LADEQ: A Fast Lagrangian Relaxation Based Algorithm for Destination-Based QoS Routing

Nitin Varyani[1], Zhi-Li Zhang[1], Muralidharan Rangachari [2] and David Dai [2]

[1]Department of Computer Science & Engineering, University of Minnesota, Minneapolis
{varya001,zhang089}@umn.edu

[2]Huawei Futurewei Technologies Inc, Santa Clara
{Muralidharan.Rangachari,david.h.dai}@huawei.com

***Abstract*—Quality of Service provisioning in today's overlay networks includes computing routes that simultaneously guarantee multiple QoS metrics like bandwidth, delay, jitter, and packet-loss rate. Lagrange relaxation-based aggregated cost (LARAC) algorithm is among the best centralized algorithms for finding a near optimal solution to the constrained shortest path (CSP) problem for the additive metrics. To take advantage of the LARAC algorithm, we transform the non-linear QoS routing problem into a linear integer programming problem by converting all constraints to additive. We then develop a multi-constrained version of the LARAC algorithm and use sub-gradient optimization to converge to a near optimal solution. As LARAC algorithm needs to solve the routing optimization problem separately for every source and destination pair, this significantly increases the total time complexity. We, therefore, modify the LARAC algorithm to destination-based QoS routing, LADEQ, to reduce the number of routing optimization problems solved. This also reduces the size of the forwarding tables. A trace-driven evaluation shows that as the network size is increased, the time taken by our algorithm, LADEQ, was significantly smaller than the state-of-the-art multi-constrained shortest path (MCSP) algorithms applied to all source and destination pairs.**

***Index Terms*—QoS, routing, Lagrange, sub-gradient, integer programming, destination-based**

I. Introduction

The Internet provides a best-effort service where the inter-domain routing is governed by business relationships rather than the needs of applications. Many applications and systems such as multimedia applications, smart grid systems and industrial systems require Quality of Service (QoS) which cannot be directly served by the Internet. Overlay networks has been an effective way to support QoS provisioning at the application layer [1]–[7]. The QoS requirements are generally expressed as the end-to-end bounds for single or multiple constraint metrics like delay, jitter, bandwidth and packet-loss rate [8].

A QoS routing problem for a single constraint is generally expressed as a constraint shortest path (CSP) problem. Minimizing a cost metric while keeping the value of a constrained metric within its bounds is called a CSP problem, which is known to be NP-hard. Lagrange relaxation-based aggregated cost (LARAC) [9] has been identified among the best central routing algorithms for finding an approximate solution to a constrained shortest path (CSP) problem [8]. LARAC relaxes the only side constraint into the objective function and then uses a single source shortest path algorithm to find the shortest path in a graph with modified costs. If the solution is not near to optimal, the costs are modified again using subgradient optimization [10] and the single source shortest path algorithm is run again to finally converge to a near optimal solution.

978-3-903176-15-7 © 2019 IFIP

When we need to compute routes which simultaneously guarantee multiple additive QoS metrics, it has to be modelled as a multi-constrained shortest path (MCSP) problem. When the CSP problem has more than one constrained metric whose bound has to be met, it is called a MCSP problem, which is known to be NP-complete. To approximate the solution of the MCSP problem using LARAC, we need to define its multi-constrained version.

As LARAC algorithm needs to solve a multi-constrained shortest path (MCSP) separately for every source and destination pair and for every QoS policy, this significantly increases the total time complexity. Thus, we also need heuristics to reduce the computation time.

In this paper, we made the following contributions:

First, we transform the non-linear QoS routing problem into a linear integer programming problem by converting all constraints to additive. This was done so that we can take advantage of the LARAC algorithm. We then developed a multi-constrained version of the LARAC algorithm and used sub-gradient optimization to converge to a near-optimal solution.

Second, we develop a destination-based QoS routing to reduce the number of linear integer programming problems solved. This also reduced the size of the forwarding tables.

The paper is organized as follows. Section II summarizes the related work. In section III, we formulate the QoS routing problem. Section IV describes the multi-constrained version of the LARAC algorithm and the destination-based QoS routing. Results from a trace-based evaluation of LADEQ is presented in Section V. The papers ends with a conclusion and ideas for future work in Section VI.

II. Related Work

Significant research has been done in solving the MCSP problem and its application in network routing optimization. A*Prune [11] solves the MCSP problem by assuming that there is a guess function available for the constraints and costs and the algorithm is made faster by pruning certain paths based on their projected constrained values. However, its run-time increases exponentially with network size and is much slower than the LARAC-based algorithms. A multi-constrained version of the LARAC algorithm, H_MCOP is proposed in [12] which searches in the direction of all constraints and cost simultaneously, but this approach does not always return a feasible solution if it exists [13]. MH_MCOP [13] finds a

closer solution to the optimal as compared to H_MCOP but it also does not always find a feasible solution if it exists [8].

In [14], the authors model the QoS routing problem as a multi-commodity flow problem which is decomposed to simpler constrained shortest path problems . They use the GEN-LARAC algorithm [15] to solve the constrained shortest path problem that satisfies all QoS constraints. The authors in [16] model the overlay routing problem as maximizing the Quality of Experience (QoE) instead of Quality of Service and performed Lagrange decomposition of their original problem. The subproblems are solved using Lagrange relaxation and sub-gradient optimization. They used k-shortest path algorithm to find shortest paths in their sub-problems. We also use a LARAC like algorithm to solve our multi-constrained shortest path sub-problems. However, the key difference is that we have developed a heuristic to reduce the number of multi-constrained shortest path sub-problems solved. This significantly reduced the time complexity of our routing algorithm.

III. Problem Formulation

This section explains our system framework and the mathematical formulation of our QoS routing problem.

A. System Framework

Our overlay routing framework is depicted in Fig. 1. Several content servers are connected to remote end users using the existing overlay network. Instead of directly connecting the end users to the source, end users connect to an ingress software router placed in the cloud. The choice of the best ingress router an end user must be connected to is handled by the DNS system. The connection between the routers are created using an Overlay tunnelling protocol. Each flow from an ingress router to an egress router is assigned a QoS policy. Each QoS policy is described as a list of bounds on the end-to-end delay, jitter, and packet-loss for a particular flow and the minimum bandwidth requirement of that flow. The overlay routers use a traffic measurement tool to regularly monitor the delay, jitter, packet-loss and bandwidth of its tunnelled links and report it to a centralized SDN controller. The SDN controller computes the least cost overlay path that meets all of the QoS constraints mentioned above to transfer the content from the content server to the end user.

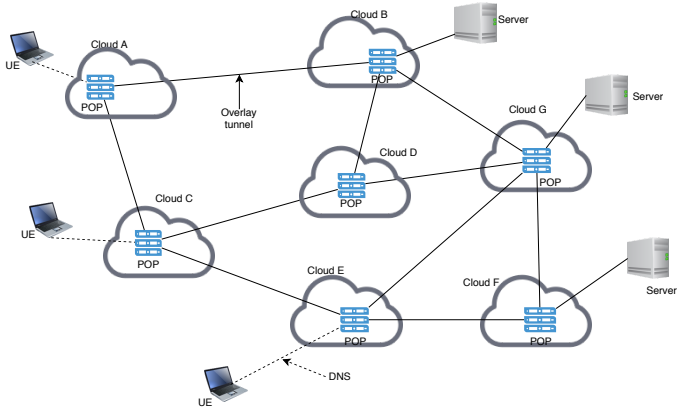

Fig. 1: System framework of our overlay network.

B. Mathematical Formulation

We model the QoS routing problem as finding a route for a new flow in a network with existing flows such that the QoS requirements of both the new flow and the existing flows are maintained. Consider a network graph $G = (V, E)$, where V is the set of overlay routers and E is the set of overlay tunnels. The row-vector of "costs" of the edges is denoted by c, $c \in R_+^{|E|}$. The "cost" denotes the price for using a link to deliver traffic and is determined by business relationships. Let d, j, b and l, $\{d, j, b, l\} \in R_+$, denote the bounds for delay, jitter, bandwdith, and packet loss probability respectively. Let D, J, B and L, $\{D, J, B, L\} \in R_+^{|E|}$, denote the row-vectors of delay, jitter, bandwidth and packet loss probability of individual edges respectively. Given a source vertex $s \in V$ and a destination vertex $t \in V$, we want to find a path of minimum "cost" from s to t which satisfies all the QoS constraints. Let F denote the $|V| \times |E|$ vertex-edge incidence matrix such that if $e = (u, v)$, then $F_{ue} = 1$, $F_{ve} = -1$ and $F_{we} = 0$ for any $w \neq u, v$. Let G, $G \in R_+^{|V|}$, be a vector such that $G_s = 1$, $G_t = -1$ and $G_v = 0$ for all $v \in V \backslash \{s, t\}$. Let x be a column-vector $(x_1, x_2 ... x_{|E|})$, $x_i \in \{0, 1\}$ of decision variables where $x_i = 1$ if the edge i belongs to the final routing path and 0 otherwise. The QoS routing problem can be written as the following combinatorial problem.

$$z = \min_{x \in \{0,1\}^{|E|}} c\ x \tag{1}$$

s.t.

$$F\ x = G \tag{1a}$$

$$D\ x \leq d \tag{1b}$$

$$\prod_{i=1}^{|E|}((1 - L_i)\ x_i) \geq 1 - l \tag{1c}$$

$$J\ x \leq j \tag{1d}$$

$$\min_{i=1}^{|E|}(B_i\ x_i) \geq b \tag{1e}$$

The constraint (1a) restricts the values of x to the all possible directed paths from s to t. Constraints (1b) and (1d) ensures that the additive metrics (delay and jitter) of the paths are below their respective maximum bounds. Constraint (1c) checks if the product of success probability of the links of the paths is greater than the minimum bound on the success probability. Finally, the constraint (1e) limits the bottleneck bandwidth of the paths to be greater than the minimum bound on the bandwidth.

IV. QoS Routing Algorithm

In this section, we first describe our multi-constrained version of LARAC algorithm and sub-gradient optimization and then describe the modifications made to it to convert it into destination-based QoS routing algorithm.

A. Linear integer programming formulation

We transform the non-linear QoS routing problem into a linear integer programming problem by removing or transforming non-additive constraints. This was done to take advantage of the LARAC algorithm. We remove the concave constraint, bandwidth, by pre-processing the graph by pruning links that

did not have sufficient bandwidth. The links were pruned by setting the delay and jitter of those links to infinity and the packet loss probability to 1. We then convert the multiplicative constraint, packet success probability, into an additive constraint by taking the negative logarithm of the packet success probability of each link. Thus, our routing problem reduces to the following form:

$$z = \min_{x \in \{0,1\}^{|E|}} c\ x \tag{2}$$

s.t.

$$F\ x = \ G \tag{2a}$$
$$D\ x \leq \ d \tag{2b}$$
$$L^{'}\ x \leq \ l^{'} \tag{2c}$$
$$J\ x \leq \ j \tag{2d}$$

We get the row-vector L' by setting $L_e^{'} = -\log(1 - L_e)$ for all $e \in E$. The constant $l^{'}$ is equal to $-\log(1 - l)$.

B. Lagrange relaxation

We relax all the inequality constraints 2b-2d, by introducing the amount of violation of these constraints and their corresponding Lagrange variables into the objective functions. Let $\gamma_1, \gamma_2, \gamma_3 \in R_+$ be the Lagrange multipliers for constraint 2b,2c,2d, respectively. Thus, our integer programming problem reduces to the following Lagrange dual problem.

$$z_L = \max_{\gamma_1,\gamma_2,\gamma_3} LR(\gamma_1, \gamma_2, \gamma_3) \tag{3}$$

s.t.

$$\gamma_1, \gamma_2, \gamma_3 \in R_+ \tag{3a}$$

where $LR(\gamma_1, \gamma_2, \gamma_3)$ is the Lagrangian dual function which is minimized subject to the non-dualized constraint.

$$LR(\gamma_1, \gamma_2, \gamma_3) = \min_{x \in \{0,1\}^{|E|}} (cx + \gamma_1(Dx - d) + \gamma_2(L^{'}x - l^{'}) + \gamma_3(Jx - j)) \tag{4}$$

s.t.

$$F\ x = \ G \tag{4a}$$

C. Single source shortest path algorithm

The Lagrangian dual function 4 can be rearranged to the following form.

$$LR(\gamma_1, \gamma_2, \gamma_3) = \min_{x \in \{0,1\}^{|E|}} ((c + \gamma_1 D + \gamma_2 L^{'} + \gamma_3 J)x - (\gamma_1 d + \gamma_2 l^{'} + \gamma_3 j)) \tag{5}$$

s.t.

$$F\ x = \ G \tag{5a}$$

Since $\gamma_1 d + \gamma_2 l^{'} + \gamma_3 j$ is a constant, the value of x corresponding to the optimal solution of the Lagrangian dual function 5 is equal to that of Lagrangian dual function 6, which is turn, is equivalent to the shortest path between s and t with the cost of the links given by the vector $c + \gamma_1 D + \gamma_2 L^{'} + \gamma_3 J$.

$$LR(\gamma_1, \gamma_2, \gamma_3) = \min_{x \in \{0,1\}^{|E|}} ((c + \gamma_1 D + \gamma_2 L^{'} + \gamma_3 J)x) \tag{6}$$

s.t.

$$F\ x = \ G \tag{6a}$$

We can use the fast single source shortest path implementations like Fibonacci based Dijkstra which has a time complexity of $O(E + VlogV)$ or can use an even faster algorithm like A* algorithm to find the shortest path between the given pair of nodes.

Algorithm 1 LADEQ: Destination based QoS routing

procedure ROUTING($V, E, t, c, D, B, L, J, d, b, l, j$)

1: $M = \phi$
2: $E^{'} = ReverseLink(E)$
3: $S = BFS(V, E^{'}, t)$
4: **while** $M\ != V$ **do**
5: Let $\gamma_1 = 2, \gamma_2 = 2, \gamma_3 = 2$
6: $s = S.pop()$
7: **for** $k \leftarrow 1$ to $iterations$ **do**
8: $X_{LR} = Dijkstra(s, c + \gamma_1 D + \gamma_2 L^{'} + \gamma_3 J)$
9: get shortest path from s to t, x_{st}, from X_{LR}
10: **if** $Dx_{st} \leq d$ **and** $L^{'}x_{st} \leq l^{'}$ **and** $Jx_{st} \leq j$ **then**
11: **if** $(\gamma_1 = 0$ **or** $Dx_{st} - d = 0)$ **and** $(\gamma_2 = 0$ **or** $L^{'}x_{st} - l^{'} = 0)$ **and** $(\gamma_3 = 0$ **or** $Jx_{st} - j = 0)$ **then**
12: **break**
13: **else**
14: $UB = cx_{st}$
15: **end if**
16: **end if**
17: **if** $UB == null$ **then**
18: $UB = c_{max}|E|$
19: **end if**
20: $LB = cx_{st} + \gamma_1(Dx_{st} - d) + \gamma_2(L^{'}x_{st} - l^{'}) + \gamma_3(Jx_{st} - j)$
21: $\theta = \frac{UB - LB}{(Dx_{st}-d)^2 + (L^{'}x_{st} - l^{'})^2 + (Jx_{st}-j)^2}$
22: $\gamma_1 = max(0, \gamma_1 + \theta \times (Dx_{st} - d))$
23: $\gamma_2 = max(0, \gamma_2 + \theta \times (L^{'}x_{st} - l^{'}))$
24: $\gamma_3 = max(0, \gamma_3 + \theta \times (Jx_{st} - j))$
25: **end for**
26: $M = M \cup nodes(x_{st})$
27: **end while**
28: reset D, L, J

end procedure

D. Sub-gradient optimization

We explore the solution space of the dual problem using a sub-gradient descent algorithm [10]. Initially, we compute a solution to the Lagrange dual problem and if the solution was feasible for the original problem, we choose the cost of the feasible path as the upper bound for our sub-gradient descent algorithm. If it was not feasible, we initialize it to the product of the number of edges in the graph and the maximum cost among all edges. As we find a feasible path while iterating, we update the upper bound with the cost of that feasible path(Lines 14-18, Algorithm 1). We use the following standard theorem to test whether the solution of Lagrangian

dual function is optimal for original problem or not (Lines 10-11, Algorithm 1).

Theorem 1. *A solution x_{st} to a Lagrangean minimization program is optimal for the original problem only if:*
(a) x_{st} is feasible for the original problem
(b) $cx_{st} = [cx_{st}+\gamma_1(Dx_{st}-d)+\gamma_2(L^{'}x_{st}-l^{'})+\gamma_3(Jx_{st}-j)]$ i.e. $\gamma_1(Dx_{st}-d)+\gamma_2(L^{'}x_{st}-l^{'})+\gamma_3(Jx_{st}-j)=0$

We initialized all the Lagrange multipliers to 2. The subgradients for the relaxed constraints 2b, 2c and 2d are $(Dx_{st}-d)$, $(L^{'}x_{st}-l^{'})$ and $(Jx_{st}-j)$ respectively.

Our scalar step size θ is given by

$$\theta = \frac{UB-LB}{(Dx_{st}-d)^2+(L^{'}x_{st}-l^{'})^2+(Jx_{st}-j)^2}. \quad (7)$$

The step size depends upon the difference between the current upper bound (UB) and the current lower bound (LB) with $(Dx_{st}-d)^2+(L^{'}x_{st}-l^{'})^2+(Jx_{st}-j)^2$ being the scaling factor. The Lagrange multipliers are updated in lines 22-24 of Algorithm 1 and the Lagrange dual function is solved again with this new set of multipliers. The algorithm is terminated if it runs a predefined number of iterations or it meets the optimality condition in Theorem 1.

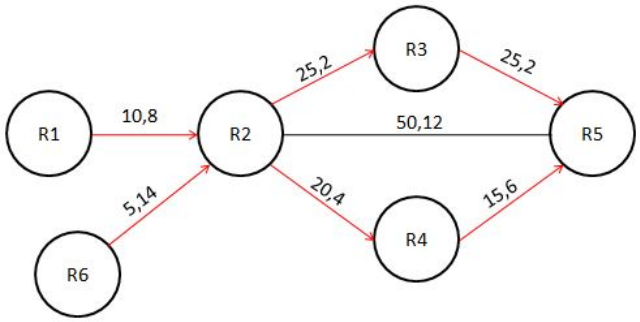

Fig. 2: Destination-based QoS DAG for the destination R5 and the QoS policy (55,18) for a network of bi-directed links.

E. QoS Directed Acyclic Graph

A simple tree structure cannot serve as a routing Directed Acyclic Graph (DAG) that meets all the QoS contraints.

For example, consider the network graph in Fig. 2, where the pair of values on each link represents the delay and jitter of the corresponding bi-directed link. The units are in milliseconds. We have considered only 2 additive constraints for simplicity but this will implicitly hold for more than 2 additive constraints. Suppose we need to compute feasible routes towards the destination router R5, which have a maximum end-to-end one way delay of 55ms and the maximum end-to-end one way jitter of 18ms, from every other router. Then, the only feasible path from router $R1$ to $R5$ is $R1 \rightarrow R2 \rightarrow R4 \rightarrow R5$ while the only feasible path from router $R6$ to $R5$ is $R6 \rightarrow R2 \rightarrow R3 \rightarrow R5$. Thus, the packets arriving at R2 from R1 should be forwarded to R4 and packets from R6 should be forwarded to R3 to meet the QoS requirements. This indicates that there is no routing tree for each destination router and policy that meets all the QoS constraints. In other words, to properly forward the packets, we will have to use source IP address, destination IP address and policy ID as the key in the forwarding tables or define a new key.

F. Forwarding tables

We need to store some extra information in the forwarding tables that removes the need of the source IP address in the key. We replace the source IP address with a key for each of the additive constraints 2b-2d. The new keys added denote the maximum permissible amount of delay or jitter or packet loss the packets can go through when it arrives at that router to become eligible to be forwarded to the corresponding next hop. For example, the forwarding table for router R2 corresponding to the DAG in Figure 2 (b) is shown in Table I.

TABLE I: Forwarding table for Router R2

Policy	*Destination*	*Maximum elapsed delay*	*Maximum elapsed jitter*	*Next hop*
1	R5	5	14	R3
1	R5	20	8	R4
1	R5	∞	∞	R3
2	R5	..	..	..

Table I denotes a few of the forwarding entries for the routing DAG shown in Figure 2 (b). For Policy 1, we have a maximum tolerable delay of 55 ms and jitter of 18ms and the delay and jitter along the path, $R2 \rightarrow R3 \rightarrow R5$, is 50ms and 4ms, respectively. Thus, if the maximum permissible delay and jitter that a packet had gone through before reaching router R2 is 5ms (55-50) and 14ms (18-4) respectively, then the packet can be forwarded to router R3. This corresponds to the first entry in the forwarding table (Table I). Similarly, we generate other forwarding entries for other feasible routes generated by Algorithm 3 and passing through router R2.

For the given example, the encapsulated overlay packets carry two additional values in their overlay headers; the delay and jitter elapsed since they left the source. At each router, that information is updated by adding the delay and jitter of the outgoing link to those values respectively. If a packet that is ingress-ed through R1 reaches R2, these two values will be 10 and 8, respectively. When we do a comparison of these two values to the entries in the forwarding table of R2, we find that only an entry that is valid for it is the second one. Similarly, only the 1st entry matches the packet which ingress-ed from R6 and reached R2. Thus, we can see that the destination router, policy ID and the amount of each additive metric spent by a packet before reaching a router can properly forward the packets. If none of the entry matches, then the packet takes a default route which generally happens during routing updates and failures. The third entry in Table I is a default entry for packets destined to router R5 and having policy ID 1. The default rule has its maximum elapsed delay and maximum elapsed jitter set to a very large number so that all related packets matches that entry. LADEQ generates lesser forwarding entries compared to source-destination based routing because it only generates entries equal to number of outgoing routes for a particular policy.

G. Generating destination-based DAG

This section describes how we generate the destination-based directed acyclic graph (DAG) for a particular destination router and policy.

We observe that any feasible route from an ingress router to an egress router also contains within it the feasible routes from the intermediary routers to the same egress router. Thus, computing feasible routes from a subset of nodes to the destination router will cover all the nodes in the graph.

To reduce those subset of nodes, we developed the following approach. We first reverse all the links in the original graph and then performed breadth-first search (BFS) starting from the destination router (Lines 2-3 of Algorithm 1). Using BFS, the nodes are stored in the decreasing order of their hop distance from the destination router. In this order, we solve the MCSP problem between other routers and the destination router. This is because the farther a router is from the destination router, a feasible route from such router will contain feasible routes for much more intermediary nodes. Finally, we check if all the routers in the graph are covered (line 4 and 26, Algorithm 1) and if it is, we terminate the algorithm. This reduces the route computation time for the LADEQ algorithm.

V. Evaluation

We perform a trace-driven evaluation of the LADEQ routing algorithm to determine its scalability and the optimality. We use traces from a real overlay network of a global communications technology solutions provider. The QoS routing algorithm is written in Python using fast libraries like Dijikstar and the traces were provided as a CSV input file to the Python program.

We use a 24 hour trace which contained the delay, jitter, packet loss and bandwidth of the overlay links connecting the service routers. The traces used for our experimentation were available for every 30 seconds. We used QoS requirements of 10 different kinds of user-applications to evaluate the algorithms. As an example, the QoS requirement used for HD1080, 30 fps video conferencing was (150ms, 30ms, 10Mbps, 0.5%). The tuple is in the format (maximum delay bound, maximum jitter bound, minimum bandwidth requirement, maximum packet loss). Thus, our plots are generated based on the average value of 28,800 ($24 \times 60 \times 2 \times 10$) instances of data points. Table II gives the mean and standard deviation of delay, jitter, packet loss and bandwidth values of the network links over the period of 24 hours. The topology of the overlay network consisted of service routers spread across multiple data centers world wide. The traces were available for different sizes of the overlay network and for different topologies like partial mesh, star, ring and tree.

TABLE II: Summary of QoS metrics statistics of the overlay links in the traces

Metrics	*Mean*	*Standard Deviation*
delay (ms)	25.49	9.27
jitter (ms)	11.53	4.75
bandwidth(Mbps)	80	10.53
packet loss(%)	0.4	0.35

We use these traces to evaluate the scalability and optimality of LADEQ compared to other state-of-the-art multi-constrained shortest path (MCSP) algorithms, A* Prune [11], H_MCOP [12] and MH_MCOP [13].

We measure four performance characteristics:

- **Route Computation Time**: This includes generating the routes for all the destination routers and policies as well as generating forwarding tables for the service routers using these routes.
- **Cost Deviation**: Cost deviation denotes the percentage deviation of the cost of the approximate solution from the cost of the solution returned by A*Prune which is optimal.
- **Percentage Reduction in Forwarding Entries**: This denotes the reduction in forwarding entries compared to those generated by source-destination-based routing algorithms such as A*Prune, H_MCOP and MH_MCOP.
- **Relative percentage difference**: This is a measure of the relative difference between the QoS of the paths found and the maximum QoS bound. It is computed using the formula $(QoS\_bound - path\_QoS)/QoS\_bound \times 100$.

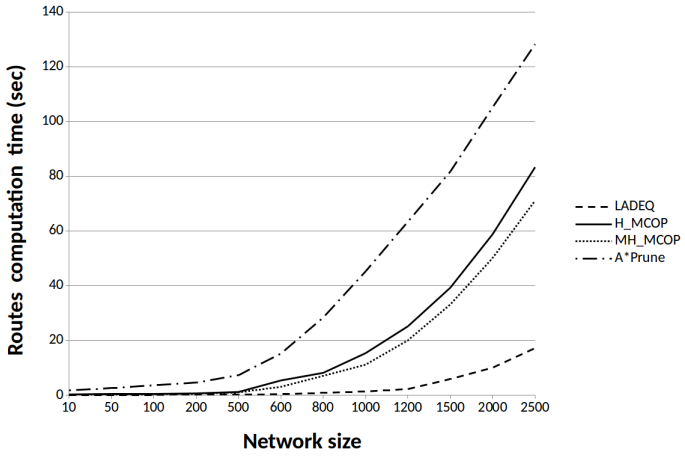

Fig. 3: Routes computation time comparison between LADEQ, A*Prune, H_MCOP and MH_MCOP

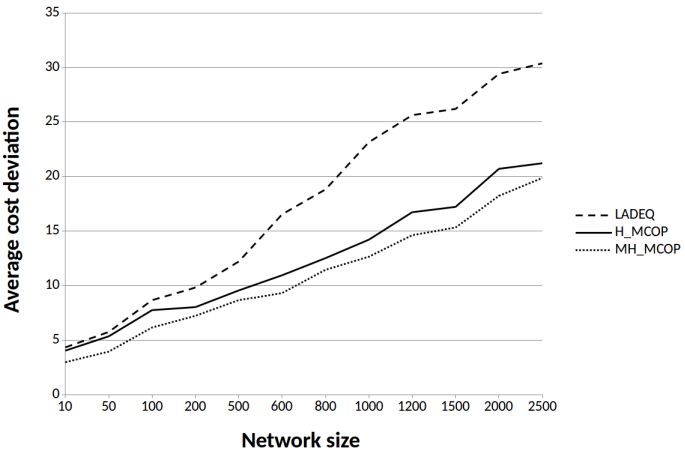

Fig. 4: Average cost deviation of LADEQ, H_MCOP and MH_MCOP from A*Prune

Fig. 3, 4, 6 and 7 were generated using a partial mesh topology and fig. 5 was generated using networks of size 1000.

Fig. 3 shows that LADEQ has a significantly lower route computation time compared to other algorithms like A*Prune, H_MCOP and MH_MCOP. The reduction in route computa-

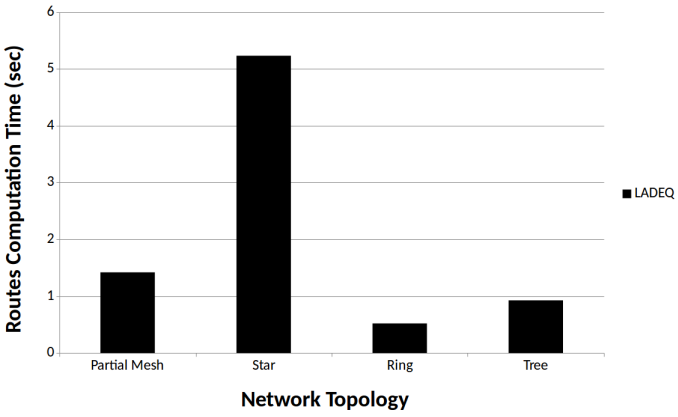

Fig. 5: Route computation time of LADEQ in different network topologies

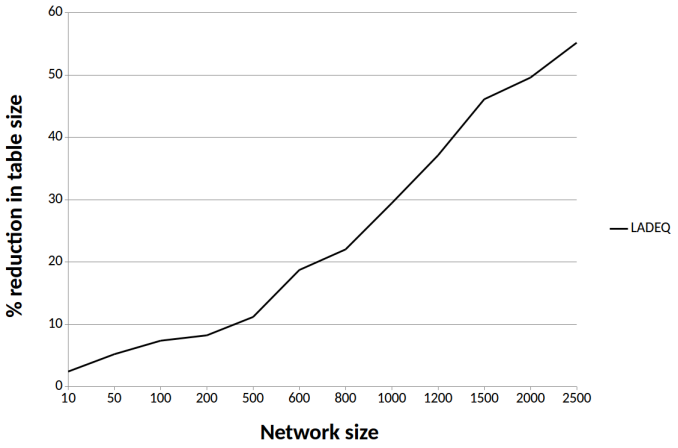

Fig. 6: Percentage reduction in size of forwarding tables generated by LADEQ compared to source-destination-based routing algorithms

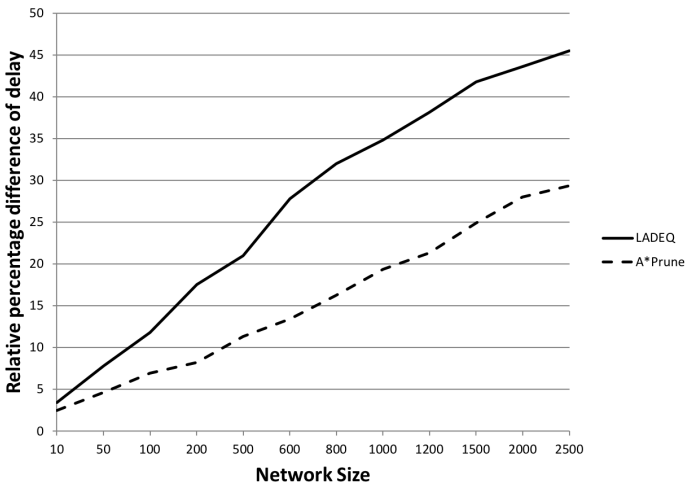

Fig. 7: Relative percentage difference of LADEQ and A*Prune for delay QoS metric

tion time is primarily because we reduce the number of multi-constrained shortest problems solved.

The time complexity of LADEQ was reduced at the cost of the optimality of the solution and thus H_MCOP and MH_MCOP perform better in terms of optimality (Fig. 4). However, unlike H_MCOP and MH_MCOP, LADEQ always returns a feasible route if it exists. If a small increase in cost of overlay links is not a major concern for overlay providers and they want a quickly adaptive route computation algorithm for a network with fast changing traffic, then the LADEQ algorithm would be highly helpful.

LADEQ computes the routes faster in the ring, tree, and mesh topology but not in star topology (Fig. 5). This is because the star topology does not benefit from our heuristic of a feasible route between two routers covering the feasible routes of many intermediary routers.

Fig. 6 demonstrates that LADEQ results in a significant reduction in size of forwarding tables compared to source-destination-based routing algorithms as network size grows. This is because we have replace the source IP address in the key of the forwarding entries with the maximum elapsed constraint values.

Fig. 7 shows the comparison of LADEQ and A*Prune in terms of relative percentage difference of the delay QoS metric. We do not use H_MCOP and MH_MCOP because they do not always generate a feasible path. We can observe that as the network size increases, the relative percentage difference of LADEQ increases faster than that of the A*Prune. This is because the feasible paths from some nodes to a destination node are contained in a feasible path from one of the nodes to the destination node.

VI. Conclusion and Future Work

In this paper, we have formulated a non-linear integer QoS routing problem for overlay networks, which is then converted to a linear integer programming problem to take advantage of a LARAC algorithm. We have developed a multi-constrained version of a LARAC algorithm and applied sub-gradient optimization to converge to a near optimal solution. We have also developed a heuristic to reduce the number of linear integer programming problems solved by modifying our algorithm to destination-based QoS routing. A trace-based evaluation shows that our algorithm, LADEQ, achieves lower route computation time and a reduction in the forwarding table size.

To implement LADEQ algorithm in a real overlay network, the routers should support the comparison and the addition operations on packet header fields along with the capabilities of OpenFlow switches [17], [18]. LADEQ algorithm can be prototyped using P4-NetFPGA switches [19]–[21] to evaluate its performance in a real test-bed. The routing algorithm should also compute the backup next hops in case some overlay link is down. This can be done by generating destination-based directed acyclic graphs (DAGs) which are completely or partially disjoint from the primary DAG.

References

[1] Li, Z. and Mohapatra, P., 2004. QRON: QoS-aware routing in overlay networks. IEEE Journal on Selected Areas in Communications, 22(1), pp.29-40.

[2] Andersen, D., Balakrishnan, H., Kaashoek, F. and Morris, R., 2001. Resilient overlay networks (Vol. 35, No. 5, pp. 131-145). ACM.

[3] Kurian, J. and Sarac, K., 2010. A survey on the design, applications, and enhancements of application-layer overlay networks. ACM Computing Surveys (CSUR), 43(1), p.5.
[4] Rahul, H.S., Kasbekar, M., Sitaraman, R.K. and Berger, A.W., 2008. Performance and availability benefits of global overlay routing. In Content Delivery Networks (pp. 251-272). Springer, Berlin, Heidelberg.
[5] Sitaraman, R.K., Kasbekar, M., Lichtenstein, W. and Jain, M., 2014. Overlay networks: An akamai perspective. Advanced Content Delivery, Streaming, and Cloud Services, 51(4), pp.305-328.
[6] Duan, Z., Zhang, Z.L. and Hou, Y.T., 2003. Service overlay networks: SLAs, QoS, and bandwidth provisioning. IEEE/ACM Transactions on Networking (TON), 11(6), pp.870-883.
[7] Mukerjee, M.K., Naylor, D., Jiang, J., Han, D., Seshan, S. and Zhang, H., 2015, August. Practical, real-time centralized control for cdn-based live video delivery. In ACM SIGCOMM Computer Communication Review (Vol. 45, No. 4, pp. 311-324). ACM.
[8] Guck, J.W., Van Bemten, A., Reisslein, M. and Kellerer, W., 2018. Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation. IEEE Communications Surveys & Tutorials, 20(1), pp.388-415.
[9] Juttner, A., Szviatovski, B., Mcs, I. and Rajk, Z., 2001. Lagrange relaxation based method for the QoS routing problem. In INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 2, pp. 859-868). IEEE.
[10] Boyd, S., Xiao, L. and Mutapcic, A., 2003. Subgradient methods. lecture notes of EE392o, Stanford University, Autumn Quarter, 2004, pp.2004-2005.
[11] Liu, G. and Ramakrishnan, K.G., 2001. A* Prune: an algorithm for finding K shortest paths subject to multiple constraints. In INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 2, pp. 743-749). IEEE.
[12] Korkmaz, T. and Krunz, M., 2001, April. Multi-constrained optimal path selection. In IEEE INFOCOM (Vol. 2, No. April, pp. 834-843). INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE).
[13] Feng, G., Makki, K., Pissinou, N. and Douligeris, C., 2002. Heuristic and exact algorithms for QoS routing with multiple constraints. IEICE Transactions on Communications, 85(12), pp.2838-2850.
[14] Medagliani, P., Paris, S., Leguay, J., Maggi, L., Xue, C. and Zhou, H., 2017, May. Overlay routing for fast video transfers in CDN. In Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on (pp. 531-536). IEEE.
[15] Xiao, Y., Thulasiraman, K. and Xue, G., 2005, December. GEN-LARAC: A generalized approach to the constrained shortest path problem under multiple additive constraints. In International Symposium on Algorithms and Computation (pp. 92-105). Springer, Berlin, Heidelberg.
[16] Calvigioni, G., Aparicio-Pardo, R., Sassatelli, L., Leguay, J., Medagliani, P. and Paris, S., 2018, April. Quality of Experience-based Routing of Video Traffic for Overlay and ISP Networks. In IEEE International Conference on Computer Communications (INFOCOM).
[17] McKeown, Nick, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38, no. 2 (2008): 69-74.
[18] OpenFlow Switch Specification, Version 1.3.0 (Wire Protocol 0x04), June 25, 2012. https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf
[19] Bosshart, Pat, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 44, no. 3 (2014): 87-95.
[20] P4 $\rightarrow$ NetFPGA project wiki. https://github.com/NetFPGA/P4-NetFPGA-public/wiki
[21] NetFPGA SUME wiki https://github.com/NetFPGA/NetFPGA-SUME-public/wiki