International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

OTE: Optimal Traffic Engineering using Hop-By-Hop Adaptive Link-State Routing

Sumayya Samreen & Dr. Asma Parveen
[1]M. Tech Student KBNCE, Kalaburagi
[2]CSE Dept., KBNCE, Kalaburagi.

Abstract

Link State routing protocols do not view networks in terms of adjacent routers and hop counts, but they build a comprehensive view of the overall network which fully describes the all possible routes along with their costs. Using the SPF (Shortest Path First) algorithm, the router creates a "topological database" which is a hierarchy reflecting the network routers it knows about. It then puts it's self on the top of this hierarchy, and has a complete picture from its own perspective. Open Shortest Path First (OSPF) and Intermediate System-Intermediate System (IS-IS), split traffic evenly over shortest paths based on link weights. However, optimizing the link weights for OSPF/IS-IS to the offered traffic is a well-known NP-hard problem, and even the best setting of the weights can deviate significantly from an optimal distribution of the traffic. In this paper, we propose a new link-state routing protocol, PEFT, that splits traffic over multiple paths with an exponential penalty on longer paths.

Keywords

OTE, Link-State Routing.

1. Introduction

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 since the early 1970s with the advent of ARPANET, the predecessor of the Internet. 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 intradomain 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 long 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. Furthermore, it is possible for even the best weight setting to lead to routing that deviates significantly from the optimal routing assignment.

Our goal in this paper is to eliminate this tradeoff between optimality and ease of implementation in routing. The result is Optimal Traffic Engineering (OTE), 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. 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 intradomain 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. 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 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 Gallagher.

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. However, since the link weights are calculated in a centralized manner with the traffic matrix asan explicit input, PEFT is not adaptive. Nor does it always guarantee optimality as claimed in the paper.

2. System Analysis

Existing System:

The information routers require to build their databases is provided in the form of Link State advertisement packets (LSAP). Routers do not advertise their entire routing tables, instead each router advertises only its information regarding immediately adjacent routers.

Link State protocols in comparison to Distance Vector protocols have:

- Big memory requirements
- Shortest path computations require many CPU circles
- If network is stable little bandwidth is used; react quickly to topology changes
- Announcements cannot be "filtered". All items in the database must be sent to neighbors
- All neighbors must be trusted
- Authentication mechanisms can be used to avoid undesired adjacencies
- No split horizon techniques are possible.

Proposed System:

Our goal in this paper is to eliminate this tradeoff between optimality and ease of implementation in routing. The result is Optimal Traffic Engineering (OTE), 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:

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

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).

Advantage:

1) Dampen update frequency
2) Target link-state updates to multicast
3) Use link-state area hierarchy for topology
4) Exchange route summaries at area borders
5) Use Time-stamps Update numbering & counters
6) Manage partitions using a area hierarchy

Future Enhancement:

Link State protocols work more efficiently, problem can arise. Usually problems occur cause of changes in the network topology (links go up-down), and all routers don't get updated immediately cause they might be on different line speeds, there for, routers connected via a fast link will receive these changes faster than the others on a slower link. 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 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.

3. System Architecture

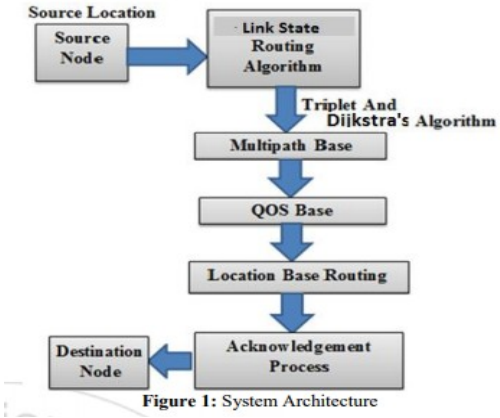

Figure 1: System Architecture

4. System Requirements

Hardware Requirements:

- System : Pentium IV 3.5 GHz
- Hard Disk : 40 GB
- Monitor : 14" Colour Monitor
- Mouse : Optical Mouse
- Ram : 1 GB.

Software Requirements:

- Operating system : Windows XP or Windows7, Windows 8.
- Coding Language : Java–AWT, Swings, Networking
- Data Base : My Sql / MS Access.
- Documentation : MS Office
- IDE : Eclipse Galileo
- Development Kit : JDK 1.6.

5. Implementation

- **Service Provider:**

  In this module, the service provider initially calculates shortest path from source to destination (service provider to end user). Later service provider browses the file and sends to the particular end users based on shortest path distance via router.

- **Router:**

  In this module, the router randomly generates the path cost between two nodes, and file will sends to particular end users. While sending the router also sends possible path details and recent routing path details to the Optimal Router. And it can

also do some operations like assign path cost, view path cost & exit.

- **Optimal Router :**

  In this module, the optimal router can stores the recent routing path details and possible routing path details those are provided by router. And it can also do some operations like view recent routing path details, view possible routing path details.

- **Remote User (End User):**

  In this module, there is n number of end users are there (A, B, C, D…). The end users receive the file by without changing the File Contents. Users may receive particular data files within the network only.

6. Conclusion

In this paper, we developed OTE, the first link-state, hop-by-hop routing algorithm that optimally solves the traffic engineering problem for intradomain 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 OTE by translating the theoretical model to a discrete implementation for numerical evaluations and then to a physical test bed 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 OTE. 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] N. Michael, A. Tang, and D. Xu, “Optimal link-state hop-by-hop routing,” in *Proc. IEEE ICNP*, 2013, pp. 1–10.
[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.
[4] B. Fortz and M. Thorup, "Increasing internet capacity using local nuity and invariance in hybrid automata," in Proc. 40th IEEE Decision search," Comput. Optim. Appl., vol. 29, no. 1, pp. 13_48, Oct. 2004. Control, 2001, vol. 1, pp. 340_345.
[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. 3, pp. 1300_1309.