Searching for Stability in Interdomain Routing

Rahul Sami
School of Information,
University of Michigan
Ann Arbor, USA
rsami@umich.edu

Michael Schapira
School of Engineering and Computer
Science, The Hebrew University
Jerusalem, Israel.
mikesch@cs.huji.ac.il

Aviv Zohar
School of Engineering and Computer
Science, The Hebrew U. Jerusalem, Israel
and Microsoft Israel R&D center
avivz@cs.huji.ac.il

***Abstract*—The *Border Gateway Protocol (BGP)* handles the task of establishing routes between the Autonomous Systems (ASes) that make up the Internet. It is known that it is possible for a group of ASes to define local BGP policies that lead to global BGP protocol oscillations. We close a long standing open question by showing that, for *any* network, if two stable routing outcomes exist then persistent BGP route oscillations are possible. This is the first non-trivial *necessary* condition for BGP safety. It shows that BGP safety *must always* come at the price of severe restrictions on ASes' expressiveness in their choice of routing policies. The technical tools used in our proof may be helpful in the detection of potential route oscillations and their debugging.**

We also address the question of how long it takes BGP to converge to a stable routing outcome. We analyze a formal measure of the convergence time of BGP for the policy class defined by Gao and Rexford, which is said to accurately depict the business structure underlying the Internet. We prove that, even for this restricted class of preferences, the convergence time might be linear in the size of the network. However, we show a much more reasonable bound if the network structure is similar to the current Internet: we prove that the number of phases required for convergence is bounded by approximately twice the *depth* of the customer-provider hierarchy.

I. Introduction

Interdomain routing is the task of establishing routes between the administrative domains, called *Autonomous Systems (ASes)*, that make up the Internet. The *Border Gateway Protocol (BGP)* handles *interdomain routing* in today's Internet. BGP enables ASes to make local decisions based on private *routing policies*, and forms global routes from these local decisions. At a very high level, the execution of BGP requires ASes to continuously make independent *greedy* route selections, based on their local preferences over routes, and announce these choices to all neighboring ASes.

However, as first observed by Varadhan et al. [1], this BGP feature may come at a costly price: The lack of global coordination between local routing policies may result in undesirable routing anomalies, in the form of persistent route oscillations. Hence, a central question regarding BGP is to characterize network instances in which BGP execution will always result in convergence to a stable solution. For this reason, researchers seek conditions that guarantee this BGP *safety* (see the seminal works of Griffin, Shepherd and Wilfong [2], [3], and also [4], [5], [6], [7], [8], [9]). Informally, BGP safety means that BGP is guaranteed to converge to a stable routing outcome for *any* possible timing update messages between ASes (see [3] II for a formal definition).

All known sufficient conditions for BGP safety impose very severe constraints on the ASes' routing policies. In particular, they all imply the uniqueness of a stable routing outcome for the network. Simple examples show that the existence of more than one stable solution in very small specific networks can lead to various routing anomalies like BGP divergence [3], [10]. However, up to this point, researchers have not yet been able to establish this fact for general networks.

Two stable solutions imply potential BGP oscillations: Our main contribution in this paper is showing that BGP safety *necessitates* the existence of a *unique* stable solution (regardless of the network's size and topology). That is, two stable solution imply that the network is unstable, in the sense that it could plausibly lead to persistent route oscillations. This result thus closes a long standing open question, first posed by Griffin and Wilfong [2].

Our proof requires a different technique than those used in previous works on interdomain routing. In particular, we analyze BGP dynamics using a convenient combinatorial structure, called the *state-transition graph*. The state-transition graph captures a simplified version of BGP dynamics which makes formal analysis easier. We prove that the results we obtain in the simplified model also hold in the standard model of Griffin et al [3]. Our proof techniques suggest that the state-transition graph might be a useful conceptual tool and heuristic for evaluating and designing network configurations. This may assist in the detection of potential route oscillations and their debugging.

Our result provides the first non-trivial *necessary* condition for BGP safety – uniqueness of the stable routing outcome. We note that there is still a big gap between this necessary condition and the known sufficient conditions (like "No Dispute Wheel" [3]). Finding useful characterizations of BGP safety remains an important open question.

Analyzing BGP's running time. Another question, closely related to the question of whether BGP is guaranteed to converge, is *how long* BGP takes to converge to a stable routing outcome. Naturally, we would like BGP to reach a stable routing outcome quickly. Answering this question requires a formal definition of how we are to measure this "convergence rate", as the Internet is asynchronous, and particular messages can be delayed or even lost.

We present a formal framework for the analysis of BGP's running time. We define a *BGP phase* to be a period of time in which all ASes are activated, and each AS receives update messages from all its neighbors. We then define BGP's running time for a given instance to be the number of phases it may require to converge.

It is known that BGP can take very long to converge on pathological instances. In this paper, we seek to analyze the convergence time of BGP under Internet-like settings. In particular, we consider the class of policies presented by Gao and Rexford [4], which are believed to be a realistic model of policies induced by the commercial relationships between ASes in the Internet. In the Gao-Rexford setting (see Appendix) every pair of neighboring ASes can have a customer-provider relationship or be peers. These business relationships induce natural constraints on ASes' routing policies.

Our first result is negative: We show that, even with this restricted class of preferences, there are instances such that BGP's convergence rate is linear in the size of the network. Specifically, we show that, in a network with n nodes, it might take BGP n time-steps to converge . We also prove that this bound is tight: BGP is always guaranteed to converge in n time-steps.

The linear bound does not bode well, as there are tens of thousands of ASes in today's Internet. Intuitively, however, one would expect BGP to converge at a much quicker rate in practice as ASes' routing policies are local in the sense that they are not influenced by ASes that are far away. We formalize this intuition to prove a positive result that may account for BGP's behavior in real-life: We prove that BGP's running time cannot be more than (roughly) twice the length of the longest customer-provider chain in the network (i.e, longest series of ASes in which each AS is the direct customer of the following AS). Since the length of the longest customer-provider route in the Internet is estimated to be around 5 [11], this means that only a small constant number of steps is required for convergence.

A. Related Work

Varadhan et al. [1] first observed that BGP policy interaction may lead to persistent route oscillations. As mentioned above, there have been a number of papers that seek to find sufficient conditions to guarantee that BGP converges. In particular, Griffin et al. [3] show that BGP is guaranteed to converge if the set of all ASes routing policies does not contain a particular structure called a Dispute Wheel. Subsequently, Gao and Rexford [4] show that, if each AS's policy conforms to certain restrictions based on its business relationships with other ASes, and the global structure of business relationships consists of a hierarchy with peering, there cannot be a Dispute Wheel, and thus, BGP is guaranteed to converge.

Griffin et al. [3] also proved that the existence of two stable solutions always implies the existence of a Dispute Wheel. Note, however, that this, in itself, is insufficient to imply the existence of a BGP oscillation, because the lack of Dispute Wheels is a sufficient but not a necessary condition for convergence: There are networks which induce a Dispute Wheel, and for which BGP safety is guaranteed.

The study of BGP's convergence rate was initiated by Labovitz et al. [12]. In a synchronous update, Feigenbaum, Sami, and Shenker [13] describe an instance with n nodes in which convergence to the stable solution takes $\Omega(n)$ time steps. However, the hardness in this instance arose because next-hop preferences led to a path of $\Omega(n)$ nodes being selected. In contrast, we show that this convergence time can arise even if no paths with more than 2 hops are ever admitted.

Karloff [14] analyzed the worst-case convergence time of lowest-cost routing policies in a different asynchrony model, in which there can be an arbitrary and adversarial update timing for each node. He shows that, in this model, the number of transitions needed for convergence can even be exponential in the number of nodes. In contrast, our model considers time in the asynchronous system differently: each BGP time step or phase includes an update by each active node (and so we count time according to the slower nodes in the network); this leads to our upper bound of n on the running time.

B. Organization of the Paper

In Section II we present the formal models we use to analyze BGP dynamics. In particular, we describe the state-transition graph, which captures a simplified version of BGP dynamics and makes formal analysis easier. In Section III we present our necessary condition for BGP safety (2 stable states imply potential BGP oscillations). In Section IV we discuss our results regarding BGP's convergence rate. Finally, in Section V we conclude and discuss directions for future work.

II. The Model

In this section, we describe the formal models that we use to analyze the dynamic behavior of BGP. At the core, this model is similar to the one proposed by Griffin, Shepherd, and Wilfong [15], [3]. However, there are important differences in the way in which the asynchronous execution of BGP is modeled. Our model of the network, policies, and dynamic evolution is described in Section II-A. In Section II-B, we present an alternative way to visualize possible system behavior in this model, by identifying the system evolution with a trajectory in a "state-transition graph". In Section II-C, we discuss the relation of our model to the standard model of Griffin, Shepherd, and Wilfong [15], [3]. We show that nonconvergence results obtained in our model are valid in the standard model as well.

A. A Formal Model of BGP dynamics

Network and policies. Our model, like the model of Griffin *et al*, abstracts away some implementation-related issues in order to make it easier to analyze BGP convergence. The network is defined by an *AS graph* $G = (N, L)$, where N represents the set of ASes, and L represents physical communication links between ASes. N consists of n *source-nodes* $\{1, \ldots, n\}$ and a unique *destination node* d (in the Internet routes to every destination AS are computed independently). P^i denotes the set of all simple (noncyclic) routes from i to d in G. Each source-node i has a *ranking function* $\leq_i$, that defines a strict order over P^i (that is, i has strict preferences over all routes from i to d). We allow ties between two routes in P^i only if they share the same first link (i,j). The *routing policy* of each node i consists of $\leq_i$ and of i's *import* and *export policies*.[1]

Updates and Activations. BGP belongs to an abstract family of routing protocols named *path-vector protocols* [3]. Basically, the routing tree to a given destination is built, hop-by-hop, as knowledge about how to reach that destination propagates through the network. The process is initialized when d announces itself to its neighbors by sending update messages. From this moment forth, every active node establishes a route to d by repeatedly choosing the best route that is announced to it by its neighbors and announcing this route to all of its neighbors. The network is assumed to be *asynchronous*: ASes can be *activated* in different timings (one can think of the *activation sequence* of ASes as chosen by some adversarial entity [3], [16]).

In our model, there are two kinds of actions that an active node may carry out that potentially change the global routing state:

- A node i may select a route (or change its selected route) from the routes to given destination d that it currently believes to be available.
- A node j may send an update message to a neighboring node i, informing i of the route that j is currently using to destination d. The update message is assumed to be delivered immediately (without propagation delay), and is immediately reflected in updated beliefs that i has about j's route.

The selection and update actions can occur at arbitrary times. In particular, note that the update messages are not required to be sent at a given time interval, or whenever j's route changes.

For convenience, we assume that only one action takes place at any given time. However, it is important to note that this assumption can be made *without loss of generality* in our model, because the update messages are completely decoupled from the route selections: The effect of a set of concurrent actions (for example, concurrent route selections at a subset S of nodes, followed by concurrent advertisement of updated routes) on the global routing state can be obtained through a particular sequence of actions (an arbitrary sequence of route selections at nodes in S, followed by an arbitrary sequence of update messages from the nodes in S).

We are interested in asking the question of when BGP converges to a stable set of routes. Informally, a stable solution is a global configuration that is not changed by any of the possible actions. Thus, once the system is in a stable solution, there is never any further change (provided the network and policies stay the same). Formally, we define a **stable solution** as an n-tuple of routes in G, $R_1, ..., R_n$, such that: (1) If for two nodes $i \neq j$ it holds that j is on R_i, then it must hold that $R_j \subset R_i$ (that is, R_j is a suffix of R_i). (2) If there is a link $(i,j) \in L$, and $R_i \neq (i,j)R_j$, then $(i,j)R_j <_i R_i$. It is easy to show [3] that a stable solution is always in the form of a tree rooted in d.

At first glance, the dynamic model described in this section appears to be unrealistic in some respects, such as the lack of message delay. Indeed, this high level of abstraction is what leads to the simple and tractable state-transition graph described in the next section, which is critical in proving our main result. However, we show in section II-C that this additional simplification does not come at the cost of realism: Any nonconvergence result in our model is true in the previous model of Griffin *et. al.* [3], in which ASes can be activated simultaneously and update messages can be arbitrarily delayed along selective links.

B. The State-Transition Graph

In this subsection, we describe the state transition graph – a tool we use to analyze the convergence of BGP on different instances. The use of graph structures for the representation of BGP configurations and dynamics is a common proof technique (see, *e.g.*, [17], [3]). We exploit the relative simplicity of our asynchronous update model to summarize all possible dynamic evolutions of BGP as paths in this graph, as defined below.

The *state-transition graph* of an instance of BGP is defined as follows: The graph consists of a finite number of states, each state s is represented by an n-dimensional *forwarding vector* of routing choices $r^s = (r^s_1, ..., r^s_n)$, and a $n * n$ *knowledge matrix* $K^s = \{k^s_{ij}\}_{i,j}$. r^s_i specifies the identity of the node to which node i's traffic is being forwarded, and k^s_{ij} specifies the (loop-free) route that node i *believes* that its neighboring node j is using. We define $k^s_{ij} = NULL$ when j is not a neighbor of i; any knowledge that i has about non-neighboring nodes' routes is irrelevant to i's route selection and advertisement decisions. We assume, naturally, that node i knows who it is forwarding traffic to: r^s_i must be the first hop in k^s_{ii}.

A directed edge from a state s to a state s' represents a transition from s to s'. We allow two kinds of atomic actions that lead to transitions:

- *Route transition (route selection actions)*: Informally, a route transition arises when a node i updates its selected route by picking its favorite route from its current knowledge set of routes used by its neighbors.
  Formally, there is an *i-route transition* from state s to state s' iff there is a node i such that: The forwarding vector in s' is identical to the forwarding vector in s with the possible exception of i. Further, $r^{s'}_i = j$ for some node j such that $v_i((i,j)k^s_{ij} \geq v_i((i,j)k^s_{iu}$ for every neighbor u such that $k^s_{iu} \neq NULL$ *and* $i \notin k^s_{iu}$. (This ensures that i never picks a route with a loop in it.)
- *Knowledge transition*: Informally, a knowledge transition is an update message sent from a specific node i to a neighboring node j announcing the route that i believes it is sending traffic along.

[1] The import and export policies can be folded into the routing policies, by modifying the preferences so that paths that are filtered out have the lowest possible value. Thus, we do not explicitly consider them in the remainder of this paper.

Formally, there is a *ji-knowledge transition* from state s to state s' iff there is a node i and a neighoring node j such that: The forwarding vectors in the two states are identical. The knowledge matrix in s' is identical to the knowledge matrix in s with the exception of i's belief about j, and $k_{ij}^{s'} = k_{jj}^{s}$. In other words, i learns of the route that j currently believes it is using.

Note that this definition reflects the restricted asynchrony of our dynamic model: We assume that updates can be held up, but when an update is delivered and processed, it reflects the *current* (at the time of delivery) belief of the node that sent the update. We can phrase this restriction equivalently as: Update messages can be delayed in transit, but when they are delivered, a fresh update message from the same sender is delivered immediately (and thus overrides the delayed update.) Thus, the state description does not have to include messages in transit.

Thus, each transition corresponds to one of a set of possible actions $\mathcal{A} = \mathcal{A}^R \cup \mathcal{A}^K$, where $\mathcal{A}^R$ is the set of route selection actions (one for each node) and $\mathcal{A}^K$ is the set of update actions (one for each ordered pair i, j with an edge between i and j). We write $s \xrightarrow{a} s'$ if action a causes a transition from s to s'; we extend the notation to sequences of actions σ, and write $s \xrightarrow{\sigma} s'$ if the eventual (single) state reached by the sequence σ is s'.

In any state, any of the actions in $\mathcal{A}$ may be activated; we include a self-loop $s \xrightarrow{a} s$ if activating action a in state s results in the system staying in the same state.

We shall call the state in which d has not yet announced itself to any of its neighbors the *Zero state*.

Definition 1: The **Zero state** is the state s in the state-transition graph in which $r_i^s = \emptyset$ for every node i, and $k_{ij}^s = \emptyset$ for every two nodes i, j.

Stability and Oscillations in the State-Transition Graph:

Definition 2: A **stable state** in the state-transition graph is one in which the forwarding vector induces a confluent routing tree T_d to the destination d, the knowledge matrix is such that every node knows its neighbors' routes in T_d, and every node prefers its assigned route over any other available routes.

In other words, a stable state is one in which the nodes forward traffic along a stable solution, and have complete and accurate knowledge about their neighbors' routes. Observe, that all outgoing transition edges from a stable state lead back to the same stable state. This is because a node will never change its route or acquire new knowledge. For this reason, we shall refer to stable states as *sinks*.

We want to prove the existence of potential BGP oscillations in the state transition graph. In many cases, oscillations occur only for specific timings of asynchronous events. Our aim is to show the existence of an infinite activation sequence that leads to an oscillation. However, while BGP routers have unpredictable delays and timings of individual updates, the activation sequences that arise in practice are not completely arbitrary. In particular, starting at any given point of time, every node *eventually* updates its route selection if its knowledge of routes has changed, and every node *eventually* receives update messages from each neighbor that has changed a route. Moreover, there can only be a finite number of other activations taking place between subsequent route selections (or updates) at a given router. For this reason, we look for oscillations that can arise through a *fair* activation sequence:

Definition 3: An infinite activation sequence σ is said to be **fair** if each transition in $\mathcal{A}$ appears infinitely often in σ.

Definition 4: A **fair cycle** in the state-transition graph is a finite cyclic path that does not contain a sink, such that every action in $\mathcal{A}$ is taken at least once in each traversal of the cycle. (Note that the cycle need not be simple, *i.e.*, a state might occur multiple times on a single traversal.)

To summarize, if an instance of BGP results in a state-transition graph (for a given destination) that has a fair cycle, we will infer that there is a plausible sequence of route selections and updates that will cause BGP to oscillate.

C. Implications for the model of Griffin et al

Our model of the static and dynamic evolution of BGP differs from the model of Griffin *et al* [3] in the simpler model of update messages that does not require the tracking of messages in transit. This simpler model allows us to use the state-transition graph to prove, in section III, that the existence of two stable solutions in an instance of BGP implies the possibility of an indefinite oscillation on a fair sequence of route selections and knowledge updates. It is natural to ask if this result is merely an artefact of the simpler model. In this subsection, we show that this is not the case. We prove that if an instance of BGP permits a fair oscillation in our model, it will also permit a fair oscillation in the model of [3].

We modify the dynamic evaluation model of Griffin *et al* [3] in two ways:

- Messages arrive at their destinations *immediately* (update messages are not delayed).
- In the execution of BGP, it is not necessary that a node inform a neighboring node of *every* new route it chooses. It must only announce a route it is using every once in a while.

These modifications are clearly simplifications. At first glance, they may appear to be oversimplifications, but we argue below that the results we obtain with the modified model hold in the original model of Griffin *et al* as well. This provides validation for the modified model, and suggests that it could be independently useful in other settings.

Consider the differences in the message model, and the corresponding difference in the notion of a fair activation sequence. In our model, a fair activation sequence is a sequence of knowledge updates and route selections that is asynchronous, but subject to two constraints: (1) Every action is taken infinitely often, and (2) A delayed update message is equated to a dropped update message, in that it is assumed to always be superseded by a fresh update message from the same sender that was not delayed at all. Constraint (1) is equivalent to the fairness constraint of Griffin *et al*. Constraint (2) is different, and at first glance, it seems unnatural. However, any activation sequence in our model that satisfies constraint (2) can be simulated by an activation sequence in the model of Griffin *et al*[2]. Non-convergence results in this restricted model of asynchrony imply the corresponding results *a fortiori* in a more general model of asynchronous activation, such as the one proposed by Griffin et al [3].

III. Two Stable Solutions ⇒ BGP Oscillation

In this section we prove our main result, i.e, that if there are two stable solutions then the network is unstable in the sense that persistent route oscillations are possible. We prove the following result:

Theorem 3.1: If the AS graph G contains two stable solutions, then there is a fair activation sequence under which BGP will oscillate on G.

That is, two stable solution imply that the network is unstable, in the sense that it could plausibly lead to persistent route oscillations. Therefore, to achieve BGP stability, the network must have a unique stable solution.

The intuition behind our proof is as follows. In the state-transition graph, each stable state will have a corresponding "attractor region": a subset of states (possibly just the stable state itself, or much larger) that, once reached, we can be certain that the system will ultimately converge to the stable state. We can visualize the state-transition graph as a map, with each attractor region a different color – red, blue, etc. However, there will also be some states that do not lie in any one attractor region, because different evolutions from that state could lead to different stable states. We label these states with a distinct color – purple, say – and show that the Zero state must belong in this subset.

The key to the proof is showing that, starting from any purple state, we can find a fair activation sequence that ends at another purple state. We use the properties of route selection and update actions to show that we can swap the order of any two consecutive activations, perhaps repeating one of them, and achieve the same result as the original order. Thus, it is not possible that any given activation a leads to a red state in the original order, but leads to a blue state in the perturbed order. Using this, we show that we can add each activation while staying within the purple region. As the graph is finite, this implies the existence of a fair cycle.

The formal proof of Theorem 3.1 is slightly more involved, because we need to consider multiple cases for the pairs of activations that are perturbed.

Proof: Assume that there are no cycles in the state-transition graph. If so, then, from any state, every fair infinite path must lead to one of the sinks (stable states). Let us assign every sink a unique color that is not purple; for concreteness, we assume that one sink has color red and another has color blue. (There may be other sinks of different colors as well.)

We now extend the coloring to all states in the state-transition graph: Consider a state s. If every fair path from s leads to the same sink, then we shall color s by the color of that sink. If there are two fair activation paths from s that lead to different sinks, we assign s the color purple.

Claim 3.2: There is at least one purple state.

Proof: We show that the Zero state is purple. It is sufficient to show that any given stable state – say the red sink – can be reached from the Zero state through a fair activation sequence. We prove this by explicitly constructing such an activation sequence.

Let T be the stable routing tree in the red sink. Starting from Zero, we activate route selections and updates such that each node has its route in T selected. We do this by growing the tree from the destination: First, the destination's neighbors in T are activated to select routes; they will naturally choose the direct link to T, which is the only route available to them. Subsequently, we activate update messages from nodes i already connected to the destination to their children in T, followed by activating the route selection in each child j. This culminates in a state in which all nodes have their routes in T selected. Note that this is not necessarily the red sink state, as nodes may not have complete information about all their neighbors' routes. However, after T has been constructed, we can activate all update and route selection messages in round-robin order, infinitely often. The stability of T guarantees that the routes will never be changed, and thus (in finite time because the graph is finite) we will ultimately reach the red sink state. The finiteness of the initial construction ensures that this activation sequence is fair. As this construction is possible for each sink state, the Zero state must be purple. ■

To prove our theorem we need only prove that from every purple node we can reach another purple node by following a path in which all possible actions are activated at least once. Since there is a finite number of purple nodes, if we start from one, construct a fair path to other purple node, and continue this process, eventually a purple node will appear for the second time. Since no purple node is a sink, a path between purple nodes cannot contain a sink, and as all actions were activated in each finite segment constructed, this forms a fair cycle.

Claim 3.3: From any purple state, there is a finite path containing all actions in $\mathcal{A}$ that leads to another purple state.

Proof: Consider a purple node p. We want to show that, starting from p, there is a finite activation sequence using every action in $\mathcal{A}$ that leads to a purple state. For contradiction purposes, suppose that such an activation sequence does not exist. Then, there must exist an action a and activation sequence σ (possibly null) such that:

1) $a \notin \sigma$
2) $p \xrightarrow{\sigma} p'$, where p' is purple (possibly $p = p'$).
3) Any finite activation sequence from p' that contains action a results in a state that is not purple.

(If no such a and σ existed, we could add each action in turn to σ while keeping it finite, and still result in a purple node.)

We will now prove that these conditions cannot be satisfied. Consider the action a at state $p_0 = p'$. By the third condition, this must lead to a state q_0 that is not purple; without loss of generality, suppose that q_0 is red. Now, because p_0 is purple,

[2] The converse of this statement is not true, but that is not relevant for our proofs, because we only rely on the construction of nonconvergent instances.

there is some finite path from it leading to a state that is neither purple nor red; without loss of generality, suppose this path is $p_1, p_2, \cdots, p_k$, where $p_1, \cdots, p_{k-1}$ are all purple, and p_k is blue. (The path maybe a single step, in which case p_1 is blue.)

Now, define the corresponding states $q_1, q_2, \cdots, q_k$ as follows: $p_i \xrightarrow{a} q_i$. In other words, q_i is the state reached by action a from p_i. Note that, by condition 3, none of the q_i can be purple. Also, q_0 is red and q_k is blue; thus, there must exist an index i such that q_{i-1} is red while q_i is another color (blue without loss of generality).

Let b be the action leading from p_{i-1} to p_i. We consider four cases, and prove a contradiction in each case:

- Case (i): a and b are both route selection actions. In this case, a and b must be route selections for distinct nodes. Further, neither a nor b changes the knowledge sets, to the knowledge profile is the same at q_{i-1} and q_i. Now, consider activating the action b from q_{i-i}. As the knowledge states are the same in q_{i-1} and p_{i-i}, we must have $q_{i-1} \xrightarrow{b} q_i$. But this implies that there is a fair activation sequence leading from q_{i-1} to the blue sink, which is impossible because q_{i-1} is red.
- Case (ii): a and b are both knowledge actions. In this case, a and b must be updates corresponding to different edges in the network, as they are distinct actions. Hence, a and b affect different components of the global knowledge state K, and thus, the activation sequences ab and ba starting from p_{i-1} lead to the same state. Again, we find that $q_{i-1} \xrightarrow{b} q_i$, leading to a contradiction.
- Case (iii): a is a knowledge action, and b is a route selection action. Let a correspond to an update from some node j to some node i. If b is a route selection at a node other than i, then ab and ba are equivalent, leading to a contradiction as in the first cases. If b is a route selection at i, the situation is a little trickier: The action b might have a different result if it occurred after the action a. Now, consider instead the two activation sequences ab and bab starting at state p_{i-i}. The former activation sequence must lead to a red state, as it passes through q_{i-1}. The latter activation sequence likewise must lead to a blue state, as t passes through q_i. However, the final state should be the same following both activations: In activation sequence bab, whatever route node i chose when action b was first activated at p_{i-1}, the route it chooses in the second activation after a will be the same route that was selected in activation sequence ab. Further, the knowledge sets of all nodes are also the same, as only i's knowledge set has changed in either activation sequence, and i's knowledge set is updated in the same way in both sequences. This leads to a contradiction, as a blue state cannot be a red state.
- Case (iv): a is a route selection action, and b is an knowledge action. This case is symmetric to case (iii); considering the activation sequences ba and aba leads to a contradiction.

Thus, no such a and σ exist; it follows that given any finite activation sequence σ leading from p to a purple node, and given any $a \notin \sigma$, we can extend the activation sequence with a finite number of steps to include a, while still reaching a purple node. Proceeding in this manner, we can construct a finite activation sequence σ that contains all activations in $\mathcal{A}$, such that $p \xrightarrow{\sigma} p*$, where $p*$ is a purple node (perhaps p). ■
(Concluding proof of theorem 3.1): Repeating this construction, and recognizing that there are finitely many purple nodes, we must eventually return to a state that has been previously visited. Thus, we can construct a finite activation sequence σ such that for some purple node p, $p \xrightarrow{\sigma} p$, and σ contains all actions. By the construction, p is reachable from the Zero state in a finite number of steps. An infinite repetition of this σ after getting to p is a fair activation sequence, and leads to an oscillation, because it never reaches a stable state. ■

This proof also suggests that the state-transition graph and attractor regions might be a useful conceptual tool for evaluating and designing network configurations. If there is a desired stable state, a routing policy change can be evaluated based on how much it grows the size of the corresponding attractor region. Unfortunately, the state-transition graph is exponential in size, so this is not a useful formal measure for large networks; however, we believe that it could still be a helpful heuristic in designing smaller subnetworks.

IV. BGP Convergence Time

In this section, we tackle the question of how long BGP takes to converge to the unique stable solution. We begin with the abstract model of BGP as described in section II-A. BGP is an asynchronous protocol, and individual messages may be lost or delayed arbitrarily. As we cannot assume a bound on the actual elapsed time of a single message, any model of convergence "time" needs to define a unit of time measurement that remains meaningful in this asynchronous setting. We propose the following definition:

Definition 5: A **BGP phase** is a period of time in which all nodes get at least one update message from each neighboring node, and all nodes are activated at least once after receiving updates from their neighbors.

We analyze the number of BGP phases it requires for the network to converge. The rationale for this definition is that, although it is difficult for the analyst to assert numerical bounds on the update frequencies at different nodes, it is reasonable to expect that all nodes are updating at similar timescales. The definition of phases admits asynchrony, thus capturing the realistic possibility that different sequences of update activations can lead to different transient behavior. At the same time, by tying the unit of measurement to the slowest node's update instead of a fixed time unit (or the fastest update), we avoid pathological worst-case time bounds that are only attained if, for example, one node's update cycle is measured in years instead of seconds or minutes.

How many consecutive phases does it take BGP to converge to a stable solution in the worst case? Routes are propagated through the network one hop at a time, so the best we can hope for is a time proportional to the lenght of the longest route in the stable solution. It is easy to construct instances with n nodes in which there are routes of length $\Omega(n)$. However, these instances are unnatural; currently, Internet routes tend to be much shorter than this.

For this reason, we focus on bounding the BGP convergence time on Internet-like. In particular, we consider the class of instances in which nodes have customer/provider or peer relationships, and their preferences satisfy the Gao-Rexford conditions (detailed in the Appendix). Sadly, we show that even in the Gao-Rexford setting, it is possible to come up with examples in which the worst-case convergence time is n BGP phases. Furthermore, this hardness is not solely due to the existence of unnaturally long routes: As the next example illustrates, even if all nodes are only interested in routes of size 2, BGP convergence might require n phases.

Example 4.1: The graph in Figure 1 depicts a network with n nodes, and a destination node d. Node 1 prefers to go directly to d. Any other node i prefers the route $i \rightarrow i-1 \rightarrow d$ over the direct route $i \rightarrow d$. All routes of length greater than 2 are less desirable to any node. This set of path preferences is compatible with the Gao-Rexford constraints for the following set of customer/provider relationships: 1 is a customer of 2, 2 is a customer of 3, *etc.*; and, additionally, d is a customer of every other node.

The figure also depicts an initial routing configuration (using arrows to describe the initial routing choice of each node). In this configuration, node 1 has no routing choice (perhaps it just came online), all even nodes route directly to the destination and all odd nodes (except 1) route through their counter-clockwise neighbor. The reader may verify that this configuration is stable with the exception of node 1.

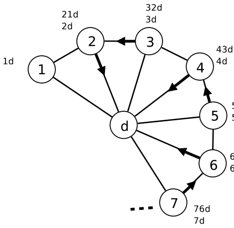

Fig. 1. A slow-converging network configuration.

Now, consider the evolution of the network under the following activation sequence. In each phase, initially all update messages go through, and then all nodes are activated. In the first phase, only node 1 will change its routing choice and will route to d. In the next phase, only node 2 will change its routing choice and will route through 1. Then node 3 will change to route through d and so on. The network will eventually converge to the routing outcome in which all odd nodes route directly to d and all even nodes route through their counter-clockwise neighbor.

Next, we show that this bound is tight for the class of instances that satisfy the Gao-Rexford conditions. In fact, we prove a slightly stronger result: The following proposition shows that our bound on BGP's convergence rate is tight on the larger class consisting of all instances in which the "No Dispute Wheel" condition of [3], [5] holds.

Proposition 4.2: If "No Dispute Wheel" holds then BGP's convergence rate is at most n phases.

Proof: Let us assume that indeed the "No Dispute Wheel" condition holds in a network graph G with a destination node d. We will show that at every phase, one of the nodes of the graph converges to a route that will not change from that point on. The first node that converges in the first phase is the destination node d, that has the empty path, and announces that path to its neighbors.

We now argue that there must exist a node in the network that is a direct neighbor of the destination d and that its most preferred path is going directly to d. To see that this is indeed the case, pick an arbitrary node v, look at its most preferred path to the destination. This path goes through a neighbor of d right before it reaches d. We shall denote this neighbor by v_1. Now, consider the most preferred path of node v_1, and the closest node to d on that path that we shall denote by v_2. In this manner we define the nodes v_i for $i = 1, 2, 3, \ldots$. At some point, nodes in the sequence we defined must start repeating. If only one node repeats infinitely then this node must have a direct route as its most preferred path, and we are done. Otherwise, the sequence of repeating nodes $v_k, v_{k+1}, \ldots, v_{k+l}$ (for some k, l) forms a dispute wheel: each node prefers to go through the next one in the sequence rather than directly to d. This contradicts our assumption.

Therefore, there exists a node that prefers to go directly to d over any other path. It will choose this path to d on the second phase, send update messages to its neighbors, and will never again change its path (since no path will be better).

We will now continue to follow the convergence process, and observe that at any phase, there must exist a node v that converges to its most preferred route *given the route of the nodes in the system that have already permanently converged.* This node will never again change its path (because unless previous nodes change, it will have no better path, and these previous nodes have also converged). To prove that such a node must exist, we fix the routes of all permanently converged nodes, and pick an arbitrary node v_1 that did not converge. We once again define the sequence of nodes $v_1, v_2, v_3 \ldots$ by defining the node v_{i+1} as the node that is closest to d on the most preferred path of node v_i that did not permanently converge. The set of paths from which we select this most preferred path, is the set of paths that are consistent with the nodes that have already permanently converged. Once again, this sequence of nodes must repeat, and since it cannot contain a dispute wheel, it must have only a single repeating node that is the closest node that did not converge on its own most preferred path. In the next phase, this node's path converges.

We have thus shown that if the AS graph contains no dispute wheels, the convergence time of BGP is bounded by the number of nodes in the entire network graph. ■

In practice however, we know that the Internet does not converge on a time scale that grows linearly with the size of the network. Our following theorem exhibits a much lower bound on convergence time assuming that the Gao-Rexford [4] economically oriented constraints hold for the network, *and* that the length of customer-provider chains is limited. This is a realistic assumption as it is usually accepted that there are around 3-5 levels in the hierarchy of providers [11]. In fact, a the 3-tier model for the network that is widely accepted as a rough approximation of the structure implies only 3 levels [18]. This in turn implies a very fast convergence time for BGP in the network – even in the worst case.

Theorem 4.3: In the Gao-Rexford setting, BGP's convergence time is at most $2\alpha + 2$ phases, where α is the length of the longest directed customer-provider chain in the AS graph.

Proof: We start with some notation and terminology that will assist with the proof. We shall say that a path is *permissible* if every link in the path exists in the network, and will also be reported (i.e., not filtered) by the node, as dictated by the Gao-Rexford constraints.

We say that a node's route is a customer route if the first hop on that route is to one of the node's customers. We define a peer route and a provider route in a similar manner. Let α denote the length of the longest directed customer-provider chain in the graph. Because of the Gao-Rexford constraints we are assured that the routing graph has a unique stable state to which the network will converge. We will now prove the following three claims:

1) A node that has a permissible customer route will converge to a customer route within $\alpha + 1$ phases.
2) A node that eventually converges to a peer route does so at most 1 phase after nodes with customer routes have converged.
3) A node that eventually converges to a provider route converges within α phases after all nodes with peer and customer routes have converged.

When put together, the three claims imply that the network will converge within $2 \cdot \alpha + 2$ phases. We now prove them:

1) First, notice that if a node has a permissible customer route then every node on this route forwards its traffic to one of its customers. This is because the Gao-Rexford constraints state that no AS publishes peer or customer routes to any of its providers. Thus, a node that in some permissible path accepts traffic from its provider will not forward it to a peer or provider.

   We will now prove that any node v that has at least one permissible customer route will converge by phase $\alpha_v + 1$ where α_v is the length of v's longest permissible customer route. The proof is by induction. Our base case will be that the convergence time for any node v that has $\alpha_v = 1$ is at most two phases. These nodes are neighbors of the destination. In the first phase, the destination node announces itself and converges to the empty path. In the second phase, any node v for which $\alpha_v = 1$ will select a direct path to the destination. This is because such a node has only a single customer route (which is of length 1) and because it must prefer customer routes to any other route (due to the Gao-Rexford constraints). As this node has no other permissible customer routes, it will never route differently.

   In the induction step, we assume that any node v for which $\alpha_v < k$ has converged by phase k. Let u be an arbitrary node for which $\alpha_u = k$. Any node that is on a customer route of u has converged by phase k at the latest, and has updated its neighbors. Therefore, node u's available customer routes will not change from this point on. In phase $k + 1$ node u chooses one of the available customer routes (because the Gao-Rexford constraints dictate that they are preferred over any other route) and never changes its next hop thereafter, as no better path will ever present itself, nor will the current path ever become unavailable (as all nodes on it have also permanently converged). This concludes the proof of the claim.
2) Due to the Gao-Rexford constraints, an AS will not export its peer routes to its peers/providers and therefore any route that begins with a peering connection must proceed as a customer route from the second node onwards (as the second node on the route would not publish any route other than a customer route to its peer). This implies that the second node on such a route must have a permissible customer route, and therefore has converged by phase $\alpha + 1$. On phase $\alpha + 2$, any node v with a peer-route that has not already converged will have heard updates from its neighbors about all the peer routes that will ever be available. Since these routes are preferred to provider routes, the most preferred peer-route will be selected in phase $\alpha + 2$ (or earlier) and will not change from that point on. Note that any peer route that is not available during phase $\alpha + 2$ will never be available, and will never be selected.
3) We denote by β_v the length of the longest customer $\rightarrow$ provider chain that appears in any permissible path of node v. Surely it for all nodes $\beta_v \leq \alpha$. We now prove the following claim: Any node v that eventually converges to a provider route does so within β_v phases after all nodes with customer and peer routes have converged. We prove this claim by induction on the phase number. Our base case is that one phase after convergence of all customer and peer routes, any node v that has $\beta_v = 1$ converges. Since v's provider on any path is using either a peer route or a customer route, then the next hop of v on any permissible path has already converged, and sent notification messages about its permanent path. v therefore chooses path it finds most appealing and never changes it from this point on.

   In the induction step, we assume that by phase $\alpha + 2 + k$ all nodes v' for which $\beta_{v'} < k$ have already converged. Nodes that have a value of $\beta_v = k$ therefore have their next hop on any permissible path already converged by this time. Thus, they choose the most appealing route, send update messages and never switch paths again.

■

V. Conclusions and Future Work

In this paper, we studied fundamental questions related to whether and how fast BGP converges to a stable solution. We showed that, in any instance in which the local routing policies can lead to two stable solutions, BGP could potentially lead to persistent oscillations; thus, the existence of a unique stable solution is a necessary condition for safe convergence. We then analyzed the worst-case convergence time of BGP on instances that satisfy the Gao-Rexford conditions. We proved that the convergence time on a graph with n nodes is $\Theta(n)$ in the worst case, but is much smaller in networks with shallow customer-provider hierarchies.

Our results suggest several interesting directions for future work. First, can we close the gap between out necessary condition and known sufficient conditions for safe convergence? Second, can we develop a compositional theory for safe policies: If we put together two subnetworks with unique stable solutions, when does the combination also have a unique stable solution? It would also be valuable to extend the convergence-time analysis to broader classes of preferences, and to characterize the average-case (instead of worst-case) convergence time following a network change. Answers to these questions could provide network operators with new principles to tradeoff the desire for flexible autonomous policies with the need for global routing efficiency.

Finally, there are practical aspects of BGP operations not considered in this paper such as MRAI (Minimum Route Advertisement Interval) and RFD (Route Flap Damping [19]), which play a significant role in BGP convergence [20], [21]. An interesting direction for future research is proposing formal models for addressing these issues and assessing their impact on our necessary condition for BGP safety.

Acknowledgements

We thank Joan Feigenbaum, Aaron Jaggard, Sharon Goldberg, Vijay Ramachandran, Jennifer Rexford, Scott Shenker, and the anonymous referees for helpful comments.

References

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

[2] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proceedings of SIGCOMM 1999*.

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

[4] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 681–692, 2001.

[5] L. Gao, T. G. Griffin, and J. Rexford, "Inherently safe backup routing with BGP," in *20th INFOCOM*. Pistacaway: IEEE, 2001, pp. 547–556.

[6] T. G. Griffin, A. D. Jaggard, and V. Ramachandran, "Design principles of policy languages for path vector protocols," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York: ACM, 2003, pp. 61–72.

[7] A. D. Jaggard and V. Ramachandran, "Robustness of class-based path-vector systems," in *Proceedings of ICNP'04*, IEEE Computer Society. IEEE Press, October 2004, pp. 84–93.

[8] N. Feamster, R. Johari, and H. Balakrishnan, "Implications of autonomy for the expressiveness of policy routing," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer comm.* New York, NY, USA: ACM Press, 2005.

[9] J. L. Sobrinho, "An algebraic theory of dynamic network routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1160–1173, 2005.

[10] T. G. Griffin and G. Huston, "TRFC 4264: BGP wedgies," 2005.

[11] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz, "Characterizing the internet hierarchy from multiple vantage points," *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Comm. Societies. Proceedings. IEEE*, vol. 2, pp. 618–627, 2002.

[12] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed internet routing convergence," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 175–187, 2000.

[13] J. Feigenbaum, R. Sami, and S. Shenker, "Mechanism design for policy routing." *Distributed Computing*, vol. 18, no. 4, pp. 293–305, 2006.

[14] H. Karloff, "On the convergence time of a path-vector protocol," in *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004, pp. 605–614.

[15] T. G. Griffin and G. Wilfong, "A safe path vector protocol," in *Proceedings of IEEE INFOCOM 2000*, IEEE Communications Society. IEEE Press, March 2000.

[16] H. Levin, M. Schapira, and A. Zohar, "Interdomain routing and games," in *Proceedings of the 40th ACM Symposium on Theory of Computing (STOC)*, May 2008.

[17] A. Fabrikant and C. Papadimitriou, "The complexity of game dynamics: BGP oscillations, sink equlibria, and beyond," in *Proceedings of SODA 2008*.

[18] G. Huston, "Interconnection, peering, and settlements," in *Internet Global Summit (INET)*. The Internet Society, 1999.

[19] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz, "Route flap damping exacerbates internet routing convergence," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2002, pp. 221–233.

[20] E. C. Jr., Z. Ge, V. Misra, and D. Towsley, "Network resilience: Exploring cascading failures within bgp," in *Allerton Conference on Communication, Control and Computing*, October 2002.

[21] K. Sriram, D. Montgomery, O. Borchert, O. Kim, and D. R. Kuhn, "Study of bgp peering session attacks and their impacts on routing performance," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 10, pp. 1901–1915, 2006.

Appendix

Studies of the commercial Internet [18] suggest two types of business relationships between pairs of ASes: *Customer-provider* or a *peering*. Customer ASes pay their provider ASes for connectivity. Peers find it mutually advantageous to exchange traffic among their customers for free. Gao and Rexford [4] suggest constraints on routing policies that are naturally induced by these business relationships, and prove that these constraints guarantee BGP stability.

- **No customer-provider cycles:** Let G_{CP} be the directed graph with the same set of nodes as G and with a directed edge from every customer to its direct provider. We require that there be no directed cycles in this graph (*i.e.*, no AS is indirectly its own provider).
- **Prefer customers to peers and providers:** A *customer route* of a node i is a route in which i's next-hop node j is i's customer. *Provider* and *peer routes* are defined similarly. We require that nodes always prefer customer routes over peer and provider routes.
- **Provide transit services to customers only:** ASes should export *all* of their routes to their customers, but should only export customer routes to their providers and peers.