

A Load Balancing of Hop-By-Hop Adaptive Link State Optimal Routing

Mrs. D. Sudha Rani, Mrs. B. Vijaya Sri
Department of Computer Science and Engineering
R.K College of Engineering,
Vijayawada, India

sudha.sathwika@gmail.com, vijaya.b81@gmail.com
DOI:10.53414/UIJES:2024.43.529

Abstract – The goal of this work is to do away with the trade-off that exists between routing's optimality and implementation convenience. Thus, I put forth my hop-by-hop routing scheme, which iteratively converges to the best routing assignment while maintaining the simplicity of link-state, hop-by-hop protocols. This is the first ideal link-state hop-by-hop routing solution that we are aware of. Not surprisingly, creating such a solution presents a number of difficulties. For convenience of presentation, we define the following key words before delving into them. Moreover, our method is adaptive: when a quasi-static network changes, it will automatically converge to the new optimal routing assignment. Along with exploring more facets of the solution and outlining a proof-of-concept implementation report numerical and experimental evaluations to validate our theoretical predictions.

Keywords – IP networks, load balancing, network management, optimal routing.

I. INTRODUCTION

Since the early 1970s with the advent of ARPANET, the predecessor of the Internet. Till today it is find that the different optimal routing algorithms developed over the last 40 years are seldom implemented. Routing term suggests the process of selecting best paths in a network. That means there are n numbers of ways to send a data packet from sender node to receiver node, the process through which we find the best path in terms of marginal cost and low traffic so that the cost of sending packet in network will be optimal. In early days the term routing was also used as means forward network traffic among networks. This has been fundamental research and practical interest from the early 1970s with the era of ARPANET, the predecessor of the Internet. Till today it is find that the different optimal routing algorithms developed over the last 40 years are seldom implemented. So, hop-by-hop packet forwarding supporting distributed link-state routing protocols like OSPF/IS-IS are the dominant intra-domain routing solutions on the Internet. The reason behind the widespread distribution and adoption of link-state, hop-by-hop algorithms has been their simplicity—here in algorithm main idea suggest to centrally assign weights to links based on input traffic statistics, then flood the link weights through the network, and then compute the shortest paths from the link weights and locally forward packets to destinations along these computed shortest paths. Because of rapid growth in our communication networks in size and complexity, this simplicity has helped OSPF to expand the boundaries of optimal routing techniques that are harder to implement. Ultimately this results into lost performance of this tradeoff [1,2,6]. Generally poor resource utilization results from OSPF. Network administrators are forced to overprovision their networks to handle peak traffic. Subsequently this resulted into on average, most network links run at just 30%–40% utilization. The problem become critical, when it realized that there has to be no way around this tradeoff

The driving force behind the widespread adoption of link-state, hop-by-hop algorithms has been their simplicity—the main idea is to centrally assign weights to links based on input traffic statistics, flood the link weights through the network, and then locally forward packets to destinations along shortest paths computed from the link weights. As our communication networks have grown rapidly in size and complexity, this simplicity has helped OSPF eclipse extant optimal routing techniques that are harder to implement.

However, the obvious tradeoff has been lost performance. For instance, due to the poor resource utilization resulting from OSPF, network administrators are forced to overprovision their networks to handle peak traffic. As a result, on average, most network links run at just 30%–40% utilization. To make matters worse, there seems to be no way around this tradeoff. In fact, given the offered traffic, finding the optimal link weights for OSPF, if they exist, has been shown to be NP-hard [4]. Furthermore, it is possible for even the best weight setting to lead to routing that deviates significantly from the optimal routing assignment [4].

1.0Hop-by-hop: Each router, based on the destination address, controls only the next hop that a packet takes.

1.1Adaptive: The algorithm does not require the traffic demand matrix as an explicit input in order to compute link weights.

Specifically, the algorithm seamlessly recognizes and adapts to changes in the network, both topology changes and traffic variations, as inferred from the network states like link flow rates.

1.2 Link-state: Each router receives the state of all the network's links through periodically flooded link-state updates and makes routing decisions based on the link states.

1.3 Optimal: The routing algorithm minimizes some cost function (e.g., minimize total delay) determined by the network operator. The problem of guiding network traffic through routing to minimize a given global cost function is called traffic engineering (TE).

The first design challenge stems from coordinating routers only using link states. This means that no router is aware of all the individual communicating pairs in the network or their traffic requirements. However, they still have to act independently such that the network cost is minimized. This is a very real restriction in any large dynamic network like the Internet, where it is not possible to obtain information about each communicating pair. If the link-state requirement is set aside, optimal distance-vector routing protocols have already been developed [2]. The idea there is to iteratively converge to the optimal routing assignment by sharing estimates of average distances to destinations among neighbors. However, distance-vector protocols have not caught on for intra-domain routing because of scalability issues due to their slow convergence and robustness issues like vulnerability to a single rogue router taking down the network as in the "Internet Routing Black Hole" incident of 1997 [5].

The hop-by-hop forwarding requirement presents the next challenge. As a result, a router cannot determine the entire path that traffic originating at it takes to its destination. Without this requirement, a projected gradient approach [6] can be used to yield optimal iterative link-state algorithms that can be implemented with source routing, where the path a packet takes through the network is encoded in its entirety at the source. However, the need for source routing means that these techniques are not practical given the size of modern networks.

Another challenge arises because the optimal routing assignment changes with the input traffic and the network. There are two aspects to this problem. The first aspect is that the algorithm needs sufficient time between network and traffic changes to calculate and assign optimal routes. This requirement is typically captured by the quasi-static model of routing problems described by Gallager [2]. The second aspect is that the algorithm should smoothly adapt the routes to changes when they do occur. Thus, ideally, the algorithm should avoid global inputs that require additional computation when performing routing updates. However, the algorithm also needs some way to track the network state to compute efficient routes. Link rates fill this gap because they are widely available and easily accessible in modern networks. The first aspect is modeled by studying a static network with static input traffic in between changes in the network. If the second stipulation is set aside, recently, significant progress was made in this direction with PEFT, a link-state protocol with hop-by-hop forwarding based on centralized weight calculations [7]. However, since the link weights are calculated in a centralized manner with the traffic matrix as an explicit input, PEFT is not adaptive. Nor does it always guarantee optimality as claimed in the paper.

II. PROBLEM FORMULATION

Under the quasi-static model, the traffic engineering problem can be cast as a Multi-Commodity Flow (MCF) problem in between topology and input traffic changes. We model the network as a directed graph $G = (V, E)$ with node/router set and edge/ link set with link capacities $C_{u,v}, \forall (u,v) \in E$. The rate required for communication from s to t is represented by $D(s,t)$. The commodities are defined in terms of their final destination t . We use f_{uv}^t to represent the flow on link (u,v) corresponding to commodity t and f_{uv}^t for the total flow on link (u,v) . The network cost function, Φ , is typically selected to be a convex function of the link rate vector $f = \{ f_{u,v} \}, \forall (u,v) \in E$. Using this notation, the MCF problem can be stated as

$$\begin{aligned} & \min_{f_{u,v}^t} \Phi(f) \\ & \text{s.t.} \quad \sum_{v:(s,v) \in E} f_{s,v}^t - \sum_{u:(u,s) \in E} f_{u,s}^t = D(s,t) \quad \forall s \neq t \\ & \quad f_{u,v} = \sum_{t \in V} f_{u,v}^t \leq c_{u,v} \quad \forall (u,v) \\ & \quad f_{u,v}^t \geq 0. \end{aligned}$$

A fact about MCF is that its optimal solution generally results in multipath routing instead of single-path routing [1]. However, finding the right split ratios for each router for each commodity is a difficult task. Our starting point is to merge the link-state feature of the source-routing protocols with the hop-by-hop forwarding feature of the distance-vector schemes. Another characteristic that we borrow is the iterative nature of these algorithms. Here, each iteration is defined by the flooding of existing link states through the network followed by every router updating its split ratios, which modifies the link states for the next iteration. In what follows, we measure time in units of iterations. With this idea in mind, in the time between network changes when the topology and the input traffic is static, we do the following.

Iteratively adjust each router's split ratios and move traffic from one outgoing link to another. This only controls the next hop on a packet's path leading to hop-by-hop routing. If instead we controlled path rates, we would get source routing.

Increase the split ratio to the link that is part of the shortest path at each iteration even though the average price via the next-hop router may not be the lowest. If instead we forwarded traffic via the next-hop router with the lowest average price, we get Gallager's approach, which is a distance vector solution.

Adapt split ratios dynamically and incrementally by decreasing along links that belong to non-shortest paths while increasing along the link that is part of the shortest path at every router. If instead split ratios are set to be positive instantaneously only to the links leading to shortest paths, then we get OSPF with weights, Wu,v.

III. GENERAL SOLUTION

We begin by defining n_u^t , the *branch cardinality*, as the product of the number of branches encountered in traversing the shortest path tree rooted at t from t to u . It makes sure that routers on the tree that are farther away from the destination shift traffic to the shortest path more conservatively than routers that are closer to the destination. At every iteration due to link-state flooding, each node u has the link-state information to run Dijkstra's algorithm to compute the shortest path tree to destination t . Here, additional care is required because every node has to locally arrive at the same shortest path tree to ensure that the algorithm proceeds as expected. Therefore, at any stage, while running Dijkstra's algorithm locally, if there is ambiguity as to which node should be added next, tie-breaking based on node index is used. In other words, if at any iteration there are multiple shortest paths to choose from, tie-breaking is used to ensure that all routers arrive at the same shortest path tree. The calculation n_u^t of proceeds as shown in Algorithm1.

Algorithm 1: Algorithm to calculate $\eta_u^t \{w_e, \forall e \in E\}$

- 1: Compute shortest path tree for destination t using Dijkstra's algorithm with tie-breaking based on node index.
 - 2: Traverse the tree from t to u .
 - 3: Initialize $\eta_u^t \leftarrow 1$.
 - 4: At every junction do $\eta_u^t \leftarrow \eta_u^t b$ where b is the number of branches from that junction.
-

IV. RELATED WORK

Over the years, due to its importance, traffic engineering has attracted a lot of research attention. We provide a brief overview of major related results from different communities such as control, optimization, and networking. Broadly, the existing work can be divided into OSPF-TE, MPLS-TE, traffic demand agnostic/ oblivious routing protocol design, and optimal routing algorithms.

The work on OSPF [4], [8], [9] has concentrated on using good heuristics to improve the centralized link weight calculations. Although these techniques have been shown to improve the algorithm's performance significantly by finding better weight settings, the results are still far from optimal.

Typically, these and other centralized traffic engineering techniques also require reliable estimates or measurements of the input traffic statistics in the form of a traffic matrix. While excellent work has been done in traffic matrix estimation from link loads, even the best results have errors on the order of 20% [10], which can lead to bad traffic engineering. Another approach is to directly measure the traffic to every destination at every router. While it is possible to globally aggregate the measurements into a traffic matrix that can be fed to a traffic engineering algorithm, it is more straightforward to use local measurements locally. Also, usually it is smoother and quicker to respond to changes locally when they do occur. Thus, we are advocating a shift to relying directly on link loads and local traffic measurements instead of computing a traffic matrix for traffic engineering.

A good way to avoid traffic matrices and a popular way to implement traffic engineering today is MPLS-TE [11], [12]. The idea is to compute end-to-end tunnels for traffic demands with the available network bandwidth being assigned to new traffic demands using techniques like Constrained Shortest Path First. However, here, the performance gained over OSPF comes at the cost of establishing multiple end-to-end virtual circuits. Moreover, as the traffic changes, the end-to-end virtual circuits that were established for a particular traffic pattern become less useful, and performance degrades.

Oblivious routing has also been proposed as a way around using traffic matrices for traffic engineering. The idea is to come up with a routing assignment that performs well irrespective of the traffic demand by comparing the “oblivious performance ratio” of the routing, i.e., the worst-case performance of the routing for a given network over all possible demands. Breakthrough work in this area includes papers by Applegate and Cohen [13] that developed a linear programming method to determine the best oblivious routing solution for the special case of minimizing maximum channel utilization and Kodialam *et al.* [14] that focused on maximizing throughput for the special case of two-phase routing. However, oblivious routing solutions do not adapt well to changes in the network.

V. CONCLUSION

In this paper, I developed the first link-state, hop-by-hop routing algorithm that optimally solves the traffic engineering problem for intra-domain routing on the Internet. Furthermore, we showed that based on feedback from the link-state updates, the protocol automatically adapts to input traffic and topology changes by adjusting router split ratios. We also provided guidelines on implementing my project by translating the theoretical model to a discrete implementation for numerical evaluations and then to a physical testbed built on NetFPGA boards. Importantly, although they did not satisfy the theoretical assumptions about continuous split ratio updates and synchronization between the routers, the numerical and experimental evaluations backed up our theoretical predictions about the performance and adaptivity of this project. In terms of future directions, there are still interesting areas to be explored. For instance, the convergence rate of the algorithm needs to be analyzed. Another direction involves developing the theory behind the performance of algorithm in the absence of synchronous link-state updates and executions.

REFERENCES

- [1] M. Wang, C. W. Tan, W. Xu, and A. Tang, “Cost of not splitting in routing: characterization and estimation,” *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1849–1859, Dec. 2011.
- [2] R. Gallager, “A minimum delay routing algorithm using distributed computation,” *IEEE Trans. Commun.*, vol. COM-25, no. 1, pp. 73–85, Jan. 1977.
- [3] L. Fratta, M. Gerla, and L. Kleinrock, “The flow deviation method: An approach to store-and-forward communication network design,” *Networks*, vol. 3, no. 2, pp. 97–133, 1973.
- [5] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 5/E. New York, NY, USA: Addison-Wesley, 2010.
- [6] D. Bertsekas and E. Gafni, “Projected newton methods and optimization of multicommodity flows,” *IEEE Trans. Autom. Control*, vol. AC-28, no. 12, pp. 1090–1096, Dec. 1983.
- [7] D. Xu, M. Chiang, and J. Rexford, “Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering,” *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1717–1730, Dec. 2011.
- [8] A. Sridharan, R. Guerin, and C. Diot, “Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks,” *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 234–247, Apr. 2005.
- [9] S. Srivastava, G. Agrawal, M. Pioro, and D. Medhi, “Determining link weight system under various objectives for OSPF networks using a lagrangian relaxation-based approach,” *IEEE Trans. Netw. Service Manag.*, vol. 2, no. 1, pp. 9–18, Nov. 2005.
- [10] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, “Fast accurate computation of large-scale IP traffic matrices from link loads,” in *Proc. ACM SIGMETRICS*, New York, NY, USA, 2003, pp. 206–217.
- [11] D. Awduche, “MPLS and traffic engineering in IP networks,” *IEEE Commun. Mag.*, vol. 37, no. 12, pp. 42–47, Dec. 1999.
- [12] A. Elwalid, C. Jin, S. Low, and I. Widjaja, “MATE: MPLS adaptive traffic engineering,” in *Proc. 20th Annu. IEEE INFOCOM*, 2001, vol. 2001, pp. 1–11.
- [13] C. E. Agnew, “On quadratic adaptive routing algorithms,” *Commun. ACM*, vol. 19, no. 1, pp. 18–22, Jan. 1976.
- [14] M. Kodialam, T. V. Lakshman, J. Orlin, and S. Sengupta, “Oblivious routing of highly variable traffic in service overlays and IP backbones,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 459–472, Apr. 2019.