BGP pySim documentation

Lorenzo Ghiro

Abstract
A python simulator has been developed to replicate the exponential path exploration problem described in [1]. The simulator workflow and kind of events, together with the BGP node logic implemented by the pySim, are described in this document.

1. Simulator high-level architecture

The simulator requires:

1. The network topology, described by a graphml file
2. The output folder

2. Initialization

The graphml is parsed to:

1. initialize node objects with their TYPE and prefixes to be exported
2. setup neighbourhood relationships. This includes peering or customer/provider role assignment and per-neigh default-MRAI assignment

3. Node implementation

Node attributes

A node has/is described by, and keeps updated the following:

1. nodeID and nodeType
2. **neighs**: a dictionry with neighID as keys and *(relation, mrai)* as neighbour attributes
3. **exportPrefixes**: a list of prefixes exported by this node
4. **RoutingTable**: an object with convenient methods to install routes and to remeber received updates, so to be ready to install backup routes

Routing table

A routing table is a dictionary indexed by known prefixes. For each prefix these info are kept updated:

1. NH and AS-PATH
2. PREFERENCE, computed according to the **policy function**[1]
3. MRAIs: a dictionary indexed by neighbours' ids. For each neigh the time after which is possible to send an update is maintained.
4. SHARED-FLAG: again a per-neigh indexed dictionary. A flag per neighbour is maintained to remember if an update has been sent or not to this neigh for this prefix. Thanks to these flags and assuming no losses in sending updates over TCP connections, we will see the network "*silent*" at convergence.
5. adjRIBin: again per-neigh dict. The last update received from the indexed neigh for the given prefix is maintained here

[1] The policy function comes as a separate py file, to ease extension and multiple versions implementation in the future

PROCESSING received updates

When a node receives an update, it schedules its own "state-transition" after a short delay. This delay model the non-zero time required to process an update. After this delay, the node run a (instantaneous) DECISION-PROCESS, deciding what to do with the received update. This process may trigger the sending of an update.

SENDING updates

The send-update routine is responsible to disseminate those routes that, compared to the last sent update, are new or have been modified (modified once or multiple time). The send-update can fail therefore if:

1. the route-to-announce has been already shared with neighbours
2. the mrai for the announced prefix (with a given neigh), has not expired

4. Decision Process

Nodes react to only one kind of event, called: DECISION-PROCESS. The event is scheduled when nodes receive an update or set an mrai. In the first case, the DECISION-PROCESS starts after a little delay, modelling the non-zero processing time [1]. In the 2nd case, the DECISION-PROCESS is triggered when an mrai expires so that, if a node has a route still not shared with neighbour, then will send updates. The full decision process is illustrated by Fig. 1.

Workflow

1. Put received updates in the adjRIBin
2. Phase 1: Compute PREFERENCE applying the policy function to all updates in adjRIBin
3. Phase 2: For each destination (in our case just one), select and install the route with higher preference. If

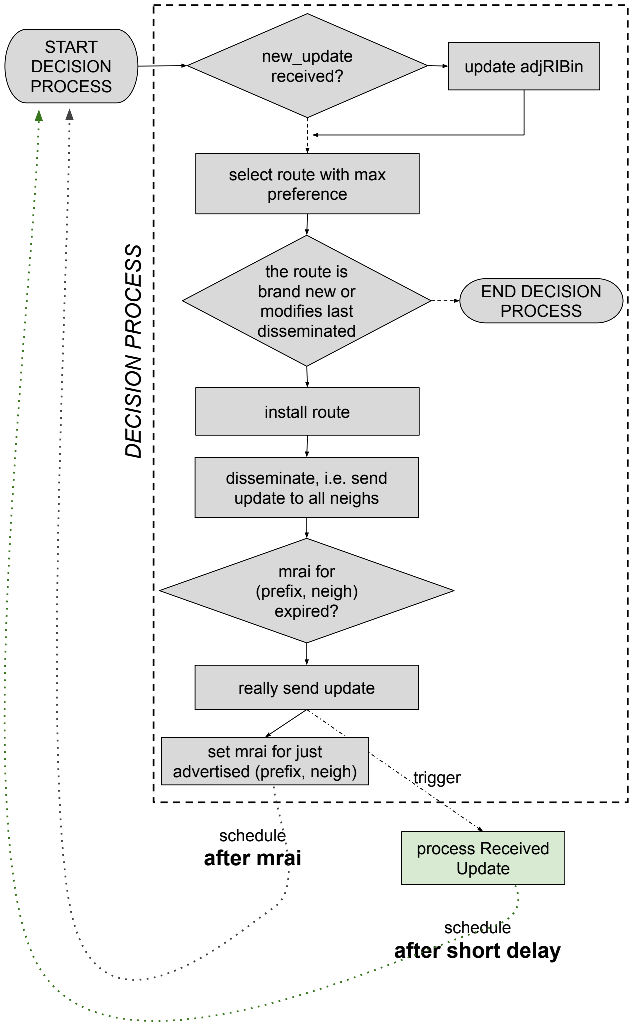

Figure 1. Decision Process. In grey the main actions of a node, in green the actions triggered for neighbouring node.

the best&installed route is related to a new or modified route, then the installing routine sets the SHARED-FLAG of this route to False (i.e. not shared yet, an advertisement must be sent).

4. Phase 3 (Dissemination): The node tries to send updates about routes that have been not shared yet. Fails if MRAI is not expired. In case of success, the SHARED-FLAG is reset to True and MRAI is reset as well.

5. First results

Based on the usual topology (Fig. 3), I tested 3 configurations:

- *a)* MRAI DECREASING (cut by 2) from d to X4. e.g. $X_1,Y_1 = 8s \quad X_2,Y_2 = 4s$...
- *b)* MRAI reversed (2x) from d to X4 e.g. $X_1,Y_1 = 2s \quad X_2,Y_2 = 4s$...
- *b)* MRAI=0 for all

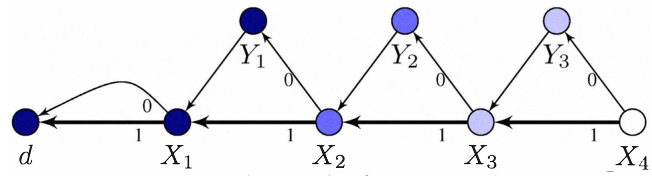

Figure 2. The usual reference topology

Scenario	#Updates	#RT INSTALLMENTs	CONV TIME [sec]
a	21	21	30
b	15	15	32
c	18	15	16

6. Processing Logs

The simulator comes with a script for processing simulation's logs. The script produces these kind of summary:

```
+----------+--------------------+------------------+
| #Updates | #ROUTE_INSTALLMENTs |     CONV_TIME    |
+----------+--------------------+------------------+
|    21    |         21         | 30.00268149750159 |
+----------+--------------------+------------------+
Time progress of route installments after link failure
+--------------------+----+----+----+-----+-----+------+
|        TIME        | Y1 | X2 | Y2 |  X3 |  Y3 |  X4  |
+--------------------+----+----+----+-----+-----+------+
|        0.0         | 1  | 11 | 11 | 111 | 111 | 1111 |
| 24.001264765485192 | "" | 10 | "" |  "" |  "" |  ""  |
| 24.001325136335424 | "" | "" | "" | 110 |  "" |  ""  |
| 24.00135543421853  | "" | "" | "" |  "" |  "" | 1110 |
| 24.002211885152565 | 0  | "" | "" |  "" |  "" |  ""  |
| 24.002282752670393 | "" | 1  | "" |  "" |  "" |  ""  |
| 24.002303690505126 | "" | "" | 10 |  "" |  "" |  ""  |
| 24.002326162358838 | "" | "" | "" | 101 |  "" |  ""  |
| 24.002394125137783 | "" | "" | "" |  "" | 110 |  ""  |
| 24.002441393334106 | "" | "" | "" |  "" |  "" | 1101 |
| 26.001353794863988 | "" | "" | "" |  "" |  "" | 1100 |
| 26.002378236209587 | "" | "" | "" |  "" | 101 |  ""  |
| 26.00252167339219  | "" | "" | "" |  "" |  "" | 1011 |
| 28.00130003782124  | "" | "" | "" | 100 |  "" |  ""  |
| 28.001379477008342 | "" | "" | "" |  "" |  "" | 1010 |
| 28.00239594806414  | "" | "" | 1  |  "" |  "" |  ""  |
| 28.00239656574221  | "" | "" | "" |  11 |  "" |  ""  |
| 28.002408363544326 | "" | "" | "" |  "" | 100 |  ""  |
| 28.002592896953434 | "" | "" | "" |  "" |  "" | 1001 |
| 30.00144331119337  | "" | "" | "" |  "" |  "" | 1000 |
| 30.002468912802247 | "" | "" | "" |  "" |  11 |  ""  |
| 30.00268149750159  | "" | "" | "" |  "" |  "" | 111  |
+--------------------+----+----+----+-----+-----+------+
--- Desktop/Link to BGPpysim ‹bgpPYSIM* M› »
```

Figure 3. Output from processing script. X_1 is not processed as it is the node that triggers the storm of updates.

References

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