Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 19, 2018

T. Li
Arista Networks
March 18, 2018

An Architecture for Dynamic Flooding on Dense Graphs
draft-li-dynamic-flooding-03

Abstract

Routing with link state protocols in dense network topologies can result in sub-optimal convergence times due to the overhead associated with flooding. This can be addressed by decreasing the flooding topology so that it is less dense.

This document discusses the problem in some depth and an architectural solution. Specific protocol changes are not described in this document.

Table of Contents

1. Introduction

In recent years, there has been increased focused on how to address the dynamic routing of networks that have a bipartite (a.k.a. spine-leaf or leaf-spine), Clos [Clos], or Fat Tree [Leiserson] topology. Conventional Interior Gateway Protocols (IGPs, i.e. IS-IS [ISO10589], OSPF [RFC5340]) under-perform, redundantly flooding information throughout the dense topology, leading to overloaded control plane inputs and thereby creating operational issues. For practical considerations, network architects have resorted to applying unconventional techniques to address the problem, applying BGP in the data center [RFC7938], however it is very clear that using an Exterior Gateway Protocol as an IGP is sub-optimal, if only due to the configuration overhead.

The primary issue that is demonstrated when conventional mechanisms are applied is the poor reaction of the network to topology changes. Normal link state routing protocols rely on a flooding algorithm for state distribution. In a dense topology, this flooding algorithm is highly redundant, resulting in unnecessary overhead. Each node in the topology receives each link state update multiple times.

Ultimately, all of the redundant copies will be discarded, but only after they have reached the control plane and been processed. This creates issues because significant link state database updates can become queued behind many redundant copies of another update. This delays convergence as the link state database does not stabilize promptly.

In a real world implementation, the packet queues leading to the control plane are necessarily of finite size, so if the flooding rate exceeds the update processing rate for long enough, the control plane will be obligated to drop incoming updates. If these lost updates are of significance, this will further delay stabilization of the link state database and the convergence of the network.

This is not a new problem. Historically, when routing protocols have been deployed in networks where the underlying topology is a complete graph, there have been similar issues. This was more common when the underlying link layer fabric presented the network layer with a full mesh of virtual connections. This was addressed by reducing the flooding topology through IS-IS Mesh Groups [RFC2973], but this approach requires careful configuration of the flooding topology.

Thus, the root problem is not limited to massively scalable data centers. It exists with any dense topology at scale.

This problem is not entirely surprising. Link state routing protocols were conceived when links were very expensive and topologies were sparse. The fact that those same designs are sub-optimal in a dense topology should not come as a huge surprise. The fundamental premise that was addressed by the original designs was an environment of extreme cost and scarcity. Technology has progressed to the point where links are cheap and common. This represents a complete reversal in the economic fundamentals of network engineering. The original designs are to be commended for continuing to provide correct operation to this point, and optimizations for operation in today's environment are to be expected.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Problem Statement

In a dense topology, the flooding algorithm that is the heart of conventional link state routing protocols causes a great deal of redundant messaging. This is exacerbated by scale. While the

protocol can survive this combination, the redundant messaging is unnecessary overhead and delays convergence. Thus, the problem is to provide routing in dense, scalable topologies with rapid convergence.

3. Requirements

A solution to this problem must then meet the following requirements:

Requirement 1 Provide a dynamic routing solution. Reachability must be restored after any topology change.

Requirement 2 Provide a significant improvement in convergence.

Requirement 3 The solution should address a variety of dense topologies. Just addressing a complete bipartite topology such as K5,8 is insufficient. Multi-stage Clos topologies must also be addressed, as well as topologies that are slight variants. Addressing complete graphs is a good demonstration of generality.

Requirement 4 There must be no single point of failure. The loss of any link or node should not unduly hinder convergence.

Requirement 5 Dense topologies are subgraphs of much larger topologies. Operational efficiency requires that the dense subgraph not operate in a radically different manner than the remainder of the topology. While some operational differences are permissible, they should be minimized. Changes to nodes outside of the dense subgraph are not acceptable. These situations occur when massively scaled data centers are part of an overall larger wide-area network. Having a second protocol operating just on this subgraph would add much more complexity at the edge of the subgraph where the two protocols would have to inter-operate.

4. Dynamic Flooding

We have observed that the combination of the dense topology and flooding on the physical topology in a scalable network is sub-optimal. However, if we decouple the flooding topology from the physical topology and only flood on a greatly reduced portion of that topology, we can have efficient flooding and retain all of the resilience of existing protocols.

In this idea, one node is elected to compute the flooding topology for the dense subgraph. This flooding topology is encoded into and distributed as part of the normal link state database. Nodes within the dense topology would only flood on the flooding topology. On links outside of the normal flooding topology, normal database

synchronization mechanisms (i.e., OSPF database exchange, IS-IS CSNPs) would apply, but flooding would not. New link state information that arrives from outside of the flooding topology suggests that the sender has a different or no flooding topology information and that the link state update should be flooded on the flooding topology as well.

Since the flooding topology is computed prior to topology changes, it does not factor into the convergence time and can be done when the topology is stable. The speed of the computation and its distribution is not a significant issue.

If a node has not received any flooding topology information when it receives new link state information, it should flood according to legacy flooding rules. This situation will occur when the dense topology is first established, but is unlikely to recur.

If, during a transient, there are multiple flooding topologies being advertised, then nodes should flood link state updates on all of the flooding topologies. Each node should locally evaluate the election of the lead node for the dense subgraph and first flood on the topology of the lead node. The rationale behind this is straightforward: if there is a transient and there has been a recent change in the elected node, then propagating topology information promptly along the most likely flooding topology should be the priority.

During transients, it is possible that loops will form in the flooding topology. This is not problematic, as the legacy flooding rules would cause duplicate updates to be ignored. Similarly, during transients, it is possible that the forwarding topology may become disconnected. To address this, nodes can perform a database synchronization check anytime a link is added to or removed from the flooding topology.

4.1. Leader election

The election of the node within the dense topology that computes the flooding topology is straightforward. A generalization of the mechanisms used in existing Designated Router (OSPF) or Designated Intermediate-System (IS-IS) elections would suffice. When a new node is elected and has distributed new flooding topology information, then the old node should withdraw its flooding topology information from the link state database.

4.2. Computing the Flooding Topology

There is a great deal of flexibility in how the flooding topology is computed. For resilience, it needs to at least contain a cycle of all nodes in the dense subgraph. However, additional links could be added to decrease the convergence time. The trade-off between the density of the flooding topology and the convergence time is a matter for further study. The exact algorithm for computing the flooding topology need not be standardized, as it is not an interoperability issue. Only the encoding of the result needs to be documented.

While the flooding topology should be a covering cycle, it need not be a Hamiltonian cycle where each node appears only once. In fact, in many relevant topologies this will not be possible. Consider K5,8. This is fortunate, as computing a Hamiltonian cycle is known to be NP-complete.

A simple algorithm to compute the topology for a complete bipartite graph is to simply select unvisited nodes on each side of the graph until both sides are completely visited. If the number of nodes on each side of the graph are unequal, then revisiting nodes on the less populated side of the graph will be inevitable. This algorithm can run in O(N) time, so is quite efficient.

While a simple cycle is adequate for correctness and resiliency, it may not be optimal for convergence. At scale, a cycle may have a diameter that is half the number of nodes in the graph. This could cause an undue delay in link state update propagation. Therefore it may be useful to have a bound on the diameter of the flooding topology. Introducing more links into the flooding topology would reduce the diameter, but at the trade-off of possibly adding redundant messaging. The optimal trade-off between convergence time and graph diameter is for further study.

Similarly, if additional redundancy is added to the flooding topology, specific nodes in that topology may end up with a very high degree. This could result in overloading the control plane of those nodes, resulting in poor convergence. Thus, it may be optimal to have an upper bound on the degree of nodes in the flooding topology. Again, the optimal trade-off between graph diameter, node degree, and convergence time, and topology computation time is for further study.

If the leader chooses to include a multi-node broadcast LAN segment as part of the flooding topology, all of the connectivity to that LAN segment should be included as well. Once updates are flooded onto the LAN, they will be received by every attached node.

4.3. Topologies on Complete Bipartite Graphs

Complete bipartite graph topologies have become popular for data center applications and are commonly called leaf-spine or spine-leaf topologies. In this section, we discuss some flooding topologies that are of particular interest in these networks.

4.3.1. A Minimal Flooding Topology

We define a Minimal Flooding Topology on a complete bipartite graph as one in which the topology is connected and each node has at least degree two. This is of interest because it guarantees that the flooding topology has no single points of failure.

In practice, this implies that every leaf node in the flooding topology will have a degree of two. As there are usually more leaves than spines, the degree of the spines will be higher, but the load on the individual spines can be evenly distributed.

This type of flooding topology is also of interest because it scales well. As the number of leaves increases, we can construct flooding topologies that perform well. Specifically, for n spines and m leaves, if m >= n(n/2-1), then there is a flooding topology that has a diameter of four.

4.3.2. Xia Topologies

We define a Xia Topology on a complete bipartite graph as one in which all spine nodes are bi-connected through leaves with degree two, but the remaining leaves all have degreee one and are evenly distributed across the spines.

Constructively, we can create a Xia topology by iterating through the spines. Each spine can be connected to the next spine by selecting any unused leaf. Since leaves are connected to all spines, all leaves will have a connection to both the first and second spine and we can therefore choose any leaf without loss of generality. Continuing this iteration across all of the spines, selecting a new leaf at each iteration, will result in a path that connects all spines. Adding one more leaf between the last and first spine will produce a cycle of n spines and n leaves.

At this point, m-n leaves remain unconnected. These can be distributed evenly across the remaining spines, connected by a single link.

Xia topologies represent a compromise that trades off increased risk and decreased performance for lower flooding amplification. Xia

topologies will have a larger diameter. For m spines, the diameter will be m + 2.

In a Xia topology, some leaves are singly connected. This represents a risk in that in some failures, convergence may be delayed. However, there may be some alternate behaviors that can be employed to mitigate these risks. If a leaf node sees that its single link on the flooding topology has failed, it can compensate by performing a database sychronization check with a different spine. Similarly, if a leaf determines that its connected spine on the flooding topology has failed, it can compensate by performing a database synchronization check with a different spine. In both of these cases, the synchronization check is intended to ameliorate any delays in link state propagation due to the fragmentation of the flooding topology.

The benefit of this topology is that flooding load is easily understood. Each node in the spine cycle will never receive an update more than twice. For n leaves and m spines, a spine never transmits more than m/n updates.

4.3.3. Optimization

If two systems have multiple links between them, only one of the links should be part of the flooding topology. Moreover, symmetric selection of the link to use for flooding is not required.

4.4. Encoding the Flooding Topology

There are a variety of ways that the flooding topology could be encoded efficiently. If the topology was only a cycle, a simple list of the nodes in the topology would suffice. However, this is insufficiently flexible as it would require a different encoding scheme as soon as a single additional link is added. In anticipation of richer flooding topologies, we recommend the advertisement of the full adjacency matrix of the flooding topology.

We can assume that all links are bidirectional, so we can represent each link with a single bit. The matrix can then be represented as the list of nodes, plus one bit per possible link. This results in N * (N -1)/2 possible bits for links. This can be further reduced by sparse matrix techniques.

5. Applicability

In a complete graph, this approach is appealing because it drastically decreases the flooding topology without the manual configuration of mesh groups. By controlling the diameter of the

flooding topology, as well as the maximum degree node in the flooding topology, convergence time goals can be met and the stability of the control plan can be assured.

Similarly, in a massively scaled data center, where there are many opportunities for redundant flooding, this mechanism ensures that flooding is redundant, with each leaf and spine well connected, while ensuring that no update need make too many hops and that no node shares an undue portion of the flooding effort.

6. Acknowledgements

The author would like to thank Zeqing (Fred) Xia, Naiming Shen, Adam Sweeney, and Olufemi Komolafe for their helpful comments.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

This document introduces no new security issues. Security of routing within a domain is already addressed as part of the routing protocols themselves. This document proposes no changes to those security architectures.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[Clos] Clos, C., "A Study of Non-Blocking Switching Networks", The Bell System Technical Journal Vol. 32(2), DOI 10.1002/j.1538-7305.1953.tb01433.x, March 1953, <http://dx.doi.org/10.1002/j.1538-7305.1953.tb01433.x>.

[ISO10589] International Organization for Standardization, "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)", ISO/IEC 10589:2002, Nov. 2002.

[Leiserson]
Leiserson, C., "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing", IEEE Transactions on Computers 34(10):892-901, 1985.

[RFC2973] Balay, R., Katz, D., and J. Parker, "IS-IS Mesh Groups", RFC 2973, DOI 10.17487/RFC2973, October 2000, <https://www.rfc-editor.org/info/rfc2973>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <https://www.rfc-editor.org/info/rfc5340>.

[RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <https://www.rfc-editor.org/info/rfc7938>.

Author's Address

Tony Li
Arista Networks
5453 Great America Parkway
Santa Clara, California 95054
USA

Email: tony.li@tony.li