Characterizing the Performance and Stability Issues of the AS Path Prepending Method: Taxonomy, Measurement Study and Analysis*

Hui Wang Rocky K.C. Chang Dah Ming Chiu John C.S. Lui

Abstract

We consider the autonomous system (AS) path prepending (ASPP) approach to perform load-balancing on inbound traffic for multihomed ASes. Although the ASPP approach has been practiced by many AS operators for a long time, it is surprising that there still lacks a systematic study of the approach and understanding of its effects and performance implications. The purpose of this paper is to provide such an analysis, which serves as a basis to further improve the approach and to understand its limitations. There are two parts to this study. The first one is based on measurement and analysis of the Routeviews data, which provides the fact of the "prevalence" of using ASPP approach in the current Internet and its possible impact on Internet routes. In the second part we propose a model of how ASes perform traffic engineering and we also present some fundamental issues of decentralized traffic engineering in the Internet. In our model, each AS performs inbound load balancing so as to optimize its traffic engineering goals. Some important questions that we address are how these local actions affect the global network performance and whether these ASPP actions interfere with each other and induce instability. We provide some examples to illustrate the performance and stability issues and suggest some guidelines that will help to avoid route oscillations.

1 Introduction

One can view the global Internet as an interconnection of ASes. In general, there are two types of AS, namely, transist AS and stub AS. Transit AS provides Internet connectivity to other ASes by forwarding all types of traffic across its network. Stub AS, on the other hand, does not provide transit service for other ASes and only sends or receives its own traffic. The interconnection of ASes can also be described by *business relationship*. Major business relationships include the *provider-to-customer* relationship and the *peer-to-peer* relationship. These business relationships play a crucial role in shaping the structure of the Internet and the end-to-end performance characteristics [1]. From the point view of AS relationship, stub ASes are the ASes which have no customer (or client AS), while transit ASes are ASes which have customers. Transit ASes without provider are called "tier-1" ASes. ASes that have more than one provider are called *multihomed* ASes. Except the tier-1 ASes, all transit ASes are always multihomed so as to provide better connectivity and performance.

Motivated by the need to improve network resilience and performance, there is an increasing number of enterprise and campus networks connecting to the Internet via multiple providers. These multihomed Autonomous Systems (ASes), therefore, must undertake the task of engineering the traffic flowing in and out of the network through these multiple links. Using different inter-AS traffic engineering approaches, ASes can distribute traffic so as to satisfy their performance or cost constraints. The focus of this paper is on the *inter-AS inbound traffic engineering* problem, which is more difficult than the outbound traffic engineering problem because an AS generally cannot control the routing path for the inbound traffic. Moreover, we restrict our attention to the ASPP approach based on the Border Gateway Protocol (BGP), which is the de-facto standard for inter-AS routing in the Internet.

There are three popular BGP-based approaches for performing inbound inter-AS traffic engineering: selective advertisements (SA), specific prefix advertisement (SPA), and AS path prepending (ASPP) [2]. Unlike the SA and the SPA approach, the ASPP does not introduce longer prefixes, and at the same time takes the advantage of resilience protection from multihomed connections. Although ASPP has been practised in the Internet for a long time, there has been no systematic study on the phenomenon and performance of this method. Also, based on BGP routing tables from routers connected to the AT&T backbone, it is reported that over 30% of the routes

*Hui Wang and Dah Ming Chiu are with the Department of Information Engineering, The Chinese University of Hong Kong. Email:{hwang3,dmchiu}@ie.cuhk.edu.hk. Rocky K. C. Chang is with the Department of Computing, The Hong Kong Polytechnic University. E-mail: csrchang@comp.polyu.edu.hk. John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. E-mail: cslui@cse.cuhk.edu.hk.

has some amount of ASPP and this indicates that ASPP has a significant impact on the current Internet routing structure[3].

This paper attempts to fill these gaps. We motivate our research through our measurement findings and then point out that ASPP policies of different ASes may affect each other. We also show ASPP does not always improve the Internet's global performance according to our performance metrics. The contribution of this paper can be summarized as follows:

- We present original findings about ASPP from the measurement of Routeview data.
- We define various local and global performance metrics so as to study the influence of ASPP.
- We propose a formal model to study the performance and implication of ASPP.
- We pinpoint the potential route oscillation problem if the ASPP policies of different ASes affect each other.
- We present general guidelines in using ASPP so as to avoid route oscillation.

The outline of the paper is as follows. In Section 2, we describe the AS path prepending approach and present results from a measurement study. In Section 3, we present our network model, performance measures as well as various complex interactions when ASes use the prepending approach to perform their local optimization. Guidelines are presented in Section 4 so as to avoid instability and route oscillation. Related work is given in Section 5 and Section 6 concludes.

2 Observations of ASPP in the Internet

ASPP is an important BGP-based inbound traffic engineering method. Under the BGP route advertisement, a route has an AS path whose format is (AS_i, AS_j, ..., AS_k, AS_n). The semantics of the above route advertisement is that there is a reachable path from AS_i to AS_n via AS_j, ..., AS_k. The length of the AS path is denoted by $|(AS_i, AS_j, \ldots, AS_k, AS_n)|$. An AS can *inflate* the length of an AS path by performing ASPP, i.e., AS_i can insert its AS number in this route again so it becomes (AS_i, AS_i, AS_j, ..., AS_k, AS_n). When an AS receives this route advertisement from its neighbor AS_i, this AS will think that the length of this route is longer by one hop, as compare to the original route advertisement, to reach AS_n via AS_i. So this AS may want to select another route with a shorter AS path so as to forward traffic to AS_n.

2.1 The growth on the use of ASPP

To motivate our study on ASPP, we first report in Fig. 1 the trend on the numbers of all ASes, stub ASes, multihomed stub ASes, and transit ASes. First, we note that there are several "dips" in this figure and others to be presented later. These dips were caused by a significant loss of routes due to an Internet-wide worm attack, e.g., the SQL Slammer Worm attack started on 25 Jan, 2003. Therefore, these valleys should only be considered as exception cases.

The data shows that in recent years the number of stub ASes comprises almost 85% of the total number of ASes. Out of these stub ASes, the share of the multihomed stub ASes increases from 40% in 1997 to 60% in 2004. Thus, 50% of the ASes today are multihomed stub ASes which are the prime candidates of using ASPP to control the inbound traffic coming into their links.

As shown in Fig. 2, the actual number of the multihomed stub ASes that use ASPP for inbound traffic control is around 33% in the most recent measurement. This percentage exhibits an increasing trend in the last few years, which supports the belief that the ASPP is a very popular inbound traffic control method. On the other hand, the number of transit ASes that use ASPP is also on the rise in recent years. The share of such transit ASes from the total number of transit ASes has been increased from 22% in 1997 to 40% this year. Combining both transit and multihomed stub ASes, there are almost 25% of ASes today that are using ASPP. If we discount those singly-homed stub ASes, then there are more than one third of the ASes that have multiple links for receiving traffic are using the ASPP method.

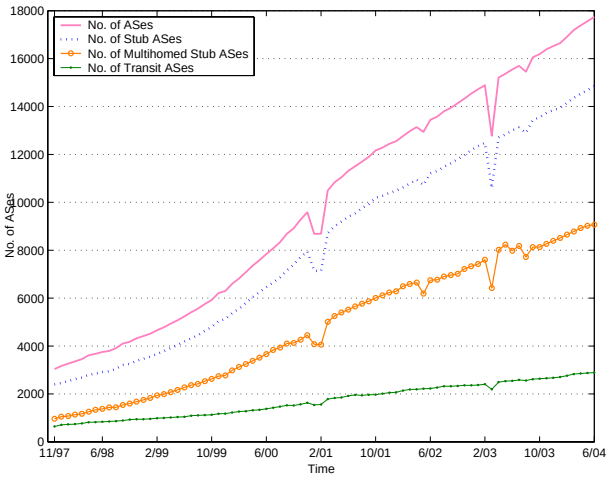

Figure 1: The trend on the numbers of stub ASes, multihomed stub ASes, and transit ASes.

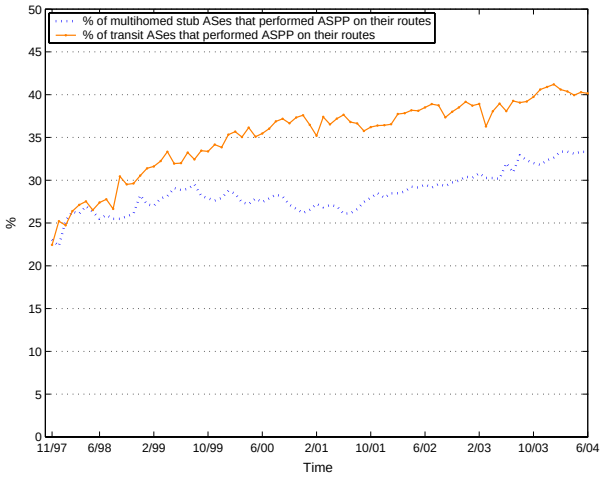

Figure 2: The percentages of multihomed stub ASes and transit ASes that perform ASPP on their routes.

2.2 Distribution of prepending number and length

Another metric of characterizing the growth of ASPP usage is based on the route statistics. We have shown in Fig. 3 that the number of *prepended routes*, i.e., the routes that have AS prependings, is also on the rise. Although the total number of routes has been increasing in the Internet, the percentage of prepended routes has been increasing steadily. In 1997, the share of prepended routes was only 7% prepended routes but it has been increased to more than 12% today.

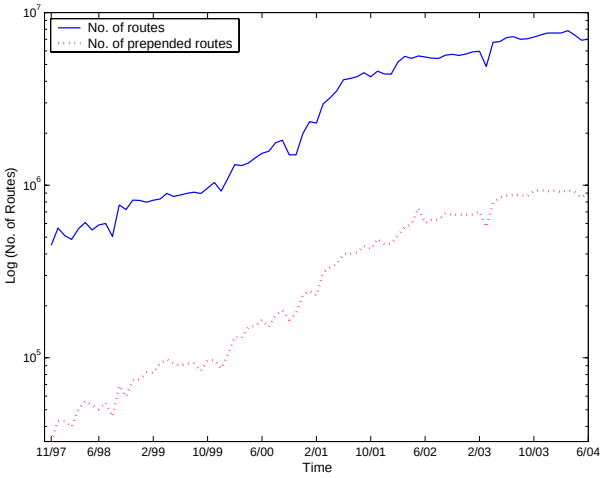

Figure 3: The trend on the number of prepended routes.

It is useful to further examine several characteristics about these prepended routes. For this purpose, we have classified the prepending into two types: *source prepending* and *intermediary prepending*. Source prepending is referred to those that are performed by the origin ASes, while the intermediary prepending are performed by non-origin ASes. For example, in the AS path $(1, 2, 2, 3, 4, 4, 4)$, AS_4's prepending is source prepending and AS_2's prepending is intermediary prepending. As noted from this example, a route could have both types of prepending or more than one intermediary prepending.

In Fig. 4, we show that more than 60% of the prepended routes have source prepending. Moreover, more than 40% of the prepended routes have intermediary prepending. Note that these two percentages do not add up to 100%, because there are routes that have both, which take up around 4% of the prepended routes. Clearly, if we pick a prepended route randomly, most likely we will see source prepending in the route. However, we caution that the actual percentage for the third case (have both types of prepending) could be much higher than those reported here, because some of these routes could have been filtered before reaching the Route View router due to a longer AS path length in these routes.

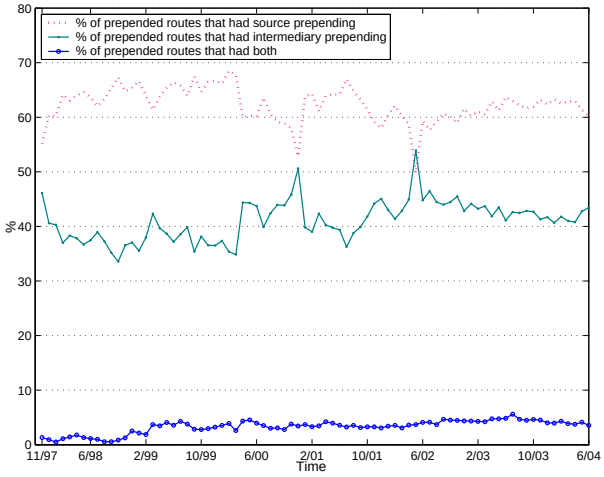

Figure 4: The percentages of source prepending, intermediary prepending, and mixed prepending in the prepended routes.

Another important statistics about the prepended routes is the distribution of the number of prependings in a route, and the distribution of the prepending length, which are shown in Fig. 5 and Fig. 6, respectively. The former counts the number of prepending in a route. In the example of $(1, 2, 2, 3, 4, 4, 4)$, there are 2 prependings in this route. The latter, on the other hand, measures the length of a prepending. In the same example, the AS2's prepending has a length of 1, while the AS4's prepending has a length of 2.

Fig. 5 shows that around 95% of the prepended routes have only one prepending and around 4% of the prepended routes have 2 prependings. Once again, many routes that have a higher number of prependings were most likely not selected by the transit ASes and therefore

filtered out. Unless we analyze the data from more vantage points, we cannot accurately estimate the number of such prepended routes.

As for the distribution on the prepending length, Fig. 6 shows some interesting trends. The prepending length of 1 is still most common. Moreover, after a careful examination on the trend for this case, we can detect a drastic shift in the percentages. Between 1998 and late 2000, the % of length equal to 1 is more or less ranged between 50% and 65%. However, this range was dropped to 40% and 50% in the period between early 2001 and today. While the exact cause for this drop is still a puzzle to us, it seems that the prepending length of 1 is not sufficient for many ASes to influence the incoming traffic. The case of length equal to 2 is also observed to drop steadily in the recent months, i.e., from 30% in late 2002 to 23% in June 2004. On the other side, the share of longer prepending length is getting higher. More noticeable cases are lengths longer than 3.

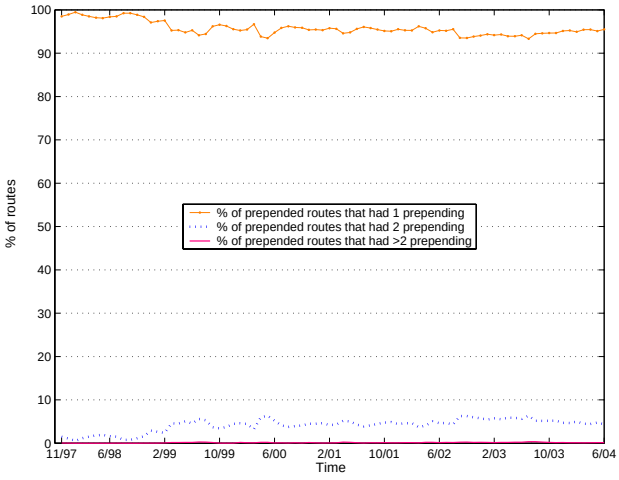

Figure 5: Distribution of the number of prependings.

2.3 Prepending policies

Another important information found from the Route View data is the prepending policy that an AS exercises. Here we classify the policies into two broad classes: *link-based prepending* and *prefix-based prepending*. An AS is said to employ a link-based prepending policy if the prepending length (including 0 for non-prepended routes) is the same for all routes announced to a specific link. However, the prepending length may be different across the links, e.g., one link without prepending and the other with a prepending length of 3. Otherwise, the AS is said to employ a prefix-based prepending policy.

Fig. 7 shows the results for the percentage of ASes that use the link-based prepending policy. All three cases in the figure show a downward trend on the percentages of

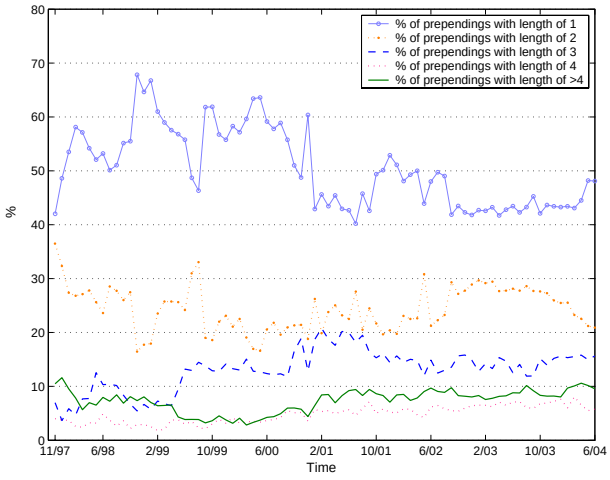

Figure 6: Distribution of the prepending lengths.

the ASes concerned that use link-based prepending. Thus, it is apparent that the prepending policies are becoming more complex than a "lazy prepending" approach. Fig. 8 shows the results for the number of links. That is, we count the links that a link-based prepending policy is applied to. Similar to the previous figure, the percentages for all three cases show a downward trend.

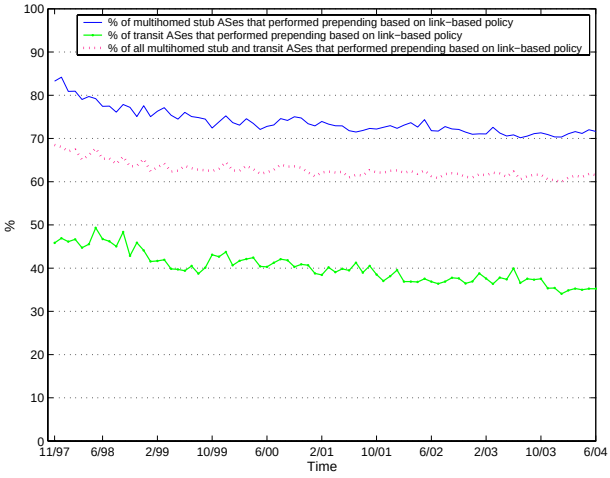

Figure 7: The percentages of ASes that employ the link-based prepending policy.

3 Analysis of AS Load Balancing Using ASPP

AS path prepending is basically an approach to selectively adjust the cost of links between ASes (for paths towards selected destinations) so as to influence the amount of traffic passing through these links. AS path prepending has both local and global effects. Locally, a multi-homed

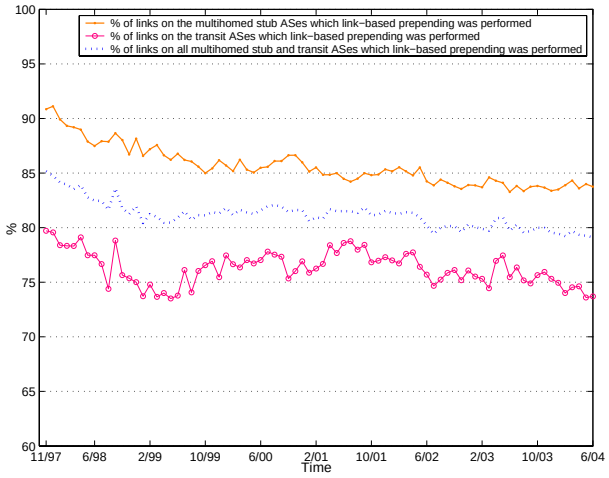

Figure 8: The percentages of links that are involved in the link-based prepending policy.

AS can better balance the traffic on incoming links from different providers. Globally, this may increase the total amount of inter-AS resource consumption in the network since traffic no longer follow shortest AS paths. On the other hand, a more interesting global effect is the degree that traffic is shifted from congested links to under-utilized links on a network-wide basis. In other words, local traffic load balancing may globally lead to a network that appears to have higher capacity, that is, able to support more users or traffic. In the following, we first define the terminologies and define suitable performance metrics to study these effects.

In order to gain some understanding and discover the fundamental principles of AS path prepending, we develop a ASPP toolkit and carry out performance evaluation so as to observe the local and global effect of route prepending. We will give a brief overview of our toolkit in later subsection.

We report our observations mainly in two steps. First, we consider a single multi-homed AS carrying out ASPP to achieve its local load balancing. We also analyze different approaches ASPP can be applied, and the local and global effects resulting from the action of a single AS. Next, we consider all ASes performing prepending and they all try to perform load balancing at the same time. Our analysis in the first step becomes handy for modeling each AS's algorithm of optimizing its local load balancing objectives. In this case, the interesting questions are whether this decentralized optimization process converges, and if it does, whether it converges to a desirable global state.

3.1 Terminology and Performance Metrics

Let a connected graph $G = (V, E)$ represents the inter-domain network topology. Each node $v \in V$ represents an entire AS; and each link $e \in E$ connects two ASes. For each link $e \in E$, let $B(e)$ define the business relationship between the two ASes for which e connects. For simplicity, we consider two specific relationships: *"provider-customer"* and *"peering"*. Fig. 9 illustrates a network of 7 ASes with links that convey different relationships. In particular, a provider-customer relationship is represented by a directed edge wherein the pointed node represents the customer. Whereas a peering relationship between two ASes is represented by a doubled-arrow link. Therefore, AS_4 is a provider for AS_3, AS_5 and AS_6 while AS_2 and AS_4 are customers of AS_0. There is one peering relationship in the figure and it is between AS_0 and AS_1.

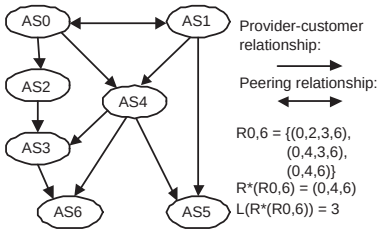

Figure 9: An example of 7 ASes with different relationships.

BGP is a policy-based routing protocol. In [4], authors illustrate two basic policies for route selection: (a) the *typical local preference policy* and (b) the *selective announcement export policy*. Under the typical local preference policy, an AS prefers to use a customer link than a peering link to forward a packet, and it prefers to use a peering link than a provider link to forward a packet, provided that these links can reach the destination AS. In other words, a customer link has a higher priority for selection than a peering link, while a peering link has a higher priority for selection than a provider link. Under the selective announcement export policy, an AS does not provide transit service for its providers. To illustrate, consider the routing selection in Fig. 9 wherein AS_5 is the source node and AS_6 is the destination node. Routes $(5, 4, 6)$ and $(5, 1, 4, 6)$ are considered legal or valid routes. On the other hand, route $(5, 1, 0, 4, 6)$ will not be announced because its violates the typical preference since AS_1 uses the peering link, rather than a customer link, for packet forwarding. Also, route $(5, 1, 4, 0, 2, 3, 6)$ will not be announced because it violates the selective export policy since AS_4 provides transit service for its providers AS_0 and AS_1.

BGP is also a path vector protocol. Among two equivalent routes based on business relationship, a BGP router picks the route with the shorter AS Path. Under BGP

convention, AS paths are represented by a sequence of AS numbers, and a prepended AS path is an AS path that has some *duplicated* AS numbers that appear consecutively. For convenience, we also represent an AS path as a sequence of links it traverses. For a prepended AS path, the corresponding links are prepended. For example, $((0,4),(4,6))$ is equivalent to $(0,4,6)$; and $((0,4),(4,6),(4,6))$ is equivalent to $(0,4,6,6)$ (when it is announced by AS_6 to AS_4. In this case, we say link $(4,6)$ is prepended."

A route r is a sequence of links $(e_1, e_2, ..., e_m)$. The length of a route r is denoted by $L(r)$ and it is equal to $m+1$, the number of AS along the path. Given a source and destination pair (s,d), for $s,d \in V$, let $R_{s,d}$ be a set that denotes all routes from s to d allowed by business relationships $B()$; and $R^*(R_{s,d})$ to denote the set of all shortest routes. So

$$L(R^*(R_{s,d})) \leq L(r) \qquad \forall r \in R_{s,d}.$$

To illustrate, consider again the graph in Figure 9 with the source node being AS_0 and the destination node being AS_6. There are three valid routes in $R_{0,6}$, they are $R_{0,6} = \{(0,2,3,6),(0,4,3,6),(0,4,6)\}$. The shortest path in this example is $R^*(R_{0,6}) = (0,4,6)$ and it has a length of $L(R^*(R_{0,6})) = 3$. Consider that AS_6 tries to reduce the amount of traffic on its incoming link from AS_4 by prepending it *twice* on this incoming link. After the prepending operation, we have $R_{0,6} = \{(0,2,3,6),(0,4,3,6),(0,4,6,6,6)\}$. The set of shortest path from AS_0 to AS_6 is now $R^*(R_{0,6}) = \{(0,2,3,6),(0,4,3,6)\}$ and $L(R^*(R_{0,6})) = 4$.

When the shortest route $R^*(R_{s,d})$ consists of a set of paths, we assume traffic from s to d is evenly divided on these paths. This assumption tends to balance traffic automatically. We are interested in studying how load balancing works even under this more *favorable* assumption.

Let $V_c \subset V$ be the subset of ASes that are multi-homed (i.e. ASes which have multiple providers), and for $v \in V_c$, let $E(v)$ be those links connecting v to its providers. ASPP is performed by a multi-homed v on its links $E(v)$. In order to define the action of ASPP, we define the following supersets of routes:

$$R_{*,d} = \bigcup_{s \in V} R_{s,d}; \qquad R_{*,*} = \bigcup_{d \in V} R_{*,d}.$$

In other words, $R_{*,d}$ denote all possible routes from any AS source to destination AS_d while $R_{*,*}$ denote all possible routes in the network G.

We consider two types of prepending actions, $\mathcal{P}$, a multi-homed AS can take:

- *Destination-specific prepending*: An AS_v repeats a link $e \in E(v)$ for all routes in $R_{*,d}$ that traverse link e and destine for AS_d. Generally, destination-specific prepending is a special case of the prefix-based prepending because it assumes that AS_v performs prepending for all prefixes of one AS in the same way. In our model, these two terminologies are interchangeable since we assume that every AS has only one prefix. An AS can obviously perform prepending for itself, i.e., when $AS_v = AS_d$, or an AS_v can perform prepending for its customer or descendant customer AS_d. We represent this action by $\mathcal{P} = e/d$. The set of routes after this prepending action is represented by $R_{*,d}(e)$.

- *Link-based prepending*: An AS_v repeats a link $e \in E(v)$ for *all* routes in $R_{*,*}$ that traverse e. We represent this action by $\mathcal{P} = e/*$, or simply $\mathcal{P} = e$. The set of routes after this prepending action is represented by $R_{*,*}(e)$.

A sequence of prepending actions is represented by concatenating these actions as a set wherein the ordering of concatenation is not important. For example, in Figure 1, if AS_6 has three prepending actions: prepends link (4,6) once and (3,6) twice, then the sequence of prepending actions can be represented as

$$P_6 = ((4,6),(3,6)^2)$$

where the subscript of P_i represents the set of prepending actions by AS_i. Similarly, the prepending actions of different ASes can be combined as well.

Furthermore, we use the notation $\mathcal{R}[s,d]$ to represent the subset of routes $\mathcal{R}$ that has source AS_s and destination AS_d. Similarly, $\mathcal{R}[s]$ is used to denote the projection of $\mathcal{R}$ to its subset with AS_s being the source node. Note, there is a subtle difference between $R_{s,d}$ and $\mathcal{R}[s,d]$. $R_{s,d}$ is always used to reference possible routes without prepending while $\mathcal{R}[s,d]$ can be any set of routes, some may be prepended. Therefore, the projection $\mathcal{R}[s,d]$ may contain prepended routes. In other words, $R_{*,*}(\mathcal{P})[s,d]$ is generally not the same as $R_{s,d}$.

Let T denote the relative traffic matrix. where T_{ij} represents the *relative* end-to-end traffic demand between the source node $i \in V$ and the destination node $j \in V$. The rationale of not defining a traffic matrix as real traffic demands is because we are not doing capacity planning in deciding how much bandwidth we need. Rather, we want to carry out scalability analysis and we use the traffic matrix to represent a *logical unit* of traffic and study what network routes will carry the maximum number of units of traffic.

We are now in the position to define various performance measures that we are interested in. They are:

1) Traffic on a link e, where $e \in E$: The traffic on each link e, before any prepending action, is simply the sum of

traffic, using the shortest path route allowed by the business relationships, that traverse link e. That is:

$$x_e = \sum_{s\in V}\sum_{d\in V}\sum_{r\in r^*(R_{*,*}(P)[s,d])}\frac{1}{k}T_{sd}I_{e\in r}. \quad (1)$$

where I is an indicating function with value of either 0 or 1 and $k = |R^*(R_{*,*}(P)[s,d])|$, which is the number of shortest paths in the set.

2) Local load balancing index $\mathcal{I}_{lb}$: Consider a single multi-homed AS_v, $\mathcal{I}_{lb}(v)$ measures the degree of load balancing of incoming traffic on providers' links $E(v)$. If an AS has two providers, the answer is quite straightforward since it is intuitive that a traffic ratio of 2:3 is better than a traffic ratio of 1:4. If an AS has more than two providers, then one has to define the degree of load balancing carefully. Let say AS_v has n provider links (i.e., $|E(v)| = n$), we have the traffic vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ where x_i is the amount of traffic on link i, for $i \in E(v)$. The local load balancing index $\mathcal{I}_{lb}(v)$ is a measure of the degree of load balancing among these n links:

$$\mathcal{I}_{lb}(v) = \frac{\left(\sum_{i=1}^n x_i\right)^2}{n\sum_{i=1}^n x_i^2}.$$

This index was first proposed for measuring the fairness of bandwidth allocation [5], but it also serves our purpose in this work. When the provider links have different bandwidths, then it is reasonable to balance the *percentage* loading on these links. Let b_i be the bandwidth for link $i \in E(v)$. Let

$$\boldsymbol{y} = \left(\frac{x_1}{b_1}, \frac{x_2}{b_2}, ..., \frac{x_n}{b_n}\right)$$

represents the fraction of loading on different provider links, then

$$\mathcal{I}_{lb}(v) = \frac{\left(\sum_{i=1}^n x_i/b_i\right)^2}{n\sum_{i=1}^n (x_i/b_i)^2}. \quad (2)$$

Note that $\mathcal{I}_{lb}(v) \in (0, 1]$. When $\mathcal{I}_{lb}v$ is close to 0, it implies the traffic loading on $E(v)$ is very skewed. When $\mathcal{I}_{lb}v$ is close to 1, it implies the loading of $E(v)$ is closely balanced. Without loss of generality, we assume all provider links have the same bandwidth unless we state otherwise.

3) Aggregated resource consumption A: Aggregated resource consumption, A, is simply the sum of traffic on all links. We have:

$$A = \sum_{e\in E} x_e \quad (3)$$

where x_e is given above. In other words, A measures the total amount of resource consumption. Note that for a given graph G, the value A can be different after a prepending action.

4) Global bottleneck traffic B: This measures the amount of traffic on the worse link in the network G:

$$B = \max_{e\in E}\frac{x_e}{b_e}. \quad (4)$$

5) Global load balancing index $\mathcal{G}_{lb}$: Like the local load balancing index, the global load balancing index also measures the closeness of traffic load on different links:

$$\mathcal{G}_{lb} = \frac{\left(\sum_{e\in E} x_e/b_e\right)^2}{|E|\sum_{e\in E}\left(x_e/b_i\right)^2}. \quad (5)$$

Again, $\mathcal{G}_{lb} \in (0, 1]$ and when the traffic on all links in the network is closely balanced (or highly skewed), $\mathcal{G}_{lb}$ is close to 1 (or 0).

As alluded to earlier, one of the interesting global effects we want to study is how much local prepending actions are able to relieve congested links so that the network as a whole can accommodate more users or traffic. Both the *global bottleneck traffic* and *global load balancing index* metrics can be used towards this end.

The global bottleneck traffic gives the *"hard"* limit of how many logical units of the traffic matrix the network can support. Further scaling up of the traffic will cause the global bottleneck link to first exceed its capacity. The global load balancing index, on the other hand, specifies a *"soft"* objective that tries to make sure traffic is as evenly spread as possible so that no link will be a specially bad bottleneck. Maximizing this latter index is more robust against uncertainties in the traffic matrix. So to maximize the effective network capacity for a certain traffic matrix, we can either minimize the global bottleneck traffic or maximize the global load balancing index.

Although we do not specifically try to optimize the aggregated resource consumption metric A, it is a useful gauge to check how much efficiency is lost due to load balancing. Note that in general, prepending will increase the value of A since ASes are selecting a longer AS path for routing.

All these local and global metrics are a function of the routing, hence influenced by the prepending policies. The interesting questions to explore are:

- What prependings are good for optimizing the global indexes?
- If each AS_v tries to choose its local prependings based on its local load balancing objective, will it generally lead to optimal (or good) routing based on global metrics?

3.2 ASPP Toolkit

Since the problem at hand is quite involved and one cannot carry out controlled experiments in the Internet, we

choose to develop a toolkit to study various scenarios. For small topologies, this toolkit can be used to visualize the network topology, business relationships between ASes, the routes for different destination (or source/destination pairs), optimal routes, and display various performance metrics as one carry out prepending operations. This is a pleasant alternative to manually examining various examples. Fig. 10 shows a screen shot of the GUI of this tool.

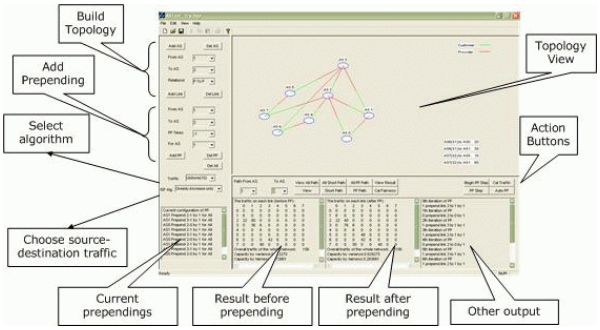

Figure 10: Screen shot of the GUI of ASPP toolkit.

The toolkit is an object-oriented toolkit with various objects performing a specific function. Some of the important objects (or modules) are:

- *Topology Generator:* The module generates various topologies based on different rules (i.e., power-law). It also allows a user to input or draw small scale topologies for testing and visualization.
- *Traffic Generator:* Given the topology, this module outputs the traffic generator T. Currently, user can specify various rules for traffic generation such as uniform, random or specific input traffic for testing.
- *Route Selection:* This modules implements the route selection under the BGP setting. In particular, different policies can be supplied such as the typical preference policy and selective export policy we mentioned above.
- *Prepending algorithms:* User can select different prepending algorithms (to be discussed in the following sub-sections). The tool can simulate the process of every ASes doing prepending in a synchronous manner and can evaluate different performance measures for different topologies and traffic matrices.
- *Statistic Gathering:* This module is to gather various traffic across all links and compute various performance measures mentioned above.
- *Visualization :* This module not only can display the topology under testing but also display various information for debugging. Some useful information include:
  - Display all paths from a specific source to a specific destination.
  - Display the set of shortest path from a specific source to a specific destination, either before or after a prepending operation.
  - Display the traffic on each link in the given topology.
  - Display various performance measures (i.e., load balancing index, aggregated resource consumption A, global bottleneck link, . . .) before or after a prepending operation.

In the following two subsections, we describe some example scenarios we studied using this toolkit, as well as some important observation we made. We believe these observation are crucial to understand the stability and convergence issues of distributed AS prepending.

3.3 Single AS Load Balancing Case

We first consider the case of a single multi-homed AS trying to do its local load balancing.

If a multi-homed AS has no customer of its own, it is known to as a *stub* AS; else a *transit* AS. We also refer to an AS with no provider of its own a *backbone* AS. In the earlier subsection, we defined link-based prepending and destination-specific prepending. For stub ASes, the only incoming traffic over the provider links are destined for the stub AS, hence we make the following observation:

Observation 1 *For a stub AS, destination-specific prepending is the same as indiscriminate link-based prepending.*

The above observation is intuitive since a stub AS does not provide transit service.

When would an AS perform destination-specific prepending? The answer is when its customer requests for it. There is a BGP attribute called *community*. A provider may be asked by its customer (or descendant customers) to implement destination-specific prepending for traffic destined for the customer using a specific community value.

Observation 2 *Using destination-specific prepending, a provider can shift a "subset" of traffic which a stub AS can shift, when that stub AS performs local prepending on that provider's incoming link.*

The implication of the above observation is that a provider AS can provide a *"finer"* granularity of load balancing on behalf of its customers. This can be illustrated by the example in Fig. 11. The only traffic in the network are from AS_4 to AS_0 and AS_5 to AS_0 with intensity of 40

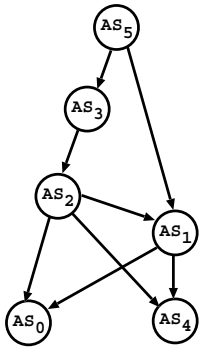

Figure 11: Example of provider doing ASPP for a customer

and 10 units of traffic respectively. The set of shortest paths from AS_4 to AS_0 is $\{(4,1,0),(4,2,0)\}$. The shortest path from AS_5 to AS_0 is $\{(5,1,0)\}$. Based on the route selection, the amount of traffic on the link $(1,0)$ is 30, the amount of traffic on the link $(2,0)$ is 20, which is shown in the second row in Table 1.

PP Configuration	traffic on (1,0)	traffic on (2,0)	$\mathcal{I}_{lb}(0)$	Shifted traffic volumn	A
$\varnothing$	30	20	0.96	-	100
(1,0)	5	45	0.60	25	105
$(1,0)^2$	0	50	0.50	30	110
$(1,0)^3$	0	50	0.50	30	110
(5,1)/0	25	25	1.00	5	105
$(5,1)^2$/0	20	30	0.96	10	110
$(5,1)^3$/0	20	30	0.96	10	110
(2,1)/0	30	20	0.96	0	100
$(2,1)^2$/0	30	20	0.96	0	100
$(2,1)^3$/0	30	20	0.96	0	100

Table 1: Effect of prepending by AS_0, or AS_1 on behalf of AS_0 on different links.

Because the load is not balanced, AS_0 considers prepending on link $(1,0)$. The next three rows in Table 1 show the results when AS_0 performs prepending one, two or three times respectively on link $(1,0)$, which are all *worse* than without prepending since $\mathcal{I}_{ld}(0)$ has a lower value. The next six rows in Table 1 show the result of AS_1 helping its customer AS_0 to balance its load by doing destination-specific prepending on link $(5,1)$ and $(2,1)$ (i.e., by prepending once, twice or three times). As we can observe, performing prepending at the provider may achieve a better local load balancing for AS_0 since $\mathcal{I}_{lb}(0)$ improves.

Most often an AS would simply do link-based prepending, since this is easier to configure and implement. So in the rest of the paper, we drop the word "link-based" when we say prepending.

The next question is, how does an AS find the prepending action that optimizes its local load balancing index? It can certainly do a search by enumeration, trying all combination of prepending values on all provider links up to some limit for each link. A more systematic algorithm is as follows.

Greedy Prepending Algorithm:
for a multi-homed AS v with links $E(v)$

1. termination_flag = FALSE;
2. Compute the local load balancing index $\mathcal{I}_{lb}(v)$ for traffic on all provider links $E(v)$;
3. Find the link $e^* \in E(v)$ such that it is the most heavily loaded;
4. **While** (termination_flag == FALSE) {
   /* `loop till no improvement.`*/
5. Evaluate the new traffic pattern if link e^* is prepended by 1;
6. **If** (new local load balancing index $\mathcal{I}_{lb}(v)$ does not improve)
7. termination_flag = TRUE;
8. **Else** Perform prepending on link e;
9. }

This algorithm assumes an AS can perform traffic measurements and based on the collected statistics predict the resulting load distribution for different prepending actions. This is a reasonable assumption, as [6] described efficient techniques for such measurements and predictions.

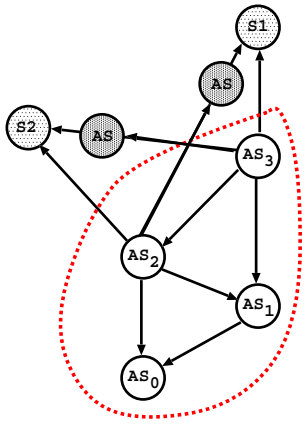

Figure 12: Example network for studying prepending algorithm convergence

We illustrate the Greedy Prepending Algorithm using the network in Fig. 12. There are four nodes, representing four ASes: AS_0, AS_1, AS_2 and AS_3. There are other

nodes solely for the purpose of generating and forwarding traffic. In particular, $S1$ and $S2$ are nodes which will generate traffic to AS_0 and AS_1. Note that in the next subsection, we will use the same network to illustrate the convergence issue when multiple ASes are doing prepending. For our purposes here, assume only AS_1 is doing load balancing via the Greedy Prepending Algorithm. Since there is no other traffic in this example, the traffic matrix T can be simply represented by a two by two matrix, specifying the relative traffic intensity from $S1$ to AS_0 and AS_1, and $S2$ to AS_0 and AS_1 respectively. Consider the following traffic matrix:

$$T = \begin{pmatrix} 140 & 10 \\ 0 & 10 \end{pmatrix}.$$

The operation of the Greedy Prepending Algorithm is summarized in the Table[1] 2:

PP Conf.	traffic on (2,1)	traffic on (3,1)	$\mathcal{I}_{lb}(1)$	A
$\varnothing$	10	56	0.67	454
(3,1)	16	3	0.68	464
(2,1)	10	10	1.00	460

Table 2: Result of executing the Greedy Prepending Algorithm by AS_1.

When there is no prepending, the amount of traffic on link $(2,1)$ is 10 and on link $(3,1)$ is 56. Note that *some* of the traffic on link $(3,1)$ are destined for AS_0. When AS_1 uses the Greedy Prepending Algorithm, it first chooses link $(3,1)$ to prepend. It then executes the algorithm again and prepend on link $(2,1)$. After this prepend operation, no further prepending is necessary since these two links are balanced and $\mathcal{I}_{lb}(1) = 1$.

In the following, we describe the global effect when multiple ASes are doing prepending.

3.4 Multiple AS Load Balancing Case

When multiple ASes are performing prepending actions at the same time, the situation becomes less predictable. First, we observe that under certain scenarios, one AS's local prepending actions *will not affect* or *interfere with* other's ASes local balancing index.

Observation 3 *If only stub ASes are performing prepending so as to balance traffic on their local provider links, then there is no interference in the network.*

[1]For the ease of presentation, we only indicate the integer part of the traffic.

This is clearly true because for a stub AS, say d, only links $E(d)$ are involved and no provider links of any other stub AS are part of the routes to d, $R_{*,d}$.

When a customer and its provider (or ancestor provider) are both doing ASPP at the same time, there can definitely be interference in the network. To study this phenomenon, we need to be more specific about the notion of *"doing ASPP at the same time"*. In here, we assume all participating multi-homed ASes start running some local algorithm in lock steps. In other words, at time interval t, they all start doing traffic measurements first; then at the end of the interval, they all run an algorithm such as the Greedy Prepending Algorithm described in the last subsection and decide their prepending decisions. The overall prepending actions will be $P(t) = \bigcup_{v \in V_c} P_v(t)$. All these ASes will then go back to do traffic measurement at time interval $t+1$ and execute the prepending algorithm again, ad nausea.

If an AS v decides not to do any prepending in time interval t, then $P_v(t) = \varnothing$ (the empty set). If for some $t = t_{last}$, $P(t_{last}+1) = \varnothing$ for all ASes, then $P(t) = \varnothing$ for all $t \geq t_{last}$ (assuming the network topology and traffic matrix are stationary). If this is true, we say the prepending algorithm has *converged* for the given network topology and traffic matrix.

Observation 4 *The Greedy Prepending Algorithm introduced in Section 3.3 does not always converge.*

We illustrate this observation by example. Consider the network in Figure 12 introduced in the last subsection. In this case, *both* AS_0 and AS_1 are performing prepending. We show that the convergence of the Greedy Prepending Algorithm actually *depends* on the traffic matrix.

1) Traffic matrix example 1: Consider this traffic matrix:

$$T_1 = \begin{pmatrix} 20 & 20 \\ 40 & 30 \end{pmatrix}.$$

Let us go over the state transitions when multiple ASes are performing the Greedy Prepending Algorithm, as summarized in Table 3. The initial load (i.e., without prepending) on each link[2] are computed in the second row, based on applying Equation (1) to the shortest paths from sources to destinations before prepending. The ensuing *prediction* step shows the local evaluation by AS_0 and AS_1 of what will happen if a prepending action is taken. The subsequent *update* step shows what prepending actually took place - only AS_0 decided to prepend link $(2,0)$. This action led to new traffic load on the various links which may be different than the prediction if someone else also took a prepending action (e.g. in AS_1's case). Then the two ASes perform prediction again, and this time neither finds

[2]Note, (2,0), (1,0) are provider links for AS_0; (2,1) and (3,1) are provider links for AS_1; (3,2) is a provider link for AS_2 but AS_2 is not doing load balancing since it has a single provider link.

any reason to prepend again. Both the global load balance index and bottleneck traffic improved slightly, at a moderate increase of total resource consumption A (from 126 to 150). This is a desirable situation, as it shows the decentralized local load balancing actions converged resulting in a reasonable global state.[3]

PP C.	AS_0			AS_1			AS_2	A	$\mathcal{G}_{lb}$	B
	(1,0)	(2,0)	$\mathcal{I}_{lb}(0)$	(2,1)	(3,1)	$\mathcal{I}_{lb}(1)$	(3,2)			
∅	6	52	0.61	30	26	0.99	6	126	0.33	52
Prediction:										
(2,0)	40	20	0.90							
(2,1)				15	41	0.82				
Update										
(2,0)	40	20	0.90	50	40	0.98	0	150	0.36	50
Prediction:										
(1,0)	6	52	0.61							
(2,1)				15	55	0.75				
No more prepending based on local predictions ⇒ convergence										

Table 3: Example showing the Greedy Prepending Algorithm will converge

2) Traffic matrix example 2: We show that a slight change in the traffic matrix can lead to a very different behavior. Consider

$$T_2 = \begin{pmatrix} 20 & 30 \\ 40 & 10 \end{pmatrix}.$$

In this case, the results are shown in Table 4, in a more condensed form (only showing the traffic on key links before and after prepending steps, without the prediction steps). This time, we have a problem. The Greedy

PP C.	AS_0			AS_1			AS_2
	(1,0)	(2,0)	$\mathcal{I}_{lb}(0)$	(2,1)	(3,1)	$\mathcal{I}_{lb}(1)$	(3,2)
∅	6	52	0.61	10	36	0.75	6
(2,0) (3,1)	32	28	0.99	58	14	0.72	18
(2,1)	6	52	0.61	10	36	0.75	6
Back to the initial state! ⇒ oscillation							

Table 4: Example showing the Greedy Prepending Algorithm does not converge

Prepending Algorithm results in a repeating pattern of prepending actions.[4]

Note that the Greedy Prepending Algorithm was defined when considering a single AS doing prepending. Now we observe that when multiple ASes are running the Greedy Prepending Algorithm, it may not converge. At first, there seems to be a simple fix. Consider this variation to the algorithm:

[3]In this example, as well as the rest of examples based on the same network, we compute the global metrics based on only the four ASes and links between them. We ignore $S1$, $S2$ and the other ASes introduced solely for sourcing traffic.

[4]In real life, if this happens, there is probably a limit beyond which prepending would be stopped.

Decrease-first Greedy Prepending Algorithm:
For a multi-homed AS v with links $E(v)$

1. Compute local load balancing index $\mathcal{I}_{lb}(v)$ for traffic on all provider links $E(v)$;
2. Let e^* be the most lightly loaded link;
3. **If** (reduce prepending on e^* can improve load index)
4. reduce the number of prepending on e^* by 1;
5. **Else**
6. execute the Greedy Prepending Algorithm;

The Decrease-first Greedy Prepending Algorithm is designed to prevent perpetual increasing levels of prepending by first trying to *remove* prepending to help balancing the load. This algorithm solves the problem for the above example. This time, it results in the following prepending steps: Note, we use the notation "$-e$" to indicate the

PP C.	AS_0			AS_1			AS_2
	(1,0)	(2,0)	$\mathcal{I}_{lb}(0)$	(2,1)	(3,1)	$\mathcal{I}_{lb}(1)$	(3,2)
∅	6	52	0.61	10	36	0.75	6
(2,0) (3,1)	32	28	0.99	58	14	0.72	18
-(3,1)	40	20	0.90	30	50	0.94	0
The network reached a new stable state.							

Table 5: Example showing that the Decrease-first Greedy Prepending Algorithm converges

removal of a previous prepending action on link e. In this example, after both AS_0 and AS_1 prepended, AS_1 removed its earlier prepending; and after that neither ASes finds it profitable to prepend any more. After convergence, the resultant global metrics are in Table 6:

PP C.	A	$\mathcal{G}_{lb}$	B
∅	116	0.29	52
(2,0) (3,1)	168	0.39	58
-(3,1)	140	0.36	50

Table 6: Global performance metrics after the Decrease-first Prepending Algorithm converges.

To appreciate what is going on, we define a *network cut* as a set of links when removed will partition the network into two halves. Further, if each link[5] in the cut is prepended by the same number of times, then we call this a *uniform cut prepending*.

Observation 5 *Let E_{cut} represent a cut of the network graph G, and $P = E^*_{cut}$ be a uniform cut prepending. Then $r^*(R(P)[s,d]) = r^*(R)$ for all s and d. In other words, a uniform cut prepending has no effect on routing.*

[5]strictly speaking, unidirectional link going toward the same half of the network.

What happens when the original Greedy Prepending Algorithm is applied to the second traffic matrix T_2 is that the consecutive prepending actions by AS_0 and AS_1 form a uniform cut, which keeps leading the routing back to the initial state. The Decrease-first Greedy Prepending Algorithm is able to explore more states. By removing a prepending action, it can sometimes break out of the repetitive cycle.

3) Traffic matrix example 3: Consider a third example:

$$T_3 = \begin{pmatrix} 20 & 30 \\ 10 & 80 \end{pmatrix}$$

This case results in the following sequences of prepending actions by the Greedy Prepending Algorithm and Decrease-first Greedy Prepending Algorithm. The

PP C.	AS_0			AS_1			AS_2
	(1,0)	(2,0)	$\mathcal{I}_{lb}(0)$	(2,1)	(3,1)	$\mathcal{I}_{lb}(1)$	(3,2)
∅	6	22	0.75	80	36	0.87	6
(2,1)	6	22	0.75	40	76	0.91	6
(2,0)	20	10	0.90	40	90	0.87	0
(3,1)	6	22	0.75	80	36	0.87	6
Back to the initial state! ⇒ oscillation							

Table 7: Another example when the Greedy Prepending Algorithm does not converge.

Greedy Prepending Algorithm still does not converge since the prepending actions on links $(2,1)$, $(2,0)$ and $(3,1)$ by AS_1, AS_0 and AS_1 respectively are considered a uniform cut preprending. Furthermore, the Decrease-first Greedy Prepending Algorithm does not converge either. In this case, the prepending actions on links $(2,1)$

PP C.	AS_0			AS_1			AS_2
	(1,0)	(2,0)	$\mathcal{I}_{lb}(0)$	(2,1)	(3,1)	$\mathcal{I}_{lb}(1)$	(3,2)
∅	6	22	0.75	80	36	0.87	6
(2,1)	6	22	0.75	40	76	0.91	6
(2,0)	20	10	0.90	40	90	0.87	0
-(2,1)	25	5	0.69	85	50	0.93	0
-(2,0)	6	22	0.75	80	36	0.87	6
Back to the initial state! ⇒ oscillation							

Table 8: The Decrease-first Greedy Prepending Algorithm fails to converge too.

and $(2,0)$ are nullified which lead the network back to the original routing. In summary, the Decrease-first Greedy Prepending Algorithm explores more routing than the Greedy Prepending Algorithm; but it may also fail to converge.

Observation 6 *Both the Greedy Prepending Algorithm and Decrease-first Greedy Prepending Algorithm may converge to a worse global state than before prepending.*

4) Traffic matrix example 4: Finally, we observe that the decentralized prepending actions may actually lead to a *"worse"* global state. Consider the following:

$$T_4 = \begin{pmatrix} 40 & 20 \\ 40 & 50 \end{pmatrix}.$$

In this case, AS_0 sees load balancing can help balance its unbalanced traffic on links $(1,0)$ and $(2,0)$. After implementing this, we observe all the global metrics are slight worse off, as shown in Table 9.

PP C.	AS_0			AS_1			AS_2	A	$\mathcal{G}_{lb}$	B
	(1,0)	(2,0)	$\mathcal{I}_{lb}(0)$	(2,1)	(3,1)	$\mathcal{I}_{lb}(1)$	(3,2)			
∅	13	66	0.68	50	33	0.95	13	188	0.36	66
(2,0)	60	20	0.80	70	60	0.99	0	210	0.35	70
Neither AS_0 nor AS_1 has reason to further prepend.										

Table 9: Convergence to worse global state

In this case, AS_0's prepending action lead to better local load balance for both AS_0 and AS_1, yet globally, the metrics B and $\mathcal{G}_{lb}$ are slightly worse off. The reason is that the link $(3,2)$ carries no traffic after the prepending; hence globally, the balance of link utilization is a little worse off. In this example, the global state is not much worse off than before, but the important lesson is that it can get worse than local prepending started.

4 Guidelines

In this section, we present some guidelines for AS operators so as to prevent the route oscillation problem.

Guideline 1 *If only stub ASes are performing prepending actions so as to balance the traffic on their local provider links, then these prepending actions will not result in route oscillations.*

Proof: This has been stated as Observation 3. ∎

Note that this guideline does not allow transit ASes to perform any prepending, hence it is quite restrictive. A less restrictive guideline is as follows:

Guideline 2 *If no AS performs prepending except on the routes originated by itself, then these prepending actions will not result in route oscillations.*

Proof: In order to prove guideline 2 works, we want to put all ASes in different "levels". Let us classify all ASes into different "*levels*". Let V_0 be the lowest level such that it contains all stub ASes only. Let V_1, $V_2, \ldots$ be the successively higher levels. An AS v belongs to V_i if all its customers come from levels V_j, where $j < i$.

Under guideline 2, prepending actions of an AS can only affect traffic destined to itself. This traffic will not traverse any inbound provider links of a lower or same level AS because all AS paths should be *"valley free"* based on the *selective announcement* export policy[1]. Therefore, any prepending by an AS will not affect the prepending decision (based on load balancing) of another

AS at the same level or a lower level. Let us call this property 1.

Now we can prove the convergence by induction. Suppose at some time t_i, all ASes in levels from 0 to i have completed their prependings and will not do any more prepending. Therefore, ASes in level $(i+1)$ will do their prependings without being affected by any prepending actions from these lower level ASes. Based on property 1, we know prepending actions from ASes at the same or higher level cannot affect the prepending decisions of ASes in this level $i+1$ either. So there must exist some time t_{i+1} so that all ASes in levels from 0 to $i+1$ will have completed their prependings and will not change any more.

Again from property 1, we know ASes in level 0 cannot be affected by any prepending actions of ASes from the same or higher level. Therefore, there must exist a time t_0 that they complete their prepending without further change. Since there are finite number of levels in the Internet, the whole process must converge. This completes our proof by induction. ∎

5 Related Work

Based on BGP routing tables from routers connected to the AT&T backbone, Feamster et al reported that over 30% of the routes had some amount of ASPP, and most of these paths were prepended with one or two ASes [3]. However, Broido et al reported that only 6.5% of the routes in the November 2001 Route Views data had ASPP [7]. This significant difference shows that it is important to observe the ASPP on different levels of the Internet routing hierarchy.

Swinnen et al used computer simulation to evaluate the ASPP method [8]. In the simulation model, each stub AS connected to two different transit ASes. When each stub AS prepended one AS to one of the route announcements, their simulation results showed that the distribution of the inter-domain paths changed for almost all stub ASes. Moreover, the impact of the ASPP was different for each stub AS. With a prepending length of 2, almost all the inter-domain paths were shifted to the nonprepending link. Beijnum studied the impact of ASPP on a doubly homed stub AS under two different scenarios [9]. The first one was when the stub AS was doubly homed to similar ISPs in the sense that the ISPs directly peered with each other via the same network access point. The second case was when the stub AS was doubly homed to dissimilar ISPs that did not directly peer with each other. He used a simple example to show that applying the ASPP to the second case had a more gradual effect on the change of the incoming traffic distribution.

Motivated by a lack of systematic procedure to tune the ASPP, Chang and Lo proposed a procedure to predict the traffic change before effecting it. They implemented and tested the procedure in an operational, doubly homed AS which was connected to two regional ISPs [6]. The measurement results showed that the prediction algorithm was fairly accurate. Moreover, the traffic shift peaked when the prepending length was changed from 2 to 3, and almost 60% of the routes were affected.

Given the AS relationship, Wang and Gao inferred BGP router's import and export routing policies [4]. For the import policies, they inferred on the local preferences that is usually the first metric used for selecting the best route. Their measurement study confirmed that the routes learned from customers are preferred over those from peers and providers. For the export policies, they have observed a large percentage of selective announced prefixes from some ASes. Furthermore, they showed that the cause for this came from selective announcement which was practiced by multihomed ASes to balance the inbound traffic.

Various kinds of BGP route oscillation problems have been studied in the past. Varadhan et al [10] studied persistent route oscillation in general whenever ASes do independent route selection under local policies. Route oscillation problems with using the MED attribute have been studied by [11][12]. In [13], Gao and Rexford proposed a set of guidelines for an AS to follow in setting its local policies to avoid route oscillations. But none of the previous work considered route oscillation caused by ASPP.

6 Conclusion

We have made several contributions towards using BGP's ASPP approach to do inbound traffic engineering. First, through careful analysis of the Routeviews data, we report a trend of increasing level of multi-homing and use of ASPP in BGP, which means a lot of network operators are doing load balancing. Our subsequent modeling and analysis take a systematic look at the fundamental issues: (a)how is AS path prepending done locally? (b) what are important global metrics? (c) does this decentralized ASPP process converge? (d) If it does, does it improve global metrics? We characterized when there is no interference between local load balancing (when only stub ASes do load balancing), and exposed some problems (convergence, optimality) when providers are also doing load balancing. Lastly, We also provide guidelines so that AS operators can avoid instability and route oscillation.

References

[1] L. Gao, "On inferring autonomous system relationships in the internet," in *Proc. IEEE Global Internet Symposium*, Nov. 2000.

[2] B. Quoitin, et al, "Interdomain traffic engineering with BGP," *IEEE Commun. Mag.*, vol. 9, no. 3, pp. 280–292, May 2003.

[3] N. Feamster, J. Borkenhagen, and J. Rexford, "Controlling the impact of BGP policy changes on IP traffic," AT&T Research, Tech. Rep. 011106-02, Nov. 2001.

[4] F. Wang and L. Gao, "On inferring and characterizing Internet routing policies," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Oct. 2003.

[5] R. Jain, D. M. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems," Digital Equipment Corp, Tech. Rep. TR301, 1984.

[6] R. Chang and M. Lo, "Inbound traffic engineering for multihomed ASes using AS path prepending," in *Proc. IEEE/IFIP Network Operations and Management Symp.*, Apr. 2004.

[7] A. Broido, et al, "Internet expansion, refinement and churn," *European Trans. Telecommun.*, Jan. 2002.

[8] L. Swinnen, "An evaluation of BGP-based traffic engineering techniques," available from http://www.info.ucl.ac.be/people/OBO/biblio.html.

[9] I. Beijnum, *BGP*. O'Reilly, 2002.

[10] K. Varadhan, et al, "Persistent route oscillations in inter-domain routing," *Computer Networks*, vol. Vol.32, no. No.1, pp. 1–16, 2000.

[11] A. Basu, et al, "Route oscillations in IBGP with route reflection." Pittsburgh, Pennsylvania, USA: SIGCOMM, August 2002.

[12] T. Griffin and G. Wilfong, "Analysis of the MED oscillation problem in BGP." Paris, France: 10th IEEE International Conference on Network Protocols (ICNP'02), November 2002.

[13] L. Gao and J. Rexford, "Stable Internet Routing Without Global Coordination," *IEEE/ACM Trans. Networking*, vol. 9, no. 6, pp. 681–692, Dec. 2001.