BGP Routing Scalability: Reflectors vs. Route Servers

Michael J. Lewchuk
Department of Electrical and Computer Enginering, University of Alberta
Edmonton, Alberta, Canada

and

Michael H. MacGregor
Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada

Abstract

Autonomous systems need to be connected using BGP servers. To reduce the complexity of connecting these systems together, researchers have suggested that route servers or routing databases be used. While both route reflectors and routing databases offer significant scalability advantages over standard full mesh BGP, they also introduce problems in routing and survivability. This paper explores the relative advantages and disadvantages of full mesh BGP, route reflectors, and routing databases.

Keywords: Routing, Databases, Servers, Reflectors, Scalability, Network, BGP.

1. Introduction

As the Internet grows, connecting autonomous systems (AS) to each other becomes increasingly complex. Each AS must have routing tables that tell it how to reach external systems. This is done using BGP, the Border Gateway Protocol [7][8][9]. A BGP node obtains its information by communicating with the external systems. Each external AS directly connected to a BGP node will inform it when a new system is reachable and when a mentioned destination is no longer reachable. In standard BGP, all BGP nodes are fully interconnected; this requires $O(N^2)$ TCP/IP connections, so the method does not scale well. Traina [T1] indicates that ASs of up to 90 BGP peers have been used, requiring 4095 TCP/IP connections for BGP route updates.

The number of connections also influences the amount of network bandwidth dedicated to routing. Depending on the physical structure of the network, each update must travel between one and (N-1) hops to reach its destination, where N is the number of nodes in the network. This results in $O(N^3)$ transmissions as a routing update traverses the network over the $O(N^2)$ connections. Worse, Floyd [4] indicates that updates may become synchronized, resulting in periods of long delay followed by a large burst of updates. The updates will then be queued, resulting in further delays.

Establishing a single TCP/IP connection is not a problem. However, every TCP/IP connection consumes time and space – keep-alives must be sent every so often, and each connection consumes kernel table space. If a connection is very lightly used, most of its traffic will be keep-alive messages. If a connection is heavily used, the quantity of routing updates being sent may impact network performance

Two techniques in which several BGP nodes are gathered together into a cluster have been suggested to improve scalability. In these techniques, each cluster has an external representative appointed by the network

administrator. The representative informs each member of its cluster of any important updates, and transmits updates generated inside the cluster to all external BGP entities.

2. Route Reflectors

One method of clustering is to use **route reflectors**[1]. Here, the representative node acts as a filter and reflector for all other nodes in its cluster. It decides which updates should be kept and which should be discarded. Given a cluster of size M in a network of size N, the number of BGP connections is reduced from $O(N^2)$ to $O(M^2)$. For example, a large BGP network may contain 80 BGP speakers – in the standard full mesh, each node would be connected to 79 others. Using route reflectors and this same maximum of 79 BGP peers for any one node, a network of 1640 BGP speakers can be constructed using 40 clusters, each cluster containing 41 nodes. Each node within a cluster peers with 40 other nodes. The route reflector in each cluster also peers with the 39 other route reflectors. This requires only 33580 TCP/IP connections, compared with over 1.3 million for the full mesh approach.

Allowing a route reflector to determine route suitability for a cluster does have some disadvantages. If a route reflector is on the edge of a cluster, it may choose to discard a path that would benefit nodes on the other side of the cluster. It may also accidentally create loops by discarding routes - see Dube[2] for an example.

Multiple route reflectors may be used in a cluster to establish connections to different parts of the AS. The duties of a single route reflector are then spread among many. Not only does this reduce the load on each reflector, it also allows each reflector to obtain data from nearby areas. Each reflector is now closer to the source of a route than any other node in its cluster, so it will likely prune only routes that the other cluster nodes would also discard. This is probably the best of all possible worlds, balancing pruning, message transmission, and loop prevention.

Route reflectors use $O(M^2)$ connections per cluster, where M is the number of nodes in a cluster. Each node has M connections, being connected to all nodes in its cluster. Each reflector is connected to all nodes in its cluster, and all clusters one level up. Thus, each reflector holds $M+\mu$ connections, where μ is the number of connections one level up. There are a total of (N/M) clusters in the AS. The minimum number of connections occurs when the sum of the connections within and between clusters is minimized. This occurs at the minimum of $(N/M)(M)(M-1)/2 + (N/M)((N/M)-1)/2$, which is at $M^3+M=2N$, or approximately $M=(\sqrt[3]{2N})$. This moves a significant amount of the connection processing from the cluster nodes to the reflectors. For example, in the 1640 node network, minimizing the number of connections results in 110 clusters of 15 nodes each. The nodes in the clusters have 14 connections each, while each reflector has 123.

3. Route Servers

Another method is to designate a representative as a **route server**, or **BGP routing database**[5]. The route server stores all incoming route information and forwards it. Incoming routes are forwarded to all nodes in the cluster, and internally generated routes are forwarded to all peers outside of the cluster. The route server performs no actual “thinking” for its cluster; it forwards all updates as-is. This method reduces the number of TCP/IP connections drastically. Nodes in the cluster are connected to the route server directly, forming a

tree structure rather than a mesh of peers. A cluster of M nodes using a route server requires only (M-1) connections. This is a large savings over both the standard $O(N^2)$ connections and the route reflector's $O(M^2)$ connections.

Route servers simply forward any routes received without processing them; they do not presume to make decisions for their cluster. The price for this is an increased number of updates relative to a cluster served by a reflector, since there is no possibility of pruning routes. However, this means that a route server will not introduce routing loops due to bad pruning decisions. Since every route that can be used is forwarded to the server's peers inside the cluster, they must also make correct decisions.

Route servers have a second plus – each node requires only a connection to the route server. Only (M-1) connections are required for a cluster of M nodes. Thus, route servers result in the absolute minimum number of connections. The numerical minimum occurs at the minimum of (N/M)(M-1)+(N/M)((N/M)-1)/2, which is at M=(2N/3). This suggests that route servers are most effective in situations that call for small numbers of large clusters, as opposed to a large number of small clusters which is more favorable to route reflectors.

Route servers also add reconfiguration robustness. As long as the logical tree of BGP connections to the server is reasonably consistent with the physical network structure, it should be easy to add a node, combine two nodes, or shift a subtree from one node to another. The number of transmissions required is also very low, generally O(M). The tree structure can have another unintended bonus: Huffman analysis[3] can be used by a network administrator to balance the network structure.

However, if only one route server is used per cluster the price of a failure could be the loss of BGP updates for that entire cluster. If redundant servers are used to share the workload, each must monitor the nodes served by the others, to make sure that a node is not receiving duplicate routing information. Each must also maintain a connection with the other local route servers to make sure that they are alive and that their states are synchronized. If one server dies, the remaining servers must make sure that all responsibilities handled by the dead node are taken up by exactly one other node.

A system of route servers is suited for sparsely connected tree-like structures, where the distance between any two nodes is approximately equal to the distance from one to the root and then to the other. Clusters based on route servers share many of the characteristics of trees: if an edge fails, the graph becomes disconnected. This actually simplifies the design of route servers systems somewhat. There is no way to reroute around a lost edge in a tree, because the graph is minimally connected, so clusters using route servers have fewer alternatives when considering how to compensate for a failed node or connection. However, this does not mean that prudence should be ignored in favor of reckless abandon. Redundant systems on separate physical lines will decrease the chance that one faulty system or line will cut the cluster off from the outside world.

4. Comparison

The question of which technique to use is most easily answered on a case-by-case basis using Table 1. Is some of the network sparsely connected? Can the occasional routing loop be tolerated? How survivable is the network, and how survivable does it have to be? How much overhead is acceptable in terms of number of TCP/IP connections for BGP, BGP message loads and BGP route processing? A standard BGP mesh is viable for small networks and is the simplest of the three. If scalability is not a problem, the standard approach will be the most worry-free.

However, if scalability is a concern, then a combination of techniques may be best. Route reflectors use fully connected clusters; thus the original scalability problem of $O(N^2)$ connections is reduced, but not eliminated. Route servers minimize the number of connections at the cost of increased fragility. Route reflectors are best implemented in networks of large, fat clusters to minimize the likelihood of routing loops. Route servers are best suited to sparsely connected networks where the inherent fragility of the physical structure matches the inherent fragility of the logical structure. Route reflectors assume control over which routes enter their clusters, introducing the potential for routing loops to occur, while route servers do not. Many route reflectors can be spread out around a cluster, with each doing its job independently. In contrast, if multiple route servers are used, they must be coordinated.

As compared to minimizing the number of connections, minimizing the number of updates can be difficult. If an external network becomes disconnected, or if dissimilar routes of similar distances exist, flapping may induce large routing changes. Standard mesh peering blares out all routing changes to all nodes in the most inefficient, yet most robust, manner possible. Route reflectors can improve upon this by eliminating unnecessary routing changes. Route reflectors can also ask the philosophical question "Is a difference that makes no difference really a difference?" Specifically, local updates that do not affect external clusters may simply be terminated at the appropriate reflectors. Route servers can also store alternatives locally to minimize the number of updates. Because the routing is tree structured, a routing change must be propagated down the tree, but it can be filtered for appropriateness when moving up the tree toward the server.

Characteristic	Full mesh	Reflectors	Route Servers
Updates	Direct	Filtered	Transparent
Communication sites	Individuals	As many as desired	Few
Crash survivability	Very high	Potentially high	Low
Route selection	Decentralized	Centralized	Decentralized
Number of connections	Very high	Moderate	Minimal
Placement in cluster	-	Cluster border	Cluster hub
Load	Universally high	RR=High, Node=Low	Configurable
Scalability	Minimal	Global, but not local	Global and local
Natural Topology	Fully connected	Hierarchical / Clustered	Hierarchical / Tree
Problems	Scalability	Loops may form	Centralized server

Table 1 -- Full Mesh vs. Reflectors and Servers

For all methods, caching routes can improve performance. Paxson[6] indicates that route flaps tend to occur every KT units of time where T is a timeout value (e.g. 30 seconds). The most annoying updates have $K \approx 0$, since they recur nearly instantaneously, averaging 1-5 seconds between updates. Paxson indicates that these updates occur due to either a hardware malfunction or overzealous load balancing. These updates should be cached or one route discarded if the traffic due to the updates is too great. By monitoring the rate of incoming updates, a network administrator can select an ideal breakpoint for caching routes. Because the rates tend to be concentrated, there should be a wide gap between two consecutive clumps of updates.

5. Conclusion

Each method has its own strengths and weaknesses, and each appears to be suited for a particular type of physical network structure and general network goals. Standard BGP is suited for small networks that can tolerate high overhead and must maintain maximum reliability. Route reflectors are suited to clusters that are relatively well connected internally, and for systems that can tolerate the occasional routing loop. Route servers are best suited for sparsely internally connected tree-like clusters that have few alternate paths. The fragility of the logical topology then matches the fragility of the physical topology. A combination of these methods in a given situation is both possible and likely the best solution.

References

[1] Bates, T., and R. Chandra, "BGP Route Reflection: An alternative to full mesh IBGP", RFC 1966, June 1996.

[2] Dube, R., "A Comparison of Scaling Techniques for BGP", *Computer Communication Review*, Vol. 29, No. 3, 1998, pp. 44-46

[3] "Huffman Algorithm", in The Electrical Engineering Handbook, Second Edition, (Dorf, R.C., Ed.) © 1997 CRC Press LLC in association with the IEEE Press, pp. 410 and 168

[4] Floyd, S., and V. Jacobson, "The synchronization of Periodic Routing Messages", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2, April 1994, pp. 122-136

[5] Haskin, D., "A BGP/IRDP Route Server alternative to full mesh routing", RFC 1863, October 1995

[6] Paxson, V., "End-to-End Routing Behavior in the Internet", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, October 1997, pp. 601-615

[7] Rekhter, Y., and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995

[8] Traina, P. "Experience with the BGP-4 Protocol", RFC 1773, March 1995

[9] Traina, P., "BGP-4 Protocol Analysis", RFC 1774, March 1995

[10] Traina, P., "Autonomous System Confederations for BGP", RFC 1965, June 1996