

Policy flooding: reaching BGP-level scalability in data centers with link-state routing

Anthony Doeraene

anthony.doeraene@uclouvain.be
UCLouvain, Belgium

Cristel Pelsser

cristel.pelsser@uclouvain.be
UCLouvain, Belgium

Olivier Bonaventure

olivier.bonaventure@uclouvain.be
UCLouvain & WELRI, Belgium

Abstract

Large data center operators have replaced traditional link-state IGP with BGP, an interdomain protocol repurposed for an intradomain usage. BGP scales well and supports expressive routing policies, but does not expose the underlying topology, restricting the diversity of paths used in the network. A BGP extension, called Add-Path, can be used to improve the diversity of BGP paths, but limit the scalability of BGP. We present *policy flooding*, a flooding-reduction algorithm for link-state protocols that borrows BGP-style policies to suppress duplicate link-state announcements. Our evaluation shows that policy flooding allows reaching BGP-level scalability, making link-state protocols a viable choice for data centers without inheriting BGP's limitations.

CCS Concepts

• **Networks** → **Routing protocols**; *Network protocol design*.

Keywords

Link-state routing, BGP, Data center routing

ACM Reference Format:

Anthony Doeraene, Cristel Pelsser, and Olivier Bonaventure. 2026. Policy flooding: reaching BGP-level scalability in data centers with link-state routing. In *The 22nd International Conference on emerging Networking EXperiments and Technologies (CoNEXT Companion '26)*, December 07–10, 2026, Utrecht, Netherlands. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/3844665.3859461>

1 Introduction

BGP has been selected as the primary protocol to distribute routing information inside data center networks [1]. While link-state protocols are typically used for such intra-domain scenario, the usage of BGP for data center networks was motivated by the fact that the flooding process used by link-state protocols to distribute routing information is highly

inefficient in densely connected networks [1]. Nowadays, data center networks are frequently densely connected to improve the load balancing capabilities of traffic [8], harming the scalability of link-state protocols. BGP, in comparison to link-state routing, benefits from a more scalable information distribution process by leveraging *policies*. Using these policies, BGP can restrict the set of paths used for message propagation, reducing the amount of duplicate messages received by routers.

While BGP is effectively more scalable than link-state protocols in data centers networks, BGP has two major shortcomings. First, BGP can generate unpredictable message bursts, called churn [5], resulting in CPU spikes in the worst case scenarios. Second, BGP does not expose the underlying network topology to routers, reducing the diversity of available paths. The limited path diversity can lead to unbalanced traffic loads and slow convergence due to path exploration when facing failures [6]. While Add-Path, a BGP extension, can be used to expose some alternate paths, its usage reduces the overall scalability of BGP [11] as the number of BGP updates and RIB size grows linearly with the number of alternate paths. Another extension, BGP-LS, can be used to encode the topology in BGP updates, but delegates the path computation to a centralized controller [10], which introduces single points of failure.

In this paper, we propose a new flooding-reduction algorithm for link-state protocols, called *policy flooding*. Our algorithm leverages BGP-like policies in the flooding process to suppress the sending of Link-State Packets (LSP) on certain links based on the port they were received on. Using simulation to measure the message rate of LSPs and BGP updates, we show that policy flooding allows reaching BGP-level scalability in terms of mean message rate, while reducing the maximum message rate by a factor 5 compared to BGP. With this improved scalability, link-state protocols can now be used in data centers, offering better path diversity without needing a centralized controller. This work does not raise any ethical issues. We used AI for spell checking.

2 Policy aware LSP flooding

Policy flooding, presented in Algorithm 1, is a new flooding-reduction algorithm [3] based on BGP-like policies. Our algorithm assumes that each link vu connecting node v to node



This work is licensed under a Creative Commons Attribution 4.0 International License.

CoNEXT Companion '26, Utrecht, Netherlands

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-3006-1/2026/12

<https://doi.org/10.1145/3844665.3859461>

u has a *type* $t_{uv} \in T$ associated by the operator, where T is the set of all link types. Using link types, each node v of the network can then define a specific policy $P_v(t_{vu}, t_{vw})$. This policy dictates whether an LSP received by node v on a link with type t_{vu} should be sent on a link with type t_{vw} .

Let us consider a classical fat-tree topology [2]. In this topology, two types of links exist from a node's perspective: up and down-links. Routers can then filter messages by using a valley-free policy, as currently used in data centers [1]. This valley-free policy indicates that a message received on an up-link should only be sent on down-links, while a message received on a down-link should be sent on all links.

Whenever an LSP is received by a router, this router uses its policy to determine the set of neighbours the LSP should be sent to. However, there are two key differences between our algorithm and the classical flooding currently used by link-state protocols. First, policy flooding may prevent the exchange of LSPs between a pair of routers if no path respecting the per-hop policies exists between these routers. This means that routers may not receive the LSPs of all other routers, but an important property is that all paths respecting the policies can still be used even with this partial view. Second, an LSP with an identical sequence number may need to be propagated again to some neighbours. In the classical flooding algorithm, routers maintain for each router the highest sequence number observed to prevent the propagation of outdated or duplicated LSPs. In comparison, policy flooding may need to propagate an LSP with an identical sequence number to some neighbours, as the policy may have blocked its propagation depending on which neighbour the LSP was received from. An identical LSP is however never sent twice on the same link. This behaviour can typically be observed on some non fat-tree topologies with the valley-free routing policy. Consider an LSP that a node v has already received from an up-link. If v receives the same LSP from a down-link, it should be propagated on all up-links as no copy of this LSP has been sent on these links.

Each router should thus maintain in its LSDB, for each LSP, both the highest sequence number and the set of link types on which the LSP was already sent. This is illustrated in Algorithm 1 at line 4, where the LSP is sent to a neighbour if policies allow it and **if this LSP has not been sent yet on this link type**. Consequently, the usage of policy flooding thus requires maintaining $O(|T|)$ propagation information per router in the network, instead of $O(1)$ for classical flooding. For the valley-free policy, note that the impact of this state is very limited, as only two types of links are used.

3 Evaluation

We compare policy flooding with the classical flooding used currently by link-state protocols and with BGP in realistic

Algorithm 1: Policy flooding LSP reception function for node v

Input: lsp , the received LSP
Input: u , the neighbour from which lsp was received

```

1 begin
2    $T \leftarrow \emptyset$ ;
3   for  $w \in adj(v)$  do
4     /* Send LSP if not already sent on
       this link type and allowed */
5     if  $t_{vw} \notin sent[lsp] \wedge P_v(t_{vu}, t_{vw})$  then
6        $T \leftarrow T \cup \{t_{vw}\}$ ;
7        $send(w, lsp)$ ;
8     end
9   end
10   $sent[lsp] \leftarrow sent[lsp] \cup T$ ;
11 end
```

scenarios. In particular, we evaluate how BGP and link-state protocols react to link failure, which are frequent events [4] due to the size of data centers. This section details our experimental setup and the preliminary results we obtained.

Experimental setup. To compare our solution with classical flooding and BGP, we implemented two simulators: a generic link-state protocol simulator and a minimalist BGP simulator, with respectively ~ 800 and ~ 600 lines of Rust. While several BGP simulators have been proposed in the literature, e.g. [7], we decided to implement our own BGP simulator with a restricted set of features to allow the simulation of large topologies in reasonable time. To ensure that our simulator was reflecting realistic BGP behaviour, we validated it with a state-of-the-art simulator: C-BGP [7], extended to count the number of messages. To study how link-state routing and BGP react to failures, we randomly generate link failures and restorations in our simulators. We generate 500 failures using an exponential distribution with a mean inter-failure time of 1 week and a repair time of 5min, modelling realistic failure rates observed in data centers [4]. Experiments are repeated over 5 runs with different failures.

In these experiments, we compare the *control plane load* of different solutions, which can be seen as the amount of stress put on the control plane. In practice, we define this load as the reception rate of LSPs or BGP updates. We do not consider messages used for maintaining sessions between nodes (i.e. OSPF/IS-IS Hellos, BGP Keepalives), as these messages are not only very small, but we can also expect similar configured message rates for both protocols. We also assume that LSP refreshes are suppressed [9].

Our experiments are carried on a fat-tree topology [2] consisting of 320 nodes, with each device having 16 ports.

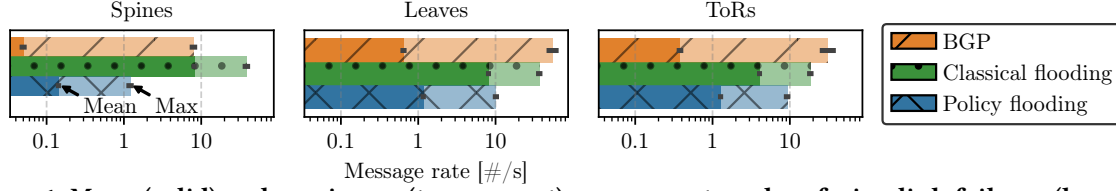


Figure 1: Mean (solid) and maximum (transparent) message rates when facing link failures (log scale).

Evaluation results Figure 1 presents our results. When facing link failures, BGP has a better mean message rate (i.e., number of messages per second) at the spines and leaf switches than classical flooding and policy flooding. By integrating shortest-path computation in its propagation process, BGP can typically suppress some updates if the best route has not changed. In comparison, link-state routing must inform all reachable nodes of the failure, to ensure a consistent view of the topology. Using this update suppression mechanism, BGP can effectively reduce the amount of messages at the spines/leaves compared to both classical and policy flooding, even if the difference is quite small for policy flooding.

However, the BGP decision process can also affect negatively the scalability of the protocol. If a link was used for many prefixes, BGP will send a burst of messages to announce for each affected prefix either a new route to reach this prefix, or a route withdrawal indicating that the prefix is now unreachable. This traffic burst not only wastes bandwidth, but also puts a burden on the control plane, which needs to handle each message appropriately. In comparison, link-state routing is much more stable, as only two LSPs are sent by adjacent nodes for each link failure/restoration, regardless of the number of prefixes using this link. This is illustrated in figure 1, where the maximum message rate seen when using BGP is several order of magnitude higher than the mean message rate. Nodes face a much more stable rate with link-state routing, represented by a 5x maximum rate reduction compared to BGP.

When comparing policy flooding with BGP, there is thus a trade-off between the mean and maximum message rates. BGP benefits from a better mean message rate, at the cost of potential message burst, increasing drastically the maximum message rate. Link-state routing is more stable than BGP in its message rates, at the cost of having a higher mean rate.

4 Conclusion

In this paper, we presented our policy flooding algorithm, a new flooding reduction algorithm leveraging BGP-like policies. Our preliminary results show that, when combined with valley-free routing, policy flooding has a slightly higher mean message rate than BGP, but our algorithm reduces by a factor 5 the maximum message rate, showing that link-state routing can be a viable choice for data center routing. As future work, instead of only considering message counts or rates, we would like to explore in more details how policy

flooding, classical flooding or BGP impacts the overall control plane scalability by measuring other routing metrics, such as CPU load, memory usage or convergence time.

Acknowledgments

Anthony Doeraene is a research fellow of the Fonds de la Recherche Scientifique - FNRS. This publication is supported by the Walloon Region as part of the FNRS WEL-T strategic axis.

References

- [1] Anubhavnidhi Abhashkumar, Kausik Subramanian, Alexey Andreyev, Hyojeong Kim, Nanda Kishore Salem, Jingyi Yang, Petr Lapukhov, Aditya Akella, and Hongyi Zeng. 2021. Running BGP in data centers at scale. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 65–81.
- [2] L Brian, M Aman, and T Niharika. 2016. A survey and evaluation of data center network topologies. (2016).
- [3] Sarah Chen and Tony Li. 2026. An Algorithm for Computing Dynamic Flooding Topologies. Internet-Draft draft-ietf-lsr-dynamic-flooding-algorithm-05. Work in Progress. Internet Engineering Task Force, (Aug. 2026). 13 pp. <https://datatracker.ietf.org/doc/draft-ietf-lsr-dynamic-flooding-algorithm/05/>.
- [4] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. 2011. Understanding network failures in data centers: measurement, analysis, and implications. en. In *Proceedings of the ACM SIGCOMM 2011 conference*. ACM, Toronto Ontario Canada, (Aug. 2011), 350–361. ISBN: 978-1-4503-0797-0. doi:10.1145/2018436.2018477.
- [5] T.G. Griffin and B.J. Premore. 2001. An experimental analysis of BGP convergence time. In *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*. IEEE Comput. Soc, Riverside, CA, USA, 53–61. ISBN: 978-0-7695-1429-1. doi:10.1109/ICNP.2001.992760.
- [6] Timothy G Griffin and Brian J Premore. 2001. An experimental analysis of BGP convergence time. In *Proceedings Ninth International Conference on Network Protocols. ICNP 2001*. IEEE, 53–61.
- [7] B. Quoitin and S. Uhlig. 2005. Modeling the routing of an autonomous system with C-BGP. en. *IEEE Network*, 19, 6, (Nov. 2005), 12–19. doi:10.1109/MNET.2005.1541716.
- [8] Costin Raiciu, Sebastian Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. 2011. Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM computer communication review*, 41, 4, 266–277.
- [9] P. Pillay-Esnault. 2005. OSPF Refresh and Flooding Reduction in Stable Topologies. RFC 4136 (Informational). RFC. Fremont, CA, USA: RFC Editor, (July 2005). doi:10.17487/RFC4136.
- [10] K. Talaulikar (Ed.) 2023. Distribution of Link-State and Traffic Engineering Information Using BGP. RFC 9552 (Proposed Standard). RFC. Fremont, CA, USA: RFC Editor, (Dec. 2023). doi:10.17487/RFC9552.
- [11] Virginie Van den Schrieck, Pierre Francois, and Olivier Bonaventure. 2010. BGP add-paths: the scaling/performance tradeoffs. *IEEE Journal on Selected Areas in Communications*, 28, 8, 1299–1307.