École Polytechnique Fédérale de Lausanne

Applying Approximate Distance Oracles to Internet Routing

by Matteo Pariset

Bachelor Semester Project Report

Prof. Bryan Ford
Project Advisor

Cristina Basescu
Project Supervisor

EPFL IC IINFCOM DEDIS
BC 210 (Bâtiment BC)
Station 14
CH-1015 Lausanne

June 5, 2020

Chapter 1

Motivation

In order to reach their destinations, IP packets must go through multiple physical devices, across several IP networks. The process of selecting their path is called *routing* and relies on information shared among routers using a *routing protocol*. In the Internet, different protocols are used at different scales. For instance, Open Shortest Path First (OSPF) is typically used for routing packets between sub-networks belonging to the same organization whereas *Border Gateway Protocol* (BGP) is the *de facto* standard for global Internet routing. It operates at the level of *Autonomous Systems* (ASes), administrative entities which are identified by unique *autonomous system numbers* (ASNs). BGP is a stretch-1 routing protocol which selects paths based on prefix announcement rules and AS-specific policies.

Over the years, the impressive growth of Internet-based services has exposed several of BGP weaknesses, making its scalability a major concern for Internet operators [8]. For instance, prefix de-aggregation, resulting from common traffic engineering practices and multi-homing [4], has provoked a faster-than-linear growth of the Routing Information Base (RIB) size [15]. Although the RIB is bounded by the address space size[1], the amount of BGP messages needed to react to network changes is not. Prefix de-aggregation "exposes the core of the network to the dynamic nature of the edges" [4], increasing BGP churn, i.e. the number of BGP UPDATE messages sent over the network. A small number of ASes and IP prefixes are responsible for a large fraction of the total UPDATE churn [7]; even worse, part of this information is often superfluous since it is generated by unstable edges [14]. The exponential growth of BGP routing table (RT), besides being a problem in itself, also entails convergence issues, since UPDATE messages can trigger re-computation of the *forwarding information base* (FIB).

Those issues seem to be inevitable, since Gavoille *et al.* [9] showed that BGP, like all stretch-1 universal routing protocols, requires $\Omega(n \log n)$ bits for the RT (n denotes the number of nodes in the network). Given that the theoretical upper-bound is also equal to $\mathcal{O}(n \log n)$, BGP achieves the worst possible performances in terms of RT size. It is thus clear that the stretch-1 requirement must be dropped to achieve sustainable, i.e. sub-linear, RT scaling.

The collection of techniques that offer sub-linear growth in RT size is known as *compact routing*.

[1] And the number of adjacent ASes

In this work, we focus on a particular universal compact routing scheme (TZ), by Thorup and Zwick [17], which ensures:

- bounded maximum stretch of $2k - 1$ (with $k \geq 1$ a constant);
- nearly optimal data structures size;
- **constant** time to answer distance queries.

The worst-case stretch of TZ (at least 3 in the non-trivial case $k > 1$) is far from being acceptable for Internet paths, but we know [11] that its average stretch on static, "Internet-like" topologies is sufficiently small (close to 1) to make it a viable inter-AS routing algorithm. It is also important to understand how TZ performs on dynamic topologies, given that the Internet graph is by no means static. Using general arguments, Afek *et al.* [1] showed that the worst-case communication cost—the number of control messages needed to reach convergence—cannot scale better than $\Omega(n)$. This result, however, does not provide estimates on the frequency of this worst-case scenario for Internet-like graphs.

Therefore, we are interested in designing and implementing TZ response to topological changes. We limit ourselves to the case of edge failures since other events, like the creation of new edges and the disappearance of an AS, can be treated with methods similar to the ones presented here.

To measure performances of this algorithm, we also build a simulator, capable of executing both BGP and our modified version of TZ on any given topology (loaded from a `csv` file). It allows to trigger edge deletions, either manually or randomly, and to log multiple metrics. Its source code, written in Go, can be found at [16]. To conduct our simulations, we use the entire AS graph rather than smaller synthetic graphs, since those would hardly be representative of the Internet. Additional details on how we derive it, as well as on its structure and evolution over time, can be found in chapter 2. In chapter 3, we present a modified version of TZ scheme, better suited to static Internet routing, which is further improved in chapter 4 to handle dynamic topologies.

Chapter 2

The AS graph

We reconstruct the shape of the Internet at the AS level, using data collected by CAIDA [2], available on a monthly basis, starting from December 2015. Inter-AS links, annotated with their types, are inferred using two methodologies, detailed in [13] and [10].

Each link between Autonomous Systems belongs to one of the following categories, which constitute the extremes of a much wider spectrum of inter-AS business relations:

- *provider-to-customer* **PC** (resp. *customer-to-provider* **CP**): it reflects the hierarchical organization of ISPs and generally implies the sale of connectivity by one of the endpoints to the other;
- *peer-to-peer* **P2P**: it indicates a bilateral agreement to exchange traffic and routes free of charge.

In what follows, we also use the symbols $\searrow$, $\nearrow$ and $\longleftrightarrow$ to indicate PC, CP and P2P links respectively.

Looking at the size of the graph, we observe that the number of ASes kept growing linearly between 2015 and 2020, as already remarked by Dhamdhere and Dovrolis [5] in 2008. A linear increase can also be seen in the the number of CP links (+30% in the same period) (figure 2.3). P2P links were, instead, subject to bigger short-term variations, likely determined by the inherent difficulties in discovering them [5].

Another meaningful property to inspect is the distribution of the degrees of nodes. That is because, as Krioukov *et al.* [11] noticed, the *scale-free* nature of a graph is crucial to reduce the number of paths with worst-case stretch. In figure 2.1, we plot the CCDF of AS degrees as measured in 2015, 2019 and 2020. Despite the shift in the Internet contents distribution paradigm which happened over the last decade, AS degrees still follow a power-law distribution. This means that business interactions among ASes can still be accurately described by a preferential attachment process.

The AS degrees distribution, while devoid of radical changes, reveals nonetheless some gradual transformations. In 2020, more nodes have degrees ranging from 100 to 1000, compared to 2015.

To understand why, in figure 2.2 we group nodes according to the most frequent type of incident edges. For example, "2015 customer" will be the 2015's degrees distribution, calculated only over ASes predominantly connected to edges of type *customer-provider*. We observe that the shapes of "customer" and "provider" ASes distributions have not evolved over the last 5 years (brown and blue lines), whereas the nodes with a majority of P2P links tend to have a lower degree in today's Internet.

A possible explanation is offered by conjecturing a progressive flattening of the Internet, as exposed by Wang and Zhang [18], according to which nodes of type "peer" are, now, more likely to be middle-sized ASes, instead that tier-1 ISPs.

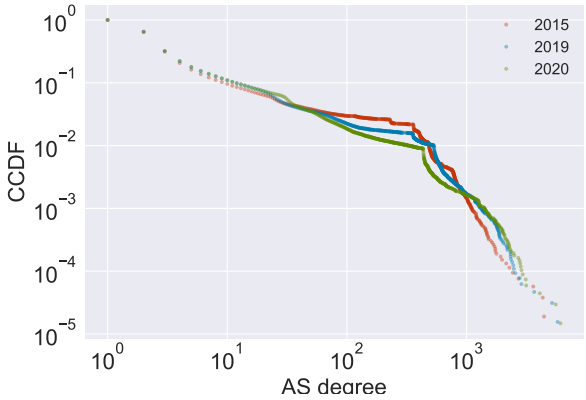

Figure 2.1: Evolution of CCDF of AS degrees

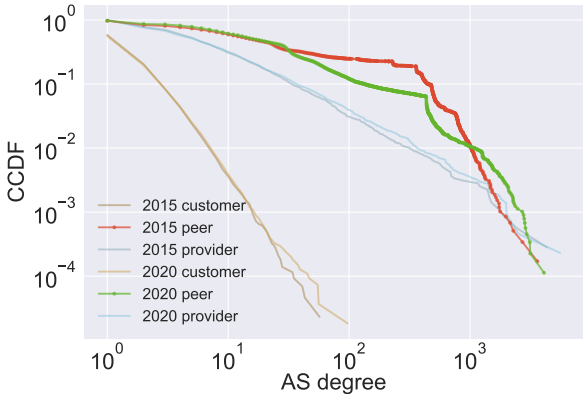

Figure 2.2: CCDF of AS degrees by type of node

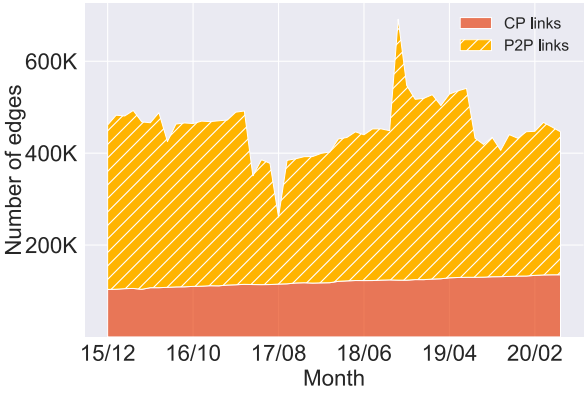

Figure 2.3: Evolution of edges by type, PC links are part of the CP category

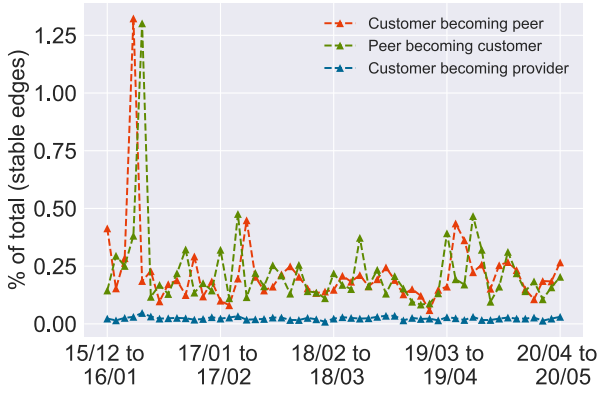

Figure 2.4: Fraction of edges in the stable sub-graph that change type

Since our simulations heavily rely on the types of edges (PC, CP and P2P), we are also interested in determining the stability of such labels. For this purpose, we consider the sub-graph induced by

the nodes that never disappear from our data for the whole period of our analysis (Dec. 2015 to Apr. 2020), which we call *stable graph*. For each month, we then plot (figure 2.4) the fraction of edges in the stable graph which change of type. On average only 0.5% of them undergo variations, mostly transitioning from CP to P2P and vice-versa. We believe that these are mostly spurious conversions, since spikes in one kind of transition are shortly followed by higher-than-usual numbers of conversions in the opposite direction.

Chapter 3

Adapt TZ to the Internet

Before describing our improvements, we briefly introduce the original TZ scheme. Given a graph $G = (V, E)$ and a parameter k, the algorithm constructs k sets (A_i) of randomly chosen nodes, called *landmarks*. They are such that:

$$A_0 = V, \qquad 1 \leq i \leq k : A_{i-1} \supset A_i, \qquad A_k = \emptyset$$

We call the elements in A_{k-1}, namely the nodes in the smallest non-empty set, *top-level* landmarks. Distances to landmarks are used to approximate distances between arbitrary nodes, which are guaranteed to know the distance to, at least, one common landmark. Every node v must maintain two structures:

- $W(v) = \{W_i(v) \mid 0 \leq i \leq k\}$, a collection of distances to *witnesses* $W_i(v)$, which (for each i) is the closest landmark in A_i to v;
- $B(v)$, its *bunch*, containing distances to other, well-chosen nodes. In particular, nodes in the bunch are closer to v than v is to some of its witnesses, i.e. $B(v) = \{s \in V \mid \exists i \leq k \text{ s.t. } \delta(v, s) < \delta(v, W_i)\}$ (where δ denotes the distance between to vertices)

Although the above structures are primarily intended to answer distance queries, they are enough to construct paths between arbitrary pairs of nodes. More on this can be found in [17].

Henceforth, we adopt the terminology introduced above and set $k = 3$.

3.1 Landmarks Selection

TZ authors suggest to select landmarks uniformly at random from nodes in the graph. This strategy, which preserves the generality of the algorithm, becomes sub-optimal when TZ is applied to the AS graph. Landmark selection strategies, in fact, do not affect the correctness of the algorithm but can worsen the quality of its path estimates and increase the size of its structures.

Since landmarks roughly serve as middle-points in approximated paths, they should preferably be selected among nodes which play that role in the Internet. We hence exploit our knowledge that the hierarchical organization of ASes favors inverted V-shaped paths, in which traffic climbs the

ISP hierarchy to then descend again. Big Internet Service Providers are in fact a good candidate for landmarks, given that they sell Internet connectivity to smaller ISPs.

We then increase the probability that important Autonomous Systems are selected as landmarks. We pay particular attention to the selection of top-level ones (which should be around $|V|^{1-2/k} = 40$), since distances to all of them are known to every node in the graph. We could define the importance of ASes as the size their customer cones, but we choose to approximate it with the number of incident PC links. We prefer this definition because, although more local, it still efficiently captures the informal notion of ISP importance. In fact, two-third of the 64 most important ISPs from CAIDA AS rank [3] match those identified according to our definition.

Our selection strategy, which we call *popularity selection*, assigns to each AS a different selection probability depending on its number of PC links. The most important (according to our definition above) ASes are several orders of magnitude more likely to become landmarks than before but every AS keeps a non-zero probability of being selected.

We compare the effects of the selection strategy in figure 3.1 which contains the distribution of stretch, when popularity (green) and random (red) landmarks selection strategies are used. We generate 4000 paths between random nodes and compare their lengths against the baseline, calculated using BGP. It can be seen that popularity selection performs better, achieving an average stretch of 1.06 compared to 1.17 of random selection.

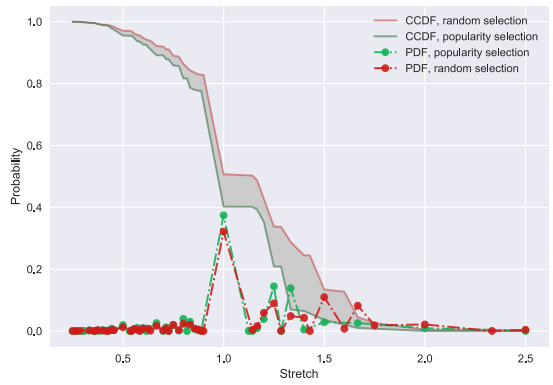

Figure 3.1: Measured stretch depending on landmarks selection strategy, 4000 routes

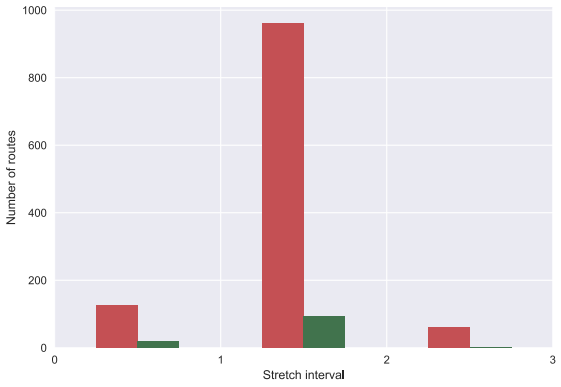

Figure 3.2: Number of routes with valleys (4000 samples), for every stretch interval

We also note that some paths have a stretch strictly smaller than 1, which might seem an anomaly. This occurs because stretch only depends on hop distance but our baseline paths are not shortest paths, i.e. they are optimal with respect to BGP criteria but non-optimal in terms of stretch. In fact, BGP also filters routes according to link types, preferring for instance a long path constituted of PC edges to a shorter one with P2P ones. To illustrate this, we provide a rather extreme example:

	Path between **AS29827** and **AS61170**	Hop-length
BGP	↗ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘ ↘	12
TZ	↗ ⟷ ↘	3

In the table, BGP does not provide the "shortest" path since the second edge of its path (in green, of type PC) is preferred to the second edge of TZ path (in red, of type P2P).

3.2 GR compliance

The example above shows why simple stretch considerations are insufficient to thoroughly assess routing algorithms performances. Traffic on the Internet is subject to complex and AS-dependent economic dynamics which heavy influence the paths it follows. These constraints, together with router-specific software differences, have posed a threat to BGP stability in the past and have prompted the formalization of guidelines which ensure routing algorithm convergence. Here we adopt those formulated by Gao and Rexford [12] (we call them GR), which strike a balance between market needs and preconditions for correct Internet routing.

We deem it necessary to understand whether paths generated using TZ scheme comply with GR or similar criteria. As a first approximation, we study the presence of *valleys* in path, which signal a specific violation of routes filtering rules. They occur when a node advertises a provider route[1] to some of its providers. This would not be cost effective for that node, and therefore not likely to happen in the real Internet. Even this partial adoption of GR rules is enough to highlight some differences between landmarks selection strategies. About a third of the paths computed using random selection contain *valleys*, whereas this fraction drops to 11.5% if we use popularity selection. Figure 3.2 breaks those figures over unitary stretch intervals.

Although popularity selection helps to fill the gap between TZ approximations and viable paths, it is unlikely to close it altogether. To guarantee that paths always respect GR requirements it is necessary to directly incorporate them in our routing scheme which, in its basic version, has no knowledge of link types.

In order to do that, we make TZ apply GR filtering rules, as proposed in [12], when it grows shortest path trees needed to construct both bunches and sets of witnesses. Distances between nodes are thus obtained using a modified version of Dijkstra's algorithm which, running from a source d, relaxes an edge e only if BGP exported the destination d over e. We test two variants of filtering rules, which we call GR and GRP. Both of them require node n:

- to export all the routes to its customers;
- to export customer routes[2] to its peers and providers.

The only difference between the two variants is that, under GRP, n can also announce peer routes to its peers, i.e. it can provide transit services to its peers. This assumption is interesting since it

[1] We use provider route as a shorthand for *route whose first hop is a customer-to-provider link*
[2] Namely the routes with a customer of n as next-hop

captures a different use of P2P links, which could be realistic under the hypothesis of decrease in transit pricing.

To restore the correctness of the modified TZ, some adjustments become necessary. In particular, since distances - calculated using GR/P filtering - do not satisfy the triangle inequality, nodes in bunches are no longer guaranteed to induce a spanning tree. For each node n, we hence restore this invariant by completing its bunch $B(n)$ with all the nodes on the shortest path between n and every $a \in B(n)$.

Another issue concerns nodes that become unreachable, i.e. those which filtering rules make invisible to some other nodes. They are a small fraction of ASes (0.5%), probably resulting from spurious noise in type annotations described in chapter 2. In those cases, we allow paths to violate route filtering rules, so that every destination is reachable. In all the other cases, this version of TZ scheme ensures that paths never contain valleys.

Chapter 4

TZ on dynamic topologies

We now examine what happens when a link between ASes becomes unavailable, following either a technical failure or a deliberate action. This translates into the disappearance of the corresponding edge from our AS graph. We suppose that, after a certain time delay, both endpoints become aware of the failure and we analyze the consequences of this circumstance on the structures of our GR-compliant TZ scheme.

It is possible that the endpoints (and other nodes) relied on the missing edge to reach some of their witnesses or some destinations in their bunches. If they simply dropped these destinations, some routing processes could fail. For each bunch, two cases can occur:

- if **all** the top-level landmarks still have valid paths, routes will sometimes incur higher stretches but their existence will be preserved;
- if one or more top-level landmarks have invalid paths (they are no more in the bunch), some path queries **can** fail if the remaining lower-level landmarks are not enough to complete them.

We set to guarantee that bunches always contain valid paths to every top-level landmarks, i.e. $\forall q \in A_2, \forall v \in V : q \in B(v)$. This ensures that, as long as the graph remains a single connected component, every query can be successfully answered. We do not place any requirements on other destinations in $B(v)$, since their original inclusion in the bunch was determined by multiple topology-related conditions which might no longer hold after an edge deletion. We do not check them again, since that could massively increase the amount of information sent over the network.

With respect to witnesses, instead, the knowledge of **one** top-level landmark could be sufficient to maintain the correctness of the algorithm. Moreover, loosing information about lower-level witnesses (i.e. the closest landmarks at each lower level) would not significantly deteriorate the quality of paths, since the average hop-distance between two random nodes is small [11] (less than 5).

Nonetheless, keeping a single landmark per node is unacceptable from a practical perspective, since it would place a burden on higher-level landmarks, which are exponentially fewer. In fact, these would increasingly be exposed to local traffic, i.e. traffic originated by and destined to nodes close to each other in the AS graph. That is why we choose to fully restore the set of witnesses, so that nodes always maintain a valid path to their closest landmark **at each level**.

4.1 Restore missing information

In the event that one of the endpoints of the missing edge (we call it z) invalidates a destination d, z must inform all its neighbors. We distinguish two scenarios.

4.1.1 $d \in W_i(z)$

We first analyze the case in which d was the i-th level witness of z. The AS z invalidates the destination d and executes UPDATE-WITNESS(z, v, d, i), for each adjacent node v (algorithm 1). z's neighbors which had learned their d-path[1] from z, should forward an invalidation message to their own neighbors (line 6) until all the nodes previously relying on the broken path to d have invalidated it.

Some of the nodes receiving the invalidation message will not be affected by it. They either:

- have d as their i-th level witness, but know a path not passing through z;
- have different i-th level witnesses.

In both cases, they have a valid route to an i-th level witness which they can supply to the invalidation message's sender (line 9), which does not need to directly reach any i-th level landmark.

Nodes having acquired new routes send them to their neighbors, which, each time, retain the better option. Broadly speaking, these messages will flow in the opposite direction of invalidation notices, directed towards the endpoints of the missing edge.

We stress that, at the end of this process, nodes will have chosen as witness whichever i-th level landmark is closer to them, without restricting to their previous witnesses W_i.

Algorithm 1 i-th level witness update

```
1:  function UPDATE-WITNESS(Caller node c, Target node v, Witness d, Witness level i)
2:
3:      if d == W_i(v).destination and W_i(v).nextHop == c then
4:          W_i(v) = ∅                                   ▷ Invalidate the witness
5:          for p ← neighbors(v) do                      ▷ Notify neighbors
6:              UPDATE-WITNESS(v, p, d i)
7:          end for
8:      else
9:          Send W_i(v) to c                             ▷ Send valid path to the caller
10:     end if
11:
12: end function
```

4.1.2 $d \in B(z)$

If d was in the bunch of z, it is added to a list of invalid destinations, called $R(z)$. The node z will then send $R(z)$ to all its neighbors: it will execute UPDATE-BUNCH$(z, v, R(z))$, for each

[1] Namely a path to d

neighbor v (algorithm 2). Those ASes will delete from their bunches the destinations that are also in $R(z)$ (line 7) and broadcast a reduced list of paths to revoke $R' \subset R(z)$ (lines 15 to 19), containing the elements in $R(z)$ they invalidated. R' is enough since, if they do not invalidate d, nodes which have learned a d-path from them will not need to invalidate d either.

At this point, however, bunches may have lost paths to some **top-level landmarks**, an occurrence that would compromise the correctness of our algorithm (as explained above). To avoid that, each AS with a valid path to a top-level landmark t sends it to the neighbors which had previously invalidated their t-paths (line 10).

The process is concluded when all the nodes have, in their bunches, valid destinations to all top-level landmarks. This is always possible because, in the worst-case, each top-level landmark will be reached and will export to its neighbors a path of length 1 to itself.

Algorithm 2 Bunch update

```
1:  function UPDATE-BUNCH(Caller node c, Target node v, Revocation List R(c))
2:
3:      R(v) = {}                                   ▷ Initialize v's revocation list
4:      for b ← R(c) do
5:          if b ∈ B(v).destinations then
6:              if B(v).nextHop(b) == c then
7:                  B(v).delete(b)                  ▷ Remove b from the bunch
8:                  R(v).add(b)
9:              else if b ∈ A_{k-1} then            ▷ b is a top-level landmark still in the bunch
10:                 Send B(v).route(b) to c         ▷ Send the valid b-path to the caller
11:             end if
12:         end if
13:     end for
14:
15:     if R(v) ≠ {} then
16:         for r ← neighbors(v) do
17:             UPDATE-BUNCH(v, r, R(v))
18:         end for
19:     end if
20:
21: end function
```

We note that some care must be taken with respect to the dispatch of alternative routes: lines 9 (algorithm 1) and 10 (algorithm 2). It is possible for a node to export a d-path to its neighbors and later receive an invalidation message concerning d. This would entail additional path revocations (of the exported d-path) and slow down the convergence of the algorithm. To avoid that, nodes should delay the execution of those lines long enough to receive invalidation messages from all their neighbors involved in the update process.

4.2 Measure deletion effects

We are now interested in gauging the amount of information that is shared among nodes, in reaction to the disappearance of an edge. We could measure the number of messages that travel on the network, but this number would be heavily dependent on the way we implement our algorithm. We adopt instead two metrics, which we call *Impact* and *Updates*, we deem to be less sensitive to implementation details.

4.2.1 Impact

Impact captures the number of nodes that receive any kind of information related to the disappearance of an edge, which could be an invalidation message (for a witness or a destination in the bunch) or an alternative path. In other words, *Impact* counts the fraction of nodes that "see" the deletion.

This number, however, can be misleading in the case in which the deletion happens close to nodes with high degree, a relevant scenario given the scale-free nature of the AS graph observed in chapter 2. In such cases, a single AS, forwarding 1 invalidation message to all its adjacent nodes, could reach up to 8000 nodes (12% of the total). Most of them, though, will rely on different paths and will hence not be affected by this information, i.e. they will not modify their structures.

4.2.2 Updates

To complement the previous metric, we also count the number of nodes which modify some of their structures—set of witnesses or bunch—in reaction to the failure of a link. *Updates* reveals how many nodes **should** be aware of this event and is not affected by the number of nodes which receive information that they do not use.

It can be argued that *Impact* and *Updates* are, respectively, the upper and lower bounds on the number of nodes that are involved in the recovery scheme triggered by the deletion of an edge. *Impact* accounts for the case in which a node broadcasts invalidation messages to all its neighbors, since it cannot know if they are using the routes it previously exported. At the other extreme, *Updates* reflects the perfect strategy, where only nodes that need to invalidate some routes are reached by the relevant information.

4.3 Simulate edge deletions

Having defined the metrics we use, we now describe our simulations.

4.3.1 Witness level change

We first explore how edge deletions locally affect TZ path approximations. In particular, since the algorithm exploits the existence of a common landmark known by any pair of nodes a and b to construct a path between them, we monitor how this landmark changes because of an edge deletion. In the first test, we only consider the special case where a and b are adjacent and the

edge to be deleted is the one connecting them, (a, b). This choice restricts the analysis to short paths and consequently highlights local variations in TZ behavior.

We compute the TZ path between a and b (we call it π^1_{TZ}), we then delete their common edge and recompute the path between them, π^2_{TZ}. Each of these routes contains a witness of one of the two nodes, which should appear in the bunch of the other. We classify this witness according to its level, i.e. the smallest index i such that $w = W_i$.

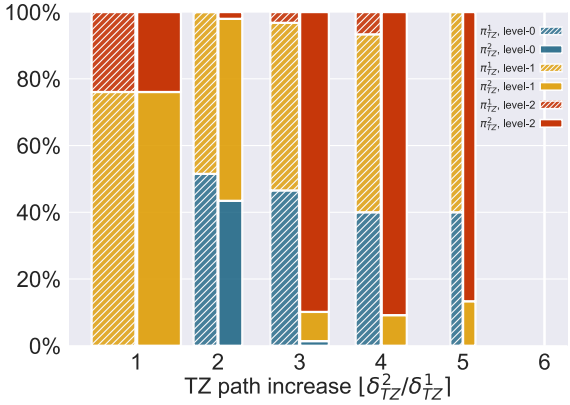

Figure 4.1: Level of witnesses used to compute paths: in blue level-0, in yellow level-1 and in red level-2 landmarks

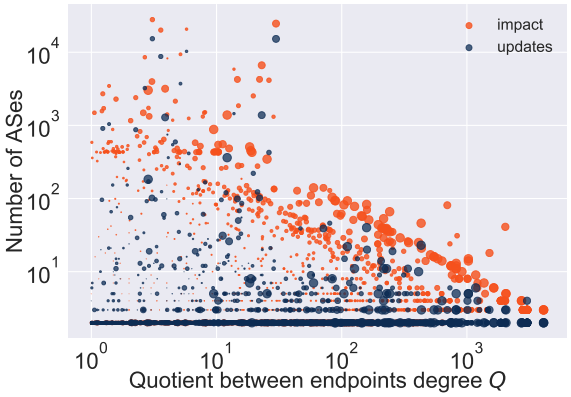

Figure 4.2: *Impact* and *Updates* of 3000 random edge deletions, ordered according to $Q = \max(\deg(e_1), \deg(e_2))/\min(\deg(e_1), \deg(e_2))$

Figure 4.1 shows the fraction of paths making use of each level of witnesses, on a sample of 3000 pairs of nodes. The hatched columns refer to the paths π^1_{TZ}, the others to π^2_{TZ} and their width is the logarithm of the number of samples belonging to each group on the x-axis. Each pair of paths (π^1_{TZ}, π^2_{TZ}) is categorized according to the ratio ρ between their lengths, $\rho = \delta(\pi^2_{TZ})/\delta(\pi^1_{TZ})$.

To motivate the choice of ρ, we first observe that π^1_{TZ} does not necessarily coincide with (a, b). GR rules, discussed in section 3.2, imply in fact that there could be a longer $(\delta(\pi^1_{TZ}) > 1)$ path, preferable to the trivial one containing only (a, b). Since we aim to measure the relative quality of π^2_{TZ} compared to π^1_{TZ}, we look at the quotient between their lengths.

The leftmost pair of columns in figure 4.1 shows the results when $\rho = 1$. The original path does not change following the deletion, which trivially implies that π^2_{TZ} and π^1_{TZ} cannot contain (a, b). We remark that this is, by far, the most common scenario, accounting for 85% of deletions.

When $\rho > 1$, the deletion affects π^1_{TZ}, which is recalculated (possibly using a different witness). The graph highlights the drastic loss of importance of level-1 landmarks (in yellow) due to the link failure. At the beginning, 46% of the paths were calculated using a level-0 witness as reference: either a or b was in the bunch of the other node. This percentage corresponds, in fact, to the weighted average of the dashed blue columns. For the remaining 54% of paths, level-1 witnesses were almost always enough (dashed yellow), with only a few paths requiring level-2 (top-level)

witnesses (dashed red). After the deletion of (a, b), the situation drastically changes: 71% of π^2_{TZ} are calculated using top-level witnesses (this is the weighted average of solid red columns). The disappearance of level-0 destinations is a direct consequence of the partial restoration of bunches, described in section 4.1. What is less evident, though, is the loss of level-1 witnesses in bunches (which are used only 10% of the time for π^2_{TZ} when $\rho \geq 3$). This means that those landmarks were often reached via (a, b).

4.3.2 Impact measurement

We now turn to the communication cost of a typical edge deletion. We delete edges selected uniformly at random and measure *Impact* and *Updates*. Figure 4.2 shows the results of this test, for 3000 deletions. The x-axis provides information about the position of the deleted edge (e_1, e_2) in the AS graph. It displays the ratio between the maximum and minimum degrees of its endpoints, i.e. $Q = \max(\deg(e_1), \deg(e_2))/\min(\deg(e_1), \deg(e_2))$. When Q is close to 1, e_1 and e_2 have about the same number of incident edges and are consequently more likely to have similar ranks in the AS hierarchy. On the other hand, when Q is big, one of the two nodes has a significantly higher number of incident edges than the other, implying that the edge (e_1, e_2) probably points from the center to the periphery of the Internet graph.

It can be seen that, when Q increases, fewer nodes are informed of the deletion. A possible explanation is that only a few, minor ASes used that link to route traffic towards bigger ASes and should therefore be informed of its disappearance.

When Q approaches 1 the maximum values of *Impact* and *Updates* increase and so do their variances. To understand why, we also need to know the absolute values of the degrees (of e_1 and e_2), which the ratio Q does not provide. We encode this information in the radius of the dots, which is proportional to the maximum degree of e_1 and e_2: $R \propto \max(\deg(e_1), \deg(e_2))$. Hence, when $Q \approx 1$ and $R \gg 1$, the edge probably connects two important ASes, e.g. two tier-1 ISPs. We see that its disappearance can imply that up to 20% of the nodes become aware of the deletion. If, instead, the edge connects two less important ASes, namely $Q \approx 1$ and R is small, the experiment shows that fewer ASes must be notified of its deletion. The maximum impact roughly follows a polynomial expression in Q and, in general, for a fixed Q, big values of R translate into higher numbers of ASes impacted. In spite of a high variance, the average impact (over all the values of Q) is low: 88 nodes, corresponding to 0.13% of the ASes. Its median is 2, meaning that most of the time no path used the edge that is deleted (the endpoints do not send any invalidation message).

Looking at *Updates*, instead, it can be seen that the number of nodes which modify their structures is, on average, about a quarter of the ones which are notified (26 nodes). Also, when $Q \gg 1$, the average difference between *Updates* and *Impact* is smaller, providing additional evidence that the endpoint most affected by those kinds of deletions is the one with lower degree. That is the case since most of the nodes that receive the invalidation message actually need it, meaning that the invalidated destinations are likely high-level landmarks (as opposed to nodes in the bunch, since those are less likely to appear in multiple bunches).

4.3.3 Stretch deterioration

In the previous tests, we analyzed the local effects of deletions and their typical communication cost. We now evaluate the global evolution of performances, in particular of the average stretch, following a significant amount of link failures. Since, as explained in section 4.1, we allow the size of bunches to decrease, we expect destinations in bunches that are not top-level landmarks to be progressively invalidated and not restored. This ignites a progressive reduction of nodes awareness of their neighborhoods because bunches contain nodes "closer" than witnesses. We look at the macroscopic implications of this phenomenon.

In this simulation, we hence remove a constant fraction of all the edges (selected uniformly at random) and then compute the average stretch on paths chosen independently from the those edges. As always, we run TZ and BGP in parallel and use BGP as baseline.

In figure 4.3, we plot the evolution of the average stretch, in relation to the fraction of edges deleted from the original graph (blue and dark blue lines). We observe that, after removing a small percentage of the edges, the increase in stretch becomes visible on a global scale: when 5% of edges are deleted the average stretch (of random paths) increases of 4%. Also, it can be seen that GRP filtering rules are less vulnerable to deletions, since P2P links can carry transit traffic and reduce the reliance on top-level landmarks.

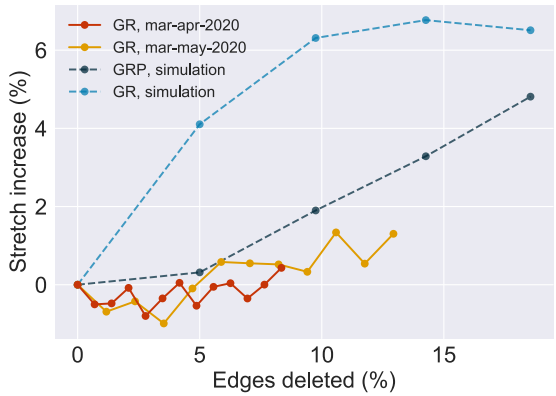

Figure 4.3: Evolution of average stretch (over 2000 paths) caused by edge deletions

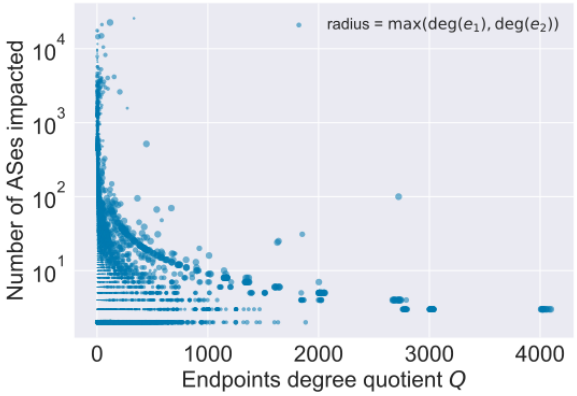

Figure 4.4: *Impact* of link failures occurring in March 2020

4.4 Simulation with real link failures

The previous simulations will now serve as the baseline to evaluate the behavior of our algorithm when considering the actual evolution of the Internet graph.

We extract the changes in the AS graph by comparing the snapshots taken in March and April by CAIDA. Since our simulator does not support live insertion/deletion of ASes, we only look at the edges. We suppose that the 42848 edges (9.2% of the total) censused in March but not in April became unavailable at some point, triggering the reaction of Internet routing algorithm.

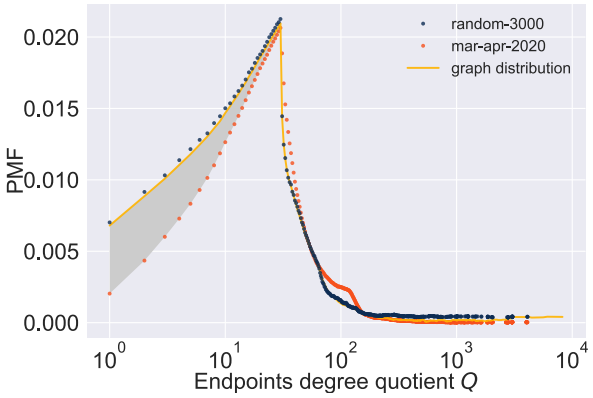

Figure 4.5: Difference in distributions of endpoints degrees

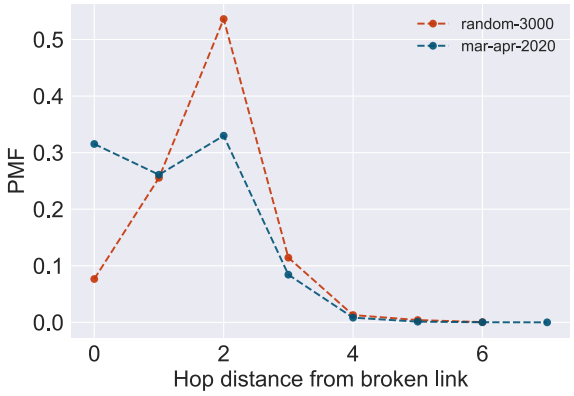

Figure 4.6: PMF of hop distance from broken link of impacted nodes

The red line in figure 4.3 shows the average stretch measured while progressively executing this sequence of link failures. The yellow line refers to the failures occurred in April and May. We remark that the first halves of these two lines do not overlap (even though they refer to the month of April) since the edges deletions were executed in different orders. It can be seen that the stretch worsens at a slower rate compared to the baseline. For instance, after replaying all the failures detected in March and April, the average stretch incurs a 1% penalty, whereas an equal number of random deletions provokes a 6.7% increase (light blue line).

The differences between the two simulations are due to the fact that link failures in the Internet are not randomly distributed. Figure 4.5 substantiates this claim by showing the distribution of Q (the quotient between the degrees of endpoints described in section 4.3.2) over three different sets: the edges of the AS graph, those which failed during March 2020 and the 3000 random sample used in our tests above. As expected, the values of Q for the random sample follow, as expected, the distribution of the entire set of edges in the graph. The shaded area highlights that links with Q between 1 and 30 fail are less likely to fail, i.e. their are under-represented among the edges which failed.

This is not surprising since the edges connecting important ASes ($Q \approx 1$ and $R \gg 1$), which are more stable than the average link, fall in the category $1 \leq Q \leq 30$. In fact, only 8.5% of link failures involve edges incident to one of the 64 biggest ASes (by customer cone size), down from 17% with randomly chosen deletions.

We note that the landmarks selection strategy presented in section 3.1 is helpful in this scenario since the topology undergoes few changes in proximity of important ASes, which are also more likely to be landmarks. Paths to those ASes, known to many other nodes, change less frequently and thus only the lower ranks of the AS hierarchy experience bunch quality deterioration.

Further proof of this can be found in the distribution of hop-distances traveled by invalidation messages, illustrated in figure 4.6. A third of the nodes that receive invalidation messages generated

by real link failures are endpoints of one of the unstable edges (they are at hop-distance 0), against 10% with random deletions. Invalidation messages reach, on average, nodes which are only 1.2 hops away from the failing link (down from 1.75 with random deletions).

Even the *Impact* of real link failures, depicted in figure 4.4, improves compared to the baseline (presented in section 4.3.3). Its average and standard deviation, respectively of 36 and 341 ASes, are a third of the values obtained with random deletions.

We conclude that the typical edge deletion happening in the Internet is "less visible" to other nodes than a random one, i.e. the ASes which become aware of this event are generally closer to it.

Concluding remarks

In this project, we design a compact routing scheme for the Internet based on previous work by Thorup and Zwick. This algorithm complies with Internet-specific requirements regarding the stretch and shape of its paths estimates. We perform simulations using the real AS graph and investigate the behavior of the scheme in the event of topology changes.

We conclude that the structure of the Internet graph makes it possible to efficiently apply compact routing even in the case of a dynamic topology. The majority of edge deletions have a bearing on the state of a limited number of Autonomous Systems.
We also show that the average communication cost for edge failures is even lower when considering the real edge failures occurring in the AS graph.

One of the limitations of our approach relates to bunch quality deterioration. We believe that it can be avoided by making ASes periodically run cleanup routines (e.g. full bunch re-computation), but we leave the analysis of this solution to future works.

Bibliography

[1] Y. Afek, E. Gafni, and M. Ricklin. "Upper and Lower Bounds for Routing Schemes in Dynamic Networks." In: *30th Annual Symposium on Foundations of Computer Science*. 30th Annual Symposium on Foundations of Computer Science. Research Triangle Park, NC, USA: IEEE, 1989, pp. 370–375. ISBN: 978-0-8186-1982-3. DOI: 10.1109/SFCS.1989.63505. URL: http://ieeexplore.ieee.org/document/63505/ (visited on 05/25/2020).

[2] *AS Graph, CAIDA*. URL: http://data.caida.org/datasets/as-relationships/serial-2/.

[3] *CAIDA ASRank*. URL: https://asrank.caida.org/.

[4] Meyer D, Zhang L, and Fall K. *Report from the IAB Workshop on Routing and Addressing*. URL: http://tools.ietf.org/id/draft-iab-raws-report02.txt,April2007.

[5] Amogh Dhamdhere and Constantine Dovrolis. "Ten Years in the Evolution of the Internet Ecosystem." In: *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement Conference - IMC '08*. The 8th ACM SIGCOMM Conference. Vouliagmeni, Greece: ACM Press, 2008, p. 183. ISBN: 978-1-60558-334-1. DOI: 10.1145/1452520.1452543. URL: http://portal.acm.org/citation.cfm?doid=1452520.1452543.

[6] Amogh Dhamdhere and Constantine Dovrolis. "The Internet Is Flat: Modeling the Transition from a Transit Hierarchy to a Peering Mesh." In: *Proceedings of the 6th International COnference on - Co-NEXT '10*. The 6th International COnference. Philadelphia, Pennsylvania: ACM Press, 2010, p. 1. ISBN: 978-1-4503-0448-1. DOI: 10.1145/1921168.1921196. URL: http://portal.acm.org/citation.cfm?doid=1921168.1921196.

[7] Ahmed Elmokashfi, Amund Kvalbein, and Constantine Dovrolis. "BGP Churn Evolution: A Perspective from the Core." In: *IEEE/ACM Trans. Netw.* 20.2 (Apr. 2012), pp. 571–584. ISSN: 1063-6692.

[8] Ahmed Elmokashfi, Amund Kvalbein, and Constantine Dovrolis. "On the Scalability of BGP: The Role of Topology Growth." In: *IEEE Journal on Selected Areas in Communications* 28.8 (Oct. 2010), pp. 1250–1261. ISSN: 0733-8716. DOI: 10.1109/JSAC.2010.101003. URL: http://ieeexplore.ieee.org/document/5586438/.

[9] Cyril Gavoille and Stéphane Pérennès. "Memory Requirement for Routing in Distributed Networks." In: *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing - PODC '96*. The Fifteenth Annual ACM Symposium. Philadelphia, Pennsylvania,

United States: ACM Press, 1996, pp. 125–133. ISBN: 978-0-89791-800-8. DOI: 10.1145/248052.248075. URL: http://portal.acm.org/citation.cfm?doid=248052.248075.

[10] Vasileios Giotsas, Shi Zhou, Matthew Luckie, and kc claffy. "Inferring Multilateral Peering." In: *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies - CoNEXT '13*. The Ninth ACM Conference. Santa Barbara, California, USA: ACM Press, 2013, pp. 247–258. ISBN: 978-1-4503-2101-3. DOI: 10.1145/2535372.2535390. URL: http://dl.acm.org/citation.cfm?doid=2535372.2535390.

[11] Dmitri Krioukov, k c claffy, Kevin Fall, and Arthur Brady. "On Compact Routing for the Internet." In: *ACM SIGCOMM Computer Communication Review* 37.3 (July 20, 2007), pp. 41–52. ISSN: 0146-4833. DOI: 10.1145/1273445.1273450. URL: https://dl.acm.org/doi/10.1145/1273445.1273450.

[12] Lixin Gao and J. Rexford. "Stable Internet Routing without Global Coordination." In: *IEEE/ACM Transactions on Networking* 9.6 (Dec./2001), pp. 681–692. ISSN: 10636692. DOI: 10.1109/90.974523. URL: http://ieeexplore.ieee.org/document/974523/.

[13] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and kc claffy. "AS Relationships, Customer Cones, and Validation." In: *Proceedings of the 2013 Conference on Internet Measurement Conference - IMC '13*. The 2013 Conference. Barcelona, Spain: ACM Press, 2013, pp. 243–256. ISBN: 978-1-4503-1953-9. DOI: 10.1145/2504730.2504735. URL: http://dl.acm.org/citation.cfm?doid=2504730.2504735.

[14] Ricardo V. Oliveira, Rafit Izhak-ratzin, Beichuan Zhang, Lixia Zhang, Ricardo V. Oliveira, Rafit Izhak-ratzin, Beichuan Zhang, and Lixia Zhang. "Measurement of Highly Active Prefixes in BGP." In: *IEEE GLOBECOM* 2005 ().

[15] *Potaroo*. URL: http://bgp.potaroo.net.

[16] *Routing Algorithms Simulator*. URL: https://github.com/dedis/student_20_compactInternet.

[17] Mikkel Thorup and Uri Zwick. "Approximate Distance Oracles." In: *Journal of the ACM* 52.1 (Jan. 1, 2005), pp. 1–24. ISSN: 00045411. DOI: 10.1145/1044731.1044732. URL: http://portal.acm.org/citation.cfm?doid=1044731.1044732.

[18] Yangyang Wang and Keyao Zhang. "Quantifying the Flattening of Internet Topology." In: *Proceedings of the 11th International Conference on Future Internet Technologies - CFI '16*. The 11th International Conference. Nanjing, China: ACM Press, 2016, pp. 113–117. ISBN: 978-1-4503-4181-3. DOI: 10.1145/2935663.2935682. URL: http://dl.acm.org/citation.cfm?doid=2935663.2935682 (visited on 05/26/2020).