A Theoretical Method for BGP Policy Verification

Raj Kumar Rajendran, Dan Rubenstein, Michael Wasserman
Dept. of Electrical Engineering
Columbia University, New York, NY 10025
Email: {raj,danr,mike}@ee.columbia.edu

***Abstract*— Since BGP broadly determines the flow of data over today's Internet, ensuring that autonomous systems (ASes) throughout the network properly implement the protocol is crucial. However, serious anomalous behavior continues to occur because BGP lacks explicit rules for enforcement or tools for policy verification. In addressing this lack, we present a distributed, theoretically grounded technique that can be used by an AS to check whether the other ASes, whom it relies on to carry its traffic, are in fact operating according to the norm. The theoretical grounding of our techniques ensures that when we reveal an inconsistency, it *must* in fact exist.**

We demonstrate the efficacy of our approach by showing how it would have detected the violation caused by the well-known AS7007 incident. Then we apply our technique to real BGP data consisting of over a quarter of a million unique routes collected from seven different sources and show that all examined sites show a significant number of violations of standard BGP policy. Significantly, in nearly all sites, more than 1% of the ASes examined were involved in errant paths. In addition we show that combining views from multiple ASes enhances violation detection.

Keywords: Network Architectures, Network Design and Planning, Network Management and Control, Routing, Traffic Engineering, BGP, Graph theory, Anomaly detection

I. Introduction

BGP broadly determines the flow of data over today's Internet. As a result, ensuring that autonomous systems (ASes) throughout the network properly implement the protocol is of paramount importance. In the past, there have been instances where small oversights in proper implementation had a catastrophic impact. The most famous example is the AS7007 incident in which AS7007 announced very short, inaccurate routes to most of the Internet. For over two hours this disrupted connectivity to large tracts of the Internet. Despite the publicity this incident generated, it is clear that serious anomalous behavior continues to occur. Most recently, AS3561 propagated more than 5000 improper route announcements again leading to global connectivity problems [12], [3]. These accidents demonstrate that BGP is also a fertile ground for deliberate attack.

These Autonomous Systems are independently operated (as the name suggests), but must cooperate for the Internet to function correctly. BGP can be thought of as the protocol that sets the rules for cooperation. However, it lacks explicit rules for enforcement, or even a means that allows an AS to confirm that the ASes it must interact with are in fact "playing by the rules".

In this paper, we investigate *distributed, theoretically grounded* techniques that can be used by an AS to check whether the other ASes, on whom it relies upon to carry its traffic, are in fact operating as they are supposed to. With respect to recent work in this area, our work offers the following advantages:

- Our technique can be implemented within a single AS, using only the information it obtains under normal BGP operations. Sharing information gathered from other ASes is not required.
- The theoretical grounding of our technique prevents the occurrence of false positives. When we reveal an inconsistency, that inconsistency *must* in fact exist.

These advantages trade off the detail of information our analysis can provide for both the flexibility of a distributed implementation and provable correctness of the technique we employ. Recent work that has addressed this same problem uses heuristics to make the best educated guess possible about the state of the network, and can incorrectly infer that there are problems when in fact none exist [6], [16], [7], [18]. Other work is willing to take advantage of simultaneous analysis of multiple AS perspectives [21], where, for instance, ASes are willing to share their routing table information. Our work contributes to a more visionary, long-term goal, which is to develop a complete methodology that will allow nodes running a distributed routing protocol to use the BGP route information they receive from their routing neighbors to simultaneously check the correctness of the distributed implementation within which they are running, and to identify offenders. We seek to develop a theory that never mistakenly identifies an offender, and we wish to identify, under this constraint, how often offenders can in fact be detected.

We begin by formalizing the rules that capture the manner in which providers running BGP are supposed to act. Essentially, we assume that all operators implement what are known as prefer-customer and/or shortest path selection algorithms and the valley-free export policy. Under this assumption, we identify a property that must hold between any pair of paths reported to an AS. Namely, if one considers only the ASes that appear along both reported paths, we show that there are restrictions on the relative orders in which these ASes can appear between the two paths - any violation is an indication that some AS is not strictly following prefer-customer and/or shortest path selection algorithms and the valley-free export policy.

We demonstrate the efficacy of our approach by first showing that we would have detected the violation caused during the AS7007 incident. We then apply our technique to real BGP data collected from seven different sites that include the Oregon Route-View project, AT&T, Optus and Columbia University, and show experimentally that all examined sites show a significant number of violations of standard policy within the current Internet. Additionally, we show that, while not required, combining the views from multiple ASes enhances the ability of detecting violations implying that well connected ASes can very effectively police their routes for violations of standard policy.

Having such a policing mechanism in addition, assists in identifying accidental misconfigurations and those who choose not to follow conventional policy. It is also a further deterrent to malicious attackers, since a careless attack can easily be detected. And, since the attacker only has a subset of vantage points, even if the attacker can construct an attack that cannot be detected at its own vantage points, this does not rule out the ability to detect the attack at alternative uncorrupted vantage points.

A. Prior Work

While several works have identified that disruption due to incorrect implementation of BGP is an important problem, the approaches to solutions have been different. In [15] the authors propose a secure tool that can be used to locate the source of routing misbehavior, while others propose authentication and encryption based schemes [1], [9]. Some bodies of work have attacked broadly similar questions. Among these is competitive routing [13] where selfish users

in a communications network attempt to maximize their flow by controlling the routing of their flows and the works attempt to find if there exist equilibrium states. This body of work differs in that they are concerned with traffic flows rather than reachability and with a competitive environment rather than a misconfigured environment.

To the best of our knowledge, this is the first work that provides a technique for detecting static inconsistencies while being grounded in a theoretical framework. It complements other studies that have sought to understand the nature of misconfigurations [11] and yet others that aim to detect misconfigurations through observance of dynamic behavior [10], [2], [19], [8], [5], [4].

The rest of the paper proceeds as follows. In Sec. II we introduce BGP. In Sec. III we formalize the rules of proper BGP behavior and in Sec. IV we present our model of BGP. In Sec. V we show how individual nodes can check for inconsistencies given the BGP model of Sec. IV. Sec.VI shows how our methodology would have detected a well-known misconfiguration that widely disrupted traffic while Sec. VII presents the results of the application of our methods on real BGP data. Sec. VIII lists further issues for consideration and Sec. IX concludes the paper.

II. BGP

In this section we briefly introduce the Border Gateway Protocol (BGP). For more details the reader is referred to [20]. The BGP network consists of a set of Autonomous Systems (AS), each of which supports a set of IP addresses. Contiguous groups of IP addresses supported by a single AS are referred to as prefixes. The ASes are linked to each other in a network through routing-relationships. The BGP network is broadly tree-structured and ASes can therefore be broadly classified as belonging to tiers where the tier-numbers decreases as we approach the root of the tree. The ASes at the highest tier form a clique.

The routing-relationships between a pair of ASes can be of three types: [1]

- Customer-Provider
- Provider-Customer
- Peer-Peer

In a Customer-Provider or Provider-Customer relationship, one AS plays the role of the Provider, and is responsible for making the prefixes (or IP addresses) of its Customer known to the other ASes that constitute the Internet, and ensuring that the prefixes supported by all other ASes are visible to the Customer. Except for ASes at the highest and lowest tiers, most ASes simultaneously play the roles of Provider and Customer while a significant fraction of ASes have peering relationships. ASes typically play the role of Provider to multiple Customers and the role of Peer with multiple Peers. However ASes typically play the role of Customer to only one Provider. Occasionally a Customer may have multiple Providers (termed multi-homing) to provide multiple-paths to destinations thereby providing resiliency against failure. An AS that plays the role of a Provider to different ASes is responsible for providing access to all prefixes in the network to its Customers. Two ASes can also have a Peer-Peer relationship where each of the peers makes visible (provides routes) all its descendant prefixes to the other peer. The set of all ASes to which an AS has a Provider, Customer or Peer relationship are called its *neighbors*.

An AS makes prefixes visible to another AS by providing it a *route* to those prefixes. The format with which these routes are provided takes the form of strings of AS identifiers than need to be sequentially traversed to reach a prefix. All routes are learned from neighboring ASes. Since an AS typically has multiple neighbors, it may learn about routes to a prefix from more than one neighbor. It has to then choose its preferred route to a prefix from the various choices provided by its neighbors, using what is termed its *route-selection* algorithm.

[1]There is also a fourth kind of relationship known as a sibling-sibling relationship, but such relationships are rare and do not affect our results.

A. Route-Selection

BGP recommends that routes be selected by the ordered criterion of Fig. 1. Our analysis involves the first two of these steps, largest local-preference, and shortest AS path-length. BGP allows an AS to set an attribute called local-preference to each connection to a neighbor. [2] When the AS needs to decide between multiple routes to a prefix, the route with the larger local-preference value is chosen. This allows an AS to indicate the preference-order among neighbors when routing choices are available.

ASes may prefer routes through customers over routes through peers and routes through peers over routes through providers since the cost of routing through customers is typically less than the cost of routing through peers which in turn is less than the cost of routing through providers. However many ASes do not set local-preference values, and therefore when multiple paths are available, the shortest-path, or the path with fewest ASes, is chosen [24].

When ASes set their local-preference attributes according to the cost-considerations stated above, a route-selection algorithm popularly known as *prefer-customer* route-selection results where routes through customers are preferred over routes through peers or providers. When ASes ignore setting their local-preference values an algorithm we call the *shortest-path* routing algorithm results, where the shortest-path is chosen to a destination irrespective of the routing-relationships.

1) Largest *local-preference*
2) Shortest AS path length
3) Lowest origin type ($IGP < EGP$)
4) Lowest MED (with *same next-hop AS)*
5) *eBGP-learned over iBGP-learned*
6) *Lowest IGP path-cost to exit point*
7) *Lowest router ID or BGP speaker*

Fig. 1. BGP Route-selection

B. Export Policy

Even though each AS will ultimately know about routes to all prefixes in the BGP network, it only makes certain routes visible to each of its neighbors. The decision about which routes are made available to which neighbor is usually determined by an AS's relationship to its neighbor due to cost considerations and responsibilities to neighbors. An AS needs to make its customer's prefixes visible to the world, so it typically exports customer-routes to its providers. Similarly, it needs to make the prefixes of all ASes visible to its customers, therefore makes all its available routes visible to its customers. In routing traffic to neighbors, traffic to a provider is usually the most expensive, so ASes form peer-peer relationships to avoid accessing each-other's customers through providers. Therefore peers export customer-routes to each other. Additionally since traffic to providers and peers do involve costs, ASes do not typically carry provider-provider traffic or peer-peer traffic, and so do not export provider-routes or peer-routes to other providers or peers.

These considerations and consequent actions lead to an export policy widely known as the *valley-free* export policy. Various studies

[2]It can also be set at a prefix level but for our purposes this is not relevant.

have shown that nearly all ASes follow this policy [6], [7], [17] in exporting routes. This policy is reiterated in Table I below:

Relationship to Neighbor	Policy
Peer-Peer	Exports only routes to prefixes learned from its Customers
Customer-Provider	Exports only routes to prefixes learned from its Customers
Provider-Customer	Exports all known routes

TABLE I

The above export-policy is called *valley-free* because if all nodes in a route adhere to this policy, the resulting route consists of a sequence of customer-provider traversals to a peak which may consist of zero or one peer-peer traversals, followed by a sequence of provider-customer traversals.

We now proceed to formalize our model of the BGP network and policies.

III. Methodology

Since BGP allows individual nodes to run practically any routing algorithm, it is not apparent what it is that we are looking for when we check or misconfigurations or inconsistencies. We first therefore formalize the notion of inconsistency as applied to BGP.

We start by distinguishing between *tight* protocols, where each node is required to run exactly the same algorithm and *loose* protocols where different nodes may run different algorithms in choosing and communicating routes. Consider distance-vector routing where the the exact algorithm that each node needs to run in order to decide the best route among available routes and how it is exported is exactly specified. This is an example of a *tight* routing-protocol. BGP on the other hand allows each node to choose its routing-algorithm. Each node independently decides how it will choose the best route among its choices and to whom it will export these routes. This is an example of a *loose* routing protocol. We start by defining these notions.

Each node i participating in a distributed routing-algorithm receives data about the network from other nodes, performs computations on it, then exports some state information to other nodes. The received and computed information is collectively called its state S_i. As part of its computations, it chooses the best route to each destination using a *route-selection* algorithm. After completing its computations it uses a second algorithm called the *export-policy* to determine what information gets distributed and to whom. We define a *routing-algorithm* f to be a combination of a route-selection algorithm and an export-policy.

We define a routing-protocol's routing-policy P to be a subset of routing algorithms from the set of all possible routing-algorithms. A node that participates in a network with a routing-protocol that has routing policy P, uses any of the routing-algorithms in P to make its routing decisions. A routing-protocol with a routing-policy P that has only one routing-algorithm ($|P| = 1$) is said to be *tight*, while a protocol that has multiple routing-algorithms in its routing policy ($|P| > 1$) is said to be *loose*.

Even though an individual node i can use all the routing-algorithms in P in making routing decisions, it may choose to use only a subset of the routing-algorithms available in P. This subset $L^i \subseteq P$ is called i's *local-policy*. Now consider a node i that is operating with a local-policy that is smaller than P, i.e. $L^i \subset P$. In this case other nodes j in the network may run with local-policy contained in the local-policy of i i.e. $L^j \subseteq L^i$, in which case we say that j is *consistent* with i, or they may run with local-policy different than i i.e. $f \in L^j, f \notin L^i$ in which case we say that j is *inconsistent* with i.

A. Application to BGP

We now define BGP in the terms defined above. Since BGP gives each node complete freedom in choosing its routing-algorithm, we set P, BGP's routing-policy, to be the set of all possible routing-algorithms. However, as BGP allows each node to have its own local-policy, it is not immediately apparent how we can arrive at a local-policy L such a $L \subset P$. Luckily experimental observation have shown that nearly all BGP nodes adhere to certain rules in making their routing decisions [6], [16], [7], [11], [17]. These experiments have noted that nearly all nodes running BGP use a route-export algorithm called the *valley-free* property and one of two route-selection algorithms: the *prefer-customer* or the *shortest-path* algorithm. From these observations we construct two local-policies. The first we call the *standard* or prefer-customer local-policy $L = \{f_{sp}, f_{pc}\}$ has two routing-algorithms in it. The first f_{sp} is the valley-free export policy combined with the shortest-path route-selection algorithm, and the second f_{pc} is the valley-free export policy combined with the prefer-customer route-selection algorithm. We will also refer to a second local-policy $L_{sp} = \{f_{sp}\}$ called the *shortest-path* local-policy with only one routing algorithm f_{sp} in it.

We now have a way to specify proper behavior in a BGP network. We can decide that proper behavior entails behaving according to either the standard local-policy or the shortest-path local-policy and we can study if other nodes in the network implement routing-algorithms consistent with this assumed norm of behavior. To do this we model the BGP network and routing algorithms.

IV. The Model

In this section, we formally define the BGP network and its routes.

Definition We represent the BGP network as a typed graph $G = (A, E, T)$ where $A = \{a_1 \dots a_n\}$ is a set of nodes that represent ASes and E is a set of edges where each edge e_{ij} uniquely connects two nodes a_i and a_j. T is a set of types t_{ij} corresponding to each edge where $t_{ij} \in R = \{c, p, r\}$. Type c indicates a customer-provider edge shown as $\overrightarrow{a_i a_j}$, type p indicates a provider-customer edge $\overrightarrow{a_i a_j}$ and type r marks a peer-peer edge $\overline{a_i a_j}$. Peer-peer types are symmetric, i.e. $\overrightarrow{a_i a_j} \Rightarrow \overrightarrow{a_j a_i}$ while customer-provider and provider-customer types are inverses, i.e. $\overrightarrow{a_i a_j} \Rightarrow \overrightarrow{a_j a_i}$.

We refer to nodes with an edge between them to be neighbors. We also refer to a provider-customer edge between two nodes to be a parent-child edge and in extension, two nodes a_1, a_2 connected to one-other by a sequence of parent-child edges to be ancestor and descendant.

We assume that the nodes of G are connected by edges in such a way that no node is its own ancestor or its own descendant. We call this the *cycle-free* property.

A. BGP Routes

Each node in a network running BGP is provided a set of routes by each of its neighbors. Routes are ordered sequences of nodes where any two consecutive nodes are neighbors.

Definition A route from a_1 to a_k is the ordered set of nodes $a_1..a_k$ such that a_{i-1} and a_i are neighbors $1 < i < k$.

We denote a route from node a_i to node a_j as $a_i..a_j$ and its length $d(a_i..a_j)$ to be a distance between a_i and a_j. Typically this will simply be a count of the number of edges in the route.

Two different routes may share nodes.

Definition Let x and y be two routes with $a_1 a_2 \dots a_r$ and $b_1 b_2 \dots b_r$ the nodes that appear in x and y in the orders they appear in x and

y. We define $c(x)$, $c(y)$ *lengths of the common sub-paths* of x and y to be $d(a_1..a_r)$ and $d(b_1..b_r)$ the distance between a_1 and a_r along x's path, and between b_1 and b_r along y's path.

A BGP route $a_1..a_k$ consists of the sequence of edges $e_{12}e_{23}..e_{(k-1)k}$ and a sequence of edge-types $t_{12}t_{23}..t_{(k-1)k}$. We call the sequence of edge-types, compactly represented as a string of symbols from $R = \{c, p, r\}$, the traversal-pattern of the route.

Definition The traversal-pattern of the route $a_1 \ldots a_k$ is $t(a_1 \ldots a_k) = t_{12}t_{23} \ldots t_{(k-1)k}$ and is represented by a string of $k-1$ symbols from $R = \{c, p, r\}$.

For example if the route $a_1..a_6$ consists of two customer-provider traversals, a peer-peer traversal, followed by two provider-customer traversals indicated as $\overline{a_1a_2}, \overline{a_2a_3}, \overline{a_3a_4}, \overline{a_4a_5}, \overline{a_5a_6}$, its traversal-pattern $t(a_1..a_6) = pprcc$.

Regular-expressions are a compact way of representing sets of symbol-patterns, and therefore we will use them to indicate sets of traversal-patterns. We will use the following two well-known regular-expression symbols:

Definition We define *m** to indicate 0 or more repetitions of the symbol m and *m?* to indicate 0 or 1 repetitions of the symbol m.

For example *p** indicates the set $\{\phi, p, pp, ppp, \ldots\}$ while *c*r?* indicates the set $\{\phi, c, cc, ccc, \ldots\} \cup \{r, cr, ccr, \ldots\}$, and so on.

B. Routing-algorithms

When a node receives multiple routes to another node from different neighbors, it chooses its best route based on its *route-selection* algorithm. It then exports some of its routes to each of its neighbors based on its relationship to its neighbor and its *export-policy* as outlined in Sec. III. We first define the export-policies.

1) Export Policies: Nearly all ASes uses the valley-free export policy as explained in Sec. II. This export-policy is called valley-free because if all nodes in a route adhere to this policy, the resulting route consists of a sequence of customer-provider traversals to a peak which may consist of zero or one peer-peer traversals, followed by a sequence of provider-customer traversals. Formally:

Definition A route $a_1..a_n$ is *valley-free* route iff its traversal-pattern $T(a_1..a_n)$ can be represented by the regular-expression *c*r?p**.

For example $cccrppp, ccc, cp$ and cr are valid expressions of the regular-expression *c*r?p**, while $ccrrpp, pc, prc, cpcp$ and $pppc$ are not.

Definition Nodes $\{a_i, a_{i+1} | t(a_i, a_{i+1}) = r\}$ and $\{a_j | t(a_{j-1}, a_j, a_{j+1}) = cp\}$ in a BGP route are defined to be *peak* nodes.

That is, in any route, the nodes on either side of a peer edge and the common node when a customer-provider edge is followed by a provider-customer edge are known as peak nodes. Furthermore, a route may have one or two peaks and if it does have two peaks, they are adjacent and joined by a peer-peer edge.

All valley-free routes $a_1..a_n$ fall into one of three categories:

- *routes with peaks* indicated as $\overline{a_1..a_n}$ with traversal patterns *c*rp** or *c*cpp**
- *customer-provider routes* indicated as $\overline{a_1..a_n}$ with traversal-pattern *c**
- *provider-customer routes* indicated as $\overline{a_1..a_n}$ with traversal-pattern *p**

That all valley-free routes fall into the above three categories can be seen when we expand the traversal-pattern *c*r?p** of valley-free routes into its components. {*c*r?p**} can be written as {*c*p** ∪ *c*rp**} which in turn can be expanded to {*c** ∪ *p** ∪ *c*cpp** ∪ *c*rp** }. The first two terms of this expanded term are the traversal-patterns of customer-provider and provider-customer routes, while the last two terms are the traversal-pattern of routes with peaks as defined above.

Note that routes that involve a provider-customer edge followed by a customer-provider edge violate the valley-free property and do not allow any traffic to pass through them (since the valley-free export policy prevents provider-provider traffic to pass through customers). We say that such routes *block* traffic.

2) Route-Selection Algorithms: Before a node exports routes, it needs to decide between alternative-routes to a single destination. It uses its route-selection algorithm to do so. Even though each node in a BGP network may employ different route-selection algorithms, as stated in Sec. II, nearly all nodes use one of two route-selection algorithms: the shortest-path algorithm or the prefer-customer algorithm. Sometimes ASes choose the shortest-route to a destination, irrespective of its relationship to the neighbor who reported the route. This is called the *shortest-path* algorithm. However when an AS x has two routes available to a descendant AS y, the first through a customer AS, and the second through a parent or peer AS, it will often take the first route through a customer AS even if it is longer. In such a case routes between the two nodes x and y may not be symmetric, since AS y may choose the shorter route to AS x involving the peer (as y's neighbor on the route x chose will be its provider according to the valley-free property). This is called a *prefer-customer* algorithm.

Definition When a node encounters multiple-routes to a destination, it is said to employ the *shortest-route* route-selection algorithm if it always chooses the route with the shortest length.

Definition If on encountering multiple-routes to a destination, a node always prefers the shortest-route through a customer over a shorter route through a provider or a peer, it is said to employ a *prefer-customer* route-selection algorithm.

3) Routing-algorithm: In Sec. III we had defined a routing-algorithm to consist of a route-selection algorithm and an export policy. We will define two such routing-algorithms. The first f_{sp} called the shortest-path routing algorithms is the valley-free export policy combined with the shortest-path route-selection algorithm, while the second f_{pc} called the prefer-customer routing algorithm is the valley-free export policy combined with the prefer-customer route-selection algorithm.

4) Local-Policy: We define the local-policy L called the standard (or prefer-customer) local policy to consist of the two routing-algorithms $L = \{f_{sp}, f_{pc}\}$. Additionally we define the shortest-path local-policy to be $L_{sp} = \{f_{sp}\}$.

When nodes adhere to the standard local-policy or the shortest-path local-policy the valid routes that can be taken from one AS to another become constrained. This allows us to formulate rules about the traversal-patterns of BGP Routes and look for violations of these rules. We study these rules in the next section.

V. Route Analysis

Now that we have defined a model of ASes running the BGP protocol, we can describe the property we will use to identify deviations from the standard local-policy inside the network. We proceed by stating this rule and illustrating it. The proof is provided in Subsection V-C.

A. Consistency of local-policy

As mentioned in Sec. III since BGP gives each node complete freedom in choosing its routing-algorithm there is no "official"

violation that can be checked for. However given a local-policy we can check for consistency with respect to this policy. Therefore we will assume that a node uses the standard local-policy $L = \{f_{sp}, f_{pc}\}$ constructed in Sec. IV and we focus on techniques that check for inconsistencies against this policy.

We now show that the standard local-policy L imposes a strong property on the routes reported by two different nodes, and violations of this rule implies inconsistency. More specifically, this property requires that if a node reports a route with ASes in a certain order, another route can *only* contain these ASes in very specific orders.

The property involves an analysis of two routes reported to a node, n, from two different neighbors, x and y. Let $x_1 \ldots x_s$ be the route reported by node x and let $y_1 \ldots y_t$ be the route reported by y. We first extract the nodes that these two routes have in common. Let $a_1 a_2 \ldots a_r$ be the nodes in x's reported route that also appear in y's route in the order in which they appear in x's route, i.e., if $a_i = x_j$, then $a_{i+1} = x_{j+k}$ for $k > 0$. Note that a_i and a_{i+1} need not be neighbors in the original AS graph since it is possible that y's route included both of these nodes but did not include the nodes on x's route that connects them. The nodes $a_1 a_2 \ldots a_r$ also appear in the route reported by y, but possibly in a different order. The property we observe limits the order in which these nodes can appear within the route reported by y, which we order as $b_1, b_2, \ldots b_r$, where each $b_i = a_j$ for some j.

We start by defininining a notion called the common sub-path:

Definition Let x and y be two routes with $a_1 a_2 \ldots a_r$ and $b_1 b_2 \ldots b_r$ the common nodes in the orders they appear in x and y. We define $c(x)$, $c(y)$, the *lengths of the common sub-paths* of x and y to be $d(a_1..a_r)$ and $d(b_1..b_r)$ which are the distances between a_1 and a_r along x's path, and between b_1 and b_r along y's path.

Note these distances need not be equal, as the lengths depend on the sub-paths that include the nodes that are not common to both paths that are traversed. In the following theorem, without loss of generality, we choose x and y such that the length of the common sub-path of x is no shorter than the length of the common sub-path of y.

Theorem 5.1: In a network with the standard prefer-customer local-policy, let node n have two neighbors, x and y, each reporting a route and let S be the intersection of nodes that appear in both routes. If the nodes from S appear in the order $a_1, a_2, \ldots, a_r$ in the route reported by x, then the route reported by y must have these nodes appear in one of the three following orders:

- same: $a_1, a_2, \ldots, a_r$
- reversed: $a_r, a_{r-1}, \ldots, a_1$
- decrease-increase: $a_r, a_{r-1}, \ldots, a_{r-k}, a_1, a_2, \ldots, a_{r-k-1}$, $0 \leq k < r-1$.

Furthermore, the decrease-increase sequence is possible only when the length of the common sub-path of x is strictly longer than the length of the common sub-path of y.

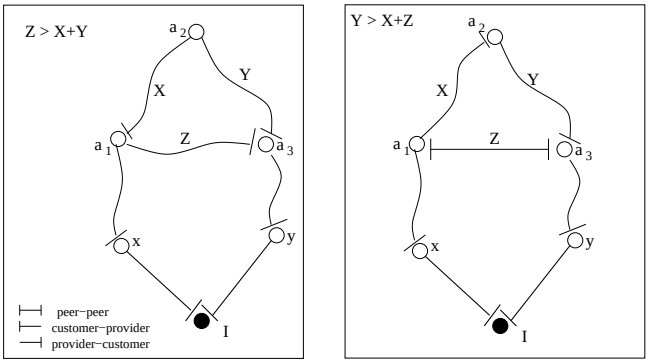
Please refer to Subsection V-C for the proof.

The above theorem says something very interesting about routes in BGP networks with a prefer-customer (or standard) local-policy. It says that if a route contains a set of ASes in a certain order and we number these ASes in increasing order, then another shorter or same-length route route can never contain the same ASes in an order that, under the numbering imposed above, first increases and then decreases. For example, if two routes contain three ASes in common, and we number the ASes such that the route that contains the longer (or equal) common sub-path reports an order $1, 2, 3$, then any other route must have these nodes appear within its common sub-path in the same order $1, 2, 3$, the inverse order $3, 2, 1$ or, when the common sub-paths have different lengths, the decrease-increase order $3, 1, 2$. If the common sub-path has any of the other orders $2, 1, 3$ or $1, 3, 2$ or $2, 3, 1$, or, when the common sub-paths have the same length, $3, 1, 2$ we can conclude that some other node in the network is running a local-policy that is non-standard.

We now illustrate the theorem with some examples where routes have the nodes a_1, a_2, a_3 in common and the first route reported by some node x has them in the order a_1, a_2, a_3. Consider Fig. 2 where three nodes a_1, a_2, a_3 have routes between each other with distances X, Y, Z as pictured. The curved lines in the illustration indicate routes possibly containing multiple edges, while the straight lines indicate single edges. Nodes x and y are the neighbors of the node of interest I and have paths to a_1 and a_2.

A situation where nodes use the shortest-path route-selection algorithm f_{sp} resulting in the inverse order a_3, a_2, a_1 being reported in a route by y is shown in Fig. 2(A). In this scenario node x reports the route a_1, a_2, a_3 to a_3 despite the route $a_1..a_3$ being available since the former is shorter ($Z > X + Y$ as indicated in the figure). Node y in turn reports the inverse route a_3, a_2, a_1 to I, since it is a shorter route than the alternative route $a_3..a_1$ to a_1.

A second situation where nodes use the standard prefer-customer route-selection algorithm f_{pc} resulting in the decrease-increase order a_3, a_1, a_2 being reported by the node y is shown in Fig. 2(B). Here x reports the route a_1, a_2, a_3 despite $a_1..a_3$ being available because the former is a customer-route while the latter is a peer-route, and customer-routes are preferred. Node y, however reports the route a_3, a_1, a_2, since this route is shorter ($X + Z < Y$) than its other route a_3, a_2 to a_2 and it prefers a peer route over a provider-route.



(A) shortest-route (B) prefer-customer

Fig. 2. Valid Node Orderings

B. Validity Checking

Theorem 5.1 gives us a powerful tool with which a node can look for inconsistencies to its local-policy. In Fig. 3, we provide a simple algorithm that can check if two routes with differing common sub-path lengths are are consistent according to Theorem 5.1. The algorithm has a time complexity of $O(l^2)$ where l is the length of the routes. A node with n routes can check the consistency of it states in $O(n^2 \times l^2)$ time. Our experiments show that only a small fraction of routes in practice have the 3 or more nodes in common needed to demonstrate validity or violation of the Theorem, and since such a check for candidacy of a route can be performed in $O(l)$ time, a node can check, in practice, the consistency of its state in $O(n^2 \times l)$ time. If a node runs the algorithm of Fig. 3 on its state, and gets reports of inconsistencies according to Theorem 5.1 it can conclude that some node or nodes do not use the standard local-policy.

- **For** each pair of routes **do**
  - Let $a_1..a_n$ and $b_1..b_n$ be the common ASes in both routes with $c(a_1..a_n) \geq c(b_1..b_n)$.
  - **For** $(i = 2, i < n, i++)$ **do**
    - Find k such that $b_{i-1} = a_k$
    - **If** $b_i = a_{k+1}$ and $b_{i+1} = a_{k+1}$ **OK**
    - **If** $b_i = a_{k-1}$ and $b_{i+1} = a_{k-2}$ **OK**
    - **If** $(c(a_1..a_n) > c(b_1..b_n))$
      - **If** $b_i = a_1$ and $b_{i+1} = a_2$ **OK**
      - **If** $b_i = a_{k-1}$ and $b_{i+1} = a_1$ **OK**
    - **else report Inconsistency**
  - **done**
- **done**

Fig. 3. Correctness Checking Algorithm

C. The Proof

We now provide the proof to Theorem 5.1. We start by proving prove some preliminary results that we will use to derive the main property. We start with a lemma about a route that has three nodes a_1, a_2, a_3 in that order. We list the conditions under which that route is preferred over a second route involving only a_1 and a_3.

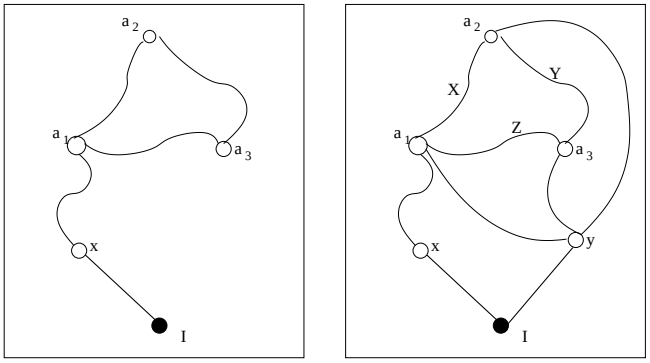


Fig. 4. Routes through 3 nodes

Lemma 5.2: Suppose a node a_1 implements the standard *prefer-customer* local-policy atop a network in which it has two paths to node a_3. Along the first path, we distinguish an intermediate node a_2 that lies along the sub-path, and along the second path, we do not distinguish any intermediate nodes, such that the two paths contain the respective ordered subsequences of nodes $a_1..a_2..a_3$ and $a_1..a_3$. If a_1 selects a route with the sub-route $a_1..a_2..a_3$, rather than $a_1..a_3$, where $d(a_1..a_3) = Z$, $d(a_1..a_2) = X$ and $d(a_2..a_3) = Y$, then:

- **Either** $X + Y \leq Z$ and the traversal-patterns is one of the following nine cases:

1	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
2	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
3	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
4	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
5	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
6	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
7	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
8	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
9	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$

- **Or** the traversal pattern is the following case:

10	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$

Proof: Consider the situation pictured in Fig. 4(A). The graph represents a node I that has a neighbor x. There are three nodes a_1, a_2 and a_3 with routes between them and distances as pictured: $d(a_1..a_2) = X$, $d(a_2..a_3) = Y$ and $d(a_3..a_1) = Z$. The traversal-patterns of the routes between the nodes x, y, a_1, a_2, a_3 is initially not known.

If neighbor x reports a route to a_3 as $x..a_1..a_2..a_3$ rather than $x..a_1..a_3$, two factors can influence the decision: the traversal-patterns $t(a_1..a_2), t(a_2..a_3), t(a_3..a_1)$ and the lengths $d(a_1..a_2), d(a_2..a_3), d(a_3..a_1)$ of the routes between a_1, a_2, a_3. Each of these three routes between a_1, a_2 and a_3 can take on the three different traversal patterns $\overline{a_m..a_n}, \overline{a_m..a_n}, \overline{a_m..a_n}$, according to Sec. IV, resulting in a total of 27 possible assignments.

Of these 27, the following assignments need not be considered either because they invalidate the route $a_1..a_2..a_3$ or they are covered by other cases.

- **Cycles (2)** *Two* possible assignments $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_3..a_1}$ and $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_3..a_1}$ are cycles in the clockwise and anti-clockwise directions and not allowed according to the cycle-free property of Sec. IV.
- **Invalid (3)** *Three* assignments involve $\overline{a_1..a_2}, \overline{a_2..a_3}$ which are invalid because they would imply that the route $a_1..a_2..a_3$ contains multiple-peaks which is not allowed under the valley-free export policy.
- **Blocked (9)** *Nine* assignments involve the traversals $\overline{a_1..a_2}, \overline{a_2..a_3}$, $\overline{a_1..a_2}, \overline{a_2..a_3}$ and $\overline{a_1..a_2}, \overline{a_2..a_3}$ which need not be considered as they are blocked at a_2 because of the valley-free property.
- **Impossible (3)** The *three* assignment $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$, $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$, $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$ will never result in x reporting the route $a_1..a_2..a_3$ because, irrespective of route-lengths, x will always prefer the customer route $a_1..a_3$.

Therefore of the 27 possible assignments, 17 need not be considered, leaving ten possible cases. Of these ten, in the 9 cases

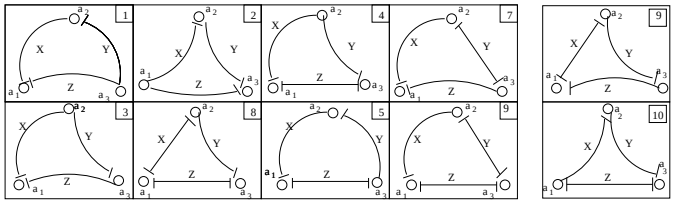
1	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
2	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
3	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
4	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
5	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
6	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
7	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
8	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
9	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$

both $a_1..a_2..a_3$ and $a_1..a_3$ are viable routes to a_3 for x. However according to the standard local-policy x will prefer $a_1..a_2..a_3$ only if $X+Y \leq Z$. In the remaining tenth assignment $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$, x will always prefer the route $a_1..a_2..a_3$ over $a_1..a_3$ irrespective of X, Y and Z as $a_1..a_2..a_3$ is a customer route and $a_1..a_3$ is not, and according to the standard local-policy customer-routes are preferred. ■

These ten possible assignments are pictured in Fig. 5 where straight-lines indicate peer-peer routes while concave and convex curves indicate provider-customer and customer-provider routes in the clockwise direction. Fig. 5(B) shows case ten where x will always prefer the route $a_1..a_2..a_3$ over $a_1..a_3$ irrespective of X, Y and Z as $a_1..a_2..a_3$ is an customer route and $a_1..a_3$ is not. In all nine other cases $a_1..a_2..a_3$ and $a_1..a_3$ are viable routes, but x will prefer $a_1..a_2..a_3$ only if $X + Y \leq Z$.

Lemma 5.3: If a neighbor x reports a route $a_1..a_2..a_3$ in a BGP network with a *prefer-customer* local-policy, then the only other valid reports involving a_1, a_2, a_3 from another neighbor y with a common sub-path whose length is no more than that belonging to x are either the same route, $a_1..a_2..a_3$, the reverse route $a_3..a_2..a_1$ and the routes $a_3..a_1..a_2$ and $a_2..a_3..a_1$.

Proof: Consider the scenario where y has a route to node $a_i, i = 1, 2, 3$ that does not pass through the other two nodes $a_j \neq a_k$,



(A) (B)

Fig. 5. Valid Labellings

$i \neq k, i \neq j$. The three sub-paths that connect y that would form each such path is depicted in Fig. 5(B). This sub-path can take on one of the three valley-free traversal-patterns $\overline{y..a_i}$,$\overline{y..a_i}$ and $\overline{y..a_i}$ from Sec. IV. Of these the two traversal-patterns $\overline{y..a_i}$ and $\overline{y..a_i}$ always end with a provider-customer edge. Since the type of the last edge determines whether routes are possible or blocked, we need only consider one of these two cases, so we only consider $\overline{y..a}$. These two traversal patterns $\overline{y..a_i}$ and $\overline{y..a_i}$ imply six possible traversal assignments for the three routes $y..a_1$, $y..a_2$ and $y..a_3$.

For each of these six cases, we consider the ten instances of Lemma 5.2 where it was possible for x to report the node sequence a_1, a_2, a_3. Of the resulting sixty possibilities we eliminate the routes that are not possible or less preferred. The remaining routes are the valid routes that y can report.

We consider these six cases for the ten instances of Lemma 5.2, which are tabulated in Fig. 6. In each of the resulting 60 cases the clockwise route and the anti-clockwise route involving all three nodes a_1, a_2, a_3 are considered in turn. The tables indicate if these routes are possible, and when both the clockwise and anti-clockwise routes are possible it indicates which one is preferred and why.

It can be seen that in all the 60 possible cases, only $a_1..a_2..a_3$, $a_3..a_2..a_1$, $a_3..a_1..a_2$ and $a_2..a_3..a_1$ are routes that y can possibly report. Moreover $a_3..a_1..a_2$ is possible only if $X + Z \leq Y$. ■

Remark It is interesting to note that of the 60 cases, the same order is allowed much more frequently than the reverse order, which is allowed with much greater frequency than the other two orders. Of the 60 cases considered, 16 allow the same order to reappear, 12 allow the reversed order to appear, and only 2 cases allows other orders to appear. This tells us that in practice we should often see the same-order, less frequently see the inverse order, and rarely see the other orders.

Lemma 5.4: In a BGP network with a prefer-customer local policy, if node x reports a route $a_1..a_2..a_3$ and another node y reports $a_3..a_1..a_2$, then $d(a_1..a_2..a_3) > d(a_3..a_2..a_1)$.

Proof: The two situations where this can occur are pictured in Fig. 5. The first from the proof of Lemma 5.3 is Case 10 with $y..a_3$. From the proof $X + Z \leq Y$, and $X + Z < Y + X$. Therefore $d(a_1..a_2..a_3) > d(a_3..a_1..a_2)$.

The second situation is not so obvious. It happens in Case 9 with $y..a_2$ if we relabel the nodes thus: $a_1 \rightarrow a_3$, $a_2 \rightarrow a_1$ and $a_3 \rightarrow a_2$. Then y reports $a_1..a_2..a_3$ and x reports $a_3..a_1..a_2$. In this case, from Lemma 5.2 $X + Y \leq Z$, or $X + Y < Z + Y$ or $d(a_1..a_2..a_3) < d(a_2..a_3..a_1)$. After relabeling $d(a_1..a_2..a_3) > d(a_3..a_1..a_2)$. ■

Remark The importance of the above lemma is that, when comparing two paths, if we ensure that the first path is the longer of the two, then the sequence $2, 3, 1$ cannot occur in the second path.

Theorem 5.5: Let two neighbor nodes x and y, in a network with the standard prefer-customer local-policy, report routes whose intersections contain the set of nodes $a_1, \ldots, a_r$ where the nodes appear in order $a_1 a_2 \ldots a_r$ in x's reported route, where the length of the common sub-path of x is no less than that of y. Then in the route reported by y, these nodes must appear in the same order, the reverse order, or in an order satisfying $a_r a_{r-1}, \ldots a_{r-k} a_1 a_2 \ldots a_{r-k-1}, 0 \leq k < r-1$.

Proof: We begin by pointing out that all our results above hold for any common set of nodes that lie on the paths obtained from x and y, i.e., we need not consider all nodes that lie on these common paths. This has the effect that some common nodes may appear on various sub-paths, but having such occurrences does not violate the above results.

Let us begin by assuming that the length of the common sub-path of x is longer than that of y's ($c(x) > c(y)$). For clarity, let us replace a_i by i, so that x's report can be written as the ordered sequence $1, 2, \ldots, r$, and let y's reported route be written as $b_1, b_2, \ldots, b_r$ where each $b_i \in \{1, 2, \ldots, r\}$ and $\cup_i \{b_i\} = \{1, 2, \ldots, r\}$. If we choose any three nodes in the original route, $i < j < \ell$, by Lemma 5.3, the order in which they must appear in the second route is either i, j, ℓ or ℓ, j, i or ℓ, i, j. Note that ℓ never appears as the middle entry in the allowable 3-node sequences. It follows that any three adjacent nodes in the second route, b_{m-1}, b_m, b_{m+1} cannot satisfy $b_m > b_{m-1}$ and $b_m > b_{m+1}$. To see this, if this were not the case, we would have $\ell = b_m$, placing ℓ is the middle node of the 3-node sequence. Hence, no 3-node subsequence can have the middle node of the subsequence being the largest. This means that a valid sequence of r nodes can only strictly increase, strictly decrease, or contain a strictly decreasing sequence followed by a strictly increasing sequence.

The unique strictly increasing sequence (namely, $1, 2, \ldots, r$) and a unique strictly decreasing sequence (namely $r, r-1, \ldots, 1$) are both permitted by the theorem. To limit the set of sequences that contain a strict decrease followed by a strict increase, consider the position in such a sequence of the node assigned identifier 1. The fact that ℓ, i, j is the only ordering permitted that is not strictly increasing or strictly decreasing, where $i < j < \ell$ implies that any node that appears before 1 in the second route must be larger than any node that appears after 1 in the second route. The only sequences that have a strict decrease followed by a strict increase and satisfy this property are those of the form $a_r a_{r-1}, \ldots a_{r-k} a_1 a_2 \ldots a_{r-k-1}, 0 <= k <= r-1$. Finally, note that if the length of the common sub-path of x is not longer than that of y, then the path-length condition is violated when considering the sub-path consisting of nodes $a_1, a_r = b_1,$ and b_r.

When the common sub-path of x has the same length as that of y ($c(x) = c(y)$), since any sub-ordering must contain nodes in the same or reverse orders, it follows that only the increasing sequence and decreasing sequence are possible orderings for nodes along y's common sub-path. ■

It is important to emphasize that the results above require common nodes are numbered with respect to their ordering in the longer of the two common sub-paths.

If ASes do not set their local-preference attribute they use, as mentioned in Sec. IV, a shortest-path routing algorithm, and what we call the shortest-path local policy. In such a case, when a node reports a route with ASes in a certain order, another route can only report routes with these ASes in the same order or the inverse order. Specifically the decrease-increase order allowed in the prefer-customer local policy is not allowed. We state this property as formally as a theorem below. The proof proceeds similarly to the proof for Theorem 5.5 and can be found in [22].

Theorem 5.6: Let two neighbor nodes x and y, in a network with the shortest-path local-policy, report routes whose intersections contain the set of nodes $a_1, \ldots, a_r$ where the nodes appear in order $a_1 a_2 \ldots a_r$ in x's reported route. Then in the route reported by

$\overline{y..a_1}$	$a_1..a_2..a_3$	$a_1..a_3..a_2$
1	*Possible*	blocked at a_1
2	*Possible*	blocked at a_3
3	*Possible*	blocked at a_1
4	*Possible*	blocked at a_1
5	*Possible*	blocked at a_1
6	blocked at a_1	blocked at a_1
7	blocked at a_1	blocked at a_1
8	blocked at a_1	blocked at a_1
9	blocked at a_1	blocked at a_1
10	*Possible*	blocked at a_1
$\overline{y..a_1}$	$a_1..a_2..a_3$	$a_1..a_3..a_2$
1	*Shorter*	Longer
2	*Possible*	blocked at a_3
3	*Shorter*	Longer
4	*Possible*	blocked at a_3
5	*Shorter*	Longer
6	*Shorter*	Longer
7	*Shorter*	Longer
8	*Shorter*	Longer
9	*Shorter*	Longer
10	*Possible*	blocked at a_3

$\overline{y..a_2}$	$a_2..a_3..a_1$	$a_2..a_1..a_3$
1	blocked at a_2	blocked at a_1
2	blocked at a_3	blocked at a_2
3	$a_2..a_1$ shorter	blocked at a_1
4	blocked at a_3	blocked at a_1
5	blocked at a_2	blocked at a_1
6	blocked at a_2	blocked at a_1
7	blocked at a_2	blocked at a_1
8	blocked at a_3	blocked at a_1
9	*Possible*	blocked at a_2
10	blocked at a_3	blocked at a_2
$\overline{y..a_2}$	$a_2..a_3..a_1$	$a_2..a_1..a_3$
1	$a_2..a_1$ shorter	blocked at a_1
2	blocked at a_3	blocked at a_2
3	$a_2..a_1$ shorter	blocked at a_1
4	blocked at a_3	blocked at a_1
5	blocked at a_2	blocked at a_1
6	blocked at a_2	blocked at a_2
7	$a_1..a_2$ shorter	blocked at a_1
8	blocked at a_3	blocked at a_2
9	$a_1..a_2$ shorter	blocked at a_1
10	blocked at a_3	blocked at a_2

$\overline{y..a_3}$	$a_3..a_1..a_2$	$a_3..a_2..a_1$
1	Longer	*Shorter*
2	blocked at a_3	blocked at a_3
3	blocked at a_1	blocked at a_3
4	blocked at a_3	blocked at a_3
5	blocked at a_3	*Possible*
6	blocked at a_3	blocked at a_3
7	blocked at a_1	blocked at a_3
8	blocked at a_3	blocked at a_3
9	blocked at a_1	blocked at a_3
10	blocked at a_3	blocked at a_3
$\overline{y..a_3}$	$a_3..a_1..a_2$	$a_3..a_2..a_1$
1	blocked at a_1	*Possible*
2	Longer	*Shorter*
3	blocked at a_1	*Possible*
4	Longer	*Shorter*
5	Longer	*Shorter*
6	blocked at a_1	*Possible*
7	blocked at a_1	*Possible*
8	blocked at a_1	*Possible*
9	blocked at a_1	*Possible*
10	*Possible* if $X+Z \leq Y$	*Possible* if $X+Y \leq Z$

Fig. 6. 60 cases of Lemma 5.3

y, these nodes must appear as one of the r sequences satisfying $a_k a_{k-1} \ldots a_1 a_{k+1}, a_{k+2} \ldots a_r, 0 < k \leq r$.

For the proof, please refer to Appendix X.

VI. Applications

A. The AS7007 Problem

Over the years, there have been several instances where a small set of misconfigured nodes in a remote region of the BGP network dramatically shifted the routing throughout the entire network. The best known among these happened, as mentioned in Sec. I, when a small AS, AS7007, through a misconfiguration, claimed unit-length routes to a large fraction of all prefixes (ASes) [12]. This caused many nodes to reroute traffic to their destinations through AS7007, which unable to handle the large volume of traffic routed through it, dropped large amounts of data, essentially shutting down a significant fraction of the Internet.

Here we show how our analysis would have detected this problem. Consider Fig. 7 where seven ASes including AS7007 and six to whom AS7007 claimed unit-length paths are shown.

Consider the reports of BGP routes from nodes 1 and 2 to node I when AS7007 and the other nodes are behaving according to standard BGP policy. AS 1 would report the path $1, 5, 6, 2, 3$ to 3 since it is a customer-path. AS 2 would report the path $2, 6, 5, 1$ to AS 1 since its other path is again blocked at AS7007. Node I would notice that these two routes have the ASes $1, 2, 5$ and 6 in common and appear in the inverse orders $1, 5, 6, 2$ and $2, 6, 5, 1$. If it were using Theorem 5.1 to check paths, this pair of routes would have passed, as they are in an inverse orders.

Now consider the situation when AS7007 was misconfigured and reported unit-length paths. AS 1 would continue to report the route $1, 5, 6, 2, 3$ to AS 3, despite AS7007 claiming that it had a unit-length path to AS 3 because AS 1 will prefer the customer route through AS 5 over the provider route though AS 4. However, now AS 2 will prefer the customer-route to AS 1 through AS 3 over the provider-route through 6, so will report the route $2, 3, 7007, 1$ to I. If node I had been using using Theorem 5.1 to check paths, it would notice that these two routes had the ASes $1, 2$ and 3 in common, $d(1..5..6..2..3) > d(2..3..7007..1)$ and these ASes appear in the orders $1, 2, 3$ and $2, 3, 1$ in the two routes. However Theorem 5.1 does not allow the second sequence to be $2, 3, 1$, when the longer sequence is labeled $1, 2, 3$ so node I would have realized that some node in its routes was violating standard BGP policy and could have taken appropriate action!

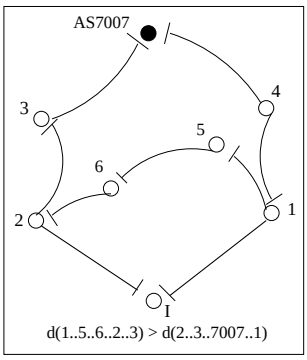


Fig. 7. The AS7007 Problem

VII. Experimental Results

In this section we present the results of experiments where we use the rule of Theorem 5.1 to test for inconsistencies to the standard prefer-customer local policy.

A. The Experimental Setup

We collected BGP route data from six different ASes: two tier-1 ASes AT&T and Global Crossing, two tier-2 ASes Optus and GT Group, and two tier-3 ASes Columbia University and EUNET Finland in and around July 2004 [24]. This data is available from [23]. We eliminated duplicate routes that result when different prefixes at the same AS each gets a route. A list of the ASes, the number of unique routes found at each site along with a count of the number of unique ASes in these routes is shown in Table II. The fanout indicated in the table is the out-degree of the AS and is an indication of the tier it belongs to; ASes closer to the root typically have larger fan-outs. In addition we collected data from the Oregon Route-View Project which peers with approximately 60 different ASes and makes the routes of these ASes available at [14].

We also created composite data-sets by combining routes from different ASes. We combined the routes from ASes at the same tier-level to create three data-sets corresponding to tier-1, tier-2 and tier-3. In addition we created a superset of routes by combining all routes from all our sources. The number of unique routes in these data-sets are given in Table III

AS id	Tier	Institution	Fanout	ASes	Routes
AS7018	1	AT&T	1,330	20,103	22,528
AS3549	1	Global Crossing	558	22,843	24,085
AS7474	2	Optus Australia	114	15,768	26,828
AS6539	2	GT Group	157	13,426	17,992
AS14	3	Columbia Univ.	3	29,392	48,224
AS6667	3	EUNet Finland	26	24,538	25,578
-	-	Oregon	-	8,643	106,563

TABLE II
DATA FROM ASES

Data-Set	ASes	Routes
2 Tier-1	27,207	45,925
2 Tier-2	16,832	44,820
2 Tier-3	32,055	73,548
All	-	305,422

TABLE III
DATA-SETS

B. Violation of Standard Policy

We first used our route-data to test for inconsistencies with regard to the standard (or prefer-customer) local-policy. We looked for pairs of routes which had three or more ASes in common, and applied Theorem 5.1 to it. The number of ASes involved in the violating routes, the number of pairs which failed the tests and the total number of pairs that were compared (and had three or more ASes in common) are shown in Table IV.

Data-Set	Violating ASes (%)	Violating pairs	Total Pairs
AT&T	44 (1.1)	196	417,874
Global	25 (1.4)	152	26,472
Optus	102 (1.9)	1,281	52,143,491
GT	10 (0.5)	21	40,761
CU	129 (1.7)	1,970	1,557,527
EuNet	39 (1.6)	278	138,243
Oregon	155 (5.8)	2,074	8,616,625
Tier-1	53 (0.7)	651	486,550
Tier-2	142 (2.3)	1,848	52,179,280
Tier-3	191 (1.9)	4,538	1,676,065

TABLE IV
VIOLATIONS

What is interesting to note is that there were violations in all the data-sets, implying that there are significant number of ASes who do not follow the standard local-policy of valley-free export combined with prefer-customer route selection. It is also interesting to note that nearly all the ASes show that a consistent fraction of the ASes are involved in the violations. This percentage varies from 0.5% at GT to 1.9% at Optus with a mean of 1.4% and standard-deviation less than 0.2. The composite Oregon site shows a larger fraction of violations. There is also no clear correlation between the number of unique paths reported and the number of violations. Five of the ASes reported about 25,000 unique routes while Columbia reported about 50,000, but the number of violations vary widely from 21 for GT to 1,970 for Columbia.

As the routes from ASes are combined, a proportionately larger number of violations are detected. This is shown in Table V where the first column indicates the number of violations detected when the algorithm is run on the combined data-sets, and the second column is the sum of the violations detected when the algorithm is run on the individual data-sets. It can be seen that there is a clear increase in the violations detected for all three tiers when the routes both ASes in the tier are combined, relative to when the detection is carried out separately. This is even more apparent when all the data-sets are combined into a single large data-set.

Data-Set	Combined detections	Sum of Violations
Tier-1	651	352
Tier-2	1,848	1,302
Tier-3	4,538	2,254
All	82,362	3,898

TABLE V
VIOLATIONS

Data-Set	same	inverse	decrease-increase
AT&T	417,678	0	0
Global	26,316	4	0
Optus	52,096,427	45,781	2
GT	40,740	0	0
CU	1,483,669	71,850	39
EuNet	137,946	19	0
Oregon	8,376,777	237,758	27
Tier-1	486,560	65	0
Tier-1	52,179,280	61,876	4
Tier-1	1,676,065	73,071	100

TABLE VI
VALID PATTERN CLASSIFICATION

In Table VI we list the frequency of occurrence of the three valid patterns allowed according to Theorem 5.1 when two routes have three or more ASes in common. The first column indicates the frequency with which the same order was found in both routes, the second column indicate the number of comparisons where common ASes were in an inverse orders, and the third column indicates the number of occurrences where the common ASes were in a relative decrease-increase order. It can be seen that by far the most common occurrence in all data-sets is the same order. The inverse order occurs reasonably often while the decrease-increase order is rarely seen. The proof of Lemma 5.3 and the remark that follows the Lemma indicate that this is generally the pattern we should expect. However the decrease-increase order occurs even less often than we would expect from the proof and remark. Since studies [24] show that a large fraction of ASes do set their *local-preference* attribute which allows the decrease-increase order to occur, our data implies that the situation shown in Fig. 2(B) which actually forces such a pattern to occur rarely occurs.

VIII. FUTURE WORK

In future work we would like to address the following issues:

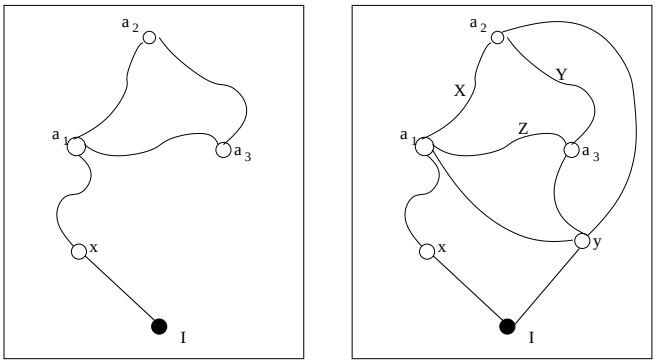
- We currently show the properties that standard BGP policy and the shortest-path BGP policy impose on routes. However there are other non-standard policies that are prevalent. We would like to tabulate all non-standard BGP policies and derive the corresponding properties they impose on an AS's state.
- We currently detect the presence of violations. If future work we would like to be able to identify the violators.
- We show how combining views of the BGP network enhances the ability to detect violations. We would like to develop techniques that specifically take advantage of multiple-views in both detecting and identifying violators.

IX. Conclusion

In this paper we proved that if ASes in a BGP network operate according to the standard BGP routing policy certain rules apply to pairs of routes. Specifically if a route contains a set of ASes in a certain order and we number these ASes in increasing order, then another route can never contain the same ASes in an order that first increases and then decreases. ASes can use this property to check if other ASes are indeed operating according to the standard BGP policy. Using this rule we showed, through experiments, that all viewed sites demonstrate violations of standard policy by ASes in the network. Additionally nearly all sites showed that more than 1% of tested nodes were involved in routes with violations. We further showed that combining views enhances the ability to detect violations.

X. Appendix

We provide the proof to Theorem 10.3. We start by proving prove some preliminary results that we will use to derive the main property. As with the proof of Theorem 5.1 we start with a lemma about a route that has three nodes a_1, a_2, a_3 in that order. We list the conditions under which that route is preferred over a second route involving only a_1 and a_3.



(A) (B)

Fig. 8. Routes through 3 nodes

Lemma 10.1: Suppose a node a_1 implements the *shortest-path* local-policy atop a network in which it has two paths to node a_3. Along the first path, we distinguish an intermediate node a_2 that lies along the sub-path, and along the second path, we do not distinguish any intermediate nodes, such that the two paths contain the respective ordered subsequences of nodes $a_1..a_2..a_3$ and $a_1..a_3$. If a_1 selects a route with the sub-route $a_1..a_2..a_3$, rather than $a_1..a_3$ then:

1	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
2	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
3	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
4	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
5	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
6	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
7	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
8	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
9	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
10	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
11	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
12	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
13	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$

Proof: Consider the situation pictured in Fig. 8(A). The graph represents a node I that has a neighbor x. There are three nodes a_1, a_2 and a_3 with routes between them and distances as pictured: $d(a_1..a_2) = X$, $d(a_2..a_3) = Y$ and $d(a_3..a_1) = Z$. The traversal-patterns of the routes between the nodes x, y, a_1, a_2, a_3 is initially not known.

If neighbor x reports a route to a_3 as $x..a_1..a_2..a_3$ rather than $x..a_1..a_3$, two factors can influence the decision: the traversal-patterns $t(a_1..a_2), t(a_2..a_3), t(a_3..a_1)$ and the lengths $d(a_1..a_2), d(a_2..a_3), d(a_3..a_1)$ of the routes between a_1, a_2, a_3. Each of these three routes between a_1, a_2 and a_3 can take on the three different traversal patterns $\overline{a_m..a_n}, \overline{a_m..a_n}, \overline{a_m..a_n}$, according to Sec. IV, resulting in a total of 27 possible assignments.

Of these 27, the following assignments need not be considered either because they invalidate the route $a_1..a_2..a_3$ or they are covered by other cases.

- **Cycles (2)** *Two* possible assignments $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_3..a_1}$ and $\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_3..a_1}$ are cycles in the clockwise and anti-clockwise directions and not allowed according to the cycle-free property of Sec. IV.
- **Invalid (3)** *Three* assignments involve $\overline{a_1..a_2}, \overline{a_2..a_3}$ which are invalid because they would imply that the route $a_1..a_2..a_3$ contains multiple-peaks which is not allowed under the valley-free export policy.
- **Blocked (9)** *Nine* assignments involve the traversals $\overline{a_1..a_2}, \overline{a_2..a_3}$, $\overline{a_1..a_2}, \overline{a_2..a_3}$ and $\overline{a_1..a_2}, \overline{a_2..a_3}$ which need not be considered as they are blocked at a_2 because of the valley-free property.

Therefore of the 27 possible assignments, 14 need not be considered, leaving 13 possible cases. In all 13 cases both $a_1..a_2..a_3$ and $a_1..a_3$ are viable routes to a_3 for x. However according to the shortest-path local-policy x will prefer $a_1..a_2..a_3$ only if $X+Y \leq Z$.

1	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
2	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
3	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
4	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
5	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
6	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
7	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
8	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
9	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
10	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
11	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
12	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$
13	$\overline{a_1..a_2}, \overline{a_2..a_3}, \overline{a_1..a_3}$

■

These thirteen possible assignments are pictured in Fig. 9 where straight-lines indicate peer-peer routes while concave and convex curves indicate provider-customer and customer-provider routes in the clockwise direction.

Lemma 10.2: If a neighbor x reports a route $a_1..a_2..a_3$ in a BGP network with a *shortest-path* local-policy, then the only other valid reports involving a_1, a_2, a_3 from another neighbor y are the same route, $a_1..a_2..a_3$ the reverse route $a_3..a_2..a_1$ or the decrease-increase route $a_2..a_1..a_3$.

Proof: Consider the scenario where y has a route to node $a_i, i = 1, 2, 3$ that does not pass through the other two nodes $a_j \neq a_k$, $i \neq k, i \neq j$. The three sub-paths that connect y that would form each such path is depicted in Fig. 5(B). This sub-path can take on one of the three valley-free traversal-patterns $\overline{y..a_i}, \overline{y..a_i}$ and $\overline{y..a_i}$ from Sec. IV. Of these the two traversal-patterns $\overline{y..a_i}$ and $\overline{y..a_i}$ always end with a provider-customer edge. Since the type of the last edge determines whether routes are possible or blocked, we need only consider one of these two cases, so we only consider $\overline{y..a}$. These two traversal patterns $\overline{y..a_i}$ and $\overline{y..a_i}$ imply six possible traversal

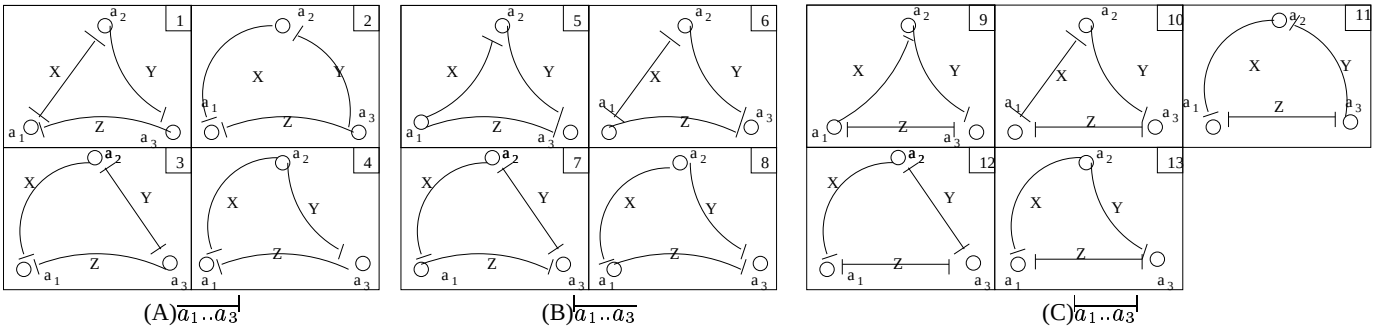


Fig. 9. Valid Labellings

assignments for the three routes $y..a_1$, $y..a_2$ and $y..a_3$.

For each of these six cases, we consider the 13 instances of Lemma 10.1 where it was possible for x to report the node sequence a_1, a_2, a_3. Of the resulting 78 possibilities we eliminate the routes that are not possible or less preferred. The remaining routes are the valid routes that y can report.

We consider these six cases for the 13 instances of Lemma 10.1, which are tabulated in Fig. 10. In each of the resulting 78 cases the clockwise route and the anti-clockwise route involving all three nodes a_1, a_2, a_3 are considered in turn. The tables indicate if these routes are possible, and when both the clockwise and anti-clockwise routes are possible it indicates which one is preferred and why.

It can be seen that in all the 78 possible cases, only $a_1..a_2..a_3$, $a_3..a_2..a_1$, $a_2..a_1..a_3$ are routes that y can possibly report. ■

Remark It is interesting to note that of the 78 cases, the same order and the reverse order are each allowed 15 times while the decrease-increase order is allowed only once. This tells us that in practice we should often see the same-order and the inverse order, and rarely see the decrease-increase order.

Theorem 10.3: Let two neighbor nodes x and y, in a network with the shortest-path local-policy, report routes whose intersections contain the set of nodes $a_1, \ldots, a_r$ where the nodes appear in order $a_1 a_2 \ldots a_r$ in x's reported route. Then in the route reported by y, these nodes must appear as one of the r sequences satisfying $a_k a_{k-1} \ldots a_1 a_{k+1}, a_{k+2} \ldots a_r, 0 < k \leq r$.

Proof: The proof is by recursion.

Let the claim be true for $r-1$ nodes and therefore y reports $a_k a_{k-1} \ldots a_1 a_{k+1}, a_{k+2} \ldots a_{r-1}, 0 < k \leq r-1$. Now consider valid sequences of length r that can be reported by y.

Construct r sequences of length r from the $r-1$ sequences of length $r-1$ as follows:

- Form $r-1$ sequences of length r by appending a_r to the *end* of the $r-1$ sequences.
- Form 1 sequence of length r by appending a_r to the *beginning* of the sequence $a_r a_{r-1} \ldots a_1$.

We next show that these indeed are the r valid sequences of length r. Sequences of length $r-1$ formed by removing a_r must also be valid. Therefore all sequences of length r must be formed by appending a_r to the $r-1$ valid sequences of length $r-1$.

There are three possibilities for adding a_r to the valid sequences of length $r-1$.

1) *In the middle* This is not possible since its neighbors and a_r would form the increase-decrease sequence $a_{r-p} a_r a_{r-q}, 0 < p, q < r-1, p \neq q$ which is not allowed by Lemma 10.1.
2) *In the end* This is always possible since any two predecessors and a_r would form either the increasing sequence $a_{r-q} a_{r-p} a_r, q > r$ or the decrease increase sequence $a_{r-q} a_{r-p} a_r, r > q$ both of which are allowed. Adding a_r to the end of $r-1$ sequences of length $r-1$ results in $r-1$ sequences of length r.
3) *At the beginning* It is only possible to place a_r at the beginning of the valid sequence of length $r-1$ $a_{r-1} a_{r-2} \ldots a_1$. Th is because a_r and any two successors form the sequence $a_r a_{r-p} a_{r-q}$ and this sequence is valid only if $q > r$ since decrease increase sequences of the form $a_r a_{r-p} a_{r-q}, p > q$ are not allowed. This produces 1 valid sequence of length r.

We now have r valid sequences of length r, and these are given by $a_k a_{k-1} \ldots a_1 a_{k+1}, a_{k+2} \ldots a_r, 0 < k \leq r$.

Consider the case when $r = 3$. Then according to Lemma 10.1 only $a_1 a_2 a_3$, $a_3 a_2 a_1$ and $a_2 a_1 a_3$ can be reported which satisfies our claim for $r = 3$.

We have shown that our claim holds for a length of 3, and if we assume it is true for lengths of $r-1$ we show that it is true for r. Therefore, by recursion, it is true for all r. ■

References

[1] W. Aiello, J. Ioannidis, and P. McDaniel. Origin authentication in interdomain routing. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 165–178. ACM Press, 2003.

[2] D.-F. Chang, R. Govindan, and J. Heidemann. An empirical study of router response to large bgp routing table load. In *Proceedings of the second ACM SIGCOMM Workshop on Internet measurment workshop*, pages 203–208. ACM Press, 2002.

[3] J. Farrar. C&w routing instability. nanog mail archives. Available at http://www.merit.edu/mail.archives/nanog/2001-04/msg00209.html.

[4] N. Feamster, D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the effects of internet path faults on reactive routing. In *Proc. of ACM SIGMETRICS 2003, San Diego, CA*, Jun 2003.

[5] N. Feamster, J. Borkenhagen, and J. Rexford. Controlling the impact of bgp policy changes on ip traffic. In *NANOG25*, 2002.

[6] L. Gao. On inferring autonomous system relationships in the internet. In *Proc. IEEE Global Internet Symposium, November 2000.*, 2000.

[7] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000. IEEE.

[8] T. G. Griffin and G. T. Wilfong. An analysis of BGP convergence properties. In *Proceedings of SIGCOMM*, pages 277–288, Cambridge, MA, August 1999.

[9] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2003 ACM workshop on Wireless security*, pages 30–40. ACM Press, 2003.

[10] C. Labovitz, G. R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Transactions on Networking*, 6(5):515–528, 1998.

[11] R. Mahajan, D. Wetherall, and T. Anderson. Understanding bgp misconfiguration. In *Proceedings of ACM SIGCOMM 2002.*, 2002.

[12] S. Misel. Wow, as7007! Available at http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html.

[13] A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Trans. Netw.*, 1(5):510–521, 1993.

[14] Oregon route-views project. Available from http://www.antc.uoregon.edu/route-views/.

$\overline{y..a_1}$	$a_1..a_2..a_3$	$a_1..a_3..a_2$
1	blocked at a_1	blocked at a_1
2	blocked at a_1	blocked at a_1
3	blocked at a_1	blocked at a_1
4	blocked at a_1	blocked at a_1
5	*Possible*	blocked at a_2
6	blocked at a_1	blocked at a_2
7	blocked at a_1	blocked at a_2
8	blocked at a_1	blocked at a_2
9	*Possible*	blocked at a_1
10	blocked at a_1	blocked at a_1
11	blocked at a_1	blocked at a_1
12	blocked at a_1	blocked at a_1
13	blocked at a_1	blocked at a_1
$\overline{y..a_1}$	$a_1..a_2..a_3$	$a_1..a_3..a_2$
1	*Shorter*	Longer
2	*Shorter*	Longer
3	*Shorter*	Longer
4	*Shorter*	Longer
5	*Shorter*	Longer
6	*Shorter*	Longer
7	*Shorter*	Longer
8	*Shorter*	Longer
9	*Shorter*	Longer
10	*Shorter*	Longer
11	*Shorter*	Longer
12	*Shorter*	Longer
13	*Shorter*	Longer

$\overline{y..a_2}$	$a_2..a_3..a_1$	$a_2..a_1..a_3$
1	*Possible*	blocked at a_2
2	blocked at a_2	blocked at a_1
3	blocked at a_2	blocked at a_1
4	blocked at a_2	blocked at a_1
5	blocked at a_3	blocked at a_2
6	blocked at a_3	blocked at a_2
7	blocked at a_2	*Possible*
8	blocked at a_2	*Possible*
9	blocked at a_3	blocked at a_2
10	blocked at a_3	blocked at a_2
11	blocked at a_2	blocked at a_1
12	blocked at a_2	blocked at a_1
13	blocked at a_3	blocked at a_1
$\overline{y..a_2}$	$a_2..a_3..a_1$	$a_2..a_1..a_3$
1	$a_2..a_1$ shorter	blocked at a_1
2	$a_2..a_1$ shorter	blocked at a_1
3	$a_2..a_1$ shorter	blocked at a_1
4	$a_2..a_1$ shorter	blocked at a_1
5	blocked at a_3	$a_2..a_3$ shorter
6	blocked at a_3	$a_2..a_3$ shorter
7	blocked at a_3	$a_2..a_3$ shorter
8	blocked at a_3	$a_2..a_3$ shorter
9	blocked at a_3	$a_2..a_3$ shorter
10	blocked at a_3	blocked at a_1
11	$a_2..a_1$ shorter	blocked at a_1
12	$a_2..a_1$ shorter	blocked at a_1
13	$a_2..a_1$ shorter	blocked at a_1

$\overline{y..a_3}$	$a_3..a_1..a_2$	$a_3..a_2..a_1$
1	blocked at a_1	blocked at a_3
2	blocked at a_1	*Possible*
3	blocked at a_1	blocked at a_3
4	blocked at a_1	blocked at a_3
5	blocked at a_3	blocked at a_3
6	blocked at a_3	blocked at a_3
7	blocked at a_3	blocked at a_3
8	blocked at a_3	blocked at a_3
9	blocked at a_3	blocked at a_3
10	blocked at a_3	blocked at a_3
11	blocked at a_3	*Possible*
12	blocked at a_3	blocked at a_3
13	blocked at a_3	blocked at a_3
$\overline{y..a_3}$	$a_3..a_1..a_2$	$a_3..a_2..a_1$
1	blocked at a_1	*Possible*
2	blocked at a_1	*Possible*
3	blocked at a_1	*Possible*
4	blocked at a_1	*Possible*
5	$a_3..a_2$ shorter	*Possible*
6	$a_3..a_2$ shorter	*Possible*
7	$a_3..a_2$ shorter	*Possible*
8	$a_3..a_2$ shorter	*Possible*
9	$a_3..a_2$ shorter	*Possible*
10	blocked at a_1	*Possible*
11	blocked at a_1	*Possible*
12	blocked at a_1	*Possible*
13	blocked at a_1	*Possible*

Fig. 10. 78 cases of Lemma 10.2

[15] V. N. Padmanabhan and D. R. Simon. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Comput. Commun. Rev.*, 33(1):77–82, 2003.

[16] P. Radoslavov, H. Tangmunarunkit, H. Yu, R. Govindan, S. Shenker, and D. Estrin. On characterizing network topologies and analyzing their impact on protocol design. Technical Report USC-CS-TR-00-731, Mar. 2000.

[17] M. Ruegg and R. Shilter. Modelling and impact of bgp routing policies in the internet. In *Masters Thesis*, 2002.

[18] G. Siganos and M. Faloutsos. Analyzing bgp policies: Methodology and tool. In *IEEE INFOCOM 2004*, Hong Kong, 2004. IEEE.

[19] N. Spring, R. Mahajan, and T. Anderson. The causes of path inflation. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 113–124. ACM Press, 2003.

[20] J. W. Stewart, III. *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[21] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the internet hierarchy from multiple vantage points. In *IEEE INFOCOM 2002*, New York, NY, June 2002. IEEE.

[22] A theoetical method for bgp policy verification. Available from http://www.ee.columbia.edu/ kumar/papers/p2004-02.pdf.

[23] Traceroute.org. Available from http://www.traceroute.org/#Route%20Servers.

[24] F. Wang and L. Gao. Inferring and characterizing internet routing policies. In *ACM SIGCOMM Internet Measurement Workshop*, 2003.