Interdomain Routing Without Instabilities

David Romão Fialho

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. João Luís Costa Campos Gonçalves Sobrinho

Examination Committee

Chairperson: Prof. António Manuel Raminhos Cordeiro Grilo
Supervisor: Prof. João Luís Costa Campos Gonçalves Sobrinho
Member of the Committee: Prof. Ricardo Jorge Feliciano Lopes Pereira

May 2017
Acknowledgments

This thesis was a major milestone in my life and it would not have been possible without the help and support of a large number of people that deserve my greatest appreciation.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. João L. Sobrinho, for his guidance and support during this adventure. Your immense knowledge, enthusiasm, clear way of thinking, and attention to detail were a source of inspiration and motivation. I am also grateful for your great availability and endless patience, without which I would not be able to complete this work. I can only hope that our future endeavors remain as interesting and enjoyable as this one.

Secondly, I would like to thank Instituto de Telecomunicações for their financial support and everyone at Instituto Superior Técnico, in particular to all my teachers who, during the last five years, contributed to enrich my knowledge and to cultivate my interest in computer science.

I would also like to address all my friends. I am extremely thankful for having such a caring and supportive group of friends, that are able to accept me in all of my idiosyncrasies and peculiarities. Particularly, to my friends at Técnico thank you for turning every boring and dreadful task into fun moments. I will never forget the hours we spent working together, particularly, the late nights before deadlines.

My family deserves a special word of praise and gratitude. They have always provided me with unfailing support and continuous encouragement throughout my life and specially during the time I was completing my degree. I would also like to thank my parents for raising me, helping me, and giving me every opportunity to pursue my goals and interests. Mum, thank you very much for helping me review the text for errors, even though you did not understand the content. Dad, thank you very much for the help you provided during this time and for the motivation you gave me without which this challenge would have been much harder.

Last but no least, I would like thank my girlfriend Sofia for her amazing support and understanding during this endeavor. I cannot thank you enough for putting up with me and my grumpiness in the most stressful moments and for all the “lost” weekends and evenings you had to endure.
Abstract

The Internet is comprised of a large set of Autonomous Systems (ASs). Every AS participates in a single routing protocol, called Border Gateway Protocol (BGP), allowing it to learn paths to communicate with each other. BGP allows AS administrators to set a wide range of routing policies autonomously. However, it does not give any stability guarantees.

This work investigates the relationship between routing policies and routing loops, and the effect they have on BGP’s ability to reach a stable state. In particular, this work focuses on stabilizing BGP for isotone routing policies. An addon to BGP is proposed, called Self-Stable BGP (SS-BGP), which guarantees termination with isotone routing policies. SS-BGP adds to BGP the ability to distinguish between temporary routing loops and persistent routing loops, and to eliminate the latter. Gradually eliminating persistent routing loops ensures termination in the case isotone routing policies, and improves the stability of BGP in general.

The performance and effectiveness of SS-BGP are evaluated through simulation. For that purpose, a new discrete-event simulator was developed able to simulate the dynamic behavior of BGP and SS-BGP in large-scale topologies, like the Internet topology, with a wide range of routing policies. A large number of simulations were conducted for both BGP and SS-BGP in realistic topologies of the Internet with realistic perturbations of the baseline interdomain routing policies. Results show that SS-BGP terminates in all cases, in contrast to BGP, with only a small percentage of ASs having to change their import policies.

Keywords: Internet, BGP, routing oscillations, protocol stability, isotonicity
Resumo

A Internet é composta por um conjunto de Autonomous Systems (ASs). Todas as ASs participam num único protocolo de encaminhamento, chamado Border Gateway Protocol (BGP), que lhes permite aprender caminhos para chegar aos vários destinos. O BGP permite que os administradores de ASs definam um alargado leque de políticas de encaminhamento de forma autónoma. Contudo, não dá garantias de estabilidade.

Este trabalho investiga a relação entre as políticas de encaminhamento e as aselhas, e o impacto que estas têm na capacidade do BGP estabilizar. Em particular, este trabalho foca-se em estabilizar o BGP para a políticas de encaminhamento isótonas. É proposto um *addon* ao BGP, chamado de Self-Stable BGP (SS-BGP), que garante terminação para políticas isótonas. O SS-BGP adiciona ao BGP, a capacidade de distinguir entre aselhas temporárias e aselhas persistentes, e de eliminar as últimas. A eliminação gradual de aselhas persistentes garante terminação com políticas isótonas e melhora a estabilidade do BGP num caso geral.

O desempenho do SS-BGP foi avaliado por simulação. Foi desenvolvido um simulador por eventos-discretos capaz simular o BGP e o SS-BGP para topologias com um elevado número de nós, como é o caso da topologia da Internet. Foram realizadas várias simulações do BGP e do SS-BGP em topologias realistas da Internet com variações realistas das políticas base de encaminhamento interdomínio. Os resultados mostram que o SS-BGP termina em todos os casos, ao contrário do BGP, sendo necessário que apenas uma percentagem muito reduzida das ASs tenha que alterar as suas políticas de importação.

**Palavras-Chave:** Internet, BGP, oscilações de encaminhamento, estabilidade do protocol, isotonicidade
Contents

Acronyms

1 Introduction
  1.1 Thesis outline

2 Interdomain Routing in the Internet
  2.1 Gao-Rexford routing policies
  2.2 Alternative routing policies
  2.3 Border Gateway Protocol (BGP) stability

3 Related Work

4 BGP Routing Model
  4.1 Routes, election, and extenders
  4.2 Pseudo-code
  4.3 Cost inflation and isotonicity

5 Elimination of Oscillations
  5.1 Shortest-path routing with negative costs
  5.2 Interdomain routing with siblings
  5.3 Interdomain routing with peer’s

6 Self-Stable BGP
  6.1 Termination and routing loops
  6.2 Pseudo-code
  6.3 Improved oscillation detection method
  6.4 Improved oscillation elimination method
  6.5 Partial deployment

7 Evaluation
  7.1 Simulator
  7.2 Input data and parameters
  7.3 Stability results
  7.4 Partial deployment
  7.5 Simulation environment

8 Conclusions
  8.1 Achievements and results
  8.2 Future work
List of Tables

2.1 Permanent state oscillations occur in the network of Figure 2.5 ........................................ 12
4.1 Mapping chart for the extenders of the GR routing policies .................................................. 20
4.2 Mapping chart for the extenders of the GR routing policies with siblings ................................. 20
4.3 Mapping chart for the extenders of the GR routing policies with peer* routes are exported to all neighbors .................................................. 21
4.4 Mapping chart for the extenders of the GR routing policies with peer* routes are exported only to customers .................................................. 21
5.1 BGP terminates for the network of Figure 5.1 ........................................................................ 23
5.2 BGP does not terminate for the network of Figure 5.2 ............................................................ 24
5.3 SS-BGP terminates for the network of Figure 5.2 .................................................................. 26
5.4 SS-BGP terminates for the network of Figure 5.3 .................................................................. 27
5.5 SS-BGP does not terminate for the network of Figure 5.4 when peer* routes are exported only to customers ........................................................................ 28
5.6 SS-BGP terminates for the network of Figure 5.4 when peer* routes are exported to all neighbors ........................................................................ 28
7.1 Percentage of destinations for which BGP does not terminate, for the GR routing policies with siblings and with peer*’s ........................................ 46
7.2 The first two columns show the number of detections, respectively, with SS-BGP and ISS-BGP for the GR routing policies with siblings and with peer*’s. The last column shows the number of false positive detections with SS-BGP ........................................................................ 46
7.3 First two columns show that termination times of BGP, for when it terminates, and of SS-BGP, for the GR routing policies with siblings and with peer*’s. The last column shows the reduction in the termination times from BGP to SS-BGP ................................................................. 47
7.4 First three entries show the percentages of ASs that belong to a persistent routing loop and that are at the tail of a peer* link, for the GR routing policies with peer*’s. The last entry shows the percentage of ASs at the tail of a sibling link for the GR routing policies with siblings ................................................................. 48
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Examples of the relationships defined in the Gao-Rexford (GR) routing policies</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Example showing the application of the export rules included in the GR routing policies</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Exporting peer(^+) routes only to customers decreases connectivity to customers</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Example with the sibling-sibling relationship</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Example of permanent state oscillations</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Example of temporary state oscillations</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>Example of network link. Routes from (v) to (u) and data flows from (u) to (v)</td>
<td>17</td>
</tr>
<tr>
<td>4.2</td>
<td>Example of a route exchange between node (u) and its neighbors (v), (w), (x), and (y)</td>
<td>18</td>
</tr>
<tr>
<td>4.3</td>
<td>Export examples using the GR routing policies model</td>
<td>19</td>
</tr>
<tr>
<td>4.4</td>
<td>Export examples using the GR routing policies model with lighter representation</td>
<td>19</td>
</tr>
<tr>
<td>5.1</td>
<td>Example of shortest-path routing. BGP reaches a stable state</td>
<td>23</td>
</tr>
<tr>
<td>5.2</td>
<td>Nodes (u) and (v) detect a routing loop and discard the routes learned from their clockwise neighbor. The state of BGP oscillates forever while SS-BGP terminates. Integers represent additive costs and the destination is announced by (t).</td>
<td>24</td>
</tr>
<tr>
<td>5.3</td>
<td>In contrast to BGP, SS-BGP terminates with the GR routing policies with siblings</td>
<td>27</td>
</tr>
<tr>
<td>5.4</td>
<td>SS-BGP does not terminate if peer(^+) routes are exported only to customer</td>
<td>28</td>
</tr>
<tr>
<td>6.1</td>
<td>When (yt) fails, (u) incorrectly signals a persistent routing loop. (R) and (C) represent peer and customer links, respectively, and the destination is announced by (t).</td>
<td>34</td>
</tr>
<tr>
<td>6.2</td>
<td>Equivalent to the network of Figure X when links (uw), (vw), and (wu) are deactivated</td>
<td>35</td>
</tr>
<tr>
<td>6.3</td>
<td>Protocol terminates with only (u) deploying SS-BGP, but still oscillates if only (w) deploys SS-BGP. Peer(^+) routes are exported to all neighbors and (t) announces the destination.</td>
<td>36</td>
</tr>
<tr>
<td>6.4</td>
<td>BGP terminates, but if only (u) deploys SS-BGP, then the protocol oscillates. Peer(^+) routes are exported to all neighbors and (t) announces the destination.</td>
<td>37</td>
</tr>
</tbody>
</table>
Acronyms

AS  Autonomous System. 3–5, 7–11

BGP  Border Gateway Protocol. vii, 3–5, 7–13

GR  Gao-Rexford. xiii 3, 7–11

ISP  Internet Service Provider. 3

ISS-BGP  Improved SS-BGP. 34

SPVP  Safe Path Vector Protocol. 15

SS-BGP  Self-Stable BGP. 4, 5
Chapter 1

Introduction

The Internet was once a research project developed by the academic and military communities with great support from the US government. It began in the 1960's with goal of developing protocols for communication across multiple networks. For some decades, the project remained restricted to a few prominent universities and some governmental research laboratories. Only in the 1990's, with the emergence of the first group of commercial Internet Service Providers (ISPs), did access to the Internet become available to the general public. Since then, the Internet evolved and spread across the globe, becoming an essential infrastructure of the modern world.

The Internet is comprised by a large collection of networks. Each one of these networks is called an Autonomous System (AS). ASs come in various forms: they may be ISPs, content provider, military networks, university networks, enterprise networks, etc. Since the commercialization of the Internet, economical interests have become the driving force which dictates its evolution. Each AS establishes commercial relationships with a few thousands of other ASs so as to attain global connectivity across the entire Internet. Two ASs which establish a relationship between each other are called neighbors. These commercial relationships determine how and what traffic the ASs exchange and thereby dictate their interdomain routing policies. Their existence and evolution are driven by the economical interests of ISPs and other players, the competitive marketplace, and the constantly changing Internet structure.

The commercial relationships considered in the GR [1] routing policies are, perhaps, the most common in the Internet. They foresee two types of commercial relationships that ASs can enter to: customer-provider and peer-peer. In the first case, the customer AS pays the provider AS to have access to the Internet, while, in the second, case the two peer agree to exchange transit from their customers free of charge. However, the structure of Internet is constantly evolving and ASs are always establishing different forms of commercial relationships, besides the two mentioned above, not only to achieve cost savings, but also to enhance service reliability and availability to their customers.

To obtain connectivity to each destination, ASs have to cooperate between each other. The process through which ASs discover paths to reach each destination is called routing. In the Internet, all ASs participate in a single interdomain routing protocol called BGP [2]. In BGP, ASs exchange pieces of routing information called routes between each other. Routes associate a set of attributes with a destination. A destination corresponds to an address assigned to a networking device connected to a single AS. Each AS owns a set of destinations, which announces to the rest of the Internet so that other ASs can reach its destinations. BGP instantiates an independent routing process for the announcement of each destination. For each destination, each AS stores a set of candidate routes learned from each neighbor and elects the best route among these. The route election process is based on the attributes of each route. In its current form, BGP supports a wide range of attributes. The first and second attributes are the LOCAL-PREF and the AS-PATH, respectively. The LOCAL-PREF is
assigned locally by each \textit{AS} and defines the degree of preference that the \textit{AS} assigns to that route. The \textbf{AS-PATH} attribute of a route contains the sequence of \textit{ASs} traversed by that route, from the destination \textit{AS} to the current \textit{AS} holding the route. This attribute is used to break ties between routes with the same \textbf{LOCAL-PREF} value: a route is preferable to another, if its \textbf{AS-PATH} is shorter. Routes are totally ordered, which means two routes with the different \textbf{AS-PATHs} are always ranked differently in a deterministic way. When an \textit{AS} elects a new route, it exports the route to its neighbors. An elected route is not necessarily exported to all neighbors of that: the export policies applied by the \textit{AS} dictate to which neighbors the elected route is exported. For each destination, this local process is repeated at each \textit{AS}, providing connectivity network-wide. The execution of the protocol for a single destination terminates when there is no more routes in transit, which indicates every \textit{AS} has settled on a route to the destination. Hopefully, at that point every \textit{AS} has elected a route to the new destination.

The routing policies applied by an \textit{AS} dictate the \textbf{LOCAL-PREF} assigned by that \textit{AS} to the routes learned from its neighbors. Therefore, these reflect the kind of commercial relationships established with the neighboring \textit{ASs}. \textbf{BGP} is a very flexible protocol that allows \textit{ASs} administrators to autonomously set a wide range of routing policies. Consequently, \textit{ASs} can set conflicting routing policies that cause the state of the protocol to oscillate permanently \cite{3} \cite{4}. In such a scenario, there is a set of \textit{ASs} which elect the same sequence of routes repeatedly in continuous succession. As a consequence, routes are continuously being exchanged which greatly impacts the Internet’s control plane. In addition, the data plane is also affected because state oscillations attract data messages to forwarding loops, causing them to be delayed or even lost.

A similar oscillating behavior can also be caused by a malfunction in some component of the network. For instance, an unstable connection that keeps turning on and off rhythmically. State oscillations caused by such conditions are usually temporary and disappear from the network as soon as the flaw is fixed. Permanent state oscillations, on the other hand, are associated with network structure and the routing configurations applied at each \textit{AS}. Since \textit{ASs} are very secretive about their routing configurations and the identity of the \textit{ASs} with which they establish relationships, the complete topology of the Internet is not fully understood, it is very hard for an \textit{AS} to distinguish between a temporary and permanent state oscillation. Therefore, oscillations caused by conflicting routing policies tend to remain in a network for a long time before they get to be eliminated from the network, if they even get eliminated.

Permanent state oscillations are related to the routing policies applied around network cycles \cite{5}. This thesis work identifies that some of them are caused by routing loops while others are not. Routing loops refer to the propagation of routes around network cycles. \textbf{BGP} already included a method to detect routing loops and to stop the propagation of routes past them: an \textit{AS} discards a route the \textbf{AS-PATH} of which contains the \textit{AS} number of that \textit{AS}. However, this is not enough to prevent the protocol from oscillating.

This work had as its starting point an embryonic idea to devise an extension to \textbf{BGP} able to detect routing loops that may cause oscillations and eliminate these loops from the network in order to improve the stability of \textbf{BGP}. After some research work, it was determined that there are two types of routing loops: temporary routing loops and persistent routing loops. Temporary routing loops occur a finite number of times over an execution of \textbf{BGP}. On the other hand, persistent routing loops may recur indefinitely in such an execution. This conclusion justifies chasing and eliminating the latter. An addon to \textbf{BGP} called \textbf{Self-Stable BGP (SS-BGP)} is proposed in this thesis work, which devises a refinement to the routing loop detection method included in \textbf{BGP}, able to distinguish between temporary and persistent routing loops, and that eliminates them. The basic idea is simple: if an \textit{AS} learns a route from a neighbor which would have been elected not for the fact that the \textbf{AS-PATH} of which identifies a routing loop, then that \textit{AS} discards all future routes learned from that neighbor. \textbf{SS-BGP} uses standard \textbf{BGP} routing messages and, therefore, can be deployed incrementally.
For the common and broad class of routing policies characterized by the isotonicity property [6][7], all permanent state oscillations are associated with persistent routing loops. SS-BGP is guaranteed to terminate for these routing policies, that is, BGP reaches a stable state whatever the initial state and whatever the delays in the delivery of routing messages. Isotonicity means that the relative preference between the LOCAL-PREF value of two routes at one AS is maintained when they are exported and learned at a neighbor. When routing policies are not isotone, SS-BGP still brings benefits to the protocol’s stability, with SS-BGP being able to reach a stable state when BGP is not.

Since BGP’s dynamic behavior is too complex to predict analytically, quite often research resorts to simulations to evaluate its performance in various configurations. The same approach is followed here to evaluate SS-BGP and to compare its performance with BGP’s. Simulating BGP in large-scale networks, such as the Internet, is usually very resource and time consuming. The required memory for BGP simulators which implement all aspects of BGP routing, such as [8][9], increases rapidly with the size of the network. The simulation of BGP like protocols in large-scale networks requires low-level details to be abstracted so that they do not consume unnecessary resources. For the evaluations performed in the thesis work, a new discrete-event simulator was developed from scratch which implements an high-level version of BGP and SS-BGP, maintaining all the important aspects required for the purposes of this work. This approach allowed a large number of simulations to be performed with realistic topologies of the Internet with limited time and resources. Furthermore, it also provided great flexibility in setting the simulation experiments and determining the information to be collected in each experiment. The simulator developed in this work is publicly available at [10] and can be freely used in other research work with the same requirements.

1.1 Thesis outline

This thesis proceeds as follows. Chapter 2 describes interdomain routing in the Internet and discusses the stability of BGP under different routing policies, highlighting the fact that conflicting routing policies prevent BGP from reaching a stable state. Chapter 3 reviews previous proposals to improve the stability of BGP, pointing out the disadvantages in comparison with the solution proposed here. Chapter 4 introduces a model for BGP routing and applies it to interdomain routing with the baseline routing policies and two other alternatives introduced in Chapter 2. Based on this model, it presents the pseudo-code of BGP and introduces the isotonicity property. Chapter 5 introduces SS-BGP through three examples, highlighting the distributed detection and elimination of oscillations and the importance of isotonicity in assuring termination. Chapter 6 describes SS-BGP with generality. It relates termination to routing loops, identifying that some routing loops are associated with oscillations and, therefore, should be eliminated. Later, it presents the pseudo-code of SS-BGP, highlighting the differences between this and BGP’s. Afterwards, this Chapter shows that SS-BGP is susceptible to false positives and presents an alternative detection method to overcome that problem. It also introduces an alternative elimination method which provides greater connectivity than the canonical version of SS-BGP when an oscillation is eliminated. The Chapter finishes by discussing partial deployment of SS-BGP. Chapter 7 introduces the simulator developed in this thesis work to evaluate SS-BGP and presents results obtained from the simulations of BGP and SS-BGP in realistic Internet topologies with realistic routing policies. Finally, Chapter 8 draws the final conclusions and provides some proposals for future work.
Chapter 2

Interdomain Routing in the Internet

BGP started as a simple path-vector protocol and evolved over time to meet the particular requirements of AS administrators. AS administrators value its flexibility and the autonomy with which they can set their own routing configurations. However, due to a lack of cooperation between ASs, they can define routing policies that prevent the protocol from reaching a stable state. Perhaps, the most common commercial relationships that ASs establish in the Internet are described by Gao and Rexford in [1]. BGP is guaranteed to terminate if all ASs configure their routing configurations to follow these policies [1][5][4]. However, commercial relationships between ASs are continuously evolving and commonly take many forms beyond the ones described in the baseline GR routing policies. These variations of the baseline GR routing policies make BGP susceptible to permanent state oscillations, which have a deep impact on the reliability of the Internet.

This Chapter is organized as follows. Section 2.1 describes interdomain routing with the baseline GR routing policies. Section 2.2 presents two realistic alternatives to the baseline routing policies which make BGP prone to permanent state oscillations. Finally, Section 2.3 discusses BGP stability, presenting examples of temporary and permanent state oscillations and illustrating the events that prevent ASs from being able to distinguish between the two.

2.1 Gao-Rexford routing policies

The routing policies set at each AS determine the preference that AS assigns to each route learned from its neighbors. Furthermore, they also dictate which routes are exported to which neighbors. As described before, the routing policies applied by AS reflect the commercial relationships they establish with their neighbors. The GR routing policies provide a baseline of understanding for interdomain routing. They consider two possible commercial relationships that two ASs can establish between each other.

- **customer-provider**: the customer pays the provider to be able to access the rest of the Internet. It is the provider’s responsibility to ensure the customer can reach and can be reached by the other ASs.

- **peer-peer**: two ASs agree to exchange traffic from their customers at no cost.

Figure 2.1 shows a simple network represented by a graph, with some examples of these commercial relationships. The peer-peer relationship is represented by a dashed line connecting the two peers and the customer-provider relationship is represented by a directed link from the provider AS to the customer AS. For instance, AS_A is a provider of AS_C and AS_D. AS_E is a customer of AS_B. AS_A and AS_B are peers, as are AS_D and AS_E.
The commercial relationships described in the GR routing policies impose an hierarchy among the ASs, dividing them into multiple tier levels. ASs without any providers are at the top of the AS hierarchy. These are called tier-1 and they do not have to pay anybody to access the Internet. Currently, there are 14 tier-1s in the Internet. Their customers lay right below them and are called tier-2. The ASs at the bottom of the hierarchy do not have any customers. These are called stubs. For instance, in Figure 2.1, AS A and AS B are tier-1 since they have no providers. AS C, AS D and AS E are tier-2, because they are direct customers of either AS A or AS B. AS F and AS G are both two levels below AS A, therefore, these are tier-3. Finally, AS H does not have any customers, which means that it is a stub.

Routes are characterized based on the relation an AS maintains with each neighbor. A route learned from a customer neighbor is called a customer route. Analogously, a route learned from a peer/provider is also called a peer/provider route. The preference order between each type of route is determined by the economic cost associated with transiting traffic through the neighbor from which the route was learned. For instance, a customer route (learned from a customer neighbor) implies an income, while a provider route is associated with an expense. Peer routes do not imply any cost for either peer. Therefore, customer routes are preferred to peer routes, which in turn are preferred to provider routes.

The export policies are also defined based on the economical interests of each AS. When an AS exports a route to a neighbor, it allows that neighbor to forward traffic through it. An AS does not export a route to a neighbor from which they do not want to receive traffic. For instance, an AS does not want to receive incoming traffic from its providers, because this implies an added expense. However, it is obligated by contract to provide connectivity to its customers. Therefore, it exports only its customer routes to its provider. On the other hand, a provider AS exports all of its routes to its customers, because exchanging traffic with customers results in profit. Finally, peering ASs agree to share their customer routes, so they can both avoid having to forward their customers’ traffic to their own providers. In sum, the export rules are as follows.

- **To customers:** export all routes.
- **To providers/peers:** export only customer routes.

Figure 2.2 shows examples of these export rules. For instance, AS A exports the customer routes learned from AS C to its customer AS D, denoted by a dashed green arrow starting at AS C, crossing AS A, and ending at AS D. AS C also exports the customer routes learned from AS F to its peer AS D. On the other hand, AS D does not export the peer routes learned from AS C neither to its provider AS A nor
its peer $AS_E$, denoted by red dashed arrows. Similarly, $AS_D$ does not export the provider routes learned from $AS_A$ either to its peer $AS_C$ nor its provider $AS_B$. However, it exports the provider routes learned from $AS_B$ to its customer $AS_G$.

### 2.2 Alternative routing policies

This Section presents two alternatives to the baseline GR routing policies. Both of them make BGP prone to permanent state oscillations.

**GR routing policies with peer’s**

The GR routing policies provide a baseline of understanding for how AS administrators configure their routing policies. However, there is compelling evidence that this policies are consistently violated by ASs for various reasons [11]. For instance, sometimes ASs prefer some peer routes over customer routes. A possible scenario where this would be reasonable, is if the connection to the customer is congested. In this scenario, electing to forward traffic through a peer, would help reduce the amount of traffic flowing through that connection, thereby, improving service reliability and availability to the customer.

Neighbors the routes learned from which are preferred over customer routes are called peers’, and those routes are called peer routes. Usually, peer routes are only exported to customers. An AS may follow the same ruling for peer routes and decide to export peer routes strictly to customers. However, it would be disregarding its contractual obligations to its customers: an AS which elects a peer route over a customer route, if it does not export that route to its providers, then it prevents all of them and their respective customers from reaching its own customers. Therefore, the provider AS is not fulfilling its contractual obligation to provide connectivity to its customers. The example in Figure 2.3 helps to clarify this point. In the network, $AS_A$ is a provider of $AS_B$, $AS_C$ is a peer of $AS_B$, and $AS_D$ is a customer of $AS_B$. To reach each destination announced by $AS_D$, $AS_B$ learns a customer route directly from $AS_D$ and a peer route from $AS_C$. In this scenario, $AS_B$ does not export the route learned from $AS_C$ to $AS_A$. As a consequence, $AS_A$ and all of its customer are left without a route to reach any of $AS_D$’s destinations.

Alternatively, an AS may opt apply the same export rules for customer and peer routes, thereby, exporting peer routes to all neighbors, as with customer routes. In that case, $AS_B$ exports the peer route learned from $AS_C$ to $AS_A$ and $AS_A$ is able to reach the destinations announced by $AS_D$ via its provider $AS_B$.
Regardless of the exportation rules applied for the peer+ routes, BGP is not guaranteed to reach a stable state if ASs violate the GR routing policies by preferring some peer routes over their customer routes. Section 5.3 discusses this matter further, presenting a reasonable case in favor of exporting peer+ routes to all neighbors.

**GR routing policies with siblings**

Violations of the GR routing policies are quite common for a great number of reasons. However, in certain scenarios violating the preference order between the routes, or the export rules defined in these policies, is not enough to fulfill the requirements of ASs. An example of this relates to the relationships between ASs which belong to the same organization. In the early stages of the Internet, each organization managed a single AS. However, today it is increasingly more common for organizations to manage multiple ASs. When two ASs belonging to the same organization want to establish a relationship, neither of the commercial relationships considered in the GR routing policies apply from an economic standpoint.

Liao et al. [12] expanded the GR routing policies to accommodate the sibling-sibling relationship. Two ASs which establish such a relationship are called siblings. Siblings share all routes they learn from their neighbors, providing total connectivity to one another. A route exported to a sibling keeps the information of how it was learned from outside the stretch of siblings (whether it was learned from a customer, a peer, or a provider), but its preference is decreased.

Figure 2.4 shows a simple example where all of, AS_C, AS_D, and AS_F, belong to the same organization. The sibling-sibling relationships established between these ASs are represented by two full lines. To reach each destination announced by AS_G, AS_C elects a customer route learned directly from AS_G and it exports this route to its sibling AS_D. Consequently, AS_D learns a customer route with one sibling hop from AS_C and elects this route over the provider route learned from AS_B. AS_F learns from AS_D a customer route with two sibling hops, which elects over the route learned from its provider AS_B. In this configuration, all three ASs which belong to the same organization, elect customer routes to reach the destinations announced by AS_F, instead of provider routes they learn from AS_A or AS_B. As a result, the organization maximizes its profit.

If, for instance, the two sibling-sibling relationships were replaced by peer-peer relationships, then AS_F elects a provider route learned from AS_B to reach AS_F, because AS_D does not export the peer route learned from AS_C to a peer. Furthermore, if AS_E became a provider of AS_F, then AS_E elects the customer route learned directly from AS_E, instead of the customer route with two sibling hops learned from AS_D, which avoids transiting traffic across the organization network for the same profit. This makes sense from an economical point of view, since transiting traffic within the organization implies some expense.
The sibling-sibling relationship is a common example of a commercial relationship established between ASs belonging to the same organization. However, it is important to note that the deployment of the GR routing policies with siblings may cause permanent state oscillations.

2.3 **BGP** stability

Network cycles are common in almost every computer network, including the Internet. However, the existence of cycles in a network may introduce some instability in a routing vector protocol, such as BGP, running on that network. In these kind of protocols, each node knows very little information about the global network and relies on the information that its neighbors provide it to find paths to each destination. This characteristic makes them very scalable protocols and, thereby, suitable for large networks such as the Internet. However, the lack of knowledge at each node prevents them from detecting some important details regarding the global network and the behavior of the protocol for that network.

A common example of instability in distance vector protocols is related to routing loops, that is, the propagation of routes around network cycles. If no action is taken to prevent the propagation of routes past routing loops, then the protocol may never reach a stable state. BGP already includes routing loop detection mechanism. Each BGP route contains an AS-PATH attribute, which carries the sequence of ASs traversed by the route, from the destination AS to the current AS holding the route. To verify if a route denounces a loop, an AS can check if its identifier is included in the route’s AS-PATH. In that case, that AS discards the looping route, preventing its propagation further around the cycle.

One might think that by preventing routes from being propagated past routing loops, then BGP eventually reaches a stable state. However, that is not the case: the routing policies defined locally at each AS may conflict, causing the state of BGP to oscillate permanently. Figure 2.5 serves to illustrate an example of a case where BGP oscillates due to conflict between the routing policies applied at each node. The links represent a unidirectional connection between the two ending ASs. Routes travel in the direction opposite to that pointed by the links. Each link is associated with an integer number which corresponds to the preference level assigned to the routes learned from that link by the AS at its tail. A lower value corresponds to an higher preference. The network contains a cycle around $AS_A$, $AS_B$, and $AS_C$. The preference values in each link show that each AS involved in the cycle prefers routes learned from around the cycle (from its clockwise neighbor) over routes learned from outside the cycle (from $AS_D$). Assume that the ASs involved in the cycle export routes with a preference of 2 (routes learned from $AS_D$) to their respective counter-clockwise neighbor, but *not* routes with a preference of 1 (routes...
learned from their clockwise neighbor). Table 2.1 shows the elected routes at each AS at successive units of time. For each route, the table shows its preference value, followed by the path traversed by the route. At $T = 0$, $AS_D$ exports a route for a new destination to its neighbors $AS_A$, $AS_B$, and $AS_C$. At $T = 1$, each of these three ASs elects a route with a preference value of 2 learned directly from $AS_D$ and they export their elected route counter-clockwise around the cycle. At $T = 2$, all of $AS_A$, $AS_B$, and $AS_C$, elect a route with a preference level of 1 learned from their respective clockwise neighbors. However, they do not export these routes further around the cycle. Instead, they all withdraw the route that they have previously exported counter-clockwise around the cycle at $T = 1$. As a consequence, at $T = 2$, ASs $AS_A$, $AS_B$, and $AS_C$ revert to the election of the route learned from $AS_D$. Therefore, BGP is back to the initial state at $T = 1$, implying permanent oscillations of its state. Notice that none of the ASs detects a routing loop. However, the protocol still oscillates. The reason for this is related to the routing policies applied by each AS around the cycle. Chapter 5 investigates this further.

Figure 2.5: Example of permanent state oscillations. Integer values indicate the preference values assign by ASs to the routes learned through the respective links. Routes travel in the direction opposite to that pointed by the links. $AS_D$ announces the destination.

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2, AS_D$</td>
<td>$2, AS_D$</td>
<td>$2, AS_D$</td>
<td>$2, AS_D$</td>
<td>$2, AS_D$</td>
</tr>
<tr>
<td>2</td>
<td>$2, AS_D$</td>
<td>$1, AS_B AS_D$</td>
<td>$2, AS_D$</td>
<td>$1, AS_C AS_D$</td>
<td>$2, AS_D$</td>
</tr>
<tr>
<td>3</td>
<td>$2, AS_D$</td>
<td>$1, AS_B AS_D$</td>
<td>$2, AS_D$</td>
<td>$1, AS_C AS_D$</td>
<td>$2, AS_D$</td>
</tr>
</tbody>
</table>

Table 2.1: Permanent state oscillations occur in the network of Figure 2.5. Elected routes are underlined. Withdrawn routes have a shaded background.

Besides conflicting policies, the state of BGP can also oscillate due to some network malfunction, such as an unreliable connection that turns on and off repeatedly. In that case, the routing protocol should not intervene, because the oscillation is only temporary and, once the failure is fixed, the oscillation disappears from the network. However, although ASs can locally verify that their state is oscillating, the information that each of them has locally available does not allow them to distinguish between temporary and permanent routing oscillations.

In the previous example, for instance, $AS_A$ oscillated between routes $(2, AS_D)$ and $(1, AS_B AS_D)$. From its state, $AS_A$ is not able to distinguish between the permanent oscillation caused by policy conflict around the cycle in Figure 2.5 and a temporary oscillation caused by the link between $AS_B$ and $AS_D$, flapping rhythmically. Figure 2.6 presents an example of that exact situation. At first, the protocol stabilizes with $AS_A$ electing a route $(1, AS_B)$ learned from $AS_B$. However, when the link between $AS_B$ and $AS_D$ starts to flap, $AS_A$ receives, alternately, an withdraw and valid route from $AS_B$. Therefore, the state of $AS_A$ oscillates between a routes $(2, AS_D)$ and $(1, AS_B AS_D)$ as before when the oscillation was caused by conflicting routing policies.
In its current version, BGP does not include any method to detect and avoid/eliminate permanent oscillations. The reason for this, is that it would require ASs to share more routing information between each other than they are willing to do: ASs are usually very secretive about their routing configurations, wanting to share as little information as possible to maintain connectivity. Including extra attributes in the routing messages of BGP would also force every AS to update their BGP-speaking devices in order to support the new attributes, which is something that they are not willing to do at this point.
Chapter 3

Related Work

The problem of BGP state oscillations is not new. Several proposals have been presented to largely modify BGP’s operation to guarantee that the protocol stabilizes. Up until now, the proposed solution all rely on some kind of route history which limits the scalability of the protocol and comprises the privacy of routing policies, which are two of the most valued aspects by AS administrators.

Griffin and Wilfong [13] were among the first authors to present a solution to the permanent state oscillation problem to which they called Safe Path Vector Protocol (SPVP), adding to the set of route attributes an history of routes: each route is exported carrying the sequence of routes which justifies the election of that route. Histories with repeated routes reveal the existence of conflicting routing policies and serve to identify routes that should be avoided: a route which is repeated in the history is no longer imported.

Ahronovitz et al. [14] proposed an improvement over the SPVP, where instead of histories being sent between ASs, each AS keeps a record of the routes elected in the recent past and uses this local history to identify a possible conflict. The instant an AS detects a repeated route in its history it sends a token to its neighbors. If that token is propagated around a cycle and arrives back at that same AS, then there is a policy conflict and the route included more than once in the history is not imported in the future.

Ee et al. [15] propose a slightly different approach to the previous two, yet it also requires an history and changing the routing messages. Routes carry an extra value called global precedence value, which is the first attribute evaluated in the route election process. An higher global precedence value denotes a less preferable route. An AS determines that it may be involved in a policy conflict when it elects a route which is less preferred than its previous elected route. Before exporting the new route, that AS stores the previously elected route in a list of past elected routes and, then, it exports the new elected route with the global precedence value incremented. Consequently, in future iterations of the protocol, more preferred and also more unstable routes will have higher global precedence values causing ASs to elect less preferred, but more stable routes.

All these three solutions involve major modifications to BGP, both at the message level and at the local level. They all require ASs to include extra information in the routing messages which compromises the privacy of their routing policies. Furthermore, modifying the messages exchanged between ASs implies that they are not compatible with BGP and, therefore, require network-wide deployment for their benefits to be earned. In addition, none of them scales as well as BGP since they all require some kind of route history to be recorded.

In contrast, the solution presented in this thesis improves the scalability properties of BGP without requiring any history of previous seen routes and keeps BGP routing messages untouched. Consequently, it preserves the privacy of routing policies and it can be deployed incrementally with BGP without affecting its scalability.
Chapter 4

BGP Routing Model

This Chapter introduces a model for BGP routing and applies the model to the baseline GR routing policies and the alternative routing policies described in Section 2.2. Based on the described model, Section 4.2 presents the pseudo-code for BGP. And finally, Section 4.3 introduces two properties of routing policies that provide some insight into the performance of BGP with different routing policies.

4.1 Routes, election, and extenders

In Chapter 1 it was said that routing is the process that ASes use to learn paths in the Internet and that BGP is the routing protocol used in the Internet. In general, a routing protocol is a distributed algorithm that nodes in a network use to find paths in order to guide data messages from sources to destinations. This Section presents a model for BGP routing based on an algebra from [6] that provides the necessary means to model any routing protocol running in any network.

A network is modeled as a directed graph, where each node represents an AS and each link $uv$ implies a connection between a node $u$, the tail of the link, and a node $v$, the head of the link. Figure 4.1 illustrates an example of a link $uv$. Node $u$ is said to be $v$’s out-neighbor, and $v$ is $u$’s in-neighbor. Data flows in the direction pointed by the link, from $u$ to $v$. Routes are sent in the opposite direction of the data, that is, routes flow from $v$ to $u$.

A destination is not a node, but an address that can be announced by any node in the network. A destination is said to be announced in unicast if single node announces the destination. In contrast, anycast refers to a destination being announced by multiple nodes.

A route associates a set of attributes to a destination. BGP supports a large set of attributes to characterize routes. The first two attributes in BGP’s election process, and therefore the most relevant ones, are the LOCAL-PREF and the AS-PATH. The former defines the degree of preference locally assigned to the route by a node. This attribute is local to each node and is not included in the exchanged routing messages. The AS-PATH contains the sequence of nodes traversed by the route, from the origin node to the current node holding the route. The AS-PATH is specially relevant in BGP, because it allows

![Figure 4.1: Example of network link. Routes from $v$ to $u$ and data flows from $u$ to $v.$](image-url)
nodes to detect routing loops, avoiding the election of routes that traverse the same node more than once.

Rather than using LOCAL-PREF explicitly, the routing model presented here characterizes BGP routes as couplets of the form \( (\alpha, P) \), where \( \alpha \) is a cost, and \( P \) is a path. The finite set \( \Sigma \) contains all possible costs that can be assigned to a route. The operation \( \prec \) establishes an order among the costs in \( \Sigma \). For two costs \( \alpha, \beta \in \Sigma \), if \( \alpha \prec \beta \), then cost \( \alpha \) is better than cost \( \beta \), and cost \( \beta \) is worse than cost \( \alpha \). Writing \( \alpha \preceq \beta \), means that either \( \alpha \prec \beta \), or \( \alpha = \beta \). Writing \( \alpha \succeq \beta \) is equivalent to \( \beta \prec \alpha \). Cost \( \bullet \) represents unreachability: a node has no knowledge of a path to the destination; thus cost \( \bullet \) is than any cost, \( \forall \alpha \in \Sigma : \alpha \prec \bullet \).

The election among costs is modeled by the binary operation \( \cap \). With two costs \( \alpha, \beta \), \( \alpha \cap \beta = \alpha \) if \( \alpha \preceq \beta \), and \( \alpha \cap \beta = \beta \) if \( \beta \preceq \alpha \). For a set of costs, the elected cost is represented by \( \cap \{ \alpha_1, ..., \alpha_n \} \).

Paths are totally ordered by \( \ll \). Order \( \ll \) respects first and foremost the path lengths, such that, given two paths \( P \) and \( Q \),

\[
P \ll Q \implies |P| \leq |Q|,
\]

(4.1)

where \( |P| \) denotes the number of links of \( P \). Paths with the same length are ordered according to an arbitrary ordering function.

A similar notation as the one used for costs, is also used for routes. Routes are ordered by \( \prec \). For two routes \( r \) and \( s \), route \( r \) is said to be: better than route \( s \), if \( r \prec s \); and, worse than route \( s \), if \( r \succ s \). Routes are totally ordered based on its cost and path. A route \( (\alpha, P) \) is better than route \( (\beta, Q) \), \( (\alpha, P) \ll (\beta, Q) \), if the following condition is verified.

\[
\alpha \prec \beta, \text{ or } \alpha = \beta \text{ and } P \ll Q
\]

(4.2)

As with costs, the election among routes is modeled by a binary operation \( \Delta \). With two routes \( r, s \), \( r \Delta s = r \) if \( r \preceq s \), and \( r \Delta s = s \) if \( r \succeq s \). Election among a set of routes is represented by \( \Delta \{ r_1, ..., r_n \} \).

The routing policies between neighboring nodes are modeled by functions. Function \( T[uv] \) describes the transformation of costs over the link \( uv \). The function \( T[uv] \) is called the extender of link \( uv \), and models the export policies of \( v \) in relation to \( u \) and the import policies of \( u \) in relation to \( v \). A route with cost \( \alpha \) exported by \( v \) through a link \( uv \) is learned at \( u \) with cost \( T[uv](\alpha) \). By modeling the transformation of costs with a function, the possibility of two routes with the same cost at \( v \) producing two routes with different costs at \( u \) is excluded.

Figure 4.2: Example of a route exchange between node \( u \) and its neighbors \( v, w, x, \) and \( y \).
which models the election policies applied by \( u \). In the third and last stage, \( u \) exports the new elected route to its in-neighbors. But, according to the routing policies of \( u \), routes with cost \( \alpha_u \) are not exported to \( v \). Thus, \( u \) sends route \((\alpha_u, P_u)\) to \( x \), which becomes route \((T[xu](\alpha_u), uP_u)\) there, and sends an withdrawal \((\bullet, \bullet)\) to \( y \). The couplet \((\bullet, \bullet)\) at \( v \), expresses the fact that \( v \) will not be able to reach the destination via \( u \).

**Gao-Rexford**

This Section applies the BGP routing model, described in the previous Section, to the baseline GR routing policies. Nodes following these policies establish three different types of relationships with their neighbors: a node can be characterized as a customer, a peer, or a provider. For a link \( uv \), the extender \( C \) models the fact that \( v \) is a customer of \( u \); the extender \( R \) models the fact that \( v \) is a peer of \( u \); and the extender \( P \) models the fact that \( v \) is a provider of \( u \). Figure 4.3 presents a network including these links. For instance, there is a customer link \( C \) between \( u \) and \( w \), and a corresponding provider link \( P \) from \( w \) to \( u \). Nodes \( u \) and \( v \) are peers, denoted by two peer links \( R \), one from \( u \) to \( v \) and another from \( v \) to \( u \).

Routes can have three different costs \( c \), \( r \), and \( p \), depending on the type of neighbor they are learned from: a route learned from a customer is called a customer route with cost \( c \); a route learned from a peer is called a peer route with cost \( r \); and a route learned from a provider is called a provider route with cost \( p \). A customer route is better than a peer route, and the latter is better than a provider route. From this follows that \( c \prec r \prec p \prec \bullet \).

The exportation rules dictate that customer routes are exported to all neighbors, but peer and provider routes are only exported to customers. Table 4.1 shows a mapping chart representing the result of applying each extender \((C, R, P)\) to each of possible route cost \((c, r, p)\) according to the export and import rules considered in the GR routing policies.

For example, the first entry of Table 4.1, which is \( C(c) = c \), encodes the fact that a customer route is exported through a customer link, originating a customer route at the tail node. Figure 4.3 shows more examples. For instance, node \( y \) learns customer routes from \( t \) and exports them to its peer \( x \), \( R(c) = r \), and provider \( v \), \( P(c) = p \). In contrast, the peer routes learned from \( y \) at \( x \) are not exported through the customer link \( ux \), represented by \( C(r) = \bullet \), but are exported through the provider link \( zr \), giving rise to a provider route at \( z \), \( P(r) = p \). And, node \( z \) does not export the provider routes learned from \( x \) to \( w \),
Section 2.2 introduces the GR routing policies with siblings, which add the sibling-sibling relationship to the baseline GR routing policies. As before, a customer link joins a provider to a customer; a peer link joins a peer to a peer; and a provider link joins a customer to a provider. The sibling link is added to these collection, joining a sibling to a sibling. The extender associated with a customer/peer/provider/sibling link is labeled as \( C/R/P/S \). The sibling-sibling relationship is modeled by two sibling links connecting the two siblings, similarly to the peer-peer relationship.

Siblings are neighbors which share all routes. A route learned from a sibling keeps the information of how it was learned from outside of the stretch of siblings, but the corresponding cost becomes slightly worse. Costs are represented as \( (x, n) \) standing for 'learned from a \( x \) through \( n \) siblings', where \( x \) is customer \( c \), peer \( r \), or provider \( p \), and \( n \) is a natural number representing the number of consecutive sibling links traversed by the route. For example, a route learned from a customer has cost \( (c, 0) \) and route learned from sibling which learned it from a customer has costs \( (c, 1) \). A route with cost \( (c, n) \) is called a customer route with \( n \) sibling hops, as well as a route with cost \( (r, n) \) or \( (p, n) \) is called a peer or a provider route with \( n \) sibling hops. Costs are ordered, first based on the type of route and, only if the types are equal, then the number of sibling hops is used. Thus, \( (x, n) \prec (y, m) \) if \( x \prec y \), or \( x = y \) and \( n < m \).

The export rules defined in the baseline GR routing policies still apply here with the addition that, all routes are exported to a sibling, as with customer routes. The extenders \( C, R, P, \) and \( S \), map costs according to the chart in Table 4.2.

<table>
<thead>
<tr>
<th></th>
<th>((c, n))</th>
<th>((r, n))</th>
<th>((p, n))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C)</td>
<td>((c, 0))</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
</tr>
<tr>
<td>(R)</td>
<td>((r, 0))</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
</tr>
<tr>
<td>(P)</td>
<td>((p, 0))</td>
<td>((p, 0))</td>
<td>((p, 0))</td>
</tr>
<tr>
<td>(S)</td>
<td>((c, n + 1))</td>
<td>((r, n + 1))</td>
<td>((p, n + 1))</td>
</tr>
</tbody>
</table>

Table 4.2: Mapping chart for the extenders of the GR routing policies with siblings.

For example, the first entry, \( C(c, n) = (c, 0) \), encodes the fact that a customer route with any number of hops is exported through a customer link, originating a customer route with 0 sibling hops at the neighbor. The entries in the last row of Table 4.2 encode the fact that any route is exported to a sibling, giving rise to a route with a worse cost there.

**Gao-Rexford with peer** 's

The peer relationship represents a violation of the guidelines in the GR routing policies. This relationship is modeled by a link assigned with the \( R^+ \) extender. The routes learned from a peer are called...
Table 4.3: Mapping chart for the extenders of the GR routing policies with peer\(^*\) when peer\(^*\) routes are exported to all neighbors

<table>
<thead>
<tr>
<th></th>
<th>(r^*)</th>
<th>(c)</th>
<th>(r)</th>
<th>(p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R^*)</td>
<td>(r^*)</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>(C)</td>
<td>(c)</td>
<td>(c)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>(R)</td>
<td>(r)</td>
<td>(r)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>(P)</td>
<td>(p)</td>
<td>(p)</td>
<td>(p)</td>
<td>(p)</td>
</tr>
</tbody>
</table>

Table 4.4: Mapping chart for the extenders of the GR routing policies with peer\(^*\) when peer\(^*\) routes are exported only to customers.

<table>
<thead>
<tr>
<th></th>
<th>(r^*)</th>
<th>(c)</th>
<th>(r)</th>
<th>(p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R^*)</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>(C)</td>
<td>(c)</td>
<td>(c)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>(R)</td>
<td>(r)</td>
<td>(r)</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>(P)</td>
<td>(p)</td>
<td>(p)</td>
<td>(p)</td>
<td>(p)</td>
</tr>
</tbody>
</table>

peer\(^*\) routes. Such routes, carry a cost \(r^+\), better than the cost \(c\) of a customer route. Thus, costs are ordered as follows, \(r^+ < c < r < p < \bullet\).

Section 2.2 considers the possibility to export peer\(^*\) routes to all neighbors, or only to customers. Depending on the export policies regarding peer\(^*\) routes, the mapping performed by extenders \(R^*\), \(C\), \(R\), and \(P\) is slightly different. Table 4.3 shows the corresponding mapping when peer\(^*\) routes are exported to all neighbors. Table 4.4 shows the different mapping performed by each extender when peer\(^*\) routes are exported only to customers.

Note the difference between the entries in first column of Tables 4.3 and 4.4 representing the export policies for peer\(^*\) routes. In the first table, each extender maps the \(r^+\) cost to a cost different from \(\bullet\). In contrast, in the second table, this is true only for the \(P\) extender, encoding the fact that peer\(^*\) routes are exported through provider links.

### 4.2 Pseudo-code

After the introduction of a routing model for BGP, this Section presents in Algorithm 1 the pseudo-code of BGP for a fixed destination, when \(u\) receives a route \((\alpha, vP)\) from an in-neighbor \(v\). The code includes three important variables \(\beta\), \(RouteT[w]\), \(Route\): variable \(\beta\) stores the cost of the route learned at \(u\) from route \((\alpha, vP)\); variable \(RouteT[w]\) stores the candidate route to reach the destination via the in-neighbor \(w\); and variable \(Route\) stores the elected route to reach the destination. The special case of \(RouteT[u]\) stores the route announced at \(u\). Anycast routing is anticipated, in which case more than one node announces a route. The functions \(Cost(r)\) and \(Path(r)\) return, respectively, the cost and path components of a route \(r\).

In Line 2, \(u\) elects its best route among the set of candidate routes excluding the route learned from \(v\). Line 3 tests two things. The first test checks if the path \(vP\) identifies a loop. And, the second test checks if \(v\) informed \(u\) that \(u\) no longer has viable route via \(v\). In either case, \(u\) stores there is no longer a candidate route for the destination via \(v\), denoted by the couplet \((\bullet, \bullet)\) assigned to variable \(RouteT[v]\) in Line 4.

#### Algorithm 1 BGP code for when node \(u\) receives routing message \((\alpha, vP)\) from its neighbor \(v\).

\[
\begin{align*}
\beta & := T[uv](\alpha) \\
Route & := \Delta \{ RouteT[w] | w \neq v \} \\
\text{if } u \in vP \text{ or } \beta = \bullet \text{ then} & \\
\quad RouteT[v] & := (\bullet, \bullet) \\
\text{else} & \\
\quad RouteT[v] & := (\beta, vP) \\
\quad Route & := Route \triangle RouteT[v] \\
\text{if } Route \text{ has changed then} & \\
\quad \text{for all } w \text{ neighbor do} & \\
\quad \quad \text{send } (Cost(Route), uPath(Route)) \text{ to } w \\
\end{align*}
\]
4.3 Cost inflation and isotonicity

Section 4.1 introduces costs and extenders. Sobrinho in [6], defines two properties of routing policies: inflation\(^1\) and isotonicity\(^2\). Each of these properties defines a class of routing policies and they each provide insight regarding the stability of BGP.

An extender \( T[uv] \) of a link \( uv \) is inflationary if the following condition is verified.

\[
\forall \alpha \in \Sigma \ T[uv](\alpha) \succeq \alpha, \tag{4.3}
\]

which means extender \( T[uv] \) inflates all possible costs, that is, any cost \( \alpha \) of a route exported through link \( uv \) can never become better at the tail node.

The extenders defined to model the GR routing policies are inflationary, which can be verified by analyzing Table 4.1. To verify that an extender is inflationary, one can see that for each input cost in each column the mapping cost is always worse or the same as the input cost. Take, for instance, extender \( C \), it maps cost \( c \) to the same cost, and all other costs to \( \bullet \) (the worst cost possible). Thus, \( C \) is inflationary. The same is true for extender \( R \). Now, take extender \( P \), it maps all costs to \( p \), since \( p \) is the worst cost (other than \( \bullet \)), this implies that \( P \) is inflationary as well.

Although, the extenders of the baseline GR routing policies are inflationary, the same is not true for the GR routing policies with either siblings or peer+’s. In the former case, the first entry in Table 4.2 shows that extender \( C \) is non-inflationary, because for \( n > 0 \), \( C(c,n) = (c,0) \prec (c,n) \). For the same reason, extender \( P \) is also non-inflationary, \( n > 0 \), \( P(p,n) = (p,0) \prec (p,n) \). In the case of peer+’s, extender \( R^+ \) is non-inflationary regardless if all routes are exported to all neighbors or just to customers. The analyses of Tables X and Y show that extender \( R^+ \) maps cost \( c \) to \( r^+ \prec c \), showing extender \( R^+ \) is non-inflationary regardless of the export policies applied for peer+ routes.

The second property of extender discussed here is isotonicity. Isotonicity means that the relative preference between any two costs at a node is not inverted between the costs learned from them at a neighbor node. The extender \( T[uv] \) of link \( uv \) is isotone if \( T[uv] \) verifies the following condition:

\[
\forall \alpha, \beta \in \Sigma \ \alpha \prec \beta \implies T[uv](\alpha) \preceq T[uv](\beta), \tag{4.4}
\]

which means the relative preference between two costs extended by \( T[uv] \) is not inverted at \( u \). Routing policies are said to be isotonoe if all the extenders used to model those routing policies are isotonoe.

To check if the GR routing policies are isotonoe, one can analyze the mapping costs in each row of Table 4.1 and verify if the cost in one column is always better or the same as the cost in the next column\(^3\). In order words, if the costs in a row, from left to right, respect the order \( \preceq \), then the corresponding extender is isotonoe. For instance, the first row contains \( c \prec \bullet = \bullet \). Thus, extender \( C \) is isotonoe. The same is true for extenders \( R \) and \( P \).

Isotonicity is also verified for the GR routing policies with siblings as can be verified from Table 4.2. The GR routing policies with peer+’s are isotonoe only if peer+ routes are exported to all neighbors. Consider Tables 4.3 and 4.4. The first row in the latter table has \( \bullet \succ r^+ \prec \bullet = \bullet \), the better cost \( r^+ \) maps to \( \bullet \), while the worse costs \( c \) maps to \( r^+ \prec \bullet \). Consequently, extender \( R^+ \) does not verify isotonicity. If, instead, routes are exported to all neighbors then \( R^+ \) verifies isotonicity, because both a peer+ route and a customer route are exported to a peer+ neighbor originating a peer+ route at the neighbor, as evidenced in the first row of Table 4.3.

---

\(^1\)Inflation is also known as absorbency.

\(^2\)Isotonicity is also known as monotonicity.

\(^3\)This is only possible because the input costs in each column are ordered according to \( \prec \).
Chapter 5

Elimination of Oscillations

Chapter 2 describes BGP routing with three different routing policies. It shows that the policies applied at each node impact the protocol's ability to get to a stable state: routing policies applied around network cycles may conflict, causing permanent state oscillations of BGP. This Chapter investigates the possibility to solve the problem of permanent state oscillations.

Section 5.1 presents an example of shortest path routing, which is used to illustrate how each node is able to detect and eliminate oscillations based strictly on their local information. Later, Section 5.2 shows that the same approach is also applicable for interdomain routing and highlights the fact that not all nodes need to be able to detect oscillations for the protocol to reach a stable state. Then, Section 5.3 highlights the importance of isotonicity in assuring termination and that even when not all nodes apply isoton routing the policies, the previously devised solution is able to terminate in some cases where standard BGP is not.

5.1 Shortest-path routing with negative costs

This Chapter starts with an example of shortest path routing because this is the most familiar form of routing, thus serving to illustrate how the problem of permanent state oscillations can be mitigated resorting strictly to the information available at each node. Shortest path routing follows the same routing model introduced in Section 4.1. The cost of a route is represented by an integer number. Lower integer values correspond to better costs. Each link is associated with its own cost, which represents the cost of

Figure 5.1: Node \( u \) detects a routing loop and discards the route learned from \( v \). BGP reaches a stable state. If link \( uv \) fails, then \( u \) elects the route learned from \( v \). Integers represent additive costs and the destination is announced by \( t \).

<table>
<thead>
<tr>
<th>( T )</th>
<th>( u )</th>
<th>( v )</th>
<th>( t )</th>
<th>( w )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0, 0</td>
<td>3, 4</td>
<td>3, 3</td>
<td>2, 2</td>
<td>1, 1</td>
</tr>
<tr>
<td>2</td>
<td>0, 0</td>
<td>4, 4</td>
<td>3, 3</td>
<td>2, 2</td>
<td>1, 1</td>
</tr>
<tr>
<td>3</td>
<td>0, 0</td>
<td>4, 4</td>
<td>3, 3</td>
<td>2, 2</td>
<td>1, 1</td>
</tr>
<tr>
<td>4</td>
<td>0, 0</td>
<td>3, 3</td>
<td>3, 3</td>
<td>2, 2</td>
<td>1, 1</td>
</tr>
</tbody>
</table>

Table 5.1: BGP terminates for the network of Figure 5.1. Routes learned from neighbors at successive instants of time. Discarded routes have a shaded background. Elected routes are underlined. The symbol • indicates no candidate route is available from the neighbor.
Figure 5.2: Nodes $u$ and $v$ detect a routing loop and discard the routes learned from their clockwise neighbor. The state of BGP oscillates forever while SS-BGP terminates. Integers represent additive costs and the destination is announced by $t$.

Table 5.2: BGP does not terminate for the network of Figure 5.2. Routes learned from neighbors at successive instants of time. Discarded routes have a shaded background. Elected routes are underlined. The symbol • indicates no candidate route is available from the neighbor.

<table>
<thead>
<tr>
<th></th>
<th>$u$</th>
<th></th>
<th>$v$</th>
<th></th>
<th>$w$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$t$</td>
<td>$v$</td>
<td>$t$</td>
<td>$w$</td>
<td>$t$</td>
<td>$u$</td>
</tr>
<tr>
<td>1</td>
<td>$0, t$</td>
<td>•</td>
<td>$3, t$</td>
<td>•</td>
<td>$2, t$</td>
<td>•</td>
</tr>
<tr>
<td>2</td>
<td>$0, t$</td>
<td>$-1, vt$</td>
<td>$3, t$</td>
<td>$3, wt$</td>
<td>$2, t$</td>
<td>$1, vt$</td>
</tr>
<tr>
<td>3</td>
<td>$0, t$</td>
<td>$-1, vt$</td>
<td>$3, t$</td>
<td>$2, wt$</td>
<td>$2, t$</td>
<td>$0, wt$</td>
</tr>
<tr>
<td>4</td>
<td>$0, t$</td>
<td>$-2, wut$</td>
<td>$3, t$</td>
<td>$1, wut$</td>
<td>$2, t$</td>
<td>$0, wut$</td>
</tr>
<tr>
<td>5</td>
<td>$0, t$</td>
<td>$-1, vt$</td>
<td>$3, t$</td>
<td>•</td>
<td>$2, t$</td>
<td>$1, vt$</td>
</tr>
</tbody>
</table>

Sending traffic through that link. The cost of a route learned by $u$ through a link $uv$ is obtained by adding the cost of that link to the costs of the route elected at $v$.

Consider the network in Figure 5.1. Remember from Section 4.1 that routes travel in the opposite direction to that pointed by the links. Assume that a route takes one unit of time to travel across a link. Table 5.1 shows the evolution of BGP at successive units of time. For each node, the table holds a column with the routes learned from each in-neighbor. Elected routes are underlined. The state of BGP oscillates forever while SS-BGP terminates. Integers represent additive costs and the destination is announced by $t$.

Table 5.1: BGP does not terminate for the network of Figure 5.2. Routes learned from neighbors at successive instants of time. Discarded routes have a shaded background. Elected routes are underlined. The symbol • indicates no candidate route is available from the neighbor.

Then, at $T = 3$, $v$ elects route $(2, wut)$ learned from $w$ over its route $(3, t)$ via $t$, and it exports the new elected route to $u$. At $T = 4$, $u$ learns from $v$ route $(3, wut)$ and notices that this routes identifies a loop. Thus, $u$ is forced to discard the route and keep electing the route via $t$. At this point, there are no routes in transit. Therefore, BGP has stabilized with $u$ electing route $(0, t)$, $v$ electing $(2, wut)$, and $w$ electing route $(1, ut)$ to reached the destination. This example illustrates the importance of detecting and preventing the propagation of routes past routing loops. Previously, at $T = 4$, node $u$ discards route $(3, wut)$ instead of promoting it to a candidate route. This action can be appreciated if, for instance, link $ut$ fails. In that scenario, and if loops are not detected, then $u$ would elect a route which loops through $u$ and propagate it further around the cycle. On the other hand, by $u$ discarding the route which identifies a loop, when link $ut$ fails, $u$ declares no route to $t$ and exports this information to $w$.

With respect to shortest path routing, it is well established that as long as the costs of links are non-negative, BGP always terminates [5]. However, the use of links with negative costs may be useful in some scenarios. For instance, consider that the operator of the network in Figure 5.1 wants traffic to flow through link $uv$ instead of link $ut$ (maybe link $ut$ is congested or less reliable). The quickest and simplest way to accomplish this would be change the cost of link $uv$ from 1 to $-4$, forcing $u$ to choose path $vt$ instead of forwarding traffic directly to $t$. 

24
Figure 5.2 shows the corresponding network after applying the suggested modification. Let us verify what is the result of BGP for the modified network when $t$ announces a new destination. The evolution of BGP can be followed with the help of Table 5.2. At $T = 1$, nodes $u$, $v$, and $w$ elect, respectively, routes $(0, t)$, $(3, t)$, $(2, t)$, and they export their respective elected routes counter-clockwise around the cycle. At $T = 2$,

- $u$ elects route $(-1, vt)$ learned from $v$;
- $v$ learns route $(3, wt)$, but still elects route $(3, t)$ since the latter corresponds to a shorter path;
- and $w$ elects route $(1, ut)$ learned from $u$.

The newly elected routes by $u$ and $w$ are exported counter-clockwise around the cycle. At $T = 3$, $v$ elects route $(2, wt)$ learned from $w$; and $w$ elects route $(0, wt)$ learned from $u$. They both export their respective newly elected route counter-clockwise around the cycle. Hence, at $T = 4$,

- $u$ learns route $(-2, vwut)$ from $v$, but path $vwut$ identifies a loop. Therefore, although $u$ prefers this route over its alternative route via $t$, it must discard the new route and elect route $(0, t)$ instead;
- similar to $u$, node $v$ also discards route $(1, wuvt)$ and elects the worse route $(3, t)$.

The routes elected by $u$ and $v$ are, once again, exported counter-clockwise around the cycle. As a consequence, at $T = 5$, the protocol returns to the same state it was at $T = 2$, with nodes $u$, $v$, and $w$ electing, respectively, routes $(-1, vt)$, $(3, t)$, and $(1, ut)$. From now on, the state of BGP will repeatedly evolve through the same sequence of states observed from $T = 2$ to $T = 5$, without any of the nodes ever sticking to a single route.

However, an argument can be made that, at $T = 4$, both $u$ and $v$ detect that something is amiss, in that, they both discard their best route to the destination because these routes identify a loop. With knowledge of the global network topology, it can be identified that the cycle $uwvw$ has cost $2(-2 = -4 + 1 + 1)$. This fact justifies routes being dispatched around the cycle and arriving back at the original node with a lower cost than the one with which they were originally exported. Based on this observations, SS-BGP adds to BGP the following test and action:

**SS-BGP:** If a route learned from a neighbor carries the best cost to reach the destination, but the path component of the route indicates a routing loop, then that neighbor is deactivated.

Deactivating a neighbor corresponds discarding all routes learned from that neighbor. Saying that a node $u$ deactivated a link $uw$ is the same as saying that $u$ deactivated $v$.

For the network in Figure 5.2 SS-BGP evolves similarly to BGP up until $T = 4$, when $u$ and $v$ detect a conflict. Therefore, they deactivate their respective neighbors $v$ and $w$, discarding all future routes coming from these nodes. Table 5.1 shows the evolution of SS-BGP from $T = 4$. At $T = 5$, $w$ elects route $(1, ut)$ and exports it to $v$, as before. Meanwhile, in contrast to BGP, $u$ does not import route $(-1, vt)$ learned from $v$. As a result, $u$ keeps its election of route $(0, t)$ and does not export any new route to $w$. Similarly, at $T = 6$, $v$ also does not import route $(-1, wuvt)$ learned from $w$ and keeps electing route $(3, t)$. As a consequence, the protocol stabilizes with $u$ electing route $(0, t)$, $v$ electing route $(3, t)$, and $w$ electing route $(1, ut)$ to reach the destination.

Notice links $uw$ and $vw$ were both deactivated. However, deactivating just one of them would eliminate the routing loop and, thereby, ensure the protocol would have reached a stable state. Since SS-BGP performs a distributed detection and elimination of oscillations, then it is susceptible to race conditions, as verified in the example of Figure 5.2. However, the simulation results in Chapter 7 show that a very small number of links are deactivated for SS-BGP to reach a stable state in the evaluated topologies.
Table 5.3: SS-BGP terminates for the network of Figure 5.2. Discarded routes have a shaded background. Elected routes are underlined. The symbol $\bullet$ indicates no candidate route is available from the neighbor.

Section 6.3 presents an alternative elimination method which allows a route learned from a deactivated neighbor to be promoted to a candidate route at the cost of an increase of its cost.

This example also suggests that not all nodes are required to use SS-BGP to ensure termination: if just $u$ or $v$ deploy SS-BGP, then the protocol still terminates. However, if only $w$ deploys SS-BGP, then the protocol can still oscillate. Partial deployment is discussed further in Section 6.5.

### 5.2 Interdomain routing with siblings

The problem of permanent state oscillations is now presented in the context of interdomain routing. The example illustrated in Figure 5.3 considers interdomain routing using the GR routing policies with siblings, which were introduced in Section 2.2. The notation used here follows the model introduced in Section 4.1. Nodes $x$ and $y$ are providers of $t$ and they are, respectively, siblings of $u$ and $v$. Furthermore, around the cycle $uvwu$, $v$ is a customer of $u$; $w$ is a customer of $v$; and $u$ is a sibling of $w$. The evolution of SS-BGP is shown in Table 5.4. Node $t$ announces the destination. At $T = 1$, both $x$ and $y$ elect a route $(c, 0, t)$ and export this route to their respective siblings $u$ and $v$. At $T = 2$, $u$ elects the route $(c, 1, xt)$ learned from $x$; and $v$ elects the route $(c, 1, yt)$ learned from $y$. Both nodes export its elected route counter-clockwise around the cycle. At $T = 3$,

- $u$ learns the customer route $(c, 0, vyt)$ from $v$ and elects this route over the customer route with one sibling hop learned from $x$;
- and $w$ elects route $(c, 2, uxt)$ learned from $x$.

The routes elected by $u$ and $w$ are then exported counter-clockwise around the cycle. At $T = 4$,

- $v$ elects a customer route with no sibling hops $(c, 0, wuxt)$ learned from $w$, which is better than the customer route with one sibling hop learned from $y$;
- $w$ elects the route $(c, 1, uvyt)$ learned from $u$.

Once again, nodes $v$ and $w$ export their elected routes to their respective counter-clockwise neighbors. Then, at $T = 5$, nodes $u$ and $v$ learn routes $(c, 0, vwuxt)$ and $(c, 0, wvyt)$, respectively. Both nodes prefer its new customer route with no sibling hops over its customer route with one sibling hop learned from outside the cycle. However, the new routes identify a loop and, therefore, must be discarded. As a consequence, nodes $u$ and $v$ elect, respectively, routes $(c, 1, xt)$ and $(c, 1, yt)$. Hence, the protocol has reverted back to the state it was on at $T = 2$, implying permanent state oscillations.

However, the checking condition included in SS-BGP is verified, once again, by $u$ and $v$ at $T = 5$: they learn a route which simultaneously is its best route and identifies a routing loop. Which means, that if they both use SS-BGP, then, at $T = 5$, $u$ deactivates its neighbor $v$, and $v$ deactivates $w$. Consequently, in the next unit of time, $u$ discards route $(c, 0, vt)$ learned from its deactivated neighbor $v$, keeping its
Figure 5.3: In contrast to BGP, SS-BGP terminates with the GR routing policies with siblings. $S$ and $C$ represent sibling and customer links, respectively, and the destination is announced by $t$.

Table 5.4: SS-BGP terminates for the network of Figure 5.3. Discarded routes have a shaded background. Elected routes are underlined. The symbol $\bullet$ indicates no candidate route is available from the neighbor.

<table>
<thead>
<tr>
<th></th>
<th>$u$</th>
<th></th>
<th>$v$</th>
<th></th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$t$</td>
<td></td>
<td>$t$</td>
<td></td>
<td>$u$</td>
</tr>
<tr>
<td>2</td>
<td>$c, 1, xt$</td>
<td>$\bullet$</td>
<td>$c, 1, yt$</td>
<td>$\bullet$</td>
<td>$\bullet$</td>
</tr>
<tr>
<td>3</td>
<td>$c, 1, xt$</td>
<td>$c, 0, vyt$</td>
<td>$c, 1, yt$</td>
<td>$\bullet$</td>
<td>$c, 2, uxt$</td>
</tr>
<tr>
<td>4</td>
<td>$c, 1, xt$</td>
<td>$c, 0, vyt$</td>
<td>$c, 1, yt$</td>
<td>$c, 0, wuxt$</td>
<td>$c, 1, uyt$</td>
</tr>
<tr>
<td>5</td>
<td>$c, 1, xt$</td>
<td>$c, 0, vwxu$</td>
<td>$c, 1, yt$</td>
<td>$c, 0, wuxt$</td>
<td>$c, 1, uyt$</td>
</tr>
<tr>
<td>6</td>
<td>$c, 1, xt$</td>
<td>$c, 0, vyt$</td>
<td>$c, 1, yt$</td>
<td>$c, 0, wuxt$</td>
<td>$c, 2, uxt$</td>
</tr>
<tr>
<td>7</td>
<td>$c, 1, xt$</td>
<td>$c, 0, vyt$</td>
<td>$c, 1, yt$</td>
<td>$c, 0, wuxt$</td>
<td>$c, 2, uxt$</td>
</tr>
</tbody>
</table>

Serves this example to illustrate that the methods to detect and eliminate oscillations included in SS-BGP apply both to shortest path routing and interdomain routing.

5.3 Interdomain routing with peer+s

Section 2.2 introduces the GR routing policies with peer+s, accounting for nodes which prefer routes learned from some of its peers over their customer routes. Such a peer is called a peer+ and the routes learned from it are called peer+ routes. Two options were proposed regarding export rules for these routes: either peer+ routes are exported only to customers, as with peer routes, or they are exported to all neighbors, as with customer routes. This Section considers both options, highlighting the implications of each one for the stability of BGP and SS-BGP.

The nodes in the network of Figure 5.4 apply the GR routing policies with peer+s. The links $R^+$ and $C$ peer+ links and customer links, as described in Section 4.1. For now, assume peer+ routes are exported only to customers. The evolution of BGP and SS-BGP with these policies is shown in Table 5.5. As always, the routing process is started when $t$ announces a destination to its providers $u$, $v$, and $w$. At $T = 1$, each provider of $t$ elects a route $(c, t)$ and exports it counter-clockwise around the cycle $uvw$. At $T = 2$, each of $u$, $v$, and $w$ elects a peer+ route learned from their clockwise neighbor. Since peer+ routes are not exported to peer+ neighbors, then they all inform their respective counter-clockwise neighbors that they no longer hold a route to reach the destination (although, they do). Consequently, at $T = 3,$
Figure 5.4: SS-BGP does not terminate if peer+ routes are exported only to customer. If peer+ routes are exported to all neighbors, then SS-BGP terminates. $R^*$ and $C$ represent peer+ and customer links, respectively, and the destination is announced by $t$.

Table 5.5: SS-BGP does not terminate for the network of Figure 5.4 when peer+ routes are exported only to customers. Discarded routes have a shaded background. Elected routes are underlined. The symbol • indicates no candidate route is available from the neighbor.

<table>
<thead>
<tr>
<th></th>
<th>$u$</th>
<th></th>
<th>$v$</th>
<th></th>
<th>$w$</th>
<th></th>
<th></th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$t$</td>
<td>$v$</td>
<td>$t$</td>
<td>$w$</td>
<td>$t$</td>
<td>$u$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$c, t$</td>
<td>•</td>
<td>$c, t$</td>
<td>•</td>
<td>$c, t$</td>
<td>•</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$c, t$</td>
<td>$r^+, vt$</td>
<td>$c, t$</td>
<td>$r^+, wt$</td>
<td>$c, t$</td>
<td>$r^+, ut$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$c, t$</td>
<td>$r^+, vt$</td>
<td>$c, t$</td>
<td>$r^+, wt$</td>
<td>$c, t$</td>
<td>•</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.6: SS-BGP terminates for the network of Figure 5.4 when peer+ routes are exported to all neighbors. Discarded routes have a shaded background. Elected routes are underlined. The symbol • indicates no candidate route is available from the neighbor.

<table>
<thead>
<tr>
<th></th>
<th>$u$</th>
<th></th>
<th>$v$</th>
<th></th>
<th>$w$</th>
<th></th>
<th></th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$t$</td>
<td>$v$</td>
<td>$t$</td>
<td>$w$</td>
<td>$t$</td>
<td>$u$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$c, t$</td>
<td>•</td>
<td>$c, t$</td>
<td>•</td>
<td>$c, t$</td>
<td>•</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$c, t$</td>
<td>$r^+, vt$</td>
<td>$c, t$</td>
<td>$r^+, wt$</td>
<td>$c, t$</td>
<td>$r^+, ut$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$c, t$</td>
<td>$r^+, vt$</td>
<td>$c, t$</td>
<td>$r^+, wt$</td>
<td>$c, t$</td>
<td>$r^+, ut$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$c, t$</td>
<td>$r^+, vt$</td>
<td>$c, t$</td>
<td>$r^+, wt$</td>
<td>$c, t$</td>
<td>$r^+, ut$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$c, t$</td>
<td>$r^+, vt$</td>
<td>$c, t$</td>
<td>$r^+, wt$</td>
<td>$c, t$</td>
<td>$r^+, ut$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nodes $u$, $v$, and $w$ revert back to elect route $(c, t)$ they each previously learned from $t$. Which means, the global state of the protocol also reverted back to the state it was on at $T = 1$, implying a permanent state oscillations.

SS-BGP requires a node to identify a routing loop to be able to detect possible permanent oscillations. But, in this example, even though the protocol oscillates, none of the nodes identified a single routing loop at any point. Serves this example to show that not all permanent state oscillations are caused by routing loops and, because of that, SS-BGP is not able to ensure termination in all cases.

Consider that $t$ announces a new destination again, but this time assume peer+ routes are exported to all neighbors. The evolution of SS-BGP with these policies is shown in Table 5.6. As before, at $T = 2$, each of, $u$, $v$, and $w$, elects a peer+ route learned from its clockwise neighbor. But now, they export their elected peer+ route to their counter-clockwise neighbors. Consequently, at $T = 3$, $u$ elects route $(r^+, vwt)$ learned from $v$; $v$ elects route $(r^+, wut)$ learned from $w$; and $w$ elects route $(r^+, wut)$ learned from $v$. Once again, the new routes elected routes by $u$, $v$, and $w$ are exported counter-clockwise around the cycle. At $T = 4$, $u$ learns route $(r^+, vwwt)$ from $v$; $v$ learns route $(r^+, wwwwt)$ from $w$; and $w$ learns route $(r^+, vwwwt)$ from $u$. Each of these nodes, by inspecting the path of its new learned route, realizes that route identifies a loop. With BGP, these routes are immediately discarded leaving the protocol to oscillate forever. However, with SS-BGP, each of $u$, $v$, and $w$ detects that the cost $r^+$ of the route that denounces a loop is better than the cost $c$ of its direct route to $t$ and, consequently, deactivates its clockwise neighbor. Thus, at $T = 4$, SS-BGP reaches a stable state with every node electing route $(c, t)$.

A small modification in the export policies was the difference between SS-BGP being able to detect and avoid the oscillations, and SS-BGP oscillating in the same way as BGP. The fact, that peer+ routes were exported to all neighbors, allowed SS-BGP to detect and prevent the oscillation, as it was able to do...
in the examples of Figures 5.2 and 5.3. The reason for this has to do with a property which characterizes the routing policies applied in the network of these two Figures and the GR routing policies with peer’s when peer’s routes are exported to all neighbors. This property is isotonicity. Isotonicity was introduced and described in Section 4.3. This Section also explains why isotonicity is verified when peer’s routes are exported to all neighbors and it is not verified when their are not. In [16], it is shown that isotonicity ensures termination of SS-BGP.

The example of Figure 5.4 can also be used to show that, in some cases, SS-BGP still terminates even when not all nodes apply isotone routing policies: if only one of the nodes does export peer’s routes to all neighbors, but the other two do so, then SS-BGP still terminates. For instance, consider $u$ is the only one which does not export peer’s routes to peer’s. In that case, $u$ still detects a routing loop and deactivates $v$ at $T = 4$, which is sufficient to ensure SS-BGP reaches a stable state.

From now on, every time a reference is made to the GR routing policies with peer’s, assume that peer’s routes are exported to all neighbors. Unless, it is explicitly said otherwise.
Chapter 6

Self-Stable BGP

This Chapter formalizes the conclusions taken from the previous Chapter. It makes a distinction between two different types of routing loops. One which occurs a finite number of times in each execution of the protocol and another which recurs an infinite number of times during such an execution. SS-BGP is presented as an addon to BGP, which adds the ability to distinguish between the two of them and eliminate the latter, which are associated with permanent state oscillations.

Therefore, Section 6.1 relates termination with routing loops. Then, Section 6.2 presents the pseudo-code for the canonical version of SS-BGP and compares it with that of BGP’s (Section 4.2), highlighting the small differences between the two. The following Sections 6.3 and 6.4 suggest, respectively, better alternatives for the detection and elimination methods of SS-BGP. Finally, Section 6.5 discusses SS-BGP in partial deployment.

6.1 Termination and routing loops

The state of BGP is composed of the candidate and elected routes at the various nodes and the routing messages in transit across the links. A state is stable if there are no routing messages in transit. BGP terminates if a stable state is eventually reached, whatever the initial state and whatever the delays of routing messages across the links.

The examples of Chapter 5 suggest that, in some cases, permanent oscillations are related to routing loops. A routing loop is a static object which refers to the propagation of a route around a network cycle. It is characterized by a triplet of the form $(C, u, \alpha)$, where $C$ is a cycle, $u$ is a node of $C$, and $\alpha$ is a cost. A route with cost $\alpha$ exported by $u$ around $C$ is learned back at $u$ with cost $T[uCu](\alpha)$, where $uCu$ is the circuit around $C$ starting and ending at $u$ and $T[uCu]$ is the composition of the extenders of all links of the circuit (see Section 4.1). If a route learned by a node $u$ from around a cycle $C$ has a better cost than the cost $\alpha$ of the dispatched by $u$, then the corresponding routing loop is persistent. In symbols, a routing loop $(C, u, \alpha)$ is persistent if the following condition is verified.

$$T[uCu](\alpha) \prec \alpha \quad (6.1)$$

In contrast, a routing loop is temporary if the route learned from around the cycle has a worse cost than or the cost as the dispatched route, $T[uCu](\alpha) \succeq \alpha$.

The definition of persistent routing loop (Condition 6.1) resembles the detection condition included in SS-BGP (see test condition defined in Section 5.1). Thus, SS-BGP extends the routing loop detection method already included in BGP, giving it the ability to distinguish between temporary and persistent routing loops. In addition, it also eliminates latter.
The examples of Chapter 5 suggest that the existence of persistent routing loops in a network may cause BGP to oscillate. The work of Sobrinho [5] provides the basis to demonstrate that persistent routing loops are in fact associated with permanent state oscillations. Sobrinho relates termination of BGP in a network to a property of the routing policies around the cycles of that network. A cycle is absorbent if, for every combination of routes learned by its nodes externally to the cycle and exported to their neighbors, at least one node elects the route learned externally to the route learned from the neighbor or, for all nodes, the route learned externally equals the route learned from the neighbor. Otherwise, the cycle is called non-absorbent. In symbols, a cycle \( C = u_0u_1...u_{n-1}u_0 \) is absorbent if

\[
\forall \alpha_0, \alpha_1, ..., \alpha_{n-1} (\exists \leq i < n \, \alpha_i \prec T[u_iu_{i+1}](\alpha_{i+1})) \lor \forall i \leq < n \, \alpha_i = T[u_iu_{i+1}](\alpha_{i+1})) \tag{6.2}
\]

where \( \alpha_0, \alpha_1, ..., \alpha_{n-1} \) represent costs of routes that nodes involved in the cycle \( C \) learn from outside the cycle.

Regarding absorbent cycles and termination of BGP, the following theorem was proven in [5].

**Theorem 1** If BGP does not terminate in a network for some destination, then there is at least one non-absorbent cycle in the network.

It can also be demonstrated that if the extender of at least one circuit around a cycle \( C \) is non-inflationary, then \( C \) is non-absorbent. Since the definition of persistent routing loop implies a non-inflationary circuit: for a routing loop \((C, u, \alpha)\) to be persistent, the extender of circuit \( uC\alpha \) must be non-inflationary. Then, from these two last statements and Theorem 1, the following proposition can be drawn.

**Proposition 1** If there is a persistent routing loop, then BGP does not terminate for some destination, either in the network or in some sub-network obtained after a number of link failures.

Thus, this proposition justifies the chase and elimination of persistent routing loops. The two features added to BGP in the form of SS-BGP.

The example of Section 5.3 highlights the fact that SS-BGP terminates if the isotonicity property is verified by the routing policies applied at each node.

In the examples of Section 5.3, SS-BGP terminated when the routing policies applied at each node verified the isotonicity property. But, it was unable to terminate when all nodes applied non-isotone routing policies, because none of the nodes detected a persistent routing loop. This proves that not all permanent state oscillations are caused by persistent routing loops. It can be demonstrated [5] that, if routing policies are isotone, then a cycle \( C \) is non-absorbent if, and only if the extender of at least one routing loop around \( C \) is persistent. This statement and Theorem 1 prompt the following proposition.

**Proposition 2** If routing policies are isotone and BGP does not terminate for some destination, then there is a persistent routing loop.

This proposition implies that, in the case of isotone routing policies, the progressive elimination of persistent routing loops eventually leads BGP into a stable state. It was proved in [16] that, in the case of isotone routing policies, if SS-BGP does not terminate, then there is a node which detects and eliminates a persistent routing loop. Since persistent routing loops are finite, it can be concluded that SS-BGP is guaranteed to terminate with isotone routing policies.

In sum, SS-BGP is an addon to BGP, which performs a distributed detection and elimination of persistent routing loops. The elimination of such loops is justified by the fact that they are associated with permanent state oscillations, which shows that SS-BGP improves the stability properties of BGP. Furthermore, for the important class of persistent routing policies, SS-BGP guarantees termination, in contrast to BGP.
6.2 Pseudo-code

Algorithm 2 presents the pseudo-code of SS-BGP for a fixed destination, when $u$ receives a route $(\alpha, vP)$ from an in-neighbor $v$. Variable $\beta$ stores the cost of the route learned at $u$ from route $(\alpha, vP)$. An in-neighbor is active if routes learned from that neighbor are imported. Otherwise, the in-neighbor is inactive. The set $N$ stores the in-neighbors that are inactive. Initially, $N$ is empty, meaning all in-neighbors are active.

Variable $RouteT[w]$ stores the candidate route to reach the destination via in-neighbor $w$, and variable $Route$ stores the route elected by $u$ to reach the destination. The special case of $RouteT[u]$ stores the route announced at $u$. The functions $Cost(r)$ and $Path(r)$ return, respectively, the cost and path components of a route $r$.

The test in Line 1 checks if $v$ is inactive. If the test is positive, then $u$ does not import the route received from $v$, setting $\bullet$ as the cost of the route learned from $v$ in Line 2. The election of a route at Line 5 runs over all active in-neighbors of $u$ with the exception of $v$. The only differences between SS-BGP (Algorithm 2) and BGP (Algorithm 1) are in Lines 1-4 and 7-8, which are absent in the latter protocol. The test in Line 7 is positive if $\beta \prec \text{cost}(Route)$ and $u \in Q$, that is, if the path learned from $v$ indicates a loop, while the cost learned from $v$ is better than the best of all candidate costs. If the test in Line 7 is positive, then $u$ deactivates $v$ in Line 8, by adding $v$ to set $N$, on the presumption that a persistent routing loop has been found.

**Algorithm 2 SS-BGP code for when node $u$ receives a routing message $(\alpha, vP)$ from its in-neighbor $v$.**

```plaintext
if $v \in N$ then
    $\beta := \bullet$
else
    $\beta := T[uv](\alpha)$
5: $Route := \bigtriangleup \{ RouteT[w] \mid w \neq v \}$
if $u \in vP$ or $\beta = \bullet$ then
    if $\beta \prec \text{Cost}(Route)$ then
        add $v$ to $N$
        $RouteT[v] := (\bullet, \bullet)$
10: else
    $RouteT[v] := (\beta, vP)$
    $Route := Route \bigtriangleup RouteT[v]$
if $Route$ has changed then
    for all $w$ in-neighbor do
        send $(\text{Cost}(Route), u\text{Path}(Route))$ to $w$
```

6.3 Improved oscillation detection method

SS-BGP detects persistent routing loops in a distributive manner. Therefore, it is susceptible to race conditions. The network in Figure 6.1 is used to illustrate an example of a race condition which causes a node to signal a persistent routing despite the loop being only temporary. This example considers routing with the GR routing policies with peer’s. Any routing loop around the cycle $wwwu$, starting and ending at $u$, is always temporary. However, in this example $u$, incorrectly, signals the routing loop as persistent. Assuming $t$ announces a new destination, BGP will always reach the same stable state, where $y$ elects route $(c, t)$ via $t$; $x$ elects route $(c, yt)$ via $y$; $u$ elects route $(c, yt)$ via $y$; $w$ elects route $(c, uyt)$ via $u$; and $v$ elects route $(c, wuyt)$ via $w$. Now, consider that, once a stable state is reached, link $yt$ fails. As consequence, $y$ is left without a candidate route to the destination announced by $t$. Thus,
Figure 6.1: When $yt$ fails, $u$ incorrectly signals a persistent routing loop. $R$ and $C$ represent peer and customer links, respectively, and the destination is announced by $t$.

$y$ sends an withdrawal to $u$ and $x$. Once $u$ receives the withdrawal from $y$, it is forced to elect another route from its current set of candidate routes. Notice that $u$ still believes, at this point, that the customer route $(c, xyt)$ learned from $x$ is valid. Therefore, $u$ elects this route and dispatches it around the cycle $uvwu$. Meanwhile, $u$ is informed by $x$ that route $(c, xyt)$ is not longer valid. Thus, $u$ elects the only candidate route it has left, route $(r, t)$ learned from $t$. Afterwards, the route dispatched by $u$ around the cycle arrives back at $u$ with a cost of $c$. Then, $u$ compares this cost with the cost $r$ of its alternative route and determines the routing loop around $uvwu$ is persistent. Therefore, $u$ deactivates its neighbor $v$. However, as stated before, any routing loop involving $uvwu$ is temporary. Then, why did $u$ detect a persistent routing loop?

By definition, a routing loop $(C', u, \alpha)$ is persistent if $T[uCu](\alpha) \prec \alpha$. The route originally dispatched by $u$ was route $(c, xyt)$. However, at the time $u$ detects the routing loop, it had already elected route $(r, t)$. Thus, $u$ did not exactly compare the cost $c$ of the dispatched route with the cost $c$ of route $(c, vwx yt)$ learned from around the cycle. If it did, then $u$ would determine the routing loop to be temporary since the two costs are exactly the same. Instead, $u$ compared the cost of route $(c, vwx yt)$ to the cost of route $(r, t)$, which misled $u$ to signal a false persistent routing loop.

Serves this example to illustrate that the test condition defined in SS-BGP does not, necessarily, compare costs $T[uCu](\alpha)$ and $\alpha$. Instead, the comparison is made between the cost $T[uCu](\alpha)$ and a cost $\alpha'$ carried by the best route at $u$ learned externally to the cycle at the moment it detects the routing loop. As with the example in Figure 6.1, it may occur that the route elected at the time a routing loop is detected, is not the same as the route originally dispatched. In this scenario, a node may be deceived to signal a persistent routing that does not exist. This is called a false positive.

For SS-BGP to be efficient, it is critical to minimize the number of nodes required to change their import policies. False positives are specially relevant because they imply an unnecessary change to the import policies of a node. At the time a node detects a routing loop, it can not be sure if its currently elected route is the same route which originated the routing loop. However, if this were true, then the tail of the looping path, that is the path without the loop, would have to match the path of the route currently elected. Therefore, the detection condition in SS-BGP can be improved, requiring the tail of the path learned from the neighbor, with the loop discounted, to match the path of the route currently elected. In Algorithm 2 we can write $vP$ in the form $CuQ$, where $uCu$ is a circuit starting and ending at $u$, and $uP$ is a path starting at $u$ and ending at the node announcing the destination. The test condition in Line 7 is then be narrowed to:

$$\text{if } \beta \prec \text{Cost(Route)} \text{ and } P = \text{Path(Route)} \text{ then}$$
$$\text{add } v \text{ to } N$$

The version of SS-BGP with the improved detection condition is called Improved SS-BGP (ISS-BGP). Back to the example in Figure 6.1 at $T = 4$, when $u$ learns route $(c, vwx yt)$. With ISS-BGP, $u$ identifies that the tail $xyt$ of path $vwx yt$ does not correspond to path $t$ of its currently elected route. Hence, in
contrast to what happens with SS-BGP, \( u \) determines the routing loop is only temporary and does not deactivate \( v \).

### 6.4 Improved oscillation elimination method

Some of the examples presented in Chapter 5 highlight that sometimes deactivating a neighbor may be an extreme measure which may have an impact on connectivity. Take the example in Figure 5.4 where peer+ routes are exported to all neighbors. After the announcement of a destination by \( t \), SS-BGP terminates at cost of each node deactivating its clockwise neighbor around \( uvwu \).

The deactivation of a neighbor \( y \) by a node \( x \) is equivalent to breaking link \( xy \) from the network. Figure 6.2 presents the equivalent for network the network of Figure 5.4 after SS-BGP terminates. Now, consider link \( ut \) fails. In this scenario, \( u \) is left with no candidate route to reach \( t \). Although, there is a peer+ route via \( v \). Imagine \( u, v, \) and \( w \) are tier-1 ASs. In such a scenario, all customers of \( u \) are suddenly unable to reach the destinations owned by \( t \). Thus, the action previously taken to ensure the protocol terminated caused a large portion of the Internet to become unconnected.

In the example of Figure 5.4, three nodes detect a persistent routing loop and deactivate their respective clockwise neighbor to eliminate the loop. However, all three persistent routing loops can be all but eliminated if only one node deactivates its clockwise neighbor. Which means that some deactivations are unnecessary to guarantee termination. SS-BGP performs a distributive detection of persistent routing loops. Therefore, it is susceptible to race conditions which lead to multiple nodes to eliminate a persistent routing loop concurrently. Despite that, even if, for instance, only \( u \) deactivates its clockwise neighbor \( v \) and link \( ut \) fails, then \( u \) is still left without a route to reach the destinations announced by \( t \). Therefore, the issue is not the extra deactivations, but the fact that routes from a deactivated neighbor are no longer viable candidate routes.

The goal behind deactivating the neighbor is to break a persistent routing loop, so that the protocol can reach a stable state. Section 6.1 shows that persistent routing loops are associated with persistent state oscillations, while temporary routing loops are not: if all routing loops are temporary then the protocol terminates. As an alternative to deactivating a neighbor, a persistent routing loop can be eliminated by transforming the loop into a temporary one. A routing loop \((C, u, \alpha)\) is said to be persistent if \( T[uCu](\alpha) < \alpha \) and temporary if \( T[uCu](\alpha) \geq \alpha \). To transform a persistent routing loop into a temporary routing loop, the extender of circuit \( uCu \) must be altered to some other extender \( T'[uCu] \), which verifies the following condition for all possible costs \( \beta \).

\[
\forall \beta \ T'[uCu](\beta) \geq \beta \tag{6.3}
\]

This is accomplished by establishing cost \( \alpha \) as a bound for all possible costs learned from around the
circuit $uCu$, which corresponds to defining extender $T'[uCu]$ as follows:

$$\forall_{\beta} T'[uCu](\beta) = \begin{cases} 
T[uCu](\beta), & \text{if } T[uCu](\beta) \geq \alpha \\
\alpha, & \text{if } T[uCu](\beta) < \alpha \end{cases} \quad (6.4)$$

ensuring condition [6.3] is always true.

The version of SS-BGP with the cost bounding elimination method is called Bounded SS-BGP (BSS-BGP) and states that, if a route learned from a neighbor denounces a persistent routing loop, then future routes from that neighbor are bounded to the best cost of all candidate routes learned from any other neighbor.

The pseudo-code for BSS-BGP adds to Algorithm 2 variable $Bound[w]$, which stores the bound cost for neighbor $w$. When the test condition in Line 7 is positive, the cost of the elected route is assigned to $Bound[v]$, replacing Line 8 with $Bound[v] := \text{Cost(Route)}$. This action affects future routes learned from $v$ when Line 4 is replaced with,

if $T[uv](\beta) < Bound[v]$ then
    $\beta := Bound[v]$
else
    $\beta := T[uv](\alpha)$

forcing the cost $\beta$ learned from $v$ to always be worse or the same as the bound cost for $v$, $Bound[v]$.

Back to the example of Figure 5.4, when at $T = 4$, nodes $u$, $v$, and $w$ detect a persistent routing loop, they change their import policies such that all routes learned from their clockwise neighbors have cost $c$ rather than cost $r^*$, effectively treating their clockwise neighbors as if they were customers. As a result, at $T = 5$, each node $u$, $v$, and $w$ learns from its clockwise neighbor the customer routes $(c,vt)$, $(c,wt)$, and $(c,ut)$, respectively, but elects its direct customer route to $t$ because this route has a shorter path. Thus, the protocol terminates. Now, if link $ut$ fails, then $u$ elects a route to reach $t$ with cost $c$, learned from $v$.

6.5 Partial deployment

Until now, the examples presented here considered that all nodes deploy either BGP or SS-BGP. An important feature of SS-BGP is the fact that it uses standard BGP routing messages. This makes SS-BGP perfectly compatible with BGP, allowing it to be partially deployed.

![Diagram](image)

Figure 6.3: Protocol terminates with only $u$ deploying SS-BGP, but still oscillates if only $w$ deploys SS-BGP. Peer routes are exported to all neighbors and $t$ announces the destination.

Figure [6.3] presents an example where only a single node deploys SS-BGP, and this is sufficient to ensure termination. The network in Figure [6.3] considers routing with the GR routing policies with
Figure 6.4: BGP terminates, but if only \( u \) deploys SS-BGP, then the protocol oscillates. Peer\(^*\) routes are exported to all neighbors and \( t \) announces the destination.

peer\(^*\)'s. Assume that only \( u \) deploys SS-BGP, that is, only \( u \) is able to detect and eliminate persistent routing loops. At \( T = 0 \), node \( t \) announces a destination to its neighbors \( u, v, \) and \( w \). At \( T = 1 \),

- \( u \) and \( v \) elect a customer route learned from \( t \) and export this route, respectively, to \( w \) and \( u \);
- \( w \) elects a peer route \((r, t)\) learned from \( t \), but does not export this route to \( v \).

At \( T = 2 \), \( u \) elects route \((r^*, vt)\) learned from \( v \); and \( w \) elects route \((c, ut)\) learned from \( u \). They each export their new elected route counter-clockwise around the cycle. At \( T = 3 \),

- \( v \) learns route \((r^*, wuvt)\) from \( w \). This route identifies a persistent routing loop. However, \( v \) is not deploying SS-BGP. Thus, \( v \) just discards the route and elects route \((c, t)\), which exports to \( u \);
- \( u \) learns route \((r^*, vwuvt)\) from \( v \), which identifies a persistent routing loop. In contrast to \( v \), node \( u \) is deploying SS-BGP. Therefore, it elects route \((c, t)\) learned from \( t \) and deactivates \( v \).

Hence, the protocol stabilizes at \( T = 5 \) with \( u, v, \) and \( w \) electing, respectively, routes \((c, t), (r^*, wuvt), \) and \((c, ut)\) to reach the destination. Thus, with only \( u \) deploying SS-BGP, the protocol was able to terminate and would have terminated regardless of any delays in the routing messages.

The example of Figure 6.4 may lead us to conclude that it is enough to have a single node in each cycle deploy SS-BGP to ensure that the protocol terminates. However, this is not true. Going back to that example, one can see that both \( u \) and \( v \) detect a persistent routing loop at \( T = 4 \). Which means that, if either one of these nodes deploys SS-BGP, then protocol assuredly terminates. However, \( w \) does not detect any routing loop. Furthermore, \( w \) never detects a persistent routing loop, because all routing loops around circuit \( wuww \) are temporary: for all possible costs learned at \( w \) and dispatched around that cycle, the cost of the route learned back at \( w \) is always worse than or the same as the cost of the dispatched route, \( T[wuww][r^*] = c \not\succ r^*, T[wuww][c] = c, T[wuww][r] = \bullet \not\succ r, \) and \( T[wuww][p] = \bullet \not\succ p \).

The conclusion to draw from this is that not every node involved in a persistent routing loop is able to detect it, therefore, the nodes required to deploy SS-BGP must be selected with care.

In the of Figure 6.3, if only \( w \) deploys SS-BGP, then the protocol behaves exactly as standard BGP, never reaching a stable state. Apparently, partially deploying SS-BGP can only improve or maintain the stability properties of BGP. Figure 6.4 presents an example that shows otherwise. Node \( t \) announces the destination. With standard BGP, the protocol reaches a stable state with \( v \) electing route \((r^*, t)\) direct to \( t \), \( u \) electing route \((r^*, vt)\) learned from \( v \), and \( w \) electing route \((r^*, uvt)\) learned from \( u \). Assuming
that only \( u \) deploys SS-BGP, then \( v \) may learn a route for the destination announced by \( t \) through the sequence \( t \rightarrow y \rightarrow x \rightarrow u \rightarrow v \), before \( v \) learns the route incoming directly from \( t \). In that scenario, \( v \) exports a route \((r^*, \text{uwyt})\) to \( u \). As a result, \( u \) deactivates link \( vu \). With link \( vu \) deactivated, the routing the cycle \( uwu \) may prevent the protocol from reaching a stable state without \( u \) being able to detect a persistent routing loop. The protocol oscillates because both \( u \) and \( w \) prefer routes learned from around the cycle to routes learned externally to the cycle: the peer* routes \( w \) learns from \( u \) are better than the customer routes that \( w \) learns directly from \( t \); and the customer routes \( u \) learns from \( w \) have the same cost as the ones \( u \) learns from \( x \), however they are shorter.

Serves the example from Figure 6.4 to illustrate that it is possible to find a network and a destination for which BGP terminates, but SS-BGP deployed at one single, selected node does not terminate. However, BGP does not terminate for all destinations in this network. For example, if \( x \) is the node announcing the destination, then BGP oscillates, due to the existence of cycle \( uwu \) and the fact that \( u \) does not learn any candidate route from \( v \) to reach the destinations of \( x \). To clarify, when \( t \) announces the destination, \( u \) always elects a route via \( v \). Thus, the cycle \( uwu \) is never enabled, that is, the cycle does not prevent BGP from reaching a stable state. In contrast, when \( x \) announces the destination, \( u \) does not learn a candidate route from \( v \). Therefore, the protocol behaves similarly to when link \( vu \) is deactivated. This observation prompts the following conclusion. If BGP terminates in a network and in all its sub-networks (for all destinations), then a partially deployed SS-BGP also terminates in that network and all its sub-networks.

A closer inspection of the example in Figure 6.4 when \( t \) announces the destination, shows that, at some point, \( w \) learns a route \((r^*, \text{uwot})\) from \( u \) and elects route \((c, t)\) learned from \( t \). Therefore, if \( w \) deploys SS-BGP, then \( w \) detects and deactivates a persistent routing loop around \( uwu \), allowing the protocol to reach a stable state with \( v \) electing route \((c, t)\), \( u \) electing route \((c, wt)\), and \( w \) electing route \((c, t)\) to reach the destination announced by \( t \). This observation rises questions about which nodes involved in a persistent routing loop in a network are absolutely required to deploy SS-BGP to ensure termination in that network. Is it possible to provide a criteria for nodes applying isotone routing policies to determine if they must deploy SS-BGP?

A generic criteria for all isotone routing policies may not be possible. However, a specific criteria can be devised for peer* and another for peer*.

With peer*’s, it can be verified from the definition of persistent routing loop in Section 6.1 that a node belongs to such a loop if, an only if it is at the tail of a peer* link and that link belongs to a cycle all links of which are peer* links or customer links. The intuition for this conclusion follows from the two following observations. First, only a customer route exported through a peer* link gives rise to a route with a better cost at the corresponding neighbor, which justifies requiring the node to be at the tail of a peer* link. And second, only customer and peer* routes are exported through a peer* or customer link, which suggests all links of the cycle must be either customer or peer* links. In the network of Figure 6.4, both \( u \) and \( w \) belong to a persistent routing loop around the cycle \( uwu \). However, only \( w \) is at the tail of a peer* link in that cycle. Therefore, \( w \) is the only node able to detect and eliminate the persistent routing loop, which explains why the protocol did not terminate when only \( u \) was deploying SS-BGP.

However, a node cannot tell based on its local information whether or not it belongs to a persistent routing loop. But, it can tell from its local information whether or not it is at the tail of a peer* link (it is at the tail of a peer* link if it has a peer* neighbor), which condition is necessary to belong to a persistent routing loop. Therefore, if every node with at least one peer* link deploys SS-BGP, then the protocol terminates. From this conclusion, follows a simple guideline for AS administrators.

**Guideline 1** If an AS violates the GR routing policies by preferring some peers over customers, then that AS must deploy SS-BGP.
This previous guideline can be widened as follows.

**Guideline 2** If an AS violates the GR routing policies in any way, then that AS must deploy SS-BGP.

The latter does not apply to every isotone routing policy. However, it is valid not only with peers, but also for siblings as described below.

Regarding the GR routing policies with siblings, a node which does not violate the baseline GR routing policies is a node which is *not* at the tail of a sibling link. That being the case, that node can only learn and export routes with costs \((c, 0)\), \((r, 0)\), and \((p, 0)\). Thus, cost component which represents the number of sibling hops can be discarded. Which means, the only component of the cost that matters is the type of route \((c, r, \text{or} p)\). Since every extender in the siblings model (Section 4.1) either keeps or inflates this component, then every route dispatched by that node always arrives back at the node with a worse cost than or the same cost as the dispatched route. Therefore, only a node at the tail of a sibling link is required to deploy SS-BGP to ensure termination with these routing policies.
Chapter 7

Evaluation

The dynamic behavior of distributed routing protocols is too complex to evaluate analytically. Thus, it is common for research work to resort to simulators to evaluate such protocols with different network topologies. This same approach was taken here to evaluate the performance of SS-BGP. A new simulator was developed to fulfill the requirements of the experiments conducted in this work. It runs standard BGP, SS-BGP, and ISS-BGP and it is capable of running each protocol in large-scale networks using very few computing resources. Resorting to this simulator, extensive simulations were conducted of BGP, SS-BGP, and ISS-BGP in realistic topologies of the Internet with the perturbations to the baseline GR routing policies described in Section 2.2.

Section 7.1 describes the implementation of the simulator with generality. It highlights some of its most important features and explains what are its inputs and outputs. Then, Section 7.2 describes the input networks and the simulator settings that were used in each experiment. Section 7.3 compares the stability properties of BGP with those of SS-BGP and ISS-BGP. Section 7.4 presents the results of applying the partial deployment guidelines presented in Section 6.5 in the evaluated topologies. Finally, Section 7.5 presents some of the challenges faced during the realization of the simulations and presents the methods applied to overcome them.

7.1 Simulator

A new discrete-event simulator was developed to fulfill the requirements of the research conducted in this thesis work. The simulator supports routing with BGP, SS-BGP, and ISS-BGP.

The input of the simulator is a formatted file, including the network topology to simulate and a set of routing configurations applied by each node in the network. The network is represented by a graph. Nodes can represent any routing entity, such as a BGP router or an AS, and are identified by a unique integer number. Links represent the existence of an unidirectional connection between the two end nodes. Links point to the direction of data-flow. Thus, routes flow in the opposite direction to that pointed by the links. To model a bidirectional connection between two nodes (as the ones established between ASs in the Internet), the input network must include two links pointing in opposite directions.

When designing a scalable simulator of BGP like protocols, it is important to determine the appropriate level of detail to take into account regarding the protocol and the network. To be able to simulate such protocols in large scale network topologies (such as the current topology of the Internet), the simulator presented here abstracts or ignores some parts which are not relevant to the subject of study of this work and focuses on the aspects that truly affect the dynamic behavior of the protocol. In real networks, routing messages are carried over various wired media and network devices. These char-
acteristics of the network induce various communication delays to every routing message. To evaluate the dynamic behavior of these type of routing protocols, the exact factors inducing message delays are not of importance, but rather the timing in which each route is received and processed at each node. In the simulator, routes are subjected to a random delay drawn from an uniform distribution within a given interval. The maximum and minimum delay values are a configurable parameters of the simulator.

A simulation run simulates the announcement of a single destination and its propagation network-wide according to the routing configurations in effect. This is possible because BGP instantiates an independent routing process for each destination. Which means that the BGP routing process performed for one destination does not affect the outcome of a similar process for a different destination. Routing processes running in parallel on the same network may induce delays to the routing messages of each process: the exchange of routing messages in one process may induce delays on the routing messages of another process running at the same time. In the simulator, this type of delays is already included in the aggregated random delay assigned to each route. Therefore, the simulator can take full advantage of this characteristic of BGP and perform an individual computation for each destination.

This approach brings some advantages over other simulators which run a full-featured implementation of BGP [8][9]. Those simulators are required to keep routing state information for all destinations during a single simulation run. In BGP, for each destination, each node keeps a record of the last route learned from each neighbor. Storing all this information when simulating BGP in large scale networks with thousands of nodes requires a large amounts of memory storage. Performing the computation for each destination separately saves a large amount of memory, allowing simulations to be conducted in computer machines with very limited resources. In addition, this approach allows the use of more efficient data structures to store the state of the protocol, which reduces processing time considerably. Another benefit from this approach is the fact the each simulation run for each destination can be across separate machines, thereby, reducing the total simulation time. Section 7.5 discusses this matter further.

The simulator implementation is based on the routing model introduced in Section 4.1. Thus, each link is associated with an extender function which models the transformation of the cost of each route exported through the link. In the input file, each link includes a label, which identifies the extender associated with that link. Depending on the routing policies being applied, the set of extenders is different and their respective implementations vary. The current version of the simulator already includes the implementation of the extenders used to model the routing policies described in this work and it can be extended to accommodate other routing policies. In contrast to the definition presented in Section 4.1, the simulator considers a destination to be an existing node of the inputted network. However, it still supports the announcement of a destination in anycast. To have multiple nodes announcing a destination the simulator requires a second input file, containing the unique identifier of the destination and the list of existing nodes that announce the destination. The cost of the route announced by each anycast node can also be configured through this file.

The MRAI timer is an enhancement mechanism included in BGP to limit the rate with which routes are exported to neighboring nodes. When a node exports a route for a given destination to a neighbor, it starts an instance of this timer. From that point on, it is not allowed to export another route concerning this destination until the associated timer has expired after MRAI seconds. While waiting for an MRAI timer to expire, a node may learn various routes and perform the election process many times. When the timer expires, it exports the best route elected at that time. This allows a node to privately enumerate many alternative choices of its best route without exposing its neighbors to every intermediate change. The use of the MRAI timer reduces the number of routes exchanged during an execution of the protocol and, thereby, impacts the dynamic behavior of the protocol. An older version of the specification of BGP
makes a distinction between advertisements\(^1\) and withdrawals\(^2\) stating that the MRAI timer should not suppress the latter. However, the most recent version of the specification \[^2\] suggests that MRAI timers should limit the rate of both. The simulator follows the most recent version, applying the MRAI timer to both, advertisements and withdrawals. The specification also suggests that the MRAI timer should be applied on a per-neighbor basis. This implies having a timer at each node for each one of its neighbors. MRAI timers can limit exported on a per-neighbor basis or for all neighbors. In the simulator, the MRAI timer at each node limits exportations to all neighbors and its value can be set individually for each node.

There are two techniques proposed in the literature \[^{18,19}\] that may improve BGP’s convergence time. They are not included in the standard specification, but even so, they are worth mentioning here.

**Horizontal-split (HS):** nodes do not export a route learned from a neighbor to that same neighbor since the exported route would be discarded at the neighbor. To comply with BGP’s specification, the simulator does not implement HS.

**Sender side loop detection (SSLD):** before exporting a route to a neighbor, a node performs loop detection on behalf of the receiving neighbor. The SSLD method cannot be used with SS-BGP since the detection of routing loops at the receiver is crucial to its oscillation detection method: only the receiver can tell the cost of the looping route, therefore, only the receiver can detect a persistent routing loop.

In its current form, the simulator reports results to a file using the standard Comma Separated Value (CSV) format. For each simulation run, the following data is collected: (i) the simulation finish time, (ii) the average of the termination times of each node, (iii) the total number of routes exchanged during the simulation run, (iv) the number of nodes which detected at least one persistent routing loop, (v) the total number of detections, and (vi) within these, the number of false positive detections (see Section \[^6.3\]). The information corresponding to each recorded detection is also stored, such as, the detected loop, the detecting node, the costs of the looping and elected routes, and a flag indicating if the detection was a false positive or not.

A node is said to have terminated when its elected route stabilizes. The termination time of a node corresponds to instant at which that node exported its last route. The protocol terminates when every node’s elected route stabilizes, that is, when every node terminates. The simulation runs until the protocol reaches a stable state. Which means that if the protocol oscillates the simulation runs forever. To prevent this situation, the simulator includes a threshold parameter: if the simulation time reaches this threshold, the simulation stops immediately and it is labeled as non-terminated. The simulation finish time only matches the protocol termination time if the protocol terminates, otherwise, the finish time corresponds to the threshold value.

As stated in Section \[^5.3\], a false positive occurs when a node incorrectly signals a routing loop as persistent when in fact the loop is only temporary. The simulator checks each detection to verify whether or not it constitutes a false positive. It does this by emulating the propagation of the route elected by the detecting node at the moment of detection, and verifying whether or not the cost learned back at that node is better than the cost of the elected route. If this is true, then the detection is labeled as a false positive and the false positive count is incremented.

### 7.2 Input data and parameters

As previously mentioned, the simulator associates each route with a random delay value to emulate the time a it takes for a route to be imported by an AS, processed, and then arrived at the neighboring AS.

---

\(^1\)An advertisement refers to routing messages which indicate availability of a route to the destination.

\(^2\)An withdrawal refers to routing messages which indicate unavailability of a route to the destination.
For the experiments conducted in this work the route delay values were limited to the interval between 0.01 and 1.0 seconds. No known documentation is available regarding the times for these delays. But, the previous values were considered plausible in [19].

In general, the optimal value for the MRAI depends on a large number of factors and is thereby very hard to estimate. The network topology and the routing configurations applied at each node are among the most significant factors with a great impact on the choice for the MRAI value. In [19], it was concluded that optimal MRAI value depended so much on a specific network topology, that it could not be used as a general mechanism to reduce termination time. In addition, [20] revealed that applying diverse MRAI values across the network may bring significant increases in both termination time and the number of routing messages exchanged during an execution of the BGP. The specification of BGP [2] suggests an interval of 30 seconds for the MRAI. The specification of BGP [2] suggests an interval of 30 seconds for the MRAI, this value has been considered far higher than the optimal by many authors [19] [18] [21] [22]. In [21] the authors suggest that the optimal value may be between 5 and 15 seconds. Further empirical work from the same authors suggests that the optimal MRAI value for the Internet may even be below 5 seconds [22]. More recent research work [23] proposes a value between 3 and 5 seconds for the MRAI. In the experiments conducted in this work, the MRAI value is set to 5 seconds for all nodes.

The input networks of the simulator are based on AS-level topologies of the Internet inferred by CAIDA using the method described in [24] and published at [25]. These topologies are updated monthly. The results reported in this work refer to the topology from July 1st of 2016.

The topology from CAIDA lists pairs of ASs labeled as either a customer-provider or a peer-peer relationship following the GR routing policies. Before using this topology some operations were performed to fix some inaccuracies found in the date [26]. The method described in [27] was used to eliminate all customer-provider cycles and to ensure the topology is policy connected, that is, there is a valid path from every AS to every other. The topology started with 54,733 ASs and 234,907, after applying the fixes the resulting topology included 53,947 ASs (98% of them) and 226,087 relationships (96% of them).

A directed graph (network) was generated from the final topology as follows, each AS was represented by a node, and each relationship was modeled by a pair of links labeled according to its type: a customer-provider relationship was modeled by a customer link from the provider AS to the customer AS and provider link in opposite direction; and a peer-peer relationship was modeled with two peer links in opposite directions. Hence, the resulting graph in twice as many links as there was relationships.

In the previous AS network most ASs (86% of them) are stubs (see Section 2.1). To save on simulation time, stub ASs were removed from the initial network, reducing the number of nodes to 8,033 (14% of them) and the number of links to 177,890 (39% of them), 21,854 of them were customer links, the same number of them were provider links, and 134,182 of them were peer links.

Assuming, as it has been done here, that ties among customer (peer, provider) routes are broken by the lengths of the paths carried in those routes, BGP terminates under the GR routing policies. Section 2.2 introduced peer’s and siblings as possible alternatives to these policies, which make BGP prone to oscillations. The GR routing policies applied in the original network were perturbed to accommodate peer* and siblings. Each perturbation was made separately, that is, peer’s and siblings were included in different network configurations.

Peer’s were added to the initial topology (without stubs) by replacing some peer links with peer* links (see Section 4.1). The peer* links were chosen randomly from the set of all peer links according to a uniform distribution. Results are presented for five percentages of peer* links: 0.25%, 0.5%, 1%, 5%, and 10% (relative to the total number of peer links). Five samples with different sets of peer* links were generated for each percentage of peer* links.

To accommodate for siblings in the initial network, some links were replaced by sibling links (see Section 4.1). Siblings are usually ASs belonging to the same organization. Thus, any link between ASs
that belong to the same organization was replaced by a sibling link. The data on CAIDA’s Inferred AS to Organizations Mapping Dataset [25] was used to identify the organizations to which each AS belongs. The final network configuration included 4,038 sibling links corresponding to 3% of all peer links.

7.3 Stability results

This section reports simulation results for the topologies described in the previous section. The basic experiment emulates the announcement of a new destination, corresponding to a simulation run, in which an AS announces a route to a destination that is propagated network-wide according to the rules of the routing protocol and the routing policies in effect. For any given destination AS, 100 simulation runs were executed with different seeds for generating random delays across links. The termination time of the routing protocol for a single destination is the average of termination times over all 100 simulation runs.

As seen before, BGP always terminates with the baseline GR routing policies. The results obtained with these policies were used here as a benchmark. The termination time of BGP with these policies was used to determine the threshold parameter (see Section 7.1): the threshold value was set to a time 300 times greater than the termination time of BGP with the GR routing policies. For a given destination, if at least one simulation run (out of 100) reaches the threshold, then the destination is declared as non-terminated. The same collection of 200 destinations was used for all destinations. These destination were randomly selected from the all collection of ASs (including stubs) with an uniform distribution.

Based on this simulation setup, the goal here is to assess the impact of peer’s and siblings on the stability of BGP, and evaluate the efficiency of SS-BGP and ISS-BGP, that is, how many changes to the AS’s import policies have to be made for the protocol to terminate and how does it perform against BGP when either protocol terminates.

Non-Termination of BGP

BGP may fail to terminate in the presence of peer’s and siblings. However, the mere presence of peer’s and siblings does not automatically imply the formation of persistent routing loops. Moreover, Section 6.1 concludes that the existence of persistent routing loops does not immediately imply non-termination of BGP for all destinations. It only implies non-termination for at least one destination, which may even need to be announced by multiple nodes for the state of BGP to oscillate. Table 7.1 shows the percentages of destinations (among 200 destinations) for which BGP did not terminate subject to the GR routing policies with peer’s and the GR routing policies with siblings. Each percentage of peer’s represents an average over five samples. The following conclusions can be drawn.

Peer’s are much likelier to induce oscillations of the state of BGP than siblings. The percentage of destinations for which BGP does not terminate is 16.0% at 0.25% of peer’s links and reaches 61.8% at 1% of peer’s links, whereas the percentage of destinations for which BGP does not terminate is only 2% for 3% of sibling links.

For higher percentages of peer’s, the stability of BGP increases with the percentage of peer’s links. Although, the percentage of destinations for which BGP does not terminate reaches 61.8% at 1% of peer’s links, this percentage decreases to 35.6% at 10% of peer’s links. Persistent routing loops come in various lengths, measured in terms of the number of links they contain. Considering each persistent routing loop in isolation, the likelihood of reaching a stable state in a shorter persistent routing loop is higher than in a longer one. Furthermore, the stabilization of a short persistent routing loop containing nodes that are simultaneously involved in longer persistent routing loops may also help stabilize those
Table 7.1: Percentage of destinations for which BGP does not terminate, for the GR routing policies with siblings and with peer+’s.

<table>
<thead>
<tr>
<th>Policies</th>
<th>Non-terminated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siblings (3%)</td>
<td>2.0%</td>
</tr>
<tr>
<td>Peer+ (0.25%)</td>
<td>16.0%</td>
</tr>
<tr>
<td>Peer+ (0.5%)</td>
<td>25.3%</td>
</tr>
<tr>
<td>Peer+ (1%)</td>
<td>61.8%</td>
</tr>
<tr>
<td>Peer+ (5%)</td>
<td>45.8%</td>
</tr>
<tr>
<td>Peer+ (10%)</td>
<td>35.6%</td>
</tr>
</tbody>
</table>

Table 7.2: The first two columns show the number of detections, respectively, with SS-BGP and ISS-BGP for the GR routing policies with siblings and with peer+’s. The last column shows the number of false positive detections with SS-BGP.

<table>
<thead>
<tr>
<th>Policies</th>
<th>Detections</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SS-BGP</td>
<td>ISS-BGP</td>
</tr>
<tr>
<td>Siblings (3%)</td>
<td>1.2</td>
<td>1.0</td>
</tr>
<tr>
<td>Peer+ (0.25%)</td>
<td>1.1</td>
<td>1.0</td>
</tr>
<tr>
<td>Peer+ (0.5%)</td>
<td>2.7</td>
<td>2.4</td>
</tr>
<tr>
<td>Peer+ (1%)</td>
<td>5.7</td>
<td>4.3</td>
</tr>
<tr>
<td>Peer+ (5%)</td>
<td>23.8</td>
<td>10.3</td>
</tr>
<tr>
<td>Peer+ (10%)</td>
<td>64.4</td>
<td>17.8</td>
</tr>
</tbody>
</table>

longer loops. As the percentage of peer+ links increases so does the number of short persistent routing loops. Therefore, increasing the percentage of peer+ links also increases the stability of BGP.

This also explains why the stability of BGP decreases abruptly with only a small percentage of peer+’s. Notice that, approximately, one quarter of the destinations did not terminate at only 0.5% of peer+ links. With a lower percentage of peer+ links, it is likelier persistent routing loops are longer. Therefore, the protocol becomes very unstable with the introduction of only a few peer+ links.

Detections

Table 7.2 under the heading ‘Detections’, presents the number of detections of persistent routing loops that it takes SS-BGP and ISS-BGP to terminate. This values represent an average over the 200 destinations. The interesting conclusion to draw from these results is that both protocols require very few detections to ensure stabilization. For instance, at 10% of peer+’s, SS-BGP terminates after 64.4 detections, corresponding to 0.038% of all links. A detection corresponds to a deactivation of a neighbor AS. The action of deactivating a neighbor by a given AS corresponds to an alteration of the import policies applied at that AS. Thus, results show that SS-BGP and ISS-BGP stabilize BGP at the expense of affecting the import policies of only a few ASs.

Under the heading ‘False Positives’, Table 7.2 presents the percentage of detections that correspond to false positives with SS-BGP. The results show that the difference in detections between SS-BGP and ISS-BGP is justified by the number of false positives, which occur in the former method and are absent in the latter. Furthermore, the percentage of false positives increases with the number of peer+’s. For instance, at 0.25% of peer+ links 0.05% of detections are false positives and this value rises to 75.6% at 10% of peer+ links.
Table 7.3: First two columns show that termination times of BGP, for when it terminates, and of SS-BGP, for the GR routing policies with siblings and with peer+’s. The last column shows the reduction in the termination times from BGP to SS-BGP.

**Termination times**

Table 7.3 compares the termination time of BGP, for those destinations for which the protocol terminates, to the termination time of SS-BGP (which terminated for all destinations). The termination time of ISS-BGP is very similar to that of SS-BGP and, for that reason, it is not presented here. The times presented under the heading ‘Termination’ correspond to an average over the termination times of the considered destinations. The results show that even for destinations where BGP terminates, SS-BGP terminates faster. For instance, at 5% of peer+’s, BGP takes 37.0 seconds to terminate while SS-BGP takes 25.6 seconds to terminate, which corresponds to a 30.8% reduction.

### 7.4 Partial deployment

Section 5.3 illustrated that SS-BGP terminates for all possible destinations in a given network without requiring every node to deploy SS-BGP. Later, Section 6.5 discussed SS-BGP in partial deployment, concluding that the termination of SS-BGP is ensured if all nodes belonging to a persistent routing loop deploy SS-BGP. However, the criteria used to determine which nodes belong to a persistent routing loop in a given network is dependent on the applied routing policies. In the case of the GR routing policies with peer+’s, it was established that a node belongs to a persistent routing loop if, and only if it is at the tail of a peer+ link and that link belongs to a cycle all links of which are peer links or customer links. This Section investigates the percentage of ASs verifying this condition.

Table 7.4 under the heading ‘Loop’ presents an average, over five samples, of the percentage of ASs that belong to a persistent routing loop in the network topologies described in Section 7.2 for each percentage of peer+ links (stubs are excluded). The results show that if 7.8% of selected ASs deploy SS-BGP, then termination is guaranteed at 1% of peer+ links. As expected, the percentage of ASs required to deploy SS-BGP increases with the percentage of peer+ links, reaching 28.0% at 10% of peer+ links.

The algorithm used to determine the set of nodes in a given network that belong to a persistent routing loop is described here with generality. The first step (1) constitutes eliminating all provider and peer links from the network, leaving only customer and peer+ link. Then, follows (2) a search for all nodes at the tail of at least one peer+ link, this can be accomplished by simply iterating over all nodes and checking if they have a peer+ out-link. The next step (3) involves performing a global depth-first search to determine all strongly connected components [28]. The collection of nodes included at least one strongly connected component, with more than a single node, constitutes the set of nodes belonging to at least one loop. Since, at this point, only customer and peer+ links exist, then these nodes are surely...
Table 7.4: First three entries show the percentages of ASs that belong to a persistent routing loop and that are at the tail of a peer+ link, for the GR routing policies with peer’s. The last entry shows the percentage of ASs at the tail of a sibling link for the GR routing policies with siblings.

An AS cannot tell from its local configuration whether or not it belongs to a persistent routing loop. Therefore, at Section 6.5 it was presented a local guideline for AS administrators to determine whether or not they must deploy SS-BGP: if an AS administrator violates the GR routing policies by preferring some peers over some customers, then it must deploy SS-BGP. In other words, every AS at the tail of a peer+ link is required to use SS-BGP. A similar guideline was also presented for siblings: if an AS has at least one sibling AS, then that AS must deploy SS-BGP.

The first three entries of Table 7.4, under the heading 'Tail', present the percentage of ASs at the tail of at least one peer+ link. Comparing the with the ones corresponding to the nodes belonging to a persistent routing loop, shows that this local condition provides a very close approximation to the actual number of nodes required to use SS-BGP, with the biggest difference being at 1% of peer+ links, where this condition shows a 0.7% more nodes are required to deploy SS-BGP. The last entry of Table 7.4 under the heading 'Tail', shows the percentage of ASs at the tail of at least one sibling link for the siblings network, showing that the deployment of SS-BGP on selected 30.8% ASs is sufficient to ensure termination.

### 7.5 Simulation environment

Simulating BGP, SS-BGP, or ISS-BGP in large-scale topologies of the Internet is a processing intensive activity. A considerable number of simulation runs need to be executed to obtain representative results and each of the of these runs requires some processing time. The research work carried in this thesis required millions of simulation runs to be executed. Due to time constraints, these needed to be performed as fast as possible.

To speedup results, the various simulations runs were distributed across ten separate machines, each executing a single simulation at a time. This approach allowed results to be produced much faster. However, it brought some challenges regarding the management and distribution of simulations across the various available machines.

To make the simulation process as efficient as possible, an automatic system was developed to manage and distribute the necessary simulations across the machines that were available. The system is composed of three different components: a single simulation queue, a single distributor, and multiple simulators. The simulation queue is a central place containing all simulations to be executed. The distributor takes simulations from the simulation and distributes them across the multiple simulators that are available. A simulator refers to a computer machine able to execute simulations.

The way the system works is very simple. Once a simulator is initialized, it informs the distributor that it is available to execute simulations. Upon the reception of this information, the distributor gets the next
simulation to be executed from the simulation queue and assigns it to the available simulator. When the simulator finishes executing, it informs the distributor that simulation is finished and the simulator assigns it a new simulation taken from the simulation queue.

This approach allows the process to be as efficient as possible because it distributes the work evenly across all available machines. The fact that all simulations are placed in a single and centralized queue allows for a much easier management of the simulations. Each simulation, before being added to the queue, is assigned with a priority value. The queue is ordered according to this value, allowing simulations to be organized in the queue according to their priority value. Simulations with an higher priority are put in the front of the queue and, thereby, they are executed first.
Chapter 8

Conclusions

8.1 Achievements and results

In this thesis work an embryonic idea was developed and refined, producing a complete solution in the form of an addon to BGP called SS-BGP, which detects and eliminates permanent state oscillations associated with routing loops. It was found that there are two types of routing loops: temporary routing loops and persistent routing loops. The existence of persistent routing loops prevents BGP from reaching a stable state, which justifies chasing and eliminating such loops. SS-BGP eliminates persistent routing loops, thereby, improving the stability properties of BGP.

Isotonicity is a property of routing policies already present in baseline interdomain routing policies and it can be enforced in a distributive way while retaining privacy, which is so deeply valued by AS administrators. For the important class of isotone routing policies all permanent state oscillations are associated with persistent routing loops, which leads to the conclusion that SS-BGP terminates for these policies. When routing policies are not isotone, SS-BGP still terminates in cases where BGP does not.

In contrast, to previous proposals to stabilize BGP, SS-BGP performs a distributive detection of persistent routing loops based strictly on the routing information already available at each node. It does not require any node to keep any record of previously seen or elected routes, maintaining all of BGP’s scalability properties. In addition, SS-BGP uses standard BGP routing messages, which makes it fully compatible with BGP, allowing for it to be deployed incrementally.

A new discrete event simulator was developed as part of the thesis work to provide information about the performance of BGP and SS-BGP in realistic topologies of the Internet. The simulator focuses on the fundamental aspects of BGP and SS-BGP, abstracting the particular aspects of their implementation that do not affect their dynamic behavior. This approach allowed simulations to be conducted in large-scale networks, such as the Internet, without the obtained results loosing any information or correctness.

Extensive simulations were conducted of BGP and SS-BGP in realistic topologies of the Internet with realistic perturbations to the baseline GR routing policies. The results show that BGP did not terminate for a considerable number of the simulated destinations. In contrast, SS-BGP terminated in all cases with only a small percentage of nodes having to change their import routing policies.

The simulations runs were spread across multiple machines to speedup results. Managing the large collection of simulations and the volume of data results obtained from them became a major challenge. An integral part of the thesis work was the development of a tool to distribute the various simulations across each machine available and to manage the high volumes of data produced. This tool may be important contribution to other research works which face similar challenges.

It was also found that only a fraction of nodes are required to deploy SS-BGP for its stabilizing effects.
to be felt network-wide. However, the set of nodes required to deploy SS-BGP must be handpicked. For each alternative routing policy (see Section 2.2), a guideline was formulated for AS administrators to determine whether or not they should deploy SS-BGP based strictly on their local routing configurations. For the topologies evaluated in this work, results shows that less than a third of the nodes are required to deploy SS-BGP.

8.2 Future work

Isotonicity allows for a wide range of routing policies. The thesis work concluded that SS-BGP terminates for isotone routing policies and evaluated its performance in these policies. However, there might a property which allows for an even broader class of routing polices in which SS-BGP terminates. Furthermore, there is still some ignorance regarding the effectiveness of SS-BGP for non-isotone routing policies. An example presented here, illustrated that there are scenarios in which SS-BGP terminates and BGP does not. Questions remain as to which scenarios does this occur. Not only that, but also, if it there is any scenario where a stable state is reached with BGP and not with SS-BGP.

The elimination method applied in SS-BGP, the way it is presented here, considers the deactivating a neighbor on a per destination basis. This implies that a node must keep record of the deactivated neighbors for each destination. An alternative solution would be for a neighbor to be deactivated for all destinations, that is, if a node detects a persistent routing loop for one destination, then deactivates the neighbor for all future destinations. With this solution, each node would only need to keep a single list of deactivated neighbors, instead of one list per destination, which requires less memory storage. At first glance, another possible advantage of this solution is that the elimination of a persistent routing loop for one destination may reduce the termination time for other destinations. However, this was not investigated and there might be some drawbacks associated with the deactivation of neighbors for all destinations that were not identified. The advantages presented here justify further investigation over the subject.

This thesis already presents better detection and elimination methods than the ones applied in the canonical version of SS-BGP. Future work should investigate this matter further and try to explore different detection and elimination methods that may bring some advantages for generic or particular applications.

The topology of the Internet changes frequently. As a consequence, a persistent routing loop, that was detected and eliminated earlier, may disappear after some ASs change their routing configurations or if some connections are turned off. No matter the reason, if the persistent routing loop no longer exists, then each neighbor, that was deactivated to eliminate the loop, can be reactivated, allowing the node which performed the deactivation to elect routes learned from that neighbor. This situation becomes more relevant when considering the global evolution of protocol's through time. Deactivated neighbors decrease the connectivity of the network and should only be deactivated if they are absolutely necessary to ensure that a stable state is reached. Future work should investigate possible solutions to reactivate neighbors when their deactivation is no longer required to ensure termination.

The dissertation describes two alternatives to the baseline GR routing policies. For each of these routing policies it presents a guideline for AS administrators to determine whether or not they should deploy SS-BGP. The guidelines presented for these routing policies probably do not apply in general. Or do they? At this time the answer is not clear. Further investigation may show that it a generic guideline can be formulated and/or provide other guidelines for other routing policies that deviate from the ones evaluated in this work.
Bibliography


