

Lecture 1: Interdomain routing

1 ISP structure and economics

Structurally, the Internet is a network of interconnected networks. Each network is called an autonomous system (AS). An AS is a network of routers managed by a single administrative entity. We will use the term *AS* and the more common term *Internet service provider (ISP)* interchangeably although large ISPs are often divided into multiple ASes for administrative purposes. The ASes may either connect to each other privately or through a network access point (NAP) where several ASes interconnect at a common exchange point.

ASes are organized as a tiered hierarchy. Tier-1 ASes are at the top level of this hierarchy and provide transit service typically to tier-2 ASes that in turn provide transit typically to tier-3 ASes and so on. At the bottom of this hierarchy are *stub* ASes that provide connectivity to end users. Tier-1 ASes are large and it is estimated that there are only about ten of them in the Internet today. The AS structure is not a strict hierarchy, i.e., there are interconnections between ASes at the same tier. A tier-3 AS may also obtain transit service directly from a tier-1 AS. In practice, tiers only provide a loose characterization of the AS hierarchy as the tier of an AS is not always known publicly nor is it an unchanging designation. Tiers merely reflect the economic relationships between interconnected ASes that themselves evolve over time.

The economic relationship between interconnected ASes commonly take one of two forms: a *provider-customer* or a *peering* relationship. In a provider-customer relationship, the customer AS pays the provider AS for transit service. Peering ASes typically do not charge each other for transiting traffic between their respective customers. The exact nature of the contract between interconnected ASes is the outcome of bilateral negotiation and is not disclosed publicly. Whether two ASes negotiate a provider-customer or a peering contract depends on their perceived value of interconnecting with each other. A regional AS is likely to perceive more value to connecting with a nationwide AS than the other way round, so the former is likely to become a customer of the latter (just like you choose to pay to connect your home network to your ISP instead of the other way round). On the other hand, two ASes with comparable size and coverage are likely to negotiate a peering relationship. The negotiation may also take into account the amount of traffic exchanged in either direction, e.g., peering ASes typically send and receive comparable volumes of traffic between each other, while traffic volumes between a provider and customer may be highly asymmetric, especially considering the download-dominated nature of Internet traffic.

Note that a customer always pays a provider irrespective of which direction (from/to the customer) the traffic flows. This aspect of Internet service pricing is fundamentally different from long-distance telephone services where typically the initiator of a call pays for the call while the recipient does not. Why the difference? The difference in the pricing schemes between Internet service and telephone service can be attributed to the nature of their traffic and resource allocation schemes. A telephone call has a call setup phase that clearly identifies the initiator and traffic flow in both directions is more or less balanced, so it suffices to charge just the initiator. Charging the initiator also avoided the scenario where a receiver of the call had to worry about the cost of the call without knowing its origin (at least before caller identification became widespread). In contrast, Internet traffic does not lend itself well to an initiator-based pricing model. Consider a typical Web transaction where an end-user downloads content from a web site. One would be inclined to identify the end-user as the initiator of the connection, but most of the traffic flows from the web server to the end-user. Who should pay for this traffic and how does the network identify this entity? Who is responsible for additional TCP connections created by programs at either end? In the Internet, both parties—the end-user and the web server—pay their respective ISPs. The network layer is stateless and does not attempt to account for resource usage on a per-connection basis. Instead, economic settlement happens on a bilateral basis between neighboring ASes based on the total volume of traffic exchanged periodically.

Provider-customer contracts may use one of several different pricing schemes such as *average*, *max*, or *committed information rate* (CIR) pricing. All of these schemes are based upon the customer’s bandwidth usage that is computed periodically, say once every five minutes, and this set of usage values is used to determine the customer’s bill at the end of the month. The average and max pricing schemes use the average and the 95th percentile usage values respectively. The CIR scheme charges a prepaid amount for any usage below a committed rate while charging a contractually agreed upon rate for rates above the CIR when excess capacity is available. Other pricing schemes determined by the usage needs of customers and profit motives of providers may exist.

Next, we look at BGP, the protocol used by ASes to determine the end-to-end interdomain route based on pairwise exchanges of routing announcements.

2 BGP overview

The border gateway protocol (BGP) is the Internet’s interdomain routing protocol. BGP has two components: eBGP or external BGP, and iBGP or internal BGP. Routers at the border of an AS run eBGP to announce and receive AS paths to/from border routers in neighboring ASes. All routers run iBGP to re-announce routes learned via eBGP to internal routers and to rank and select routes based on policy criteria as described below.

A route announcement in BGP include the following attributes: (1) **NLRI** or network layer reachability information that identifies the address prefix of the destination, (2) **Next_Hop** that identifies the next-hop router along the path, (3) **AS_Path**, an ordered list ASes traversed along the path, (4) **Local_Pref** that indicates how much the AS prefers the path and is only used internally by routers in an AS (and not passed between ASes), (5) **MED** or multiple exit discriminator that is used by an AS to tell a neighboring AS which of the multiple peering points should be used by the latter to send traffic to the former.

Routes get transformed as they propagate through the network in three steps. First, when a route propagates from a node w to a neighboring node u , node w applies its export policy to the route. Second, w applies a *path vector transformation* by adding itself to the route’s AS_Path, setting Next_Hop accordingly, and filtering the route if its AS_Path contains u . Finally, node u applies its *import policy* to the route. In particular, the import policy sets the Local_Pref attribute. We refer to the three steps in combination as a *peering transformation*.

Routers compute routes by ranking routes received from neighboring routers and choosing the best route. Interdomain concerns take precedence over intradomain concerns in this ranking procedure. The most important criterion is *local preference* that is a policy decision, which is why interdomain routing is also referred to as policy routing. For example, an AS prefers to route via a customer over a peer or a provider, and a peer over a provider, in keeping with its economic objectives. The second criterion is the length of the AS path. Although the length of the AS path is somewhat correlated with its delay performance, the ordering of the first two criteria attest to the fact that policy takes precedence over user-perceived performance in interdomain routing. The MED attribute is relevant only when there are multiple peering points between two ASes, which is typically the case between tier-1 ASes. The fourth criterion is to check if the route was learned from a router in a neighboring AS via eBGP or from an internal router via iBGP. The fifth criterion is commonly referred to as *hot potato* or *early-exit* routing. Just like you would not like to hold a hot potato in your hands for long, ASes try to get packets out of their network quickly by picking a route with the lowest IGP cost to the egress router. The last criterion is a tie-breaker so as to ensure a strict ordering of preferences for routes.

2.1 The stable paths problem: A model for policy routing

This section presents a simple model for eBGP referred to as the *stable paths problem* (SPP) as introduced by Griffin et al. [1]. The model treats each AS as a single node. Let $G = (V, E)$ be an undirected graph representing the physical AS topology where $V = \{0, 1, 2, \dots, n\}$ is the the set of nodes and E is the set of edges. Since routes to different destinations are computed independently, we explain the protocol assuming that a single *origin* node that serves as the destination to which all other nodes attempt to establish routes. A path is said to be *permitted* at a node u if a sequence of peering transformations results in a route being received at u .

Priority	Attribute	How used
1	Local_Pref	Highest Local.Pref, e.g., prefer customer routes over peer or provider routes.
2	AS_Path	Shortest AS path
3	MED	Lowest MED preferred. Typically ignored unless explicitly negotiated by contract.
4	eBGP > iBGP	Routes learned via eBGP are preferred to routes learned via iBGP.
5	IGP path	Pick route with lowest IGP cost to egress router.
6	Router ID	A tie-breaker based on the smallest router IP address.

Table 1: BGP route selection criteria.

Let P^v denote the set of *permitted paths* from a node v to the origin. Let $\lambda^v(P)$ denote a ranking function that assigns a nonnegative integer rank to a permitted path P . If $P_1, P_2 \in P^v$ and $\lambda^v(P_1) < \lambda^v(P_2)$, then P_1 is said to be preferred over P_2 . We assume that for all nodes v other than the origin (1) the empty path ϵ is permitted and is the lowest-ranked path, (2) the ranking among paths is strict, i.e., paths with different next-hop nodes have different ranks, (3) the paths do not have cycles or repeated nodes¹.

A *path assignment* is a function π that maps each node $u \in V$ to a path $\pi(u) \in P^u$. Note that $\pi(0) = \{0\}$. The set of paths *choices*(π, u) for $u \neq 0$ is defined to be

$$\text{choices}(\pi, u) = \{(u, v)\pi(v) \mid \{v \in \text{neighbors}(u)\} \cap P^u\} \quad (1)$$

i.e., the set of permitted paths formed by extending the paths assigned to neighbors of u . The path assignment π is stable at node u if

$$\pi(u) = P \in \text{choices}(\pi, u) \text{ with the highest } \lambda^u(P) \quad (2)$$

The path assignment π is stable if it is stable at all nodes. Note that the strictness of rankings implies that the above condition is well defined and that if $\pi(u) = (u, w)P$, then $\pi(w) = P$. The path $\pi(u)$ is empty iff *choices*(π, u) is empty. Thus, any stable path assignment implicitly defines a tree rooted at the origin. However, this may not be a spanning tree as some nodes may not have a nonempty path.

2.1.1 Examples

Next, we illustrate several examples of the stable paths problem. Figure 1(a) shows an example where a unique stable path assignment exists. The ranked set of permitted paths is shown as a vertical list next to each node. The bold arrows in Figure 1(b) reflect the stable path assignment, i.e., the paths (1,3,0), (2,0), (3,0), (4,3,0) respectively for nodes 1, 2, 3, and 4. The example in Figure 1(c), referred to as NAUGHTY_GADGET, adds just one additional permitted path at the top of node 3’s ranked list in the previous example. This gadget has the same unique stable path assignment as the previous example, but a BGP-like protocol may diverge for this problem, i.e., it may oscillate for arbitrarily long between other configurations before converging to the stable configuration. Finally, Figure 1(d) shows the BAD_GADGET, which does not have a stable path assignment, so a BGP-like protocol can never converge in this scenario.

Some stable paths problems may permit multiple solutions. For example, Figure 2(a) shows the DISAGREE_GADGET that has two solutions as shown in Figures 2(b) and (c).

2.2 Simple Path Vector Protocol: A model for BGP

The shortest path vector protocol (SPVP) is a simple model for how BGP addresses SPP. In SPVP, adjacent nodes exchange messages that are simply paths. When a node adopts a path P , it sends the path P to each of its neighbors. Each node maintains a route information base that consists of two data structures. The

¹“Path prepending”, a commonly used traffic engineering mechanism does not satisfy this assumption.

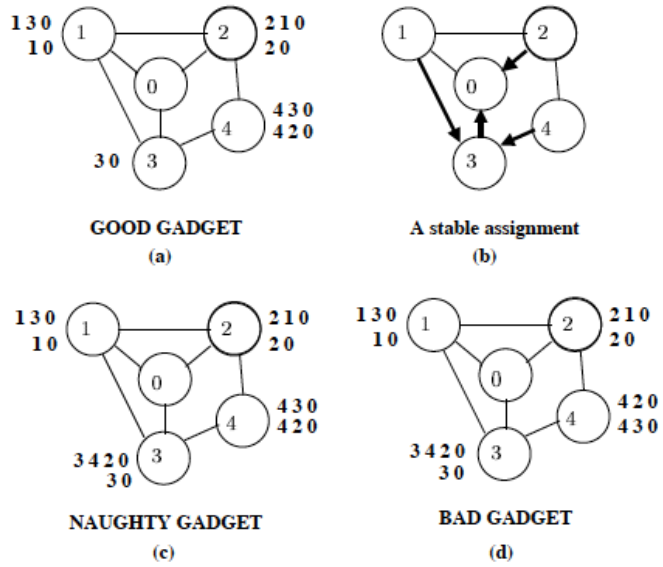


Figure 1: Examples of stable paths problems that are not shortest path problems

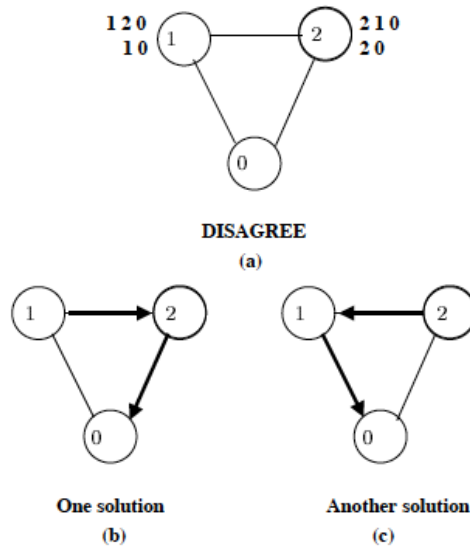


Figure 2: DISAGREE_GADGET and its two solutions.

path $rib-in(u)$ is u 's current path to the destination. The path $rib-in(u \leftarrow w)$ stores the most recent path processed at u and received from w . The set of path choices available at u is defined to be

$$choices(u) = \{(u, w)P \in P^u \mid P = rib-in(u \leftarrow w)\} \quad (3)$$

All messages are assumed to be processed using a reliable FIFO message queue between neighbors. A node updates $rib(u)$ upon receiving a path P from some neighbor w by executing the following two steps atomically: (1) Set $rib-in(u \leftarrow w) = P$; (2) Let P' be the path with the highest rank in $choices(u)$. If $rib(u) \neq P'$, set $rib(u) = P'$ and send P' to all neighbors.

The *network state* of the system is the collection of values $rib(u)$ and $rib-in(u \leftarrow w)$ and all messages queued at nodes or in transit on communication links. The network state implicitly defines a path assignment $\pi(u) = rib(u)$. The network state is *stable* if there are no queued or in-transit messages.

With the above assumptions, we are ready to state some important theoretical results characterizing SPP and SPVP. The proofs of all of the theorems in this section are described in the paper by Griffin et al. [1].

Theorem 2.1 *The problem of determining whether an instance of Stable Paths Problem is solvable is NP-complete.*

Theorem 2.2 *The path assignment associated with any stable state in SPVP is a stable path assignment, and thus a solution to the Stable Paths Problem. Therefore, if the Stable Paths Problem is unsolvable, then SPVP never converges to a stable state.*

Note that the converse of the above theorem is not true, i.e., solvability of the Stable Paths Problem does not imply that SPVP will converge to a stable state. The NAUGHTY_GADGET and DISAGREE_GADGET are both examples showing the same.

2.3 Dispute Wheels

Given the NP-completeness of the solvability problem, we look for a simple sufficient condition for SPVP to converge to a stable state (implying solvability of the Stable Paths Problem). This condition is the absence of a preference structure referred to as a *dispute wheel*.

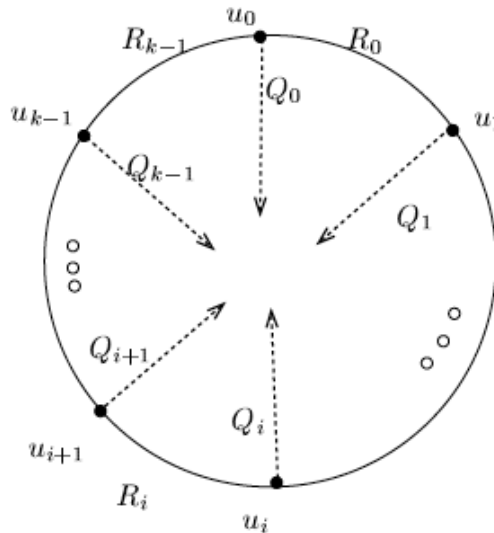


Figure 3: A dispute wheel.

Formally, a *dispute wheel* is a 3-tuple consisting of a sequence of nodes u_0, u_1, \dots, u_{k-1} , and sequences of nonempty paths Q_0, Q_1, \dots, Q_{k-1} and R_0, R_1, \dots, R_{k-1} , such that for each $0 \leq i \leq k-1$ we have 1) R_i is

a path from u_i to u_{i+1} , 2) $Q_i \in P^{u_i}$, 3) $R_i Q_{i+1} \in P^{u_i}$, and 4) $\lambda^{u_i}(Q_i) \leq \lambda^{u_i}(R_i Q_{i+1})$, where all subscripts are interpreted modulo k . Figure 3 shows an illustration.

Both NAUGHTY_GADGET and BAD_GADGET have a dispute wheel. Can you identify them?

Theorem 2.3 *If a Stable Paths Problem has no dispute wheel, then it has a unique solution and SPVP converges to the corresponding stable state.*

Furthermore, any Stable Paths Problem obtained by deleting a subset of edges and permitted paths containing those edges in the original problem, also does not have a dispute wheel.

Note that examples such as BAD_GADGET and NAUGHTY_GADGET and others where SPVP diverges involve some nodes preferring longer paths (in terms of hop count) over shorter ones. It can be formally shown that such preferences are indeed necessary for SPVP to diverge. Suppose edges were annotated with weights and the length of a path defined as the sum of the weights of its edges. Then, assuming that the graph does not contain any negative cycles (noting that positive weights trivially satisfy this assumption), we have

Theorem 2.4 *If each node always prefers a shorter path over a longer path to a destination, then the Stable Paths Problem does not have a dispute wheel.*

3 BGP in practice

3.1 Typical routing policies

Routing policies in the real-world may not always be consistent with some intuitive notion of shortest paths. However, persistent instability is rarely observed suggesting that dispute wheels are rare in practice. Why is this? Gao and Rexford [2] investigated this question and formalized a set of policy guidelines to ensure that BGP converges to a stable state. Nicely, these guidelines and their implicit assumptions are also believed to be largely obeyed in practice because of the nature of business relationships between ISPs in practice.

Recall that AS relationships are of two types: *customer-provider*, or *peer-to-peer*. The customer-provider and peer-to-peer agreements translate into the following rules governing BGP export policies:

- **Exporting to a provider:** In exchanging routing information with a provider, an AS can export its routes and the routes of its customers, but can not export routes learned from other providers or peers.
- **Exporting to a customer:** In exchanging routing information with a customer, an AS can export its routes as well as routes learned from its providers and peers.
- **Exporting to a peer:** In exchanging routing information with a peer, an AS can export its routes and the routes of its customers, but can not export the routes learned from other providers or peers.

The above rules are also referred to as *valley-free* routing. This term likely originated from the first rule above that precludes “valleys” or path segments of the form A-B-C where B is a customer of both A and C.

Recall that customer-provider relationships between ASes form a hierarchy. The hierarchical structure arises in practice because an AS typically chooses a much larger AS as its provider, so it is unlikely that a nationwide AS is a customer of a metropolitan AS. Similarly, if an AS u is a customer of v and v is a customer of w , then w can not be a customer of u . We formalize this observation into a more general rule: *any direct or indirect provider of an AS can not also be that AS’s customer*. An indirect provider is any AS that can be reached by following more than one provider link in sequence, e.g, AS w is an indirect provider of u above. Formally, the subgraph consisting of only provider-to-customer links (or only customer-to-provider links) is a DAG. Gao and Rexford [2] show the following theorem.

Theorem 3.1 *For a BGP system that has only customer-provider and peer-to-peer relationships, if all ASes strictly prefer customer paths over peer or provider paths, then the BGP system converges to a stable state.*

The restriction that ASes strictly prefer customer paths over peer paths can be relaxed in the theorem above allowing a customer path to have the same preference as a peer path provided the AS hierarchy satisfies

the following condition: an AS can not have a peer-to-peer relationship with a direct or indirect provider. Note that customer paths must still be strictly preferred over provider paths.

The above theorem provides a sufficient condition for BGP convergence. Note that if the AS topology and relationships were completely known, then the conditions in the theorems above can be checked efficiently. For example, cycles in a directed graph can be detected in time linear in the size of the graph. It can be verified that the assumptions above about—(1) export policies, (2) the hierarchical structure precluding an AS from being its indirect provider or peer-provider, and (3) ASes preferring customer paths over non-customer paths—in conjunction imply the absence of a dispute wheel that is a sufficient condition for SPVP convergence. Can you explain which of the above assumptions NAUGHTY_GADGET or BAD_GADGET fail to satisfy?

Customer-provider and peer-to-peer are the two most common relationships between neighboring ASes. However, an AS may also have a backup relationship with a neighboring AS. Having a backup relationship with a neighbor is important to maintain connectivity in face of link failures.

Theorem 3.2 *If all ASes (1) strictly prefer a route containing no backup links to a route containing a backup link, (2) strictly prefer a customer route over a provider route, and (3) prefer (possibly not strictly) a customer route over a peer route, then the BGP system converges to a stable state.*

3.1.1 Relationship inference

AS relationships are not publicly disclosed, but they can be inferred in practice. Although we do not have a complete map of the Internet’s AS-level *physical topology*, we do have the ability to observe a sizable portion of the AS *routing topology* by conducting measurements from distributed vantage points [9, 10].

These observed routes can be used to infer relationships between AS using a heuristic inference algorithm [8]. If ASes obey the valley-free export rules listed above, then the resulting AS paths must follow zero or more customer-to-provider links, zero or one peer-to-peer link, and zero or more provider-to-customer links. The heuristic inference algorithm attempts to discover inter-AS relationships such that the number of instances where valley-free routing is violated is minimized. Similarly, if one can monitor BGP routing announcements from distributed vantage points, this information combined with the assumption that ASes prefer customer routes over peer routes and peer routes over provider routes can be used to further improve the accuracy of the inference algorithm.

3.2 Message complexity, convergence delay, and timers: The devil in the details

The discussion so far has focused on stability and eventual convergence ignoring the underlying message complexity and delay. We implicitly assumed that the message processing overhead in BGP is small, i.e., each node has sufficient resources to process all messages sent (reliably) by its neighbors. However, the number of messages generated by BGP during convergence can be prohibitively high [3]. Labovitz et al. [4] investigated the message complexity of BGP and showed it to be super-exponential.

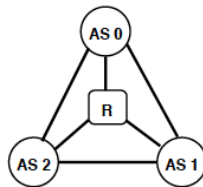


Figure 4: Example of an AS topology to illustrate BGP’s bouncing problem.

Consider the four-node topology as shown in Figure 4 where ASes 0, 1, and 2 are computing a route to a prefix owned by AS R. Suppose that in the steady-state, each AS adopts the direct, shortest route through R to reach the prefix. Now suppose that that R withdraws all routes to the prefix, say because of a link failure. Clearly, none of the other 3 ASes can have a valid route to the prefix at this point. Unfortunately,

the ASes can take up to 48 iterations to realize that that is indeed the case, and in the meantime attempt to use spurious routes that result in forwarding loops or blackholes.

Stage	Time	Routing Tables	Messages Processing	Messages Queued in System
0	N/A	steady state 0(*R, 1R, 2R) 1(0R, *R, 2R) 2(0R, 1R, *R)		
1	N/A	R withdraws its route 0(-, *1R, 2R) 1(*0R, -, 2R) 2(*0R, 1R, -)	R->0 W R->1 W R->2 W	0->1 01R 1->0 10R 2->0 20R 0->2 01R 1->2 10R 2->1 20R
2	N/A	1 and 2 receive new announcement from 0 0(-, *1R, 2R) 1(-, *, 2R) 2(01R, *1R, -)	0->1 01R 0->2 01R	1->0 10R 2->0 20R 1->0 12R 2->0 21R 1->2 10R 2->1 20R 1->2 12R 2->1 21R
3	N/A	0 and 2 receive new announcement from 1 0(-, *, 2R) 1(-, -, *2R) 2(*01R, 10R, -)	1->0 10R 1->2 10R	2->0 20R 1->0 12R 2->0 21R 0->1 02R 2->0 201R 2->1 20R 1->2 12R 2->1 21R 0->2 02R 2->1 201R
4	N/A	0 and 1 receive new announcement from 2 0(-, -, -) 1(-, -, *20R) 2(*01R, 10R, -)	2->0 20R 2->1 20R	1->0 12R 2->0 21R 0->1 02R 2->0 201R 0->1 W 1->0 120R 1->2 12R 2->1 21R 0->2 02R 2->1 201R 0->2 W 1->2 120R
5	N/A	0 and 2 receive new announcement from 1 0(-, *12R, -) 1(-, -, *20R) 2(*01R, -, -)	1->0 12R 1->2 12R	2->0 21R 0->1 02R 2->0 201R 0->1 W 1->0 120R 0->1 012R 2->1 21R 0->2 02R 2->1 201R 0->2 W 1->2 120R 0->2 012R
6	N/A	0 and 1 receive new announcement from 2 0(-, *12R, 21R) 1(-, -, -) 2(*01R, -, -)	2->0 21R 2->1 21R	0->1 02R 2->0 201R 0->1 W 1->0 120R 0->1 012R 1->0 W 0->2 02R 2->1 201R 0->2 W 1->2 120R 0->2 012R 1->2 W
(steps omitted)		steady state		
48	N/A	0(-, -, -) 1(-, -, -) 2(-, -, -)		

Figure 5: Step-wise exploration of spurious paths of increasing length resulting in a factorial number of messages.

Figure 5 shows in detail the steps that cause the long period of spurious path exploration. In the column marked “Routing Table”, the asterisked path denotes the currently chosen path. In the last two columns, the notation $X \rightarrow Y P$ means that X announces to Y the route P , and the route “W” means a withdrawal, i.e., no route is available. The figure shows that the ASes may go through 48 rounds exploring all possible loop-free AS paths (of up to length 4) before realizing that no path is available. Note that the example illustrates a theoretical worst-case scenario, however the super-exponential message complexity combined with the fact that peering cliques like the one shown in this example are common among tier-1 ASes motivate mechanisms to explicitly limit spurious path exploration.

3.2.1 Minimum route advertisement interval

In order to limit the exploration of spurious paths and the associated message complexity, BGP uses a hold-down timer called the `MinRouteAdvertisementInterval` (MRAI). The MRAI timer has a recommended default value of about 30 seconds. After a route to a prefix has been advertised to a peer, subsequent updates are held back from that peer until the MRAI timer expires. The MRAI timer tries to ensure that a node processes an update from all of its peers before processing a second update from the same peer. This reduces the exploration of spurious paths. For example, in the complete graph example above, the MRAI timer ensures that there are no pending announcements for paths of length k when a $(k + 1)$ -length path has already been announced by some node. Thus, only paths increasing monotonically in length are explored by each node reducing the message complexity from $O(n!)$ to $O(n)$ [4].

3.2.2 Route flap damping

Although the MRAI timer reduces the number of messages during convergence, it can not suppress route instabilities caused by extraneous factors such as unstable links or policy decisions. Route flap damping (RFD) was designed to reduce route flux at more coarse-grained time scales. A simple way to think about the two mechanisms is that MRAI is designed to reduce the number of messages for each new routing event (as identified by some root cause, e.g., a link failure, recovery, or policy change) to converge while the RFD timer is designed to limit the rate at which such routing events associated with the same prefix occur.

RFD works by suppressing routes to unstable prefixes. For each prefix D and each peer N , a router maintains a penalty $p[D, N]$ that is incremented every time the router receives an update from N for D . As shown in Figure 6, when the penalty crosses a threshold called the *suppression threshold*, the route is

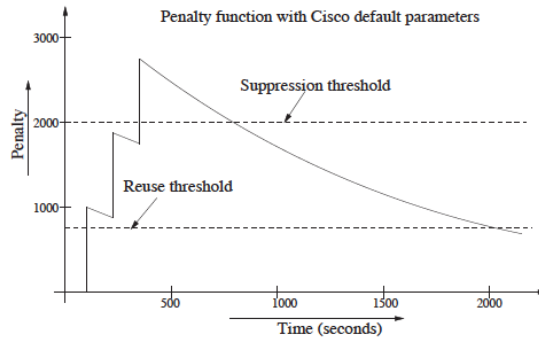


Figure 6: An example showing how route flap damping exacerbates convergence delays even under typical scenarios.

suppressed. At all times, the penalty value exponentially decays with a configured half life (typically 15 minutes). After exceeding the suppression threshold, when the penalty value decays to a lower threshold called the *reuse threshold*, the routes to D from N are allowed to be reused.

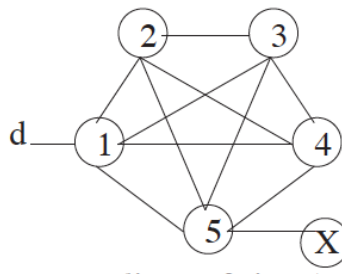


Figure 7: An example topology to show how route flap damping exacerbates convergence delays even under typical scenarios.

Although RFD timers are intended to limit route flux caused by unstable prefixes, Mao et al. [6] show that route flap damping can actually significantly delay Internet convergence times even for relatively stable prefixes. This example is shown in Figure 7 where a prefix d is homed with an AS 1 that is part of a 5-node clique one of which, AS 5, is connected to AS X. Even if AS 1 flaps the route to d twice, once by withdrawing it and again by re-announcing it (say, because of a brief link failure), AS X will suppress the route to d rendering d unavailable for several minutes. The steps showing why this can happen are listed in Figure 8. Essentially, the withdrawal results in multiple rounds of spurious path exploration before X realizes that no route to d is available. The changing routes to d during this period of exploration suffice to cause the penalty to exceed the suppression threshold.

3.3 Inconsistency: Why BGP suffers from unavailability

The examples in the previous sections make it clear that BGP is a complex protocol. The underlying topology, routing policies, and timers can interact in counterintuitive ways to cause prefixes to become unreachable. How can we tame this complexity? Does the complexity lie in the problem being solved or in the design of the particular protocol, BGP, being used to solve it? One school of thought suggests that it is the latter, i.e., unavailability in BGP fundamentally stems from an inconsistent view between routers [11], as we describe next.

Stage	Time	Routing Tables	Messages Processed	Messages Queued in System
0	N/A	steady state 2(*1, 31, 41, 51) 3(21, *1, 41, 51) 4(21, 31, *1, 51) 5(21, 31, 41, *1)		steady state
1	N/A	1 withdraws the route 2(-, *31, 41, 51) 3(*21, -, 41, 51) 4(*21, 31, -, 51) 5(*21, 31, 41, -)	1 → {2,3,4,5} W	2 → {1,3,4,5} [231], 3 → {1,2,4,5} [321], 4 → {1,2,3,5} [421], 5 → {1,2,3,4,X} [521]
2	N/A	announcement from 2 2(-, *31, 41, 51) 3(-, -, *41, 51) 4(231, *31, -, 51) 5(231, *31, 41, -)	2 → {1,3,4,5} [231]	3 → {1,2,4,5} [321], 4 → {1,2,3,5} [421], 5 → {1,2,3,4,X} [521]
3	N/A	announcement from 3 2(-, -, *41, 51) 3(-, -, *41, 51) 4(231, 321, -, *51) 5(231, 321, *41, -)	3 → {1,2,4,5} [321]	4 → {1,2,3,5} [421], 5 → {1,2,3,4,X} [521]
4	N/A	announcement from 4 2(-, -, -, *51) 3(-, -, 421, *51) 4(231, 321, -, *51) 5(*231, 321, 421, -)	4 → {1,2,3,5} [421]	5 → {1,2,3,4,X} [521]
MRAI timer expires	30	announcement from 5 2(-, -, -, -) 3(-, -, *421, 521) 4(*231, 321, -, 521) 5(*231, 321, 421, -)	5 → {1,2,3,4,X} [521]	2 → {1,3,4,5} W, 3 → {1,2,4,5} [3421], 4 → {1,2,3,5} [4231], 5 → {1,2,3,4,X} [5231]
6	N/A	withdrawal from 2 2(-, -, -, -) 3(-, -, *421, 521) 4(-, *321, -, 521) 5(-, *321, 421, -)	2 → {1,3,4,5} W	3 → {1,2,4,5} [3421], 4 → {1,2,3,5} [4231], 5 → {1,2,3,4,X} [5231]
7	N/A	announcement from 3 2(-, -, -, -) 3(-, -, *421, 521) 4(-, -, *521) 5(-, 3421, *421, -)	3 → {1,2,4,5} [3421]	4 → {1,2,3,5} [4231], 5 → {1,2,3,4,X} [5231]
8	N/A	announcement from 4 2(-, -, -, -) 3(-, -, *521) 4(-, -, *521) 5(-, *3421, 4231, -)	4 → {1,2,3,5} [4231]	5 → {1,2,3,4,X} [5231]
MRAI timer expires	60	announcement from 5 2(-, -, -, -) 3(-, -, -, *5231) 5(-, *3421, 4231, -)	5 → {1,2,3,4,X} [5231]	3 → {1,2,4,5} W, 4 → {1,2,3,5} [45231], 5 → {1,2,3,4,X} [53421]
10	N/A	withdrawal from 3 2(-, -, -, -) 3(-, -, -, *5231) 5(-, -, *4231, -)	3 → {1,2,4,5} W	4 → {1,2,3,5} [45231], 5 → {1,2,3,4,X} [53421]
11	N/A	announcement from 4 2(-, -, -, -) 3(-, -, -, *5231) 5(-, -, -, -)	4 → {1,2,3,5} [45231]	5 → {1,2,3,4,X} [53421], 5 → {1,2,3,4,X} W
12	N/A	announcement from 5 2(-, -, -, -) 3(-, -, -, -) 4(-, -, -, -) 5(-, -, -, -)	5 → {1,2,3,4,X} [53421]	5 → {1,2,3,4,X} W, 4 → {1,2,3,5} W

Figure 8: Steps illustrating how route flap damping exacerbates convergence delays for the 5-node clique topology above.

3.3.1 A case for consistency

Internet routing, especially interdomain routing, has traditionally favored responsiveness, i.e., how quickly the network reacts to changes, over consistency, i.e., ensuring that packets traverse adopted routes. A router applies a received update immediately to its forwarding table before propagating the update to other routers, including those that potentially depend upon the outcome of the update. Responsiveness comes at the cost of availability: a router A thinks its route to a destination is via B but B disagrees, either because 1) B's old route to the destination is via A, causing loops, or 2) B does not have a current route to the destination, causing blackholes.

Below are several examples where inconsistent forwarding tables cause transient unavailability in interdomain routing today. In each case, the unavailability could last several tens of seconds (and sometimes minutes) due to BGP message processing and propagation delays [4].

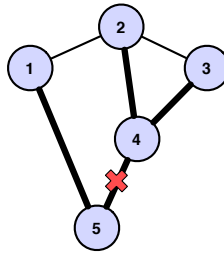


Figure 9: Link failure causing BGP loops at 2 and 3.

BGP link failures. Figure 9 shows how link failures cause transient loops in BGP. Bold lines show selected paths. If link 4-5 goes down, 4 would immediately send a withdrawal to 2 and 3. However, because both 2 and 3 know of alternate paths 3-4-5 and 2-4-5 respectively, they start to forward traffic to each other causing loops. The MRAI timer prevents 2 and 3 from advertising the new paths even though they have adopted them to forward traffic. The timer is believed necessary to prevent a super-exponential blowup in

message overhead, and its recommended value is 30 seconds. Eventually, when the timer expires, both 2 and 3 discover the alternate path to 5 through 1 that existed all along.

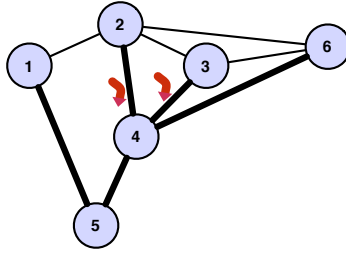


Figure 10:
Policy change causing BGP loops at 2 and 3 when 4 withdraws a prefix from 2 and 3 but not 6.

BGP policy changes. Figure 10 shows an example of how policy changes cause routing loops. AS4 may wish to engineer its traffic by withdrawing a prefix from 2 and 3 while continuing to advertise it to 6 for load balancing purposes. (For instance, by diverting traffic to arrive from 6 instead of 2, internal congestion within AS4 might be decreased.) If 2 and 3 each prefer the other over 6, routing loops would result like in the previous example. A similar situation also occurs if 5 wishes to switch its primary (backup) provider from 4 (1) to 1 (4); in this case, 5 is forced to either withdraw the route advertised (and potentially being used) to 4, or wait for a reliable indicator of when all traffic has completely moved over to the new primary provider 1. Other gadgets involving longer unavailability due to policy changes have been found [6, 7].

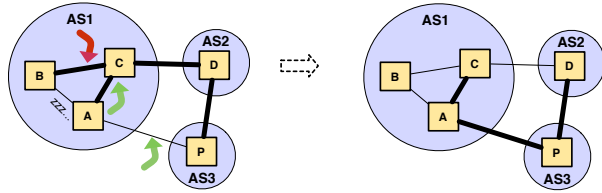


Figure 11: iBGP link recovery causing blackholes.

BGP policy cycles Figure 11 is similar to the classic “bad gadget” [1] involving cyclic preference dependencies. Each of ASes 1, 2, and 3 prefer to route via its clockwise neighbor over the direct path to AS 0, and does not prefer a path of length 3. The routes will never stabilize because there is no configuration where no AS wants to change its route to AS 0. Furthermore, the system goes through many repeated states involving routing loops causing chronic unavailability.

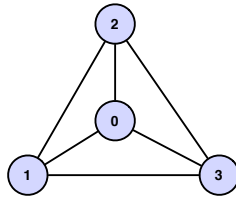


Figure 12: BGP policy cycles causing forwarding loops

iBGP link recovery Figure 12 shows a transient blackhole caused by iBGP inconsistency. Routers A, B, and C belong to AS1 while D belongs to the adjacent AS2. iBGP is a BGP protocol that runs between

routers inside an AS (in this case, A, B, and C). All routers route via D to the destination P in AS3. Suppose the previously failed link A-P recovers and is preferred by all AS1 routers over the route via AS2. If the AS1 routers all peer with each other, C will withdraw C-D-P from both A and B when it hears from A that A-P is available, but will leave it to A to announce AP to B directly because of the full-mesh design. If A is waiting upon its iBGP timer, B experiences a transient blackhole. The current BGP spec recommends an iBGP timer shorter than interdomain timers, and typical values range from 5-10 seconds. Previous work has shown that [12] such blackholes can cause packet loss for tens of seconds. If AS1 routers use route reflection as opposed to a full-mesh configuration [5], similar consistency problems can cause unavailability [5].

3.3.2 Consensus routing

John et al. [11] describe a fundamentally different approach to policy routing called *consensus routing*. The key idea in consensus routing is to treat consistency of routes as a safety property, i.e., routes adopted by routers in their forwarding tables must not result in loops or blackholes. Responsiveness, i.e., reacting quickly to routing events, should be treated as a liveness property without compromising consistency safety. How can one maintain consistency without hurting availability? For example, if a link fails, it seems natural for the router to adopt and attempt to forward along an alternate available route immediately even though it is possible that this adopted route is inconsistent with a downstream router's adopted route. If the router waits for all routers along the route to adopt consistent routes to the prefix in a synchronized manner, what should it do with packets that need to be forwarded to the destination in the meantime?

To achieve high availability without compromising consistency, consensus routing employs two modes: a *stable mode* where routers adopt a route only after all dependent routers have explicitly agreed upon the outcome of the corresponding update using a consensus protocol; and a *transient mode* where routers attempt to use (possibly inconsistent) backup routes only when a consistent route is unavailable. John et al. argue that this bimodal approach is not only simpler compared to BGP, but also significantly improves route availability compared to BGP as well as proposed alternatives that augment BGP with backup routes. A detailed description of the consensus routing approach can be found in the paper [11].

4 Internet interdomain traffic today

The above discussion focussed on the interdomain routing problem and the BGP protocol, but said little about the nature of traffic flowing between ASes in the Internet. A recent paper by Labovitz et al. [13] presents a measurement study detailing the evolution of Internet interdomain traffic in the last few years. For example, an interesting finding of that study is that enterprises such as Google and Comcast, traditionally considered as customers of larger ISPs are now among the top five contributors of Internet traffic, i.e., they contribute to more traffic than several tier-1 ISPs. The changing nature of such traffic patterns has significant implications for the evolution of provider-customer relationships in the Internet. For example, given the amount of traffic flowing in and out of Google's network, is it in a position to negotiate to peer with tier-1 ASes as opposed to being their customer (and paying them for transit)? Refer to the paper [13] for details of the measurement study and a discussion of implications for the evolution of interdomain relationships.

References

- [1] T. G. Griffin, F. B. Shepherd and G. Wilfong, "The Stable Paths Problem and Interdomain Routing", IEEE Transactions on Networking, April 2002.
- [2] Lixin Gao and Jennifer Rexford, Stable Internet routing without global coordination, Proc. ACM SIGMETRICS, June 2000.
- [3] Craig Labovitz, G. Robert Malan, Farnam Jahanian: Internet routing instability. IEEE/ACM Trans. Netw. 6(5): 515-528, 1998.
- [4] Craig Labovitz, Abha Ahuja, Abhijit Bose, Farnam Jahanian: Delayed internet routing convergence. SIGCOMM 2000: 175-187.

- [5] T. G. Griffin and G. Wilfong. On the correctness of iBGP configuration. SIGCOMM, 2002.
- [6] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, Randy H. Katz: Route flap damping exacerbates internet routing convergence. SIGCOMM 2002: 221-233.
- [7] D. Pei, X. Zhao, D. Massey, and L. Zhang. A study of BGP path vector route looping behavior. In ICDCS, 2004.
- [8] Lixin Gao: On inferring autonomous system relationships in the internet . IEEE/ACM Trans. Netw. 9(6): 733-745 (2001)
- [9] <http://www.routeviews.org/>
- [10] Harsha Madhyastha, Tomas Isdal and Mike Piatek and Colin Dixon and Thomas E. Anderson and Arvind Krishnamurthy and Arun Venkataramani, iPlane: An Information Plane for Distributed Services, USENIX OSDI 2006.
- [11] John P. John, Ethan Katz-Bassett and Arvind Krishnamurthy and Thomas E. Anderson and Arun Venkataramani, Consensus Routing: The Internet as a Distributed System, USENIX NSDI 2008.
- [12] Feng Wang, Zhuoqing Morley Mao, Jia Wang, Lixin Gao, Randy Bush: A measurement study on the impact of routing events on end-to-end internet path performance. SIGCOMM 2006: 375-386
- [13] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, Farnam Jahanian, Internet inter-domain traffic. 75-86, ACM SIGCOMM 2010.