 4$ is much more significant that in tier 2.

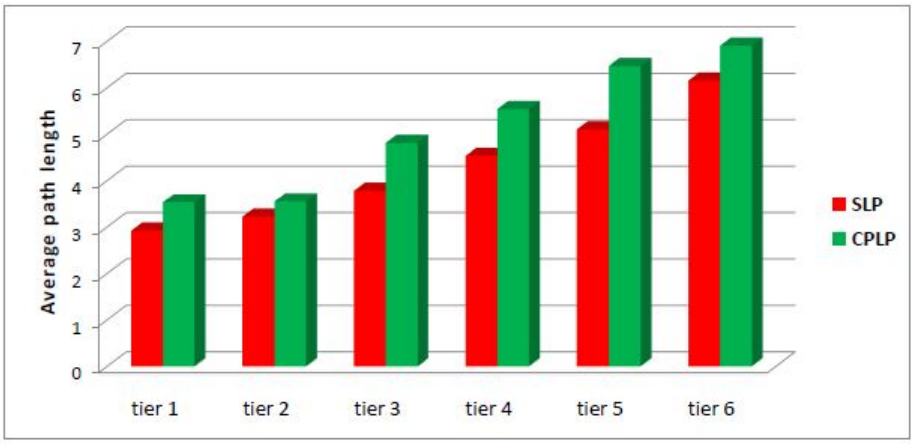


Figure 4.1: Average path lengths in SLP and SCLP by the node's tier.

Fig. **??** clearly shows that the average path length, both in SLP and in SCLP, increases with the node's tier. From this we conclude that the majority of the paths go through lower tier nodes (than the source and destination), as expected. This also explains the improvement of SLP since SCLP enforces the use of a customer's path whenever it is possible.

Fig 4.2 shows the average path lengths in SLP and SCLP by the number of customers for each node.

Fig 4.3 shows a histogram of path lengths in SLP and SCLP. The histogram shows that most of paths found by SLP are 3-4 hops long while most of the paths found by SCLP are 4-5 hops long.

Fig. **??** shows the path length improvement of SLP compared to SCLP in percentage as a function of the number of customers a node has. Each bar represents a range of number of customers in increasing order and is comprised of different colors; each represents the weight of the corresponding tier in the total. From the figure we can conclude that the improvement at nodes with many customers (51+) is larger (although there are fewer such nodes) and is mostly due to tier 2 nodes. The improvement in nodes with a lower number of customers is mostly due to tier 3 nodes, which hold the majority of the nodes.

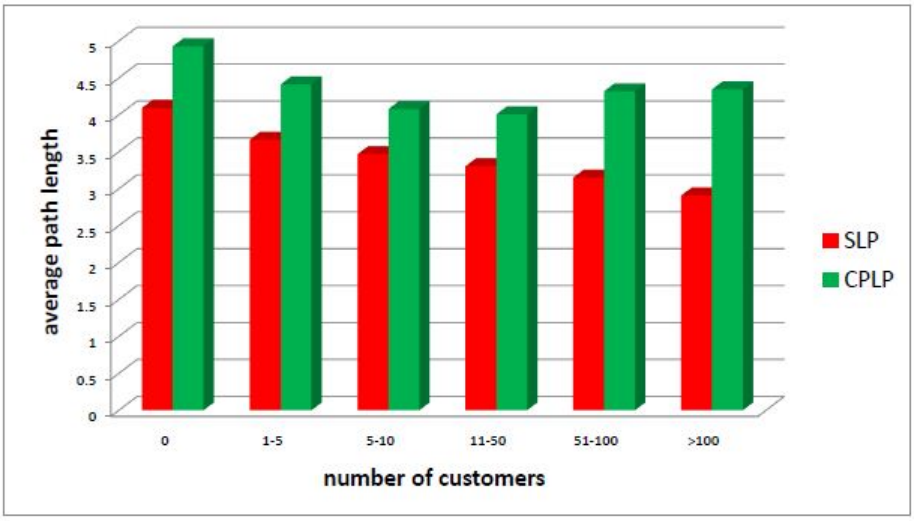


Figure 4.2: Average path lengths in SLP and SCLP by the number of customers for each node.

4.2 Regions and Protocol Messages

In this section we check the number of protocol messages each node transmits to get the network from a zero state (no node knows any other node) to a steady state (every node knows every other node) in various partitionings of the network into regions. For simplicity of presentation we used an aggregation method in which each region is advertised externally as a single message.

First we partitioned the network into roughly even size regions in a BFS-based method (we skip the details here). Let X be the number of regions in a specific partitioning of the network, then we calculated the number of messages each node transmits in the following way: In a region i with E_i edges and B_i border nodes (nodes that belong to more than one region), the protocol messages can be divided into 2 types:

1. Internal protocol messages - the link-state advertisements inside the region in which each edge in the region is advertised to all the region's nodes. Since every link (edge)

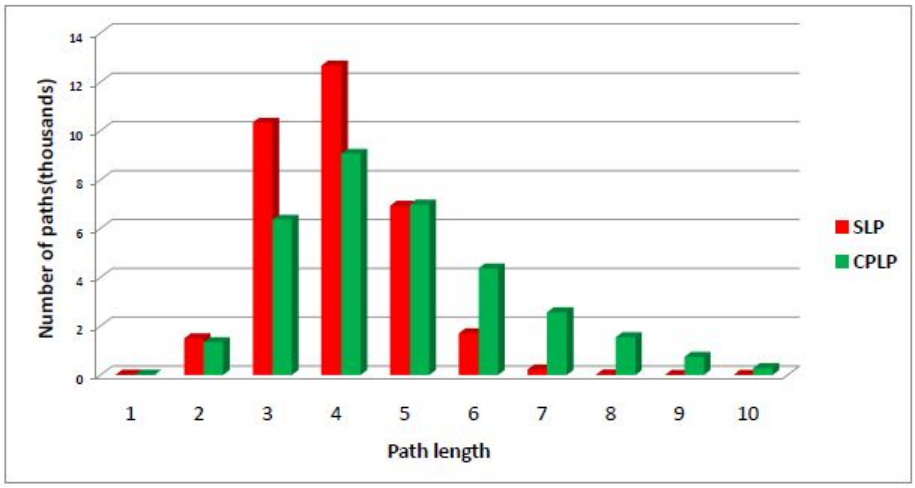


Figure 4.3: A histogram of path lengths in SCLP and in SLP.

is advertised to every node in the region the number of internal protocol messages in each region i is $M_{int}(i) = 2E_i * ((V_i - 1) + 2(E_i - (V_i - 1))) =$ `number-of-LSA`$_i$ * `broadcast-cost`$_i$.

2. External protocol messages - the path-state advertisements in which external destinations are advertised to the region's nodes by the border nodes of the region. Since we assumed here an aggregation method in which each region is aggregated as a single message, the number of external protocol messages in a region i is upper bounded as $M_{ext}(i) = X * B_i * ((V_i - 1) + 2(E_i - (V_i - 1))) =$ `regions * borders`$_i$ * `broadcast-cost`$_i$.

Note that we assume here that LSAs are broadcasted to the entire region via IS-IS-like flooding, i.e., at each node the LSA is forwarded once to each neighbor except the neighbor from which the LSA was first received. We also consider here that LSAs are of constant size and contain one link/path entry per LSA. In total, the number of transmitted protocol messages in each region i is $M_{total}(i) = M_{int}(i) + M_{ext}(i)$ and therefore the average number

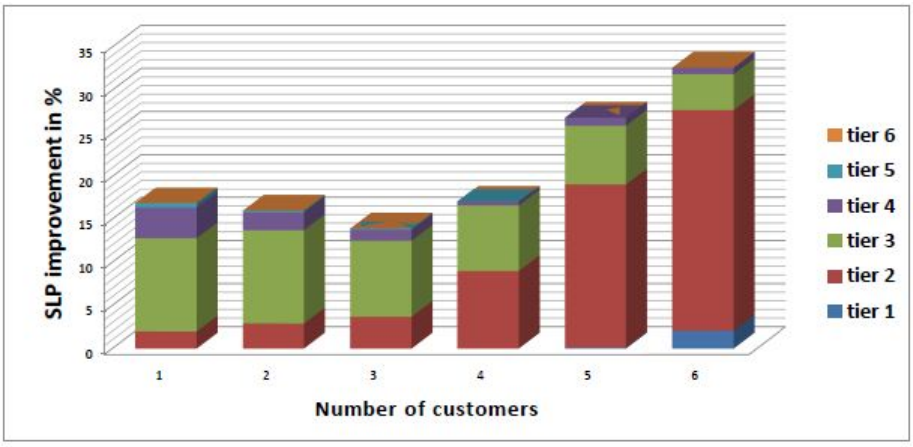


Figure 4.4: Improvement percentage of SLP in relation to SCLP by the number of customers a node has.

of transmitted protocol messages per node is:

$$\overline{M}_{node} = \sum_{i} \frac{M_{total}(i)}{|V|}$$

The results of the simulation are shown in Fig. 4.5, which gives the average number of messages per node as a function of number of regions. The left end of the graph represents the extreme case in which the network is partitioned into one region, which means it is effectively unpartitioned. In this case PSP actually behaves as a pure link-state protocol and the number of total protocol messages is maximal, $2|E| * ((|V| - 1) + 2(|E| - (|V| - 1)))$. The right end of the graph represents the other extreme case in which each edge in the network is a region. In this case the behavior of PSP would be similar to BGP since all the advertisements will be external and contain destinations and costs, as in a path-vector protocol.

The optimal partitioning of the network is achieved when the network is partitioned into 62 regions with an average size of 1077 nodes each. In this case the average number of protocol messages each node will transmit is 76.7% less than in an unpartitioned network and 57% less than in a fully partitioned network. We can see from the graph that if we

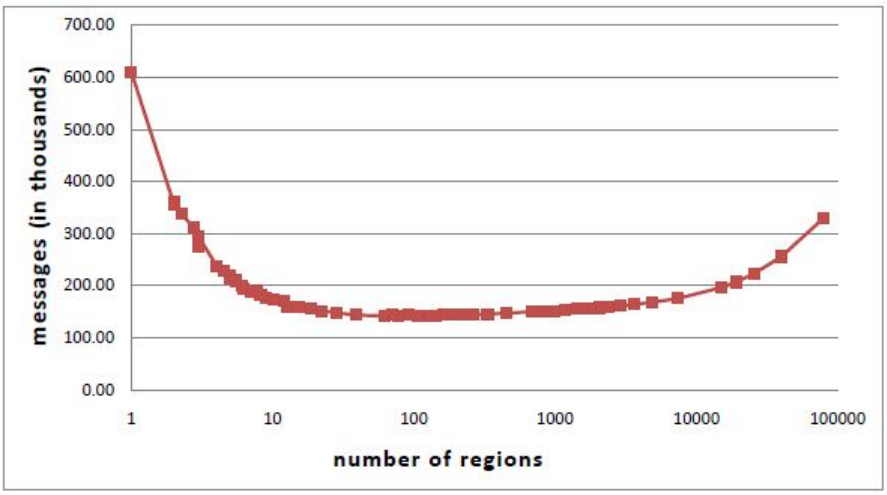


Figure 4.5: Number of protocol messages per node by the number of regions.

increase the number of regions, thus decreasing their size, the number of messages increases at a slow rate; therefore, instead of constructing regions of 1077 ASes, which may be hard to accomplish because of the complex business relations between so many ASes, we can partition the network into 1000 regions with an average size of 76 nodes each, for example. This will cost us 7% more protocol messages per node than in the optimal partitioning, but it will still be 75% less than in an unpartitioned network and 54% less than in a fully partitioned network.

Chapter 5

Conclusions and Future Work

In this thesis we present a novel type of routing protocol, Path-State Protocol (PSP), that combines both link-state and path-vector approaches. PSP supports policy-based routing and fits for inter-domain routing. We provide an initial study of PSP and acknowledge that further study is required.

Future research should reference the following issues:

- A full implementation of PSP that will give the possibility of testing it against other protocols, such as BGP and HLP. A full implementation can also provide more realistic results on the properties of the network partitioning into regions than the analytical estimations presented in this thesis and also better understanding of the quality of PSP's convergence time and stability, which are not yet well understood (and proved).

- Policy flexibility. Currently PSP does not give any freedom to nodes in choosing their routing policies as they are determined arbitrarily according to rules R1-R3, but by changing the protocol packets and the way G^* is built, policy flexibility can be improved.

- Interactions with intra-domain protocols and reference to router level. Issues such as

how PSP propagates the inter-domain information into an AS, how an internal router in an AS knows towards which of the PSP routers in its AS it should forward an inter-domain packet, and how PSP routers in the same AS synchronize between themselves should be addressed.

Overall, despite the issues mentioned above, the presentation of the protocol and the basic idea of path-state approach, together with our initial theoretical and simulation results, are important and could contribute to future discussion.

Bibliography

[1] [Online]. Available: http://www.cisco.com

[2] Cidr report, march 2010. [Online]. Available: http://www.cidr-report.org/

[3] The CAIDA AS relationships dataset. january, 2010. [Online]. Available: http://www.caida.org/data/active/as-relationships/

[4] P. F. Tsuchiya, "An architecture for network-layer routing in osi," *SIGCOMM Comput. Commun. Rev.*, vol. 17, no. 5, pp. 185–190, 1987.

[5] D. Mills, *RFC 0904, Exterior Gateway Protocol Formal Specification*, April 1984.

[6] L. T. REKHTER, Y. and S. HARES, *A Border Gateway Protocol 4 (BGP-4) Internet Draft draft-ietf-idr-bgp4-25.txt*, September 2004.

[7] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 681–692, 2001.

[8] *ISO/IEC 10589. Intermediate System to Intermediate System Intra-Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)*, 2nd ed., July 2000.

[9] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, 2002.

[10] N. Feamster, H. Balakrishnan, and J. Rexford, "Some foundational problems in interdomain routing," in *In Proc. of Workshop on Hot Topics in Networks*, 2004, pp. 41–46.

[11] M. Yannuzzi and X. Masip-bruin, "Open issues in interdomain routing: a survey," *IEEE Network*, vol. 19, pp. 49–56, 2005.

[12] B. Premore, "An experimental analysis of BGP convergence time," in *ICNP '01: Proceedings of the Ninth International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2001, p. 53.

[13] Y. Wang, M. Schapira, and J. Rexford, "Neighbor-specific BGP: more flexible routing policies while improving global stability," in *In SIGMETRICS '09*. New York, NY, USA: ACM, 2009, pp. 217–228.

[14] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, "HLP: a next generation inter-domain routing protocol," in *In SIGCOMM '05*, 2005, pp. 13–24.

[15] C. Huitema, *Routing in the Internet*. Upper Saddle River, NJ: Prentice-Hall, Inc, 2000.

[16] R. C. C. Villamizar and R. Govindan, "Bgp route flap damping, rfc 2439," Novenmebr 1998.

[17] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 277–288, 1999.

[18] M. Caesar, "Root cause analysis of BGP dynamics," Master's thesis, University of California at Berkeley, 2004.

[19] R. Sami, M. Schapira, and A. Zohar, "Searching for stability in interdomain routing," in *INFOCOM*, 2009, pp. 549–557.

[20] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," *Computer Networks*, vol. 32, no. 1, pp. 1 – 16, 2000.

[21] [Online]. Available: http://www.jgrapht.org/