Real-time Model and Convergence Time of BGP

Davor Obradovic

***Abstract*—BGP allows routers to use general preference policies for route selection. This paper studies the impact of these policies on convergence time. We first describe a real-time model of BGP. We then state and prove a general theorem providing an upper bound on convergence time. Finally, we show how to the use the theorem to prove convergence and estimate convergence time in three case studies.**

I. Introduction

BGP [1], [2] is the main inter-domain routing protocol today. Unlike most of the intra-domain routing protocols, such as RIP and OSPF, BGP routers can be configured to use independent and general preference policies for route selection. Routers keep exchanging reachability information while trying to maximize their local route preference. This process, called *convergence*, continues until all routers agree on a stable set of routes. *Fast* convergence is generally desired from most routing protocols. In the case of BGP, it has been shown that diverse policies can interact in a way that increases convergence time or even causes the system to diverge [3], [4].

This paper studies the impact of general preference policies on *convergence time*. Section II reviews the formal model of "timeless" BGP designed by Griffin et al. [4]. In Section III we extend this model with some real-time information that enables us to observe the evolution of the protocol in time. In Section IV we use the real-time model to derive a fundamental theorem about an upper bound on convergence time. Section V applies the theorem to three case studies. Section VI concludes and gives some future prospects.

II. "Timeless" Model of BGP

We will study convergence on a formal model of BGP used in [4], [5]. The model consists of two parts: the *Stable Paths Problem (SPP)* and the *Simple Path Vector Protocol (SPVP)*. SPP provides a formal semantics for BGP policies, while SPVP describes the protocol dynamics. SPVP is a distributed algorithm that attempts to solve the Stable Paths Problem, just like RIP and OSPF try to solve the Shortest Path Problem. Since BGP allows routing policies that are more general than "shortest-path-first", the Stable Paths Problem will be a generalization of the Shortest Paths Problem.

A network is represented as a simple undirected graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is the set of nodes connected by edges from E. Nodes represent the routers and edges represent BGP sessions between them. For a node u, its set of *peers* is $\text{peers}(u) = \{v \mid \{u, v\} \in E\}$. We assume that there is a single destination (node 0) to which all other nodes are trying to establish paths. A *path* in G is a sequence of nodes $(v_k\ v_{k-1}\ \ldots v_0)$, such that $\{v_i, v_{i-1}\} \in E$, for all $i, 1 \le i \le k$. We will assume that there exists a a special *empty path*, denoted by ϵ, which will be used indicate the absence of any route to the destination. Nonempty paths $P = (v_1\ v_2\ \ldots\ v_k)$ and $Q = (w_1\ w_2\ \ldots w_m)$ can be concatenated in a natural way if $v_k = w_1$. In that case, we define

$$PQ = (v_1\ v_2\ \ldots\ v_k\ w_2\ \ldots\ w_m).$$

For every path P, concatenation with the empty path produces the empty path:

$$P\epsilon = \epsilon P = \epsilon.$$

A path is called *simple* if it does not contain multiple instances of the same vertex. We will be consider only simple paths, since BGP immediately discards paths which contain loops. For a simple path $P = (v_1\ v_2\ \ldots\ v_k)$ and any two of its nodes $u = v_l$ and $w = v_m$ ($l \le m$), we denote by $P[u \ldots w]$ the corresponding sub-path $(v_l\ v_{l+1}\ \ldots\ v_m)$. Each node $v \in V - \{0\}$ has the corresponding set of *permitted paths* from v to the destination, denoted by $\mathcal{P}^v$. This is a subset of the set of all paths from v to 0, since a node may consider certain paths as unacceptable. Let $\mathcal{P} = \{\mathcal{P}^v \mid v \in V - \{0\}\}$ denote the set of all permitted path sets. For each $v \in V - \{0\}$, there is a *ranking function* $\lambda^v : \mathcal{P}^v \to \mathbf{N}$. For $P \in \mathcal{P}^v$, $\lambda^v(P)$ denotes the degree of preference that the node v gives to the path P. More preferable paths will have higher values of λ^v. Let $\Lambda = \{\lambda^v \mid v \in V - \{0\}\}$ stand for the set of all ranking functions.

We say that a triple $S = (G, \mathcal{P}, \Lambda)$ is an instance of the *Stable Paths Problem* (SPP) if the following holds:

- (SP1) Empty path is permitted: $\epsilon \in \mathcal{P}^v$.
- (SP2) Empty path is lowest ranked: $\lambda^v(\epsilon) = 0$.
- (SP3) Strictness: If $\lambda^v(P_1) = \lambda^v(P_2)$, then either $P_1 = P_2$, or $P_1 = (v\ u)P_1'$ and $P_2 = (v\ u)P_2'$ for some node u (i.e. P_1 and P_2 are either equal or have the same next hop).
- (SP4) Simplicity: If $P \in \mathcal{P}^v$, then P is a simple path (i.e. P does not contain loops).

Let $S = (G, \mathcal{P}, \Lambda)$ be an instance of the SPP. Given a node u and a set of paths $W \subseteq \mathcal{P}^u$ with distinct next hops, we define the *maximal path* in W with respect to u to be

$$\max(u, W) = \begin{cases} P \in W \text{ with max. } \lambda^u(P), & \text{if } W \neq \emptyset \\ \epsilon, & \text{otherwise.} \end{cases}$$

A *path assignment* is a function π that maps each node $u \in V$ to a permitted path $\pi(u) \in \mathcal{P}^u$. In the BGP terminology, path assignments correspond to Loc-RIB routing tables. Given a path assignment π and a node u, we define the set of *choices* for u as

$$\text{choices}(u, \pi) = \{(u\ v)\pi(v) \mid \{u, v\} \in E\} \cap \mathcal{P}^u.$$

University of Pennsylvania, davor@saul.cis.upenn.edu

This is the set of one-hop extensions of the paths selected by u's neighbors. The path assignment π is *stable at node* u if u has the optimal path among its choices:

$$\pi(u) = \max(u, \text{choices}(u, \pi)).$$

The path assignment π is *stable* if it is stable at every node $u \in V$. An SPP instance $S = (G, \mathcal{P}, \Lambda)$ is *solvable* if there exists a stable path assignment π for S. We call every such assignment a *solution* for S and write it as $(P_1, P_2, \ldots, P_n)$, where $\pi(u) = P_u$. An instance of SPP may have zero, one or more solutions. [1]

We will now describe the Simple Path Vector Protocol (SPVP). Each node maintains two data structures:

- $\text{rib}(u)$ is u's current path to the destination.
- $\text{rib\_in}(u \Leftarrow w)$ denotes the path most recently advertised by w to u.

Similarly as before, we define the set of path choices available to u as

$$\text{choices}(u) = \{(u\ w)\text{rib\_in}(u \Leftarrow w) \mid w \in \text{peers}(u)\} \cap \mathcal{P}^u,$$

and the best choice as $\text{best}(u) = \max(u, \text{choices}(u))$. Neighboring nodes keep exchanging paths that they have currently stored in the rib field. As the node u receives path advertisements from its peers, it uses them to recompute the set of available paths $\text{choices}(u)$. The node tries to maintain the best path from that set stored in the field $\text{rib}(u)$. Every once in a while, the node recomputes its best path and notifies its peers in case of a change. This may cause the peers to send further advertisements to their peers, and so on. The process continues as long as there are unprocessed advertisements.

Unprocessed advertisements are stored in reliable FIFO queues at the receiving end. Each node has one such queue for each peer—the queue where u stores advertisements from its peer w is denoted by $\text{mq}(u \Leftarrow w)$. A *state* of the protocol is defined by states of all the routers (denoted by $\mathcal{S}$) and contents of all the queues mq (denoted by $\mathcal{Q}$). A router's state consists of the values of rib and rib_in fields. Therefore, a pair $(\mathcal{S}, \mathcal{Q})$ can be regarded as a global protocol state. The global state changes when an event happens. There are two kinds of events—receipt of a route ($\text{receive}(\text{v} \Leftarrow \text{u})$) and recomputation of the best route ($\text{recompute}(\text{u})$). The first event simply stores the received route in the appropriate rib_in. The second event recomputes the best route and notifies the neighbors in case it changed. Precisely, these events change the global state in the following way:

- $\textbf{receive}(\textbf{v} \Leftarrow \textbf{u})$:
  $\text{rib\_in}(v \Leftarrow u) :=$ dequeue $(\text{mq}(v \Leftarrow u))$.
- $\textbf{recompute}(\textbf{u})$:
  if $\text{rib}(u) \neq \text{best}(u)$ then {
  $\text{rib}(u) := \text{best}(u)$
  $\forall v \in \text{peers}(u).\ \text{enqueue}(\text{mq}(v \Leftarrow u),\ \text{rib}(u))$
  }

Function *dequeue* removes the front element of the queue and returns it as the result. It signals an error if the queue is empty. Function *enqueue* adds an element at the back of the queue. An *activation sequence* is an arbitrary sequence of these events. Each event modifies the global protocol state. Therefore, given an initial state and a valid activation sequence, we can generate the corresponding sequence of states (*run of the protocol*) that describes the evolution of the system. An activation sequence is *valid* with respect to the initial state $(\mathcal{S}_0, \mathcal{Q}_0)$ if the following holds when we run it on that initial state:

1) An event of the form $\text{receive}(v \Leftarrow u)$ never happens when $\text{mq}(v \Leftarrow u)$ is empty.
2) Every route advertisement sent during a 'recompute' event is eventually received during the corresponding 'receive' event.

When we run SPVP from some initial state using a valid activation sequence, the system exhibits either a convergent or a divergent behavior. We say that the system *converges* if the routes eventually stabilize (i.e. there is some point in time after which rib fields do not change). Otherwise, rib fields perpetually change and we say that the system *diverges*. It is easy to see that in the case of a convergent behavior, final routes form a stable solution to the corresponding SPP. In that sense, SPVP is a sound algorithm for solving SPP. However, it is not complete, because there are configurations where solutions (i.e. stable routes) exist, but SPVP is not guaranteed to find them. There are also configurations where it is impossible to simultaneously pick stable routes for all nodes (i.e the corresponding SPP is unsolvable). SPVP necessarily diverges on such configurations.

Griffin, Shepherd and Wilfong [4] established a sufficient condition for ensuring convergence. Their technique is based on the notion of a *dispute digraph*, which can be constructed for every SPP instance. Let $S = (G, \mathcal{P}, \Lambda)$ be an instance of SPP. The corresponding dispute digraph is a directed graph whose nodes are all permitted paths ($\{P \mid P \in \mathcal{P}^v,\ v \in V\}$). There are two types of arcs: transmission and dispute arcs. Suppose u and v are peers.

- Transmission arcs: If $P \in \mathcal{P}^v$ and $(u\ v)P \in \mathcal{P}^u$, then there is a *transmission arc* $P \longrightarrow (u\ v)P$.
- Dispute arcs: Suppose that v has two available paths to the destination: P and Q ($P, Q \in \mathcal{P}^v$). Then u potentially also has two paths: $(u\ v)P$ and $(u\ v)Q$. Disputes arise when u and v disagree on which path to use. Namely, if v prefers P over Q, while u prefers $(u\ v)Q$ over $(u\ v)P$, [2] then there is a *dispute arc* $P \longrightarrow (u\ v)Q$. The situation is shown on Figure 1.

The major result proved in [4] states that convergence is guaranteed if the dispute digraph is acyclic.

III. Real-time Model of BGP

Ensuring convergence in practice is a hard problem. BGP is designed to allow maximum flexibility in route preference policies. This makes it possible to set up a divergent BGP system while completely respecting the protocol standard. The problem of divergence is studied in several papers [3], [4], [5], [6], [7]. We will focus our attention on a different problem—the *length* of the convergence process. While conflicting policies

[1] See [4] for examples.

[2] Or u simply rejects $(u\ v)P$, while accepting $(u\ v)Q$

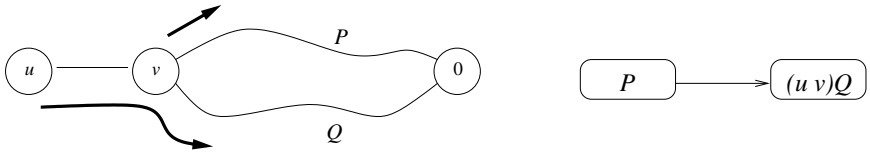

Fig. 1. A dispute between u and v

that would cause divergence are theoretically possible, many practitioners claim that such policies are rarely (if ever) used in practice. However, even then, it is important to know how much time would convergence take. Just the fact that routes would *eventually* stabilize does not provide much assurance to the users. However, knowing that convergence would happen in, say, one hour, as opposed to one day is a more valuable information. Hence, our main goal would be establishing upper bounds on convergence time of SPVP.

As the first step, we need to extend the model with real-time. We will do that by associating, with each edge, an upper bound on route propagation time across that edge. Formally, instead of representing the network with an ordinary graph $G = (V, E)$, we will use a weighted graph $G' = (V, E, d)$, where $d : V \times V \rightharpoonup \mathbf{R}$ (called *delay*) is a strictly positive partial real function such that $d(u, v)$ is defined if and only (u, v) is an edge. Delay function constrains the speed of route propagation in the following intuitive sense: if u changes its route at time t, v should find out about that change and appropriately adjust its route by the time $t+d(u, v)$. Thus augmented instance $S' = (G', \mathcal{P}, \Lambda)$ is called an instance of TSPP (Timed SPP). [3] Notice that establishing any kind of upper bound on convergence time requires us to have finite edge delays. Indeed, if an edge (u, v) can have unbounded delay, then v can indefinitely "ignore" the route advertisements from u and hence indefinitely postpone the convergence. This is why edge delays in some sense represent the minimal information that one needs to have in order to bound convergence time.

As the next step, we need to extend the notion of activation sequences to include real-time. We do that by simply attaching time stamps to the events. Formally, we say that a *timed activation sequence* is any finite or infinite sequence of pairs $(e_i, t_i)_{i=1,2,\ldots}$ where each e_i is either of the form $\mathrm{recompute(u)}$ or of the form $\mathrm{receive(v \Leftarrow u)}$ and where the sequence of time stamps $(t_i)_i$ is strictly increasing. We want to consider as valid only those timed activation sequences that respect the constraints imposed by the delay function:

A timed activation sequence $(e_i, t_i)_i$ is *valid* with respect to an initial state $(\mathcal{S}_0, \mathcal{Q}_0)$ if the corresponding "untimed" activation sequence $(e_i)_i$ is valid with respect to the same initial state *and* the following condition holds:

- For every i and every pair of peers (u, v), if $e_i = \mathrm{recompute}(u)$ is an event that causes a route P to be advertised (i.e. $P = \mathrm{best}(u) \neq \mathrm{rib}(u)$), then there exist j and k $(i < j < k)$ with the following properties:
  1) $e_j = \mathrm{receive}(v \Leftarrow u)$ is an event that puts P into $\mathrm{rib\_in}(v \Leftarrow u)$.
  2) $e_k = \mathrm{recompute}(v)$.
  3) $t_i < t_j < t_k \leq t_i + d(u, v)$.

This condition precisely expresses the intended intuition of edge delays. Notice that t_i is the time when u changes its route, t_j is the time when its peer v receives the new route and t_k is the time when v finally recomputes the best route. The condition simply requires that the time between t_i and t_k be bounded by $d(u, v)$.

When measuring convergence time, we are interested in the elapsed time since the first event. Because of that, it is convenient to have $t_1 = 0$. In addition to that, the destination node 0 should initially announce itself by advertising the trivial path (0) to its neighbors. In our model, this is achieved by having $e_1 = \mathrm{recompute}(0)$. Timed activation sequences which satisfy these two conditions are called *initialized timed activation sequences*. We will study convergence time by looking at protocol runs (i.e. state sequences) generated by valid initialized timed activation sequences.

IV. Convergence Time of SPVP

In section II we described how dispute digraphs can be used to prove convergence of SPVP. Here we show how they can be extended and used to analyze convergence *time*.

Given a TSPP instance $S' = ((V, E, d), \mathcal{P}, \Lambda)$, its *timed dispute digraph* $\mathcal{TDD}(S)$ is a weighted directed graph which is structurally identical to the corresponding dispute digraph. An arc from $P = (u \ldots)$ to $Q = (v \ldots)$ in the dispute digraph has the weight of $\Delta(P, Q) = d(u, v)$.

A timed dispute digraph is therefore a weighted dispute digraph, where weights correspond to certain edge delays. What is the intuition behind assigning weights in this way? Nodes u and v are peers and it is therefore possible that u adopts or rejects the path P at some time t_u, which causes v to adopt or reject the path Q at some later time t_v. This "chain reaction" can keep spreading further. Intuitively, $\Delta(P, Q)$ is meant to represent a bound on the "reaction time" between these two events. The first event (at time t_u) triggers the second one (at time t_v). So, $\Delta(P, Q)$ should represent an upper bound on $t_v - t_u$. Notice, however, that we have already introduced $d(u, v)$ to represent exactly this bound.

We will now proceed towards a proof of the key lemma that describes the fundamental connection between the timed dispute digraph and timing of the events. As a notational convenience, for two paths $P, Q \in \mathcal{P}^u$, we will use notation $P < Q$ to denote the fact $\lambda^u(P) < \lambda^u(Q)$. Similarly, $P \leq Q$ denotes the fact $\lambda^u(P) \leq \lambda^u(Q)$.

Lemma 1—Real-time correspondence: Let S be a TSPP instance and $\mathcal{TDD}(S)$ its timed dispute digraph. If we run SPVP on S using a valid initialized timed activation sequence $(e_i, t_i)_i$, then the following holds for any permitted path P: If a node

[3] We assume that all the conditions for SPP instances continue to hold.

goes up [4] to P or goes down [5] from P at time t, then there exists a path P^+ with the following properties:

1) $P \leq P^+$.
2) There is a path in $\mathcal{TDD}(S)$ from (0) to P^+ of the length at least t.

Before the proof, let us briefly discuss the significance of the lemma. The lemma says that, when real-time constraints are respected, if a node changes its route at time t, there must be a path in the timed dispute digraph of the length at least t. Moreover, that path starts with the node (0). [6] We say that *diameter* of a given timed dispute digraph is the length of the longest path starting at (0). Therefore, the lemma implies that no node can change its route after the time τ, where τ is the diameter of the timed dispute digraph. This means that the system necessarily stabilizes by the time τ:

Theorem 2—Bound on SPVP convergence time: Let S be a TSPP instance whose timed dispute digraph has the diameter τ. SPVP is then guaranteed to converge in time τ on S.

This theorem gives a general bound on convergence time of SPVP. Notice that if the timed dispute digraph has a cycle which is reachable from (0), its diameter is infinite and we do not have a bound on convergence time. We now present the proof of Lemma 1.

Proof: Since nodes can change their paths only at times t_i, we can prove the lemma by induction on i. Formally, we will prove the following statement by induction on i:

$$\forall i\ \forall u\ \forall P.$$
$$u \text{ goes (up to)/(down from) } P \text{ at time } t_i \Rightarrow \exists\ P^+ \ldots$$

Assume first $i = 1$. Since the activation sequence is initialized, we know that $t_1 = 0$ and $e_1 = \text{recompute}(0)$. Therefore, only node 0 can be active at this time and P must be (0). But then we can take the empty path starting at (0) as a path of length $t_1 = 0$ from (0) to P.

Now assume that the statement holds for all $i < k$, where $k > 1$. Let u be the node that goes up to P or down from P at time t_k. Since $t_k > 0$ (because $k > 1$), we know that $u \neq 0$, because node 0 does not change its path after the initial moment. This means that $P = (u\ v\ \ldots)$ for some neighbor v of u. There are two possible cases:

Case 1: u goes up to P at time t_k.
Then, because of the definition of valid activation sequences, we conclude that v must have advertised the path $P[v \ldots 0]$ at some time $t' \geq t_k - d(v,u)$ (the delay from v advertising the route till u adopting it is bounded by $d(v,u)$). There are two subcases:

- Case 1a: v went up to $P[v \ldots 0]$ at time t'.
  Since $t' < t_k$, this means that $t' = t_i$ for some $i < k$. By the induction hypothesis, we know that some path $R \geq P[v \ldots 0]$ is reachable in $\mathcal{TDD}(S)$ from (0) by a path of the length at least t'. There are two possible subcases with respect to u's preference:
  - Case $P \leq (u\ v)R$. Since there is a transmission arc from R to $(u\ v)R$ with the weight $d(v,u)$, we conclude that $(u\ v)R$ is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ := (u\ v)R$. Notice that $P \leq P^+$ because of the assumption of this subcase.
  - Case $(u\ v)R < P$. In this case we have a dispute arc from R to P with the weight $d(v,u)$, so we conclude that P is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ = P$.
- Case 1b: v went down to $P[v \ldots 0]$ (from some path Q) at time t'. As in case 1a, we conclude that $t' = t_i$ for some $i < k$. By the induction hypothesis, we know that there exists a path $Q^+ \geq Q$, which is in $\mathcal{TDD}(S)$ reachable from (0) by a path of the length at least t'. Also, since v went down from Q, we know that
  $$P[v \ldots 0] \leq Q \leq Q^+.$$
  As before, there are two possible subcases with respect to u's preference:
  - Case $P \leq (u\ v)Q^+$. Since there is a transmission arc from Q^+ to $(u\ v)Q^+$ with the weight $d(v,u)$, we conclude that $(u\ v)Q^+$ is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ := (u\ v)Q^+$.
  - Case $(u\ v)Q^+ < P$. In this case, we have a dispute arc from Q^+ to P with the weight $d(v,u)$, so we conclude that P is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ := P$.

Case 2: u goes down from P at time t_k.
Then it must be the case that v switched from the route $P[v \ldots 0]$ to some other route at some time $t' \geq t_k - d(v,u)$. Indeed, if v had held $P[v \ldots 0]$ throughout the interval $[t_k - d(v,u), t_k]$, then u would not have gone down from P. Also, if v had held routes different from $P[v \ldots 0]$ throughout the whole interval, u could not have had P just before the time t_k (u would have learned some of those other routes, since the delay time is bounded by $d(v,u)$).
There are two cases with respect to the direction in which v moved:

- Case 2a: v went up to some path Q from $P[v \ldots 0]$ at time t'. As before, we know that $t' = t_i$ for some $i < k$. By the induction hypothesis, we know that some path $Q^+ \geq Q$ is reachable from (0) by a path of the length at least t'. Also, since v went up to Q, we know that
  $$P[v \ldots 0] \leq Q \leq Q^+.$$
  There are two possible subcases with respect to u's preference:
  - Case $P \leq (u\ v)Q^+$. Since there is a transmission arc from Q^+ to $(u\ v)Q^+$ with the weight $d(v,u)$, we conclude that $(u\ v)Q^+$ is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ := (u\ v)Q^+$.

[4] We say that a node *goes up to* P if it switches to P from some less preferred path Q.
[5] We say that a node *goes down from* P if it switches from P to some less preferred path Q.
[6] Recall that nodes of the timed dispute digraph are paths of the original graph.

- Case $(u\ v)Q^+ < P$. In this case, we have a dispute arc from Q^+ to P with the weight $d(v,u)$, so we conclude that P is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ := P$.

- Case 2b: v went down from $P[v \dots 0]$ at time t'. As before, we know that $t' = t_i$ for some $i < k$. By induction hypothesis, we know that some path $R \geq P[v \dots 0]$ is reachable in $\mathcal{TDD}(S)$ from (0) by a path of the length at least t'. There are two possible cases with respect to u's preference:
  - Case $P \leq (u\ v)R$. Since there is a transmission arc from R to $(u\ v)R$ with the weight $d(v,u)$, we conclude that $(u\ v)R$ is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ := (u\ v)R$.
  - Case $(u\ v)R < P$. In this case, we have a dispute arc from R to P with the weight $d(v,u)$, so we conclude that P is reachable by a path of the length at least $t' + d(v,u) \geq t_k$. Therefore, we can take $P^+ := P$.

■

V. Applications

In this section we illustrate the power of Theorem 2 by applying it to three case studies:

- The first study shows how to derive convergence time when "shortest-path-first" policies are used. In fact, we prove a bound for more general *delay-cost consistent* policies.
- The second case study shows how to use Theorem 2 to prove the earlier result by Griffin, Wilfong and Shepherd about the convergence of "timeless" SPVP when the dispute digraph is acyclic.
- The third case study analyzes Gao and Rexford's proposal for ensuring convergence of SPVP [6]. The proposal represents some of the current best practices in configuring BGP policies. It uses the provider-customer hierarchy of the Internet to restrict the preference functions and the flow of advertisements in a way that results in provable convergence. We use Theorem 2 to establish an upper bound on convergence time for this case.

It is important to note that bounds on convergence time derived in these studies can be used *without* explicitly computing the timed dispute digraph. The most that is needed is topology and delay information. This is essential for usefulness of the results. Explicit computation of the dispute digraph is almost never feasible—partly because of its size and partly because of the often proprietary nature of BGP policies. Nevertheless, the bounds are derived from Theorem 2, which explicitly uses the diameter of the timed dispute digraph. Our case studies show that diameter of the timed dispute digraph can often be calculated (or at least bounded) without explicitly constructing the digraph.

A. Delay-cost Consistent Policies

One way to avoid inconsistent routing policies is to have the routers agree on a common *cost function* which is used to assign route preferences. This idea is presented in [4]. For a given TSPP (or an SPP) instance, a cost function is a strictly positive partial function $c : V \times V \rightharpoonup \mathbf{R}$ such that $c(u,v)$ is defined if and only if (u,v) is an edge. [7] Any cost function naturally extends to paths—if $P = (u_1\ u_2\ \dots\ u_k)$ is a path, then we define

$$c(P) := \sum_{i=1}^{k-1} c(u_i\ u_{i+1}).$$

We say that S is *consistent* with the cost function c (shortly, *cost-consistent*) if all routers prefer cheaper paths. Formally, for every node u and every two paths $P, Q \in \mathcal{P}^u$ the following holds:

$$c(P) < c(Q) \ \Rightarrow\ \lambda^u(Q) < \lambda^u(P).$$

For TSPP instances, we can also extend the delay function to paths. For $P = (u_1\ u_2\ \dots\ u_k)$, we define

$$d(P) := \sum_{i=k}^{2} d(u_i\ u_{i-1}).$$

Notice that the summation goes in the opposite direction from the one used in the cost formula. The intuition behind this is that delays bound the propagation time of advertisements that go from the destination *outwards*, while cost is supposed to measure how expensive is to send traffic along the route *towards* the destination.

It is natural to expect that delays will be reflected in the cost function in the sense that routes with longer delays should be considered more expensive for carrying traffic and vice-versa. This is captured by the following condition about consistency between delays and costs:

A TSPP instance S consistent with a cost function c is *delay-cost consistent* if for every node u and every two paths $P, Q \in \mathcal{P}^u$ the following condition holds:

$$d(P) < d(Q) \ \Rightarrow\ c(P) < c(Q).$$

One way to achieve delay-cost consistency is, for instance, by assigning costs according to the following rule:

$$c(u,v) = f(d(v,u)),$$

for some strictly increasing positive function f.

The following lemma establishes an upper bound on the length of a path in the timed dispute digraph of a delay-cost consistent instance. The bound is interesting in the sense that it depends only on the last node of the path.

Lemma 3: Let S be a delay-cost consistent instance with the cost function c. If $\Pi = (0) \rightarrow P_1 \rightarrow \dots \rightarrow P_k$ is a path in $\mathcal{TDD}(S)$, then its length is at most $d(P_k)$.

Proof: Let $L(.)$ denote the length function on paths in $\mathcal{TDD}(S)$. We will prove the lemma by induction on k.

If $k = 0$, we have a trivial path $\Pi = (0)$ which does not have edges and hence $L(\Pi) = 0 = d((0))$.

Assume the statement is true for every path in $\mathcal{TDD}(S)$ with less than m edges, for some $m \geq 1$. Let $\Pi = \Pi' \rightarrow P_m$ be

[7] The requirement for strict positivity can be weakened into the requirement that c does not result in any non-positive cycles in the underlying graph [4].

a path in $\mathcal{TDD}(S)$ with m edges, where $\Pi' = (0) \rightarrow P_1 \rightarrow \ldots \rightarrow P_{m-1}$ is its sub-path consisting of the first $m-1$ edges. Let us denote the first node of P_i by u_i (i.e. $P_i = (u_i \ldots)$). Then by the induction hypothesis, we know that

$$L(\Pi) = L(\Pi') + \Delta(P_{m-1}, P_m) \leq d(P_{m-1}) + d(u_{m-1}, u_m). \quad (1)$$

There are two possible cases with respect to the nature of the arc $P_{m-1} \rightarrow P_m$:

1) $P_{m-1} \rightarrow P_m$ is a transmission arc. Then $P_m = (u_m\ u_{m-1})P_{m-1}$. By the definition of path delays we have
$$d(P_m) = d(P_{m-1}) + d(u_{m-1}, u_m).$$
This is exactly the right-hand side of the equation 1, which means that $L(\Pi) \leq d(P_m)$.
2) $P_{m-1} \rightarrow P_m$ is a dispute arc. Then $P_m = (u_m\ u_{m-1})Q_{m-1}$, where $Q_{m-1} \leq P_{m-1}$. Because of the cost consistency, we know that $c(P_{m-1}) \leq c(Q_{m-1})$. Delay-consistency then implies that $d(P_{m-1}) \leq d(Q_{m-1})$. If we plug this into the right-hand side of the equation 1, we get:
$$L(\Pi) \leq d(Q_{m-1}) + d(u_{m-1}, u_m) = L(P_m).$$

■

We can now derive an upper bound on convergence time:

Corollary 4—Conv. time for delay-cost consist. instances: Let S be a delay-cost consistent TSPP instance with the maximum path delay $\delta = \max\{d(P) \mid P \in \mathcal{P}^u,\ u \in V\}$. Then S converges in time δ.

Proof: Because of Lemma 3, the diameter of $\mathcal{TDD}(S)$ is bounded by δ. But then Theorem 2 guarantees that S converges in time δ. ■

As a special case of delay-cost consistent policies, we look at shortest-path-first policies. BGP routers are often configured to simply prefer routes with smaller number of hops. Formally, we say that an SPP instance has *shortest-path-first* policies if it is consistent with the cost function that assigns unit costs to all edges:

$$c(u, v) = 1, \text{ for each edge } (u, v).$$

The following theorem estimates convergence time for instances with shortest-path-first policies:

Theorem 5—Conv. time for shortest-path-first policies: Let S be a TSPP instance with shortest-path-first policies where all edge delays are equal to ω. [8] Then S converges in time $D\omega$, where D is the length of the longest permitted path.

Proof: First notice that having all edge delays be equal to ω yields a delay-cost consistent instance, since $c(u, v) = \frac{1}{\omega} d(v, u)$ and consequently, $c(P) = \frac{1}{\omega} d(P)$ for every path P. The delay of a path P is equal to $|P| \cdot \omega$, where $|P|$ is the number of edges in P. Therefore, the maximum path delay will be $D\omega$, which by Corollary 4 implies that S converges in time $D\omega$. ■

This theorem is due to Labovitz et al. [8], but here we derived it as a special case of the more general Corollary 4 (which is ultimately based on Theorem 2).

[8] If edge delays are different, we can take ω to be the largest edge delay.

B. Convergence of "timeless" SPVP

Recall the sufficient condition for convergence of the "timeless" version of SPVP proved by T. Griffin, F. B. Shepherd and G. Wilfong in [4]:

Theorem 6—Convergence of SPVP: If S is an SPP instance whose dispute digraph is acyclic, then SPVP is guaranteed to converge on S under any valid initialized activation sequence.

Proof: Theorem 2 is essentially a refinement of this theorem. We will use it to prove this "timeless" version by showing that the timed protocol can simulate any scenario from the timeless protocol by simply adding the timing information in a consistent way.

Suppose we are given a valid initialized activation sequence $(e_i)_{i=1,2,\ldots}$ for S. We first construct the corresponding TSPP instance S' from S by simply making all edge delays equal to, say, 1. Then we use the activation sequence for S to construct an equivalent valid initialized timed activation sequence $(e_i, t_i)_{i=1,2,\ldots}$ for S' by appropriately adding the time stamps:

$$t_i = 1 - \frac{1}{2^{i-1}}.$$

Notice that these sequences produce the same protocol runs, since we did not change the order of events, but merely added the time stamps. Because of that, in order to show convergence under the original activation sequence, it suffices to show a finite bound on convergence time for the timed activation sequence.

Since $t_j - t_i < 1$ for any $j > i$, we know that all edge delays will be respected, so that $(e_i, t_i)_i$ really constitutes a *valid* timed activation sequence. Also, since $t_1 = 0$, the new sequence will be initialized. Because of the assumed acyclicity of the dispute digraph, the timed dispute digraph has a *finite* diameter. But Theorem 2 then guarantees that the sequence $(e_i, t_i)_i$ converges on S'. ■

C. Hierarchical SPVP

One approach for achieving convergence of SPVP (and BGP) is described in [6] by L. Gao and J. Rexford. Their idea is based on the provider-customer hierarchy of the Internet. This hierarchy is used as a basis for configuration guidelines which, if respected, guarantee convergence. Our goal is to estimate the convergence time in this case.

To define the hierarchy, we place *every* BGP peering edge in exactly one of the following two classes:

- *Customer-provider edges* exist between a *customer* and a *provider*. The provider is typically a larger AS (Autonomous System) that provides connectivity to the rest of the Internet for the customer. Providers may have even larger providers of their own and customers may have further customers.
- *Peer-to-peer edges* exist between two AS's of a comparable size, who mutually agree to exchange traffic between their respective *customers*. In particular, this means that peer-to-peer edges can only be included in routes that go from a (direct or indirect) customer of one peer to a (direct or indirect) customer of the other peer.

An instance of SPP which has this hierarchy is called a *hierarchical instance* of SPP. We will view the customer-provider edges as directed edges, oriented from the provider to the customer. In that sense, they form a directed graph which we always assume to be *acyclic*. Peer-to-peer edges are naturally undirected, since the peering relationship is symmetric. Figure 2 shows an example of this hierarchy. Customer-provider edges are represented with full lines, while peering edges are dashed.

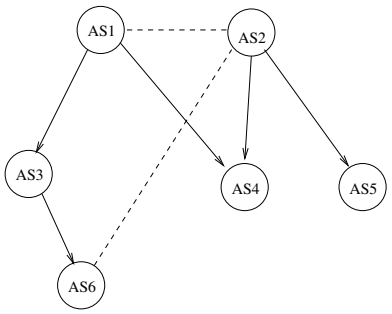

Fig. 2. Hierarchical topology

We can assume that we have a single BGP router for every AS. If r is a router, then provider(r), customer(r), and peer(r) respectively denote the sets of (immediate) *providers*, *customers* and *peers* of r. Let $P = (r\ r_1\ r_2\ \ldots\ r_k)$ be a route that r learns from another router. The notation $\mathrm{next}(P)$ is used for the next hop router on P. In this case, $\mathrm{next}(P) = r_1$. All of r's routes can be classified into three categories based on the next hop:

- Customer routes: $\{P \mid \mathrm{next}(P) \in \mathrm{customer}(r)\}$.
- Provider routes: $\{P \mid \mathrm{next}(P) \in \mathrm{provider}(r)\}$.
- Peer routes: $\{P \mid \mathrm{next}(P) \in \mathrm{peer}(r)\}$.

Convergence is assured by restricting two aspects of SPVP: flow of advertisements and routing policies. Figure 3 shows selective export rules that are used to restrict the flow of advertisements [9], [10]. The rules specify, for any given router, which routes should be advertised to which neighbors.

Fig. 3. Selective export rules

Type	Export to: provider	customer	peer
prov. route	NO	YES	NO
cust. route	YES	YES	YES
peer route	NO	YES	NO

The table says that no router should advertise its peer and provider routes to other peers and providers. Selective export rules impose a certain shape on the paths that can be produced:

A path $(r_1\ r_2\ \ldots\ r_k)$ is said to be *valley-free* if every provider-to-customer or a peer-to-peer edge can only be followed by a provider-to-customer edge. Formally, if for some $1 < i < k$, (r_{i-1}, r_i) is a provider-to-customer or a peer-to-peer edge, then (r_i, r_{i+1}) is a provider-to-customer edge.

In other words, a valley-free path consists of a sequence of zero or more customer-to-provider edges, followed by zero or one peer-to-peer edge, followed by a sequence of zero or more provider-to-customer edges.

In order to guarantee convergence, we need to supplement selective export rules by guidelines for configuring routing policies. Policy guidelines restrict the set of legal preference functions. The following basic policy guideline is suggested in [6]:

Guideline A: Each router should strictly prefer customer routes over non-customer routes.

Notice that this guideline is *local*, which means that each router can follow it independently of other routers, without disclosing its policies.

It turns out that guideline A, combined with selective export rules, suffices to guarantee convergence of SPVP. This was proved in [6].

In order to calculate convergence time by using Theorem 2, we need to analyze the shape of the dispute digraph. In particular, our analysis will be based on estimating the length of the paths in the dispute digraph. The first step is the following lemma:

Lemma 7: Let S be a hierarchical instance of SPP. Let $P = eP'$ and $Q = fQ'$ be permitted paths whose first edges are e and f respectively. Assume that $P \rightarrow Q$ is a (dispute or transmission) arc in the dispute digraph for S. Then the following holds: If f is a provider-to-customer or a peer-to-peer edge, then e is a provider-to-customer edge.

Proof: Let $f = (u\ v)$. There are two possible cases, corresponding to the type of the arc $P \rightarrow Q$:

1) $P \rightarrow Q$ is a transmission arc. Therefore, $Q' = P$. There are three subcases with respect to the edge f:
   - $f = (u\ v)$ is a customer-to-provider edge. We do not have to prove anything in this case.
   - $f = (u\ v)$ is a provider-to-customer edge. Because the transmission arc $P \rightarrow (u\ v)P$ exists, v can advertise P to its provider u. Selective export rules in that case guarantee that P is a customer route, which means that e is a provider-to-customer edge.
   - $f = (u\ v)$ is a peer-to-peer edge. Again, since we have a transmission arc $P \rightarrow (u\ v)P$, v can advertise P to its peer u. But then P must be a customer route, which means that e is a provider-to-customer edge.
2) $P \rightarrow (u\ v)Q'$ is a dispute arc. There are, again, three subcases with respect to the edge f:
   - $f = (u\ v)$ is a customer-to-provider edge. Again, we do not have to prove anything in this case.
   - $f = (u\ v)$ is a provider-to-customer edge. Since v can advertise Q' to its provider u (u would otherwise have no way of learning $Q = (u\ v)Q'$, so we could delete Q from the dispute digraph), Q' must be a customer route. However, since $P \rightarrow (u\ v)Q'$ is a dispute arc, v prefers P over Q'. In that case P must be a customer route as well, since a node can not prefer a non-customer route over a customer route (Guideline A). Therefore, e is a provider-to-customer edge.
   - $f = (u\ v)$ is a provider-to-customer edge. Again, since v can advertise Q' to its peer u, Q' must be a customer route. However, because there is a dispute, v prefers P over Q', so P must be a customer route

as well. Therefore, e is a provider-to-customer edge. ■

The claim about edges e and f from the lemma matches exactly the defining condition for valley-free paths. If we use that fact and inductively extend the lemma from arcs to arbitrary paths in the dispute digraph, we immediately get the following corollary:

Corollary 8: Let S be a hierarchical instance of SPP. Let $P_1 = e_1P_1'$, $P_2 = e_2P_2$, $\ldots, P_k = e_kP_k'$ be permitted paths, such that the dispute digraph for S contains the path $P_1 \to P_2 \to \ldots \to P_k$. Then the sequence of edges $e_ke_{k-1}\ldots e_1$ represents a valley-free path in S.

Let us try to interpret the result informally. Given a path in the dispute digraph, we can extract the first edges of all the all the nodes along that path (remember that nodes in the dispute digraph are *paths*). These edges, considered in the reverse order, will themselves form a path between some two routers. The corollary says that every path obtained this way will be valley-free. It is easy to see that with n routers, every valley-free path will contain at most $2n-1$ edges (at most $n-1$ customer-to-provider edges followed by at most one peer-to-peer edge followed by at most $n-1$ provider-to-customer edges). Therefore, the corollary implies that the length of every path in the dispute digraph is less than $2n$:

Corollary 9: If S is a hierarchical instance of SPP, every path in $\mathcal{DD}(S)$ has the length less than $2n$.

When delay constraints are added to hierarchical SPP instances, we talk about *hierarchical TSPP instances*. As a direct consequence of the Corollary 9, we derive the following theorem which gives an upper bound on convergence time:

Theorem 10—Convergence time of hierarchical SPVP: Let S be a hierarchical TSPP instance with n routers and let $M = \max\{d(u,v) \mid (u,v) \text{ is an edge}\}$ be the maximum edge delay. Then SPVP converges on S in time $2Mn$.

Proof: Because of Corollary 9, the diameter of $\mathcal{TDD}(S)$ is bounded by $2Mn$, so Theorem 2 guarantees convergence in time $2Mn$. ■

VI. Conclusions

The work presented in this paper has two major points of interest: real-time model of BGP and a general theorem providing an upper bound on convergence time.

Previous attempts at analyzing BGP convergence time [8], [11] assumed that all routers use shortest-path-first policies. While this is a valid assumption in many practical instances, BGP standard [2] specifically allows more general preference policies which do not fall in this category.

On the other hand, the authors who did study BGP with general preference policies [3], [4], [5], [6] focused solely on convergence vs. divergence, rather than on convergence *time*.

Our results in some sense unify these two research directions. As we show in Section V, our Theorem 2 can be used to estimate convergence time as well as to purely prove convergence. It is important to note that, although Theorem 2 is based on a rather abstract notion of a timed dispute digraph, we often do not need to compute the graph in order to use the theorem. Probably the best illustration of this fact is Theorem 10: the proof is crucially based on timed dispute digraphs, but the final bound depends only on the number of nodes and the maximum edge delay—even the exact topology is irrelevant.

Our real-time model is an extension of the SPP/SPVP model used in [3], [4]. It is designed to be general enough to capture most aspects of BGP-like protocols and at the same time simple enough to allow feasible formal reasoning. In order to be able to accomplish both tasks, the model makes certain simplifying assumptions about the protocol. As the authors point out in [3], the model ignores address aggregation and internal BGP. Basically, the model represents the network abstractly as a set of interconnected autonomous systems (with unknown internal structure), rather than a set individual routers. Also, we model route propagation by messages carrying a single route. This does not appear to be a substantial simplification, as BGP *update* messages containing multiple routes can be modeled as sequences of single-route messages. In addition to that, route withdrawals can be modeled by messages containing an empty path ϵ.

The main challenge in practically applying the real-time aspects of the model is estimating edge delays. While it is clear that *existence* of edge delays is necessary for estimating convergence time (as argued in Section III), it is not so clear how should one go about calculating them. An edge delay depends on the underlying network, traffic congestion, and particular BGP implementations used at its endpoints. A plausible strategy for estimating edge delays would most likely involve measuring route propagation time across the edge under heavy-traffic conditions. It is probably hard to estimate the absolutely worst propagation time possible (or it would turn out to be infinite), but measurement could provide a reasonable idea of how bad can it get. At the present time, the author has not conducted any experiments of that kind.

Finally, we should comment on our assumption about the static topology and static policies of the routers. Although this may appear to be a strong constraint, in most cases this is the best one can do. Indeed, topology changes can be so frequent that even the most efficient protocol would not have enough time to converge. Instead, we take the practical stand that topology and policies in reality change only every so often. Our results are meant to be applied to periods *in between* these changes. Convergence is guaranteed as long as the estimated convergence time is shorter than the time between topology/policy changes. This is a common assumption for studies of convergence time of routing protocols (e.g. see [12], [13]).

The best way to ensure convergence of BGP while allowing general routing policies is still an open problem. It is likely that more solutions to this problem would be proposed in future. The author believes that the analysis of timed dispute digraphs (or some equivalent structures) can be used as a general technique for evaluating these proposals from the standpoint of convergence time. A more in-depth analysis of the subject can be found in [14].

Acknowledgements

The author would like to thank Rajeev Alur, Karthikeyan Bhargavan, Alwyn Goodloe, Timothy G. Griffin, Roch Guerin,

Carl A. Gunter, Pankaj Kakkar, Insup Lee, Michael McDougall, and Alvaro Retana for their suggestions and support.

REFERENCES

[1] John W. Stewart III, *BGP4 (Inter-Domain Routing in the Internet)*, Addison-Wesley, 1998.

[2] Y. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4)," RFC 1771, IETF, March 1995.

[3] Timothy G. Griffin and Gordon Wilfong, "An analysis of BGP convergence properties," in *Proceedings of ACM SIGCOMM '99 Conference*, Guru Parulkar and Jonathan S. Turner, Eds., Boston, August 1999, pp. 277–288.

[4] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong, "Policy disputes in path-vector protocols," in *Proceedings of ICNP '99 Conference*, Toronto, Canada, October 1999.

[5] Timothy G. Griffin and Gordon Wilfong, "A safe path vector protocol," in *Proceedings of INFOCOM 2000 Conference*, Tel Aviv, Israel, March 2000.

[6] Lixin Gao and Jennifer Rexford, "Stable internet routing without global coordination," in *ACM SIGMETRICS*, 2000.

[7] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," ISI Technical Report 96-631, USC/Information Sciences Institute, 1996.

[8] Craig Labovitz, Roger Wattenhofer, Srinivasan Venkatachary, and Abha Ahuja, "The impact of internet policy and topology on delayed routing convergence," in *Proceedings of INFOCOM 2001*, Anchorage, Alaska, April 2001.

[9] C. Alaettinoglu, "Scalable router configuration for the internet," in *Proceedings of the 1996 International Conference on Networking Protocols.*, October 1996.

[10] G. Huston, "Interconnection, peering, and settlements," in *Proceedings of INET*, June 1999.

[11] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "An experimental study of internet routing convergence," Technical Report MSR-TR-2000-08, Microsoft Research, 2000.

[12] Karthikeyan Bhargavan, Carl A. Gunter, and Davor Obradovic, "RIP in SPIN/HOL," in *Theorem Proving in Higher-Order Logics (TPHOLs)*, Portland, OR, August 2000.

[13] Dimitri P. Bertsekas and Robert Gallager, *Data Networks*, Prentice Hall, 1991.

[14] Davor Obradovic, *Formal Analysis of Routing Protocols*, Ph.D. thesis, University of Pennsylvania, 2001.