International Journal of Science Engineering and Advance Technology

Hop-by-Hop Adaptive linking A Novel Approach for Finest routing

Sunkara Mounika [1], Md Amanatulla [2]
[1] M.Tech (CSE), Nimra Institute of Science and Technology, A.P., India.
[2] Assistant Professor, Dept. of Computer Science & Engineering, Nimra Institute of Science and Technology, A.P., India.

***Abstract* — Using Hop-by-Hop Adaptive linking for achieving finest routing is an unprecedented approach. And it is the first link-state routing solution carrying traffic through packet-switched networks. At each node, for every other node, the algorithm independently and iteratively updates the fraction of traffic destined to that leaves on each of its outgoing links. At each iteration, the updates are calculated based on the shortest path to each destination as determined by the marginal costs of the network's links. The marginal link costs used to find the shortest paths are in turn obtained from link-state updates that are flooded through the network after each iteration. For stationary input traffic, we prove that our project converges to the routing assignment that minimizes the cost of the network. Furthermore, I observe that our technique is adaptive, automatically converging to the new optimal routing assignment for quasi-static network changes. I also report numerical and experimental evaluations to confirm our theoretical predictions, explore additional aspects of the solution, and outline a proof-of-concept implementation of proposal.**

***Keywords* — IP networks, load balancing, network management, optimal routing.**

I. INTRODUCTION

Ever since ARPANET [3], the predecessor of the Internet, has been introduced, the concept of Optimal routing i.e., finding routing assignments that minimize the cost of sending traffic through packet-switched networks, has been of fundamental research and practical interest. Yet today, we find that the different optimal routing algorithms developed over the last 40 years are seldom implemented. Instead, distributed link-state routing protocols like OSPF/IS-IS that support hop-by-hop packet forwarding are the dominant intra-domain routing solutions on the Internet.

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].

My goal in this paper is to eliminate this tradeoff between optimality and ease of implementation in routing. So, I proposed this hop-by-hop routing solution, a routing solution that retains the simplicity of link-state, hop-by-hop protocols while iteratively converging to the optimal routing assignment. To the best of our knowledge, this is the first optimal link-state hop-by-hop routing solution. Not surprisingly, there are multiple challenges to overcome when designing such a solution. Before getting into them, we define the following important recurring terms for ease of exposition.

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

Adaptive: 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.

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.

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.Wemodel the network as a directed graph G = (V,E) with node/router set and edge/ link set with link capacities $C_{u,v}$, ¥(u,v) € 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 $f^{t}_{u,v}$ to represent the flow on link (u,v) corresponding to commodity t and $f^{t}_{u,v}$ 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 $f = \{ f_{u,v} \}$,¥(u,v) € E. Using this notation, the MCF problem can be stated as

$$\min_{f^t_{u,v}} \Phi(f)$$

$$\text{s.t.} \sum_{v:(s,v)\in\mathbb{E}} f^t_{s,v} - \sum_{u:(u,s)\in\mathbb{E}} f^t_{u,s} = D(s,t) \qquad \forall s \neq t$$

$$f_{u,v} = \sum_{t\in\mathbb{V}} f^t_{u,v} \leq c_{u,v} \qquad \forall (u,v)$$

$$f^t_{u,v} \geq 0.$$

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, $w_{u,v}$

III. Special Cases

In order to develop an intuitive understanding of why our solution takes the form that it does, it is helpful to consider a few concrete special cases first. These four cases, each of which clearly highlights the reason for including a particular factor in our solution, progressively lead us to the final algorithm. In each example, our algorithm design will exploit the fact that the KKT optimality conditions [15] of the MCF problem require that at the optimal solution the traffic rate is positive only along paths with the lowest price. The overall idea behind these examples is to design an algorithm that reduces the network cost at each iteration by moving to a routing assignment that satisfies this condition. In Section V, we will extend these ideas and show that the final algorithm that iteratively reduces the network cost will also always lead to the optimal routing assignment.

Finding the Right Split Dynamically

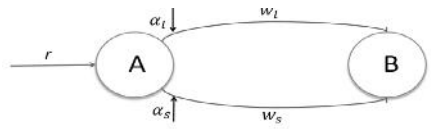
First, let us consider a very simple example illustrated in Fig. 1(a). Here, there is traffic demand of rate with the choice of two links, *l* and *s*, to go from *A* to *B*. Assuming initially $w_l > w_s$, a simple strategy to reach optimality will be to dynamically shift traffic at some rate $\delta > 0$ from the more expensive link to the cheaper link till the prices of the two links become the same. At node *A*, this would be equivalent to

α_l decreasing α_s and increasing at rate δ / r.

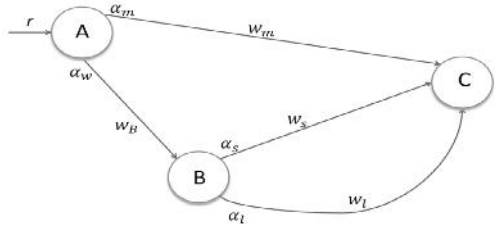
There are two ways to interpret and generalize the intuition gained from this scenario. Both give the same solution for this very simple example, but in general will lead to different dynamics (see Fig. 2) and possibly different split ratios. One interpretation, which underpins the distance-vector algorithms, is that the router should shift traffic away from neighbors with higher average price to the neighbor with the lowest average price. A different interpretation, which is the basis of our protocol, is that the router should shift traffic from links along more expensive paths to the link along the path with the lowest price. Mathematically, we reach the following update rule for the split ratios:

$$\dot{\alpha}^t_{u,v} = -\frac{\delta}{r^t_u}$$

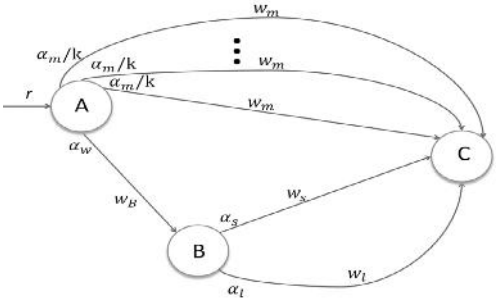
where $(u,v) \in E$ but is not on the shortest path from to.



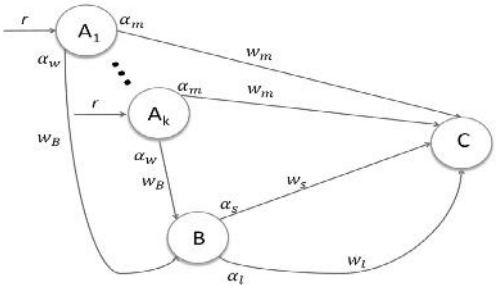
(a)



(b)



(c)



(d)

Figure 1 Four illustrative examples

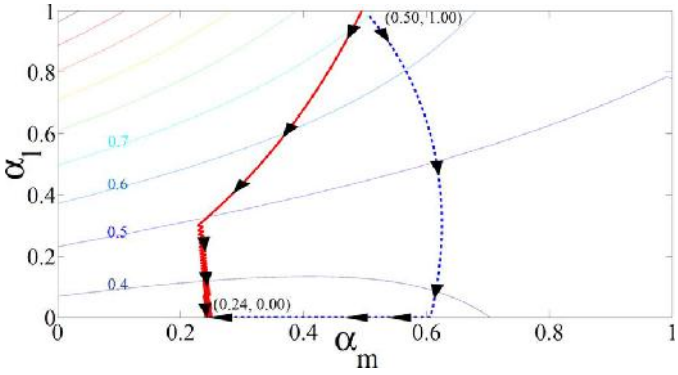


Figure 2 Trajectories taken by Gallager's algorithm

IV. General Solution

We begin by defining n^t_u, 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^t_u of proceeds as shown in Algorithm1.

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

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 $\eta^t_u \leftarrow 1$.
4: At every junction do $\eta^t_u \leftarrow \eta^t_u b$ where b is the number of branches from that junction.

V. 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

VI. 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 adaptively 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/ACMTrans.
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.

[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. 2009.

[15] S. Boyd and L. Vandenberghe, Convex Optimization. NewYork,NY, USA: Cambridge Univ. Press, 2004.