Multi-constraint QoS Disjoint Multipath Routing in SDN

Manan Doshi, Aayush Kamdar and Krishna Kansara

Abstract Efficient path computation that sustains varying quality of service requirements is a key networking concern. Even though modern networks turned to multipath routing schemes as a first step in this path, existing solutions still resulted in sub-flows being directed to the same paths. Moreover, maintaining the quality of service subject to multiple criterions while selecting the paths and handling connection requests dynamically has proved to be challenging tasks. Addressing all these issues requires a centralized, real-time- and fine-grained control of the network facilitated by Software Defined Networks (SDN) that have emerged as a revolutionary networking paradigm. In this paper, we deal with the former issue by computing k-max min disjoint paths and for the latter we use an analytic hierarchy process. The proposed solution combines the two approaches for deployment in an SDN environment.

Keywords SDN · QoS routing · Analytic hierarchy process · Disjoint multipath routing · Multi-constraints

1 Introduction

Today's network research is directed towards achieving a single shared physical network supporting different heterogeneous applications with distinct traffic characteristics and QoS requirements. QoS routing considers key networking parameters such as bandwidth, cost, delay, energy consumption, etc., in the process of path computation [1]. Achieving network service quality that adapts to varying user requirements is a primary networking concerns [2, 3].

M. Doshi (✉) · A. Kamdar · K. Kansara
Shri Bhagubhai Mafatlal Polytechnic, Mumbai, India
e-mail: manandoshi1607@gmail.com

A. Kamdar
e-mail: ashbkamdar@gmail.com

K. Kansara
e-mail: kansarakrishna@rediffmail.com

B. Iyer et al. (eds.), *Computing, Communication and Signal Processing*,
Advances in Intelligent Systems and Computing 810,
https://doi.org/10.1007/978-981-13-1513-8_40

Compared to traditional routing schemes, multipath routing reduces congestion in the network by shifting traffic to alternate paths thereby improving network resource utilization. This load equalization is considered to be effective in the improvement of network reliability and reducing energy consumption. A common solution to split flows across multiple paths is ECMP protocol. However, sub flows may be directed to the same paths, even with ECMP enabled [4, 5]. This problem is addressed by obtaining multiple paths with a certain level of disjoint-ness. Disjoint multipath routing offers several added advantages such as increased reliability, resilience to failure, reduced congestion and higher scalability as packets are forwarded through distinct paths [6].

All these requirements call for higher level management of the network for optimization of user-centric and programmable paths. This administration needs to adopt a software-oriented approach [6–8]. As a new and emerging network technology, software-defined networks provide a centralized control of the network by separation of data and control planes. The programmable control plane creates a dynamic, open and controllable network environment. Thus, it serves as the basis for dynamic path planning subject to multiple constraints that meet the QoS requirements. This makes load balancing of the core network more efficient.

The structure of this paper is as follows: Sect. 2 gives an overview of related work and our contributions. Section 3 describes the analytic hierarchy process for meeting QoS requirements. In Sect. 4 we explain the procedure for obtaining k-max min disjoint QoS paths. In Sect. 5 we present the benefits of SDN integration and finally Sect. 6 concludes the paper.

2 Literature Review

Achieving a common solution that meets QoS demands and as well provides efficient path optimization has become the focus of current research accelerated by the growth of SDN. Many solutions have been proposed considering different aspects of this problem. The work in [1] addresses two main issues associated with QoS routing in end to end communication, i.e., which links to develop to meet certain connectivity requirements and selection of paths that meet QoS demands. In [3], the authors use analytic hierarchy process to obtain a single value for multiple QoS requirements and then apply a heuristic procedure for obtaining optimized communication paths and backup paths. Also, solutions have been proposed to solve the QoS problem by using variations of routing metrics (as in [9]). The work in [9] focuses on isotonicity, a key property of algebra.

A multipath solution has been discussed in [6] where they compute k-maximally disjoint paths instead of k-max min disjoint paths. Also, authors in [10], aim to improve the throughput in shared bottlenecks by forwarding sub-flows from a same MPTCP connection through multiple paths. Suurballe [11] and Bhandari [12] conducted studies on the basic problems of link disjoint. However, they do not consider

the QoS requirements. Disjoint path computation is NP-hard for the min-max problem [13, 14].

Also previous works have shown the benefits of SDN in improving network performance. Sonkoly et al. [15] use a test bed to show improved MPTCP connections by choosing the best path for the first sub flow in an OpenFlow network.

Our contributions can be summarized as follows:

(a) We target two NP-hard problems—multi-constrained optimal routing problem and disjoint path selection.
(b) Combining with the analytic hierarchy process AHP to obtain link weights that meet varying QoS demands.
(c) Convert the multi-criterion problem to obtain a single link value thereby reducing complexity.
(d) Then use a heuristic algorithm for obtaining k-max min disjoint QoS paths.

The concepts of AHP and k-max min paths and many terminologies have been borrowed from previous works [3, 10].

3 Analytic Hierarchy Process

We address the QoS problem by using the analytic hierarchy process, a decision analysis method. The network service quality requirements may be subject to multiple constraints that vary from person to person. AHP helps users assign accurate weights based on their focus on different criteria such as bandwidth, delay, cost, energy consumption, etc. The first step would be construction of hierarchies according to the requirements. Next is to perform a pair wise comparison of factors and weigh them on a scale of 1–9. We assume this scale as it is widely used however actual range may differ. Based on these comparisons we prepare a matrix A as illustrated in Eq. (1) that records the weights or ratings for N factors.

$$A = \begin{bmatrix} a_{11} & L & a_{1N} \\ M & 0 & M \\ a_{N1} & L & a_{NN} \end{bmatrix} \tag{1}$$

Here, $a_{ii} = 1$ and $a_{ij} = 1/a_{ji}$. We perform consistency checking using the following formulas. Our obtained values have passed the consistency checking when CR is less than 0.1 or else we need to reconsider the process.

$$\text{C.I.} = \frac{\lambda_{\max} - n}{n - 1} \tag{2}$$

$$\text{C.R.} = \frac{(a_1)(CI_1) + (a_2)(CI_2) + \cdots + (a_m)(CI_m)}{(a_1)(RI_1) + (a_2)(RI_2) + \cdots + (a_m)(RI_m)} \tag{3}$$

For our example we consider three criteria—bandwidth, delay, and energy consumption denoted as (α, β, ϒ) respectively. To reduce the complexity of processing multiple values we convert them into a single value called Multi-Criterion Cost MCC. We assign it as the link weight for path computation and are given as:

$$\text{MCC} = \text{b(e)} = \frac{\alpha * b(e)}{B} + \frac{\beta * d(e)}{D} + \frac{\gamma * g(e)}{G} \tag{4}$$

Here, b(e), d(e) and g(e) denote the remaining bandwidth, delay and energy consumption of the corresponding edge e. B, D and G are pre-defined constraints for the three factors respectively. In this way we assign new multi-criterion costs to links using AHP that meet QoS requirements of different users. The next phase involves computation of k-max min QoS disjoint paths.

4 *K*-Max Min Disjoint QoS Paths

We use a two-step algorithm for path computation. At first, a set of candidate paths are obtained between a source and destination pair using modified Dijkstra's algorithm. The next step is to use a greedy technique to select the *k*-max min QoS disjoint paths from the candidates of Step 1. The network is represented as a weighted graph *G (V, E)* where, *V* is a set of vertices (SDN Switches) and *E* is a set of edges (Switch links), and *b (u, v)* is the Multi-Criterion Cost (delay and bandwidth) for each edge *(u, v)* ϵ *E* allocated using AHP. Let $s \epsilon V$ and $t \epsilon V$ be the source and destination nodes respectively. Assuming there is a path from *s* to *t*, the minimum remaining MCC (Multi-Criterion Cost) of an edge along the path is the *Bottleneck Constraint* of that path. MBC Maximum Bottleneck Constraint is the maximum among all bottleneck constraints of the multiple paths between node *s* and node *v*. MHC indicates the minimum hop count of the shortest path from *s* to *v*.

In Fig. 1 each node v has a node ID indicated by the first alphabet and the next two numbers indicate v.MBC and v.MHC respectively. Initially for root node s, s.MBC = 0 and s.MHC = ∞, and for all other nodes, v.MBC = ∞ and v.MHC = 0. A node may belong to either of the following—visited, unvisited or marked.

Starting with root node *s*, as *s*.MHC <= *a*.MHC, we execute one-way relaxation of edge *(s, a)* thereby updating *a*.MBC and *a*.MHC to 8 and 1, respectively. Also, we record the parent node *s* and the bottleneck constraint 8 of the path from *s* to *a* as *a* (*s*: 10). Similarly, nodes *b* and *c* are visited and their corresponding MHC and MBC are updated. As all the neighbors of *s* are processed, we change the status of *s* as marked. Next, we select the neighbor of *s* with maximum bottleneck constraint, i.e., *a*. We continue with this process and the Fig. 2 shows the status of the nodes after *a*, *b,* and *d* relax their outgoing edges.

In Fig. 3, we illustrate two-way relaxation operation for nodes *c* and *e* as *e*.MHC > *c*.MHC. This results in edges *(e, c)* and *(c, e)* being relaxed simultaneously.

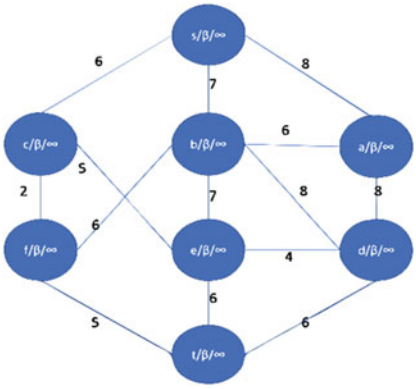


Fig. 1 SDN network with eight switches

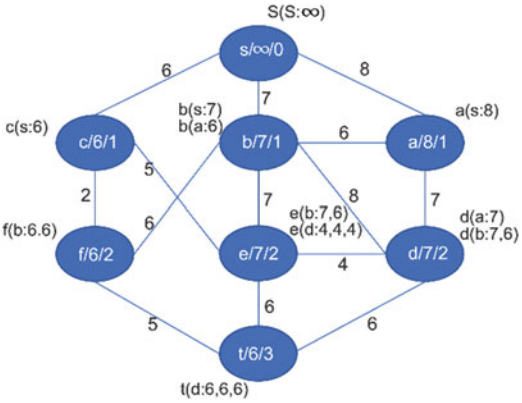


Fig. 2 Nodes *a*, *b* and *d* relaxes their outgoing edges

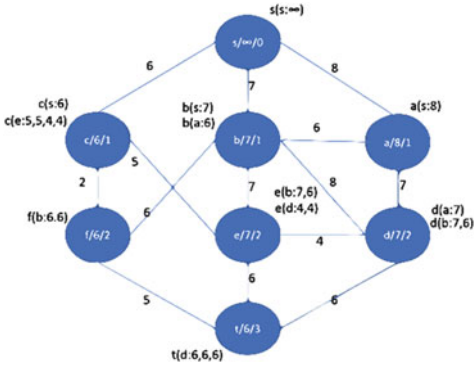
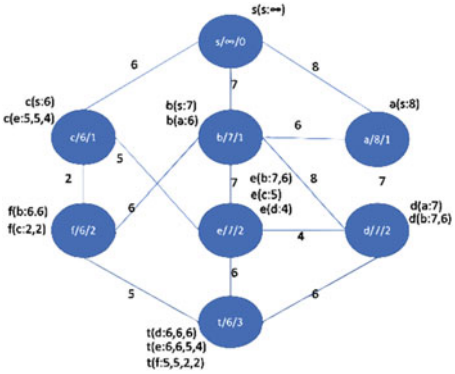


Fig. 3 Two-way relaxation of *(e, c)* and *(c, e)*

While relaxing edge *(e, c)* only the four paths with largest bottleneck constraints are selected. The selection of four paths only is an assumption we make for our algorithm. The final result of Step 1 is as show in Fig. 4.

Fig. 4 Final result of Step 1



Algorithm 1: Obtain a set of candidate paths

```
1: procedure MAIN(s, t):
2: for all neighbours of u: v_i (1 <= i <= n) in increasing order of MHC do
3:          repeat steps 8: to 20:
4:          if multiple nodes with same MHC then
5:                select randomly
6:          end if
7: end for
8: if u.MHC <= v.MHC then
9:          call One-way relaxation (u, v)
10:  else
11:         call Two-way relaxation (u, v)
12:  end if
13:  set node x with maximum MBC (then least MHC) from v_1 to v_i (1 <= i <= n)
14:  add u to Marked
15:  u ← x
16:  if all nodes in Marked then
17:         stop
18:  else
19:         goto step 2
20:  end if
21: end procedure
22: function One-way relaxation(u, v):
23:  if v.MBC < min (u.MBC, b (u, v)) then
24:         set v.MBC = min (u.MBC, b (u, v))
25:  end if
26:  if v.MHC > u.MHC + 1 then
27:         set v.MHC = u.MHC + 1
28:  end if
29:  store parent node as {u: (Bottleneck Bandwidth of paths from s to u)}
30: return
31: function Two-way relaxation (u, v):
32:  call One-way relaxation (u, v)
33:  call One-way relaxation (u, v)
34: return
```

Algorithm 2: Obtain *k*-max min QoS disjoint paths from candidates

Let *x* be the number of candidate paths obtained at destination T at the end of algorithm 1, *k* be the number of outgoing edges of the root node S, *max* be the maximum of the MBC of a set of disjoint paths and *P_temp* holds the set of disjoint paths.

```
1: procedure MAIN(s, t):
2: for i, j in range(1,x) do
3:         if cp_i ≠ cp_j then     /*if cp_i and cp_j are disjoint paths
4:             T_disj [i][j] = True
5:         end if
6: end for
7: for i in range(1,x) do
8:         p_1 = cp_i              /* Candidate path: cp_i
9:         for j in range(1,k) do
10:            Obtain a path p_j disjoint from paths p_1 to p_(j-1)
11:            check (j-1) columns:
12:            T_disj [1][p_1] to T_disj [x][p_1] and;
13:            T_disj [1][p_(j-1)] to T_disj [x][p_(j-1)]
14:            add p_j to P_temp
15:            if path not found then break
16:            end if
17:            Obtain max of paths p_1 to p_k
18:            if new max > current value then
19:                update max and P_temp to new values
20:            else
21:                retain old values of max and P_temp
22:            end if
23:        end for
24:  end for
25:  Obtain k-max min disjoint QoS paths in P_temp
26: end procedure
```

We now use the greedy technique to select *k* disjoint QoS paths from the set of candidate paths and try to maximize the minimum bottleneck constraint path of the k disjoint paths. Let *x* be the number of candidate paths $cp_1, cp_2, \ldots, cp_x$ obtained at the end of Step 1 to reach destination node *t* from the source node *s*. The candidate paths are arranged in the decreasing order of bottleneck constraints. *T_disj [] []* is a two-dimensional array that stores the disjoint path relations among the candidates. This step uses *x* iterations where we obtain *x* sets of *k* disjoint paths, stored in *P_temp []*, and their minimum bottleneck constraints. The maximum of the minimum bottleneck constraints is recorded and the corresponding set of *k*-max min QoS disjoint paths are obtained in *P_temp []*. The final result of Step 2 is as illustrated in Fig. 5 analyzing the time complexity of the algorithm, Step 1 takes $O(|E| \log |V| + |V| \log |V|)$ while Step 2 runs in $O(|V|^2)$.

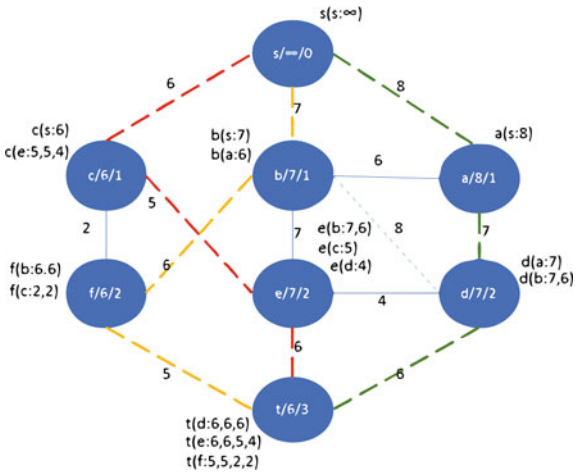


Fig. 5 Final result of Step 2

5 Benefits of SDN

The Open Networking Foundation (ONF) defines SDN as an emerging network architecture where the network is directly programmable and where the control and forwarding planes are decoupled [16, 17]. The network intelligence and states are handled by controllers that globally regulate network states. The separation allows data plane devices to be designed as simple, unintelligent devices (SDN switches) that are solely left with the task of forwarding packets based on decisions taken by the controllers [18]. A set of north-bound application programming interfaces (APIs) serve as the Application-Controller Plane Interface (A-CPI) while the Data-Controller Plane Interface (D-CPI) is handled by south-bound APIs such as OpenFlow protocol. The OF protocol allows the logically centralized controller to dynamically modify the forwarding table of routers and switches. OF uses match-action abstraction to aggregate flows across the data plane [19–21]. The logical view of SDN is illustrated in Fig. 6.

Our intuition for better performance of the proposed system is based on the fact that the two techniques we use have been investigated in previous works [3, 10]. AHP and k-max min disjoint path computation were realized as promising solutions that improved overall network performance. We combine these techniques for implementation in SDN so that the benefits of the promising technology can further enhance performance.

The proposed algorithms can be designed in Mininet. The Mininet creates a realistic virtual network, running the real kernel, switches and application code, on a single machine (VM, cloud or native). This makes us customize our SDN environment. OpenFlow can be used to manage all flows while decision-making and setting of rules would be handled by the SDN controller. The simulations should be preferably carried out in real SDN topologies to obtain accurate results. Initially, generate some random traffic in the network. Next, for a random pair, monitor the end-to-end

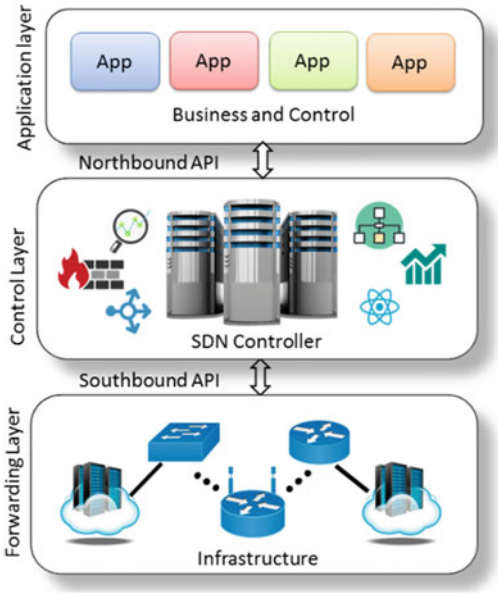


Fig. 6 SDN architecture

communication based on various QoS constraints such as bottleneck bandwidth, jitter, delay, throughput etc. using IPerf for a stipulated time period. Also traffic flow can be analyzed real time using tools such Wireshark. VeriFlow can be used to distinguish between elephant and mice flows for better traffic handling. Finally analyze the collected statistics and verify whether the results satisfy the QoS requirements [10].

6 Conclusion

With SDN accelerating the innovation and evolution of modern networks, development of a highly scalable and intelligent TE system that sustains varying QoS requirements and maintains efficiency of path selection can be envisioned. Therefore, in this paper we propose a two-phase procedure for achieving these objectives. The first phase uses the analytic hierarchy process to capture the varying QoS requirements and reduce it to a new cost function that is used to assign link weights. This reduces the complexity of processing and improves quality control of the network. The second phase obtains k-max min disjoint paths that makes the load balancing of the core network efficient, reduces congestion and improves reliability.

In the future, we focus on implementing the proposed solution in an SDN environment. SDN has a flexible and open architecture that allows dynamic regulation of network behavior. It provides software oriented control of the network and facilitates centralized path routing. As traffic patterns change, dynamic computation of paths

can be done to meet QoS needs. With these benefits, the efficiency of our proposal can be embodied to the maximum degree.

References

1. Hung, M.-H., Wang, C.-H., He, Y.: A real-time routing algorithm for end-to-end communication networks with QoS requirements. In: 3rd International Conference on Computing Measurement Control and Sensor Network (2016). https://doi.org/10.1109/cmcsn.2016.44
2. Orda, A., Sprintson, A.: Author. Efficient algorithm for computing disjoint QoS paths. In: Proceedings of IEEE INFOCOM 2004, vol. 1, pp. 727–738. IEEE Press (2004)
3. Pan, Q., Zheng, X.: MULTI-path sdn route selection subject to multi-constraints. In: 3rd International Conference on Cyberspace Technology (2015). https://doi.org/10.1049/cp.2015.0818
4. Sandri, M., Silva, A., Rocha, L.A., Verdi, F.L.: On the benefits of using multipath TCP and Openflow in shared bottlenecks. In: 29th International Conference on Advanced Information Networking and Applications. IEEE (2015)
5. Chiesa, M., Kindler, G., Schapira, M.: Traffic engineering with equal-cost-multipath: an algorithmic perspective. In: INFOCOM, 2014 Proceedings IEEE, pp. 1590–1598, Apr 2014
6. Abe, J.O., Mantar, H.A., Yayimli, A.G.: k-Maximally disjoint path routing algorithms for SDN. In: International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (2015)
7. Campista, M.E.M., Rubinstein, M.G., Moraes, I.M., Costa, L.H.M.K., Duarte, O.C.M.B.: Challenges and research directions for the future internetworking. Commun. Surv. Tutor. IEEE **16**(2),1050– 1079, Second 2014
8. Akyildiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W.: A roadmap for traffic engineering in SDN-OpenFlow networks. Comput. Netw. **7**(**1**), pp. 1–30, 4 Oct 2014. ISSN 1389-1286
9. Geng, H., Shi, X., Yin, X., Wang, Z., Yin, S.: Algebra and algorithms for multipath QoS routing in link state networks. J. Commun. Netw. **19**(2) (2017)
10. Sheu, J.-P., Liu, L.-W., Jagadeesha, R.B., Chang, Y.-C.: An efficient multipath routing algorithm for multipath TCP in software-defined networks. In: European Conference on Networks and Communications (EuCNC) (2016)
11. Suurballe, J.W.: Disjoint paths in a network. Networks **4**(2), 125–145 (1974)
12. Bhandari, R.: Survivable networks: algorithms for diverse routing. In: Springer Science & Business Media (1999)
13. Guo, Longkun, Shen, Hong: On finding min-min disjoint paths. Algorithmica **66**(3), 641–653 (2013)
14. Li, C.-L., McCormic, S.T., Simchi-Levi, D.: The complexity of finding two disjoint paths with min-max objective function. Discret. Appl. Math. **26**(1), 105–115 (1990)
15. Sonkoly, B., Nemeth, F., Csikor, L., Gulyas, L., Gulyas, A.: SDN based testbeds for evaluating and promoting multipath TCP. In: 2014 IEEE International Conference on Communications (ICC), pp. 3044–3050, June 2014
16. Mendiola, A., Astorga, J., Jacob, E., Higuero, M.: A survey on the contributions of Software-Defined Networking to Traffic Engineering. In: IEEE Communications Surveys & Tutorials, vol. 19, Issue 2, Second quarter 2017. https://doi.org/10.1109/comst.2016.2633579
17. Open Networking Foundation, Software-Defined Networking: The new norm for networks, Technical. report, 2012, white paper
18. Michel, O., Keller, E.: SDN in wide-area networks: a survey. In: 4th International Conference on Software Defined Systems (SDS) (2017). https://doi.org/10.1109/sds.2017.7939138
19. Akyildiz, I.F., Lee, A., Wang, P., Luo, M., Chou, W.: Research challenges for traffic engineering in software defined networks. In: IEEE Network, vol. 30, Issue 3, May-June 2016. https://doi.org/10.1109/mnet.2016.7474344

20. OpenFlow Switch Specification v1.0-v1.4. www.opennetworking.org/sdn-resources/onf-specifications
21. Akyildiz, F., et al.: A roadmap for traffic engineering in software defined networks. Comput. Netw. **71**, 1–30 (2014)