Modelling the BGP Protocol

FED4FIRE
FEDERATION FOR FIRE PLUS

UNIVERSITÀ
DI TRENTO

A technical annex of the Internet of Fire (IoF) project, financed by the Fed4FIRE+ Open Call #5

Mattia Milani, Luca Baldesi, Michele Segata, Leonardo Maccari, Renato Lo Cigno
January 17, 2020

1 Introduction

An integral component of the IoF project was to develop a formal model that describes a network of routers running the BGP protocol. BGP evolved in an extremely complex and feature rich protocol, which we simplified in an abstraction that could be used to create a virtualized network of routers with the minimal information necessary to study BGP performance.

For the purposes of the project, we limit the kinds of messages that a BGP node (router) can use to two kinds of messages:

- UPDATE;
- WITHDRAWAL.

We omit other messages that are devoted to connection management. We use the terms prefix and destination interchangeably, they indicate the network address of an IP subnet exported by one of the routers.

A route is a pair $r = (d, \xi)$ of destination and attributes. ξ contains information such as:

- path, which is a list of autonomous systems (ASs);
- originator id, which is the IP address for the entry point of the originating AS.

2 The BGP graph

We model a BGP network as a connected, non directed graph $G(V, E)$. Following the canonical model used in the literature [1] links can be either peer-to-peer or customer-provider depending on the nodes involved and their commercial agreements. We indicate with $\Lambda = \{\pi, c, s\}$ the labels indicating peer, consumer and provider respectively. The function $\lambda : E \to \Lambda$ assigns to a node edge (i, j) the role i has with respect to j. Hence, $\lambda(i, j) = \pi \iff \lambda(j, i) = \pi$ and $\lambda(i, j) = c \iff \lambda(j, i) = s$

We call $N_i = \{j \in V : \exists (i, j) \in E\}$ the set of neighbors for node i and $C_i = \{j \in N_i : \lambda(i, j) = s\}$ the set of customers of node i.

We indicate as $G^k(V^k, E^k)$ the fully connected sub graph of tier-one BGP nodes, such that, $V^k \subset V$, $E^K = V^k \times V^k$. Note that, $\lambda(i, j) = \pi \; \forall i, j \in V^k$. When generating Internet-like graphs we always include a sub graph G^k.

3 Advertisement propagation

When a node $i \in V$ generates an advertisement (ADV) for the route r, it is propagated to all nodes using the links of $G(V, E)$. Propagation considers:

- $\lambda()$;
- node policies blocking ADV forwarding (which we don't consider for the time being)

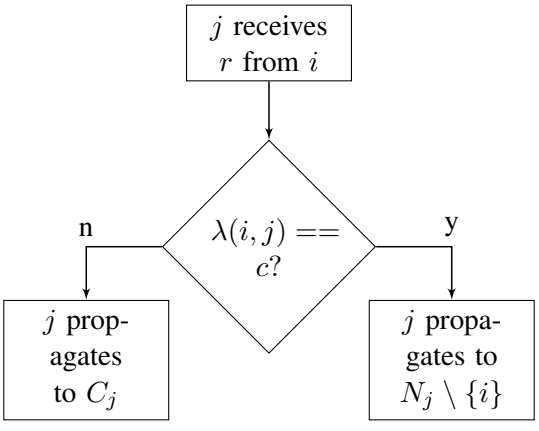

Fig. 1: Flow chart for ADV forwarding.

We use the standard assumption of *no-valley* and *prefer-customer*, depicted in fig. 1 [1], which means that routes learned from customers are announced to all neighbors, while routes learned from peers or providers are announced only to customers.

Propagation policies together with G structure produce a very specific pattern of route propagation from one node $i \in V \setminus V^k$ to the rest of the network (we assume that the majority of the modifications to the BGP graph come from nodes outside of G^k). We model this pattern by sub-dividing the graph G in three components as the propagation happens in three phases:

- Phase 1: node $i \in V \setminus V^k$ generates an ADV for route r and this spreads toward a node $z \in V^k$; we build a sub-graph $G^{(1)}(V^{(1)}, E^{(1)})$ made of all nodes and edges that are involved in this phase;
- Phase 2: ADV propagates toward tier one nodes V^k; the resulting sub-graph is $G^{(2)} = G^k$;
- Phase 3: ADV propagates from $z \in V^k$ to all the nodes $j \in V \setminus V^k \setminus V^{(1)}$; we call the resulting sub-graph $G^{(3)}(V^{(3)}, E^{(3)})$.

Phase 1 graph Let i be the ADV originator and $j \in V$ another node in G. $j \in V^{(1)} \iff \exists p_{ij}$ a path in G between i and j, with $p_{ij} = (i, k_0) \dots (k_l, j)$ such that:

- $k_x \notin V^k,\ \forall x = 0, \dots, l$;
- we have one of the following conditions:
  - $\lambda(\rho, t) = c\ \forall (\rho, t) \in p_{ij}$;
  - $\exists z \leq l : \lambda(\rho, t) = c\ \forall (\rho, t) \in p_{ik_z}, \lambda(k_z, k_{z+1}) = \pi, \lambda(\rho, t) = s\ \forall (\rho, t) \in p_{k_{z+1}, j}$

Phase 3 graph Phase 3 graph is the one connecting the nodes V^k to all the nodes $V^{(3)} = V \setminus V^k \setminus V^{(1)}$. $j \in V^{(3)} \iff \exists p_{ij}$ a path in G between $i \in V^k$ and j, with $p_{ij} = (i, k_0) \dots (k_l, j)$ such that:

- $j, k_x \notin V^k \cup V^{(1)},\ \forall x = 0, \dots, l$;
- $\lambda(\rho, t) = s\ \forall (\rho, t) \in p_{ij}$;

Propagation graph Given an originator $i \in V \setminus V^k$, we call propagation graph $G^P = (V^{(1)} \cup V^{(2)} \cup V^{(3)}, E^{(1)} \cup E^{(2)} \cup E^{(3)})$ the graph comprising all the feasible links for ADV propagation.

3.1 Node Policy

Nodes evaluate route ADVs with a function $\Gamma : R \to [0, 100] \cap \mathbb{N}$, where R is the set of routes. Given two routes r_1, r_2, a node prefers r_1 if $\Gamma(r_1) \geq \Gamma(r_2)$ and r_2 otherwise.

As noted before, $r \in R$ is a couple (d, ξ) of destination and attributes. The simplest form of policy, that we use in our experiments, returns the (possibly weighted) length of the AS path indicated in the attributes ξ (hop distance to d). Policies are implemented defining the Γ_i function for each $i \in V$.

4 MRAI timers

For each neighbour j, every node i consider an MRAI timer value T_{ij}. The typical value for it is 30 seconds. Every time an MRAI timer is setted undergoes a variation that could reduce the timer up to 25% of T_{ij} according to the RFC 4271.

A router continuously receives ADV messages containing information about routes, which can differ for the destination d or for attributes ξ. Different routes are always processed separately, and if needed, a new ADV message is generated to propagate the change of information to the rest of the nodes.

For each neighbor j, every node i uses a MRAI timer value T_{ij}, which is typically set to 30 seconds. For destination d, node i will not send more than one message every T_{ij} seconds. Whenever node i receives a route UPDATE for destination d it updates its Adj-RIBs-In[1], and if the new route r obtains a score $\Gamma(r)$

[1] the information structure pertaining route ADV

higher than the current route, node i selects ξ_m as the new preferred attribute for destination d, and it schedules a new ADV to be sent according to MRAI. When the MRAI timer fires only the new best option is advertised.

For each destination d, node i has a queue of route advertisements $r_1, \ldots, r_m$, and, for each $j \in N_i$ it keeps a function $\tau(i, j, d) : V \times V \times D \rightarrow [0, T_{ij}]$ expressing the minimum remaining time for the next transmission.

4.1 Exponential Path Exploration

Fabrikant et al. describe a potential issue related to timers and path exploration given a certain configuration of MRAI timers and some specific (but realistic) topologies [2]. This issue happens when there are multi-paths in the propagation graph and a specific sequence of timers T_{ij} in the propagation path. In the following we suppose that all nodes i, in their initial state have $\tau(i, j, d) = 0,\ \forall j \in N_i$ (they are ready to fire a new ADV).

Duplicate ADV. Let $i \in V$ originate a route advertisement r for destination d and let $j \in V$ receive it at certain time t_0. Node j processes the ADV, if j decides that the new route is the best one to reach d through another node, let's say $z \in V$, it will implement that in its routing table and will propagate the ADV with the modified ξ. However, node z also receives the ADV from i at t_0 and changes its routing table. Node z then propagates the ADV at time greater than $t_0 + \epsilon$. Then j receives the updated route from z (which may change again its routing table) right after having propagated an ADV message, but it cannot propagate the new information as $\tau(j, k, d) > 0,\ \forall k \in N_j$. Node j will have to wait for the MRAI timer to expire.

This behavior makes j send two ADVs for the same destination d, the first of which was computed on outdated information. Worse than that, the updated ADV cannot be propagated before T_{ik} seconds to the neighbors $k \in N_j$.

Duplicate duplication. Suppose that $k \in N_j$ receives the first ADV from j at time $t_0 + \epsilon$ (we omit considering he propagation time). Node k generates an ADV and propagates the wrong information. The duplication can happen again and k receives at t_1 an ADV from a node h in a path from k and j. So k has to wait $\tau(j, x, d)$ seconds before sending the updated information to its neighbor $x \in N_k$. Now, if $T_{kx} > T_{jk}$ then k will receive the update from j and send only one duplicate ADV. But, if $T_{kx} < T_{jk}$ it means that the aforementioned step can be repeated twice and k will send a total of 4 ADV messages.

Exponential explosion. In general and in presence of multi-parts, if $p_{ij} = (i, k_0), \ldots, (k_l, j)$ is a propagation path, and the MRAI timers decrease on the path, then the number of ADV messages (and route oscillation) is exponential on the overall number of decreasing timers on the path.

5 Sketch of a Solution

Ideally, we would like to have increasing values of T_{ij} along the propagation paths, but on the other hand, we can not indefinitely increase MRAI timers. We propose two approaches, in increasing order of complexity that we will test in our experiments.

5.1 Increasing Timers on Propagation Path

We propose an extension to the ADV UPDATE message that includes the MRAI timer of the sending node. This information is used by the next node on the path to set its timer accordingly.

1. Node i advertise route r;
2. Node i sends an UPDATE to $j \in N_i$ for r including the value for MRAI timer T_{ij};
3. If node j accepts r than j sets its MRAI for its neighbor $z \in N_j$, $T_{jz} = T_{ij} + c$.

The value of c is a reasonable enough, small value.

5.2 Centrality-based Approach

The defect of the previous approach is that it indefinitely increases the MRAI timer, and thus may enlarge the propagation time indefinitely. The intuition at the base of the second approach is that it is useful to apply the first approach only on $V^{(1)}$, but when the updated information reaches some node V^k than, we need to speed up the update again. We can use some centrality metric to infer the position of a node and tune the timers T_{ij} accordingly.

Considering a graph-wide maximum timer $T = 30$ and Destination Partial Centrality (DPC) $c_i \in [0, 1]$ for node i, one strategy could be to assign:

- $T_{ij} = \frac{Tc_i}{2},\ \forall i \in V^{(1)}$
- $T_{ij} = \frac{T}{2},\ \forall i \in V^k$
- $T_{ij} = \frac{T(1-c_i)}{2} + \frac{T}{2} = \frac{T(2-c_i)}{2},\ \forall i \in V^{(3)}$

With this approach as the timers increase in $V^(1)$ and then decrease again in $V^(3)$.

References

[1] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the scalability of bgp: The role of topology growth," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1250–1261, October 2010.

[2] A. Fabrikant, U. Syed, and J. Rexford, "There's something about mrai: Timing diversity can exponentially worsen bgp convergence," *2011 Proceedings IEEE INFOCOM*, pp. 2975–2983, 2011.