would be mutually recursive routes). This check ensures that a BGP speaker does not install routes in the Routing Table that will be removed and not used by the speaker. Therefore, in addition to local Routing Table stability, this check also improves behavior of the protocol in the network.

Whenever a BGP speaker identifies a route that fails the resolvability check because of mutual recursion, an error message SHOULD be logged.

9.1.2.2. Breaking Ties (Phase 2)

In its Adj-RIBs-In, a BGP speaker may have several routes to the same destination that have the same degree of preference. The local speaker can select only one of these routes for inclusion in the associated Loc-RIB. The local speaker considers all routes with the same degrees of preference, both those received from internal peers, and those received from external peers.

The following tie-breaking procedure assumes that, for each candidate route, all the BGP speakers within an autonomous system can ascertain the cost of a path (interior distance) to the address depicted by the NEXT_HOP attribute of the route, and follow the same route selection algorithm.

The tie-breaking algorithm begins by considering all equally preferable routes to the same destination, and then selects routes to be removed from consideration. The algorithm terminates as soon as only one route remains in consideration. The criteria MUST be applied in the order specified.

Several of the criteria are described using pseudo-code. Note that the pseudo-code shown was chosen for clarity, not efficiency. It is not intended to specify any particular implementation. BGP implementations MAY use any algorithm that produces the same results as those described here.

a) Remove from consideration all routes that are not tied for having the smallest number of AS numbers present in their AS_PATH attributes. Note that when counting this number, an AS_SET counts as 1, no matter how many ASes are in the set.

b) Remove from consideration all routes that are not tied for having the lowest Origin number in their Origin attribute.

c) Remove from consideration routes with less-preferred MULTI_EXIT_DISC attributes. MULTI_EXIT_DISC is only comparable between routes learned from the same neighboring AS (the neighboring AS is determined from the AS_PATH attribute). Routes that do not have the MULTI_EXIT_DISC attribute are considered to have the lowest possible MULTI_EXIT_DISC value.

This is also described in the following procedure:

```
for m = all routes still under consideration
    for n = all routes still under consideration
        if (neighborAS(m) == neighborAS(n)) and (MED(n) < MED(m))
            remove route m from consideration
```

In the pseudo-code above, MED(n) is a function that returns the value of route n's MULTI_EXIT_DISC attribute. If route n has no MULTI_EXIT_DISC attribute, the function returns the lowest possible MULTI_EXIT_DISC value (i.e., 0).

Similarly, neighborAS(n) is a function that returns the neighbor AS from which the route was received. If the route is learned via IBGP, and the other IBGP speaker didn't originate the route, it is the neighbor AS from which the other IBGP speaker learned the route. If the route is learned via IBGP, and the other IBGP speaker either (a) originated the route, or (b) created the route by aggregation and the AS_PATH attribute of the aggregate route is either empty or begins with an AS_SET, it is the local AS.

If a MULTI_EXIT_DISC attribute is removed before re-advertising a route into IBGP, then comparison based on the received EBGP MULTI_EXIT_DISC attribute MAY still be performed. If an implementation chooses to remove MULTI_EXIT_DISC, then the optional comparison on MULTI_EXIT_DISC, if performed, MUST be performed only among EBGP-learned routes. The best EBGP-learned route may then be compared with IBGP-learned routes after the removal of the MULTI_EXIT_DISC attribute. If MULTI_EXIT_DISC is removed from a subset of EBGP-learned routes, and the selected "best" EBGP-learned route will not have MULTI_EXIT_DISC removed, then the MULTI_EXIT_DISC must be used in the comparison with IBGP-learned routes. For IBGP-learned routes, the MULTI_EXIT_DISC MUST be used in route comparisons that reach this step in the Decision Process. Including the MULTI_EXIT_DISC of an EBGP-learned route in the comparison with an IBGP-learned route, then removing the MULTI_EXIT_DISC attribute, and advertising the route has been proven to cause route loops.

d) If at least one of the candidate routes was received via EBGP, remove from consideration all routes that were received via IBGP.

e) Remove from consideration any routes with less-preferred interior cost. The interior cost of a route is determined by calculating the metric to the NEXT_HOP for the route using the Routing Table. If the NEXT_HOP hop for a route is reachable, but no cost can be determined, then this step should be skipped (equivalently, consider all routes to have equal costs).

This is also described in the following procedure.

```
for m = all routes still under consideration
    for n = all routes in still under consideration
        if (cost(n) is lower than cost(m))
            remove m from consideration
```

In the pseudo-code above, cost(n) is a function that returns the cost of the path (interior distance) to the address given in the NEXT_HOP attribute of the route.

f) Remove from consideration all routes other than the route that was advertised by the BGP speaker with the lowest BGP Identifier value.

g) Prefer the route received from the lowest peer address.

9.1.3. Phase 3: Route Dissemination

The Phase 3 decision function is invoked on completion of Phase 2, or when any of the following events occur:

a) when routes in the Loc-RIB to local destinations have changed

b) when locally generated routes learned by means outside of BGP have changed

c) when a new BGP speaker connection has been established

The Phase 3 function is a separate process that completes when it has no further work to do. The Phase 3 Routing Decision function is blocked from running while the Phase 2 decision function is in process.

All routes in the Loc-RIB are processed into Adj-RIBs-Out according to configured policy. This policy MAY exclude a route in the Loc-RIB from being installed in a particular Adj-RIB-Out. A route SHALL NOT