World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:6, 2008

Optimized Delay Constrained QoS Routing

P. S. Prakash, and S. Selvan

***Abstract*—**QoS Routing aims to find paths between senders and receivers satisfying the QoS requirements of the application which efficiently using the network resources and underlying routing algorithm to be able to find low-cost paths that satisfy given QoS constraints. The problem of finding least-cost routing is known to be NP-hard or complete and some algorithms have been proposed to find a near optimal solution. But these heuristics or algorithms either impose relationships among the link metrics to reduce the complexity of the problem which may limit the general applicability of the heuristic, or are too costly in terms of execution time to be applicable to large networks. In this paper, we concentrate an algorithm that finds a near-optimal solution fast and we named this algorithm as optimized Delay Constrained Routing (ODCR), which uses an adaptive path weight function together with an additional constraint imposed on the path cost, to restrict search space and hence ODCR finds near optimal solution in much quicker time.

***Keywords*—**QoS, Delay, Routing, Optimization.

I. Introduction

DELAY sensitive application such as video conferencing, video streaming VoIP etc. require packets to be delivered to the destination within the stipulated time period. At the same time, it is highly desirable to reduce the path cost as much as possible; this may be monetary cost or cost of utilizing network resources. Performance of operational networks can be improved by engineering Internet traffic so as it is routed over resource-efficient constrained based paths [1].However constrained shortest path problem or multi constrained optimization path selection problem is highly challenging and has been proved to be NP-complete [2].In this paper, we analyze the problem of finding a least-cost path subject to an end-to-end delay constraint. It is called as a Delay-constrained Least Cost (DCLC) unicast routing problem, or broadly, a constrained optimization problem. An optimal solution proposed in [3] which performs breath-first search to find the optimum path, thus its running time might grow exponentially. Algorithm proposed by H. Salama [4] try to compute the path distributively in order to overcome the centralized computation overhead, but paths returned by these algorithms may be costly and the path set up time may be too long. Some earlier studies mainly focus on a simpler problem; the multiple-constraints path(MCP), which does not optimize the value of any of the metrics, instead, it only seeks a feasible path that satisfies all the constraints and this problem is NP-hard if more than one metric is additive and takes real values[5]. In [6] a non- linear function of link cost and delay is proposed to convert the problem into the much simpler single-metric routing problem, and so as to efficiently find a path that is far away from all the metric bounds.

Since heuristic of MCP problem is easier in terms of execution time than DCLC problem and it appears attractive to convert DCLC into a MCP problem. Based on this, we propose Delay Cost Constrained Routing (DCCR) to rapidly generate a near optimal delay-constrained path in large networks with asymmetric link metrics namely delay and cost. This algorithm first introduces a cost bound according to the network state then it employs the shortest path algorithm [7] with a new non-linear weight function of path delay and cost to search for a path subject to both the requested delay constraint and the cost constraint. The search space is reduced as paths that do not satisfy both constraints are pruned-off. In our algorithm, weight function is designed to give more priority to lowest cost paths, and this algorithm is more suitable to solve DCLC problem. As an improvement, we employ an algorithm ODCR to refine the search space. The complexity of this algorithm is asymptotically in the same order as a regular single metric shortest-path algorithm. We observe by simulation that the cost of the path found by ODCR algorithm is very near to that of the optimal path generated by the much more computationally expensive CBF Algorithm.

II. Problem Description

We represent the network by directed graph G = (V,E) where V is the set of all vertices (nodes), representing routers, E is the set of edges (links) representing physical or logical connectivity between nodes. All links are bidirectional which means that the existence of a link e=(u,v) from node u to node v implies the existence of another link e' =(u,v) for any u,v $\in$ V. Any link e $\in$ E has a cost c(e):E $\rightarrow \Re^+$ and a delay d(e): E$\rightarrow \Re^+$ associated with it, where $\Re^+$ is set of non-negative real numbers. The function c(.) defines the measure we want to constrain. Since computer networks are asymmetrical, it is possible that c (e) $\neq$ c (e') and d (e) $\neq$ d (e').

For a given source node s $\in$ V and destination node d $\in$ V, P (s,d) = P_1 ……..P_m is the set of all possible paths from s to d. The cost and delay of a path P_i is defined as:

Manuscript received 10 June 2008.

P. S. Prakash is with Computer Science and Engineering (PG) Department, Sri Ramakrishna Engineering College, Coimbatore, TamilNadu, India (phone:+91 422 2312021, 99945 25625; e-mail: prakashpsrajan@rediffmail.com)

S. Selvan is with St. Peter's Engineering College, Chennai, India.

$C(P_i)= \sum_{e \in Pi} c(e)$ and $D(P_i) = \sum_{e \in Pi} d(e)$ respectively.

A delay constraint Δ_d is specified by the application as a performance guarantee.

A. Delay Constrained Least Cost Problem (DCLC)

Given a directed network G, a source node s, a destination node d, a non-negative link delay function d(.), a non-negative link cost function c(.).For each link $e \in E$ and a positive delay constraint Δ_d, the constrained minimization problem is to find a path satisfies following conditions:

(i) min $C(P_i)$ and

(ii) $P_i \in P'(s,d)$ if and only if $D(P_i) \le \Delta_d$

Where $P'(s,d) \subseteq P(s,d)$ is the set of paths from 's' to 'd' for which the end-to-end delay is bounded by Δ_d .

The DCLC problem involves optimizing one or more variables and imposing constraints on other variables. A variant called Multi Constraint Path (MCP) problem only searches for a feasible solution for which all variables are bounded by the constraints. A variant of MCP is Delay Cost Constrained (DCC) which can be stated as the DCLC problem except the objective is to find a path $P_i \in P'(s,d)$ if and only if $D(P_i) < \Delta_d$ and $C(P_i) \le \Delta c$ where Δc is the application specified cost bound.

B. ODCR Algorithm

We transform DCLC into a DCC problem by defining a sufficiently loose cost bound so that original DCLC could be easily solved. Now we solve a DCC problem by taking least - delay path is selected as the cost-bound. Now we search for a feasible path of possibly highly delay and lower cost. If such a path exists the algorithm returns that path, else it returns the least-delay path itself. Thus we can convert the original DCLC problem into the problem of searching for near-optimal path in the solution space of this new DCC problem. Now we need to examine the paths that satisfy both the requested delay and introduced cost bound. For this, we define a weight function which combines all features of the link metrics so that by optimizing the weight, we arrive a solution that optimizes all link metrics simultaneously.

Since linear weight functions are *slow convergence* especially if number of paths is more, we now define a non-linear weight function to overcome this difficulty. By using a path weight function $\max[C(P)/\Delta_c, D(P)/\Delta_d]$, the algorithm finds the shortest path with both cost and delay are far from their bounds. With a non-linear weight function, it is now the weight of a path is no longer the sum of the weight of all links on this path. i.e; $W(P) \neq \sum_{e \in p} w(e)$ .But since it is easy to record the cumulative delay and cumulative cost of a path, we can easily solve this problem by computing the path weight as a function of $F(.)$ of the delay and cost of the path. i.e; $W(P)=F[C(P),D(P)]$. In non-linear functions sub sections of least- weight paths are not necessarily shortest path themselves. This is called '*optimal sub-structure property*'. This may result in a shortest path algorithm, may sometimes fail to find least-weight (shortest path).

The weight function used in this algorithm is

$$W(P_i^u) = \begin{cases} D(P_i^u)/1 - C(P_i^u/\Delta_c) \\ \quad \text{if } D(P_i^u) \le \Delta_d \text{ and } C(P_i^u) \le \Delta_c \\ \infty \qquad \text{otherwise} \end{cases}$$

$$\text{i.e; path weight} = \frac{PathDelay}{1 - \frac{PathCost}{CostBound}} \quad (1)$$

Where $p_i^u \in P(s,u)$ is the i^{th} path from source node s to node u found by the algorithm. Since our objective is to find a path with least cost, we are using this weight function that gives priority to low-cost paths. With our definition the path weight has an exponential growth with the path cost, and is only linearly proportional to path delay.

C. Algorithm

Our algorithm adopts greedy strategy and uses a non-linear weight function in searching for best solution. Since non-linear weight function does not have *optimal-substructure property* we first employ k-shortest path algorithm [8]which finds shortest path in increasing weight order, for each node and we can choose the path with lowest cost in the final stage as the best feasible solution.

Our algorithm restricts the search space by only examining paths that satisfy the requested delay bound and cost bound. Here the cost bound is taken to be the cost of the least delay path. This is justifiable since if there is no path with lower cost than that of the least-delay path, then the least-delay path itself is the optimal path and is returned by our algorithm. However this cost bound may be too loose especially when the relationship between cost and delay is inversely proportional to each other. Since the weight of all infeasible paths to be infinity, it is easy to see that if we use a tighter cost bound, the number of possible feasible solution decreases and the opportunity that this algorithm finds the optimal least-cost solution increases.

We use another heuristic, to search for a tighter cost bound, proposed by handler. It uses a linear function of the link delay and cost to compute link weight but it adjusts the weights given to cost and delay in the weight function according to the quality of the current path, thus it iteratively approaches the optimal solution.

The algorithm has two parts namely Least-Delay Path (LDP) and the Least Cost Path (LCP) computed using any shortest-path algorithm with the weight function being link delay and cost respectively. If LDP is a feasible path, then the algorithm returns this feasible path. If it is not feasible, then at

each iteration, the algorithm maintains two paths, the current best feasible path LDP and the current best infeasible path LCP. It then defines two parameters α and β to construct a new linear path weight function W(p)= α x D(p) + β x C(p) for each path p. Using this new linear function of link cost and delay, the algorithm tries to find a new path Least Weight Path (LWP) with least weight so as to reduce both path cost and delay. When W(LWP) < ρ where ρ is current least path weight and LWP is feasible (i.e; D(LWP) ≤ Δ_d), LWP replaces LDP to become the best feasible path, thus the weight given to link cost increases in the next round which means that lower cost paths are given most preference. If LWP is infeasible, LWP replaces LCP in the next iteration, thus the weight given to link delay increases, which means that lower delay paths are given more preference. The algorithm stops when W(LWP) = ρ and returns the best feasible path out of LWP and LDP as the near-optimal solution.

The path found by Handler algorithm is still not the optimal path due to this inherent weakness of the linear weight function. But its cost is close enough to the optimal cost to be efficiently used as a tight cost-bound for DCCR and this enhanced algorithm named Optimized Delay Constrained Routing (ODCR) since using a tighter cost bound is a mechanism to reduce the search space.

III. Algorithm Analysis

A. Validity of ODCR

(i) There exists k such that ODCR always returns a delay constrained path for a given source 's' and destination 'd', if such a path exists.

If no feasible path exists, i.e; the delay of each path that connects 's' and 'd' is greater than the delay bound, then the minimum path weight computed at node 'd' will have a weight of infinity and ODCR returns no path when the search is completed.

We prove by contradiction that ODCR return a path if one or more feasible paths exist. If there are one or more feasible paths, the possible reason for ODCR to return no paths is if the algorithm finds no feasible path leading to an intermediate node along the feasible path from s to d. In other words, let $P_i^d = \{ s, v_1, v_2, \ldots v_m, d\} \in P(s,d)$ be a feasible path from s to d and v_1 to v_m are intermediate nodes, we would have the following two conditions satisfied

$$\exists\, P_i^d \text{ such that } D(P_i^d) \le \Delta d \quad (2)$$

$$\exists\, V_j \in P_i^d \text{ such that } \forall\, P_i^{Vj} \in P\ (s, v_j),\ D(P_i^{Vj}) > \Delta d \quad (3)$$

Since delay is an additive metric and non-negative, it is not possible that the sub-path of a feasible path is not feasible. This shows that ODCR can always find a feasible solution if it exists.

(ii) The final path returned by ODCR for a given source s and destination d is loop-free.

Since the algorithm does not visit dominated paths, a path that contains a loop is never recorded and thus the final k shortest paths recorded at node d are loop-free, and so is the final path returned.

IV. Simulation Model

A discrete-event C++ simulator is used to investigate the performance of different algorithm in a realistic communication environment. We used the graph generation process as in [9] where the position of the nodes are lie in a stipulated area.. We fixed the position of the source node 's' and the destination node 'd' such that 'Manhattan distance' between 's' and 'd' is the longest possible distance in the graph. The average node degree is 3 which is approximately what the situation is in current networks.

The link delay function consists of the propagation delay function T_p, the transmission delay T_t and the queuing delay T_q. Since the network is high speed in nature the transmission delay is negligible. $\tau = T_q/T_p$,the ratio between the queuing delay and propagation delay and this parameter shows the traffic condition in the network. The link delay is defined as $d(e) = (1+ \tau) \times T_p$. We let τ be uniformly distributed in [0,T] where T is maximum queuing delay allowed at each switch. Larger the value of T, the more likely the generated network is asymmetric. Assigning link cost is a challenging job since it affects the difficulty in finding the optimal path. If link and cost are directly proportional to each other, then it is enough to just use a single metric shortest path algorithm.

In our simulation model we consider the negative correlation between cost and delay and we define link cost as $c(e)=\frac{M}{c+d(e)}$ .We choose M=500 and c=1 in our simulation and d(e) varies from 0.1 to15. Since tightness of the delay bound might affect the performance of the algorithms under investigation, we choose the delay bound based on the configuration of the graph. Each time a new graph is generated, a shortest path algorithm is used to find least-delay path and least- cost path, then compute the delay of these two paths, denoted by D(LDP) and D(LCP) respectively. We define the delay bound Δ_d as Δ_d = D(LDP) + ψ[D(LCP)-D(LDP)] where ψ is called *delay bound ratio* which reflects the tightness of the delay bound and lies between 0 and 1.In our simulation we chose ψ = 0.5.

We selected k=3 and m=5 for the network size starting from 50 nodes where, 'k' is the number of shortest paths maintained from the source to each node and 'm' is the number of iterations executed to compute a tight cost bound for our algorithm. Here 'k' and 'm' are much smaller than the network size and such a short value is enough to produce good performance. To measure the inefficiency and speed of the algorithm, CBF is used since it provides the optimal solution in terms of path cost. Thus we define the inaccuracy (i.e; inefficiency) of an algorithm as the cost different relative to the ratio of CBF path.

Inefficiency (A)= $[C(P_A)-C(P_{CBF})]/C(P_{CBF})$. We also measure actual execution time of each investigated algorithm.

V. Simulation Results

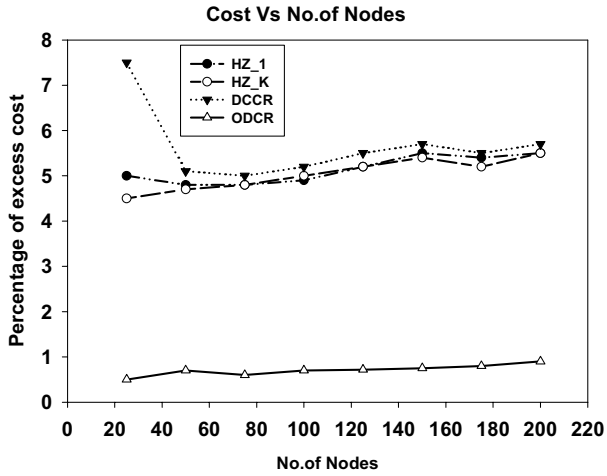

Fig. 1 Percentage of excess cost w.r.t CBF Vs Number of Nodes

Fig. 1 and Fig. 2 show the performance measures of different heuristics. Since link cost and delay are inversely proportional the least-delay path can cost as high as three times that of the optimal path. Our improved ODCR algorithm shows an attractive cost performance; the relative excess cost of ODCR always remain under 1%.This percentage of excessive cost with reference to CBF is shown in Y-axis. We can also see that the relative order and scale of cost difference does not change much with the network size. In our algorithm value of k can be kept small even for a large network.

Fig. 2 shows the data for all algorithms except the CBF algorithm.HZ_1 can converge very fast to the final solution even though an analytical bound does not exist. The speed of DCCR is slightly slower than HZ_1 since DCCR uses a non-linear path weight function and requires a k-shortest path algorithm. The proposed ODCR algorithm runs in almost the same speed as the original DCCR algorithm, which implies a more efficient search under ODCR. We also compared the speed of the optimal CBF solution and ODCR algorithm in Fig. 3. It is clear that the CBF algorithm has an exponential growth with the network size in terms of execution time, as opposed to the polynomial growth of ODCR algorithm. Fig. 4 shows the effect of delay bound on the performance. We can see that the relative excess cost of HZ_1 and DCCR is increasing on the delay bound gets looser.

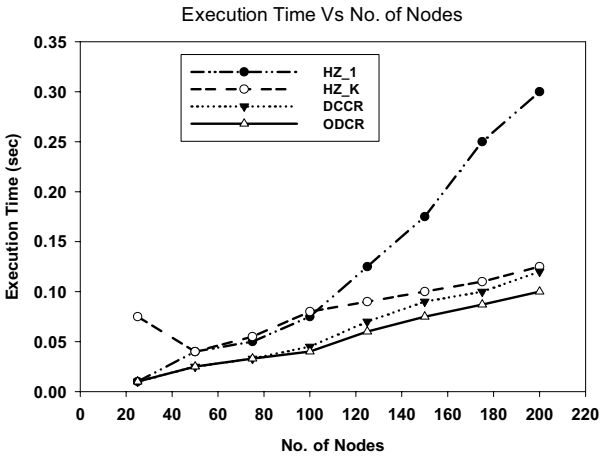

Fig. 2 Execution time Vs Number of Nodes

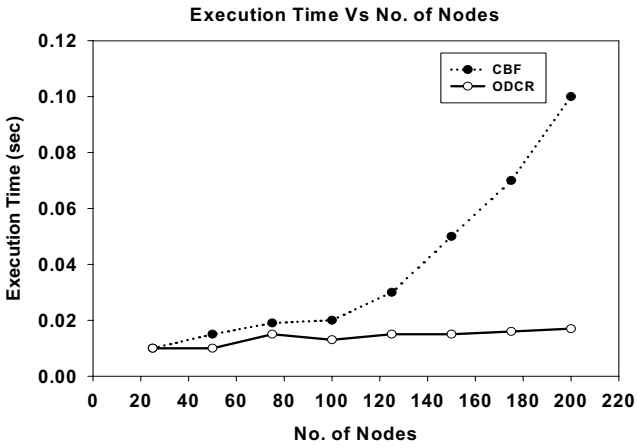

Fig. 3 Execution Time Vs Number of Nodes

This is because a looser bound will enlarge the solution space, thus the capability of these algorithms becomes limited by either the weakness of linear weight function or fixed value of k. However, the performance of ODCR is less sensitive to the delay bound since the cost bound given by HZ_1 heuristic is tight enough to restrict the number of feasible paths. All the above simulations assume that the link cost and link delay are inversely proportional to each other and increases the degree of difficulty in finding the optimal cost.

This ODCR algorithm could be applied in multicast routing protocols to build a low-cost tree. It is very hard to maintain all the time an optimal cost multicast tree since the underlying network is dynamic. One possible solution to this problem is to, whenever a new group member joins or an existing member becomes out-of-bound, add or replace the old path with a new delay- bounded path and there by reducing the cost of this delay bounded path can further reduce the cost of the whole tree.

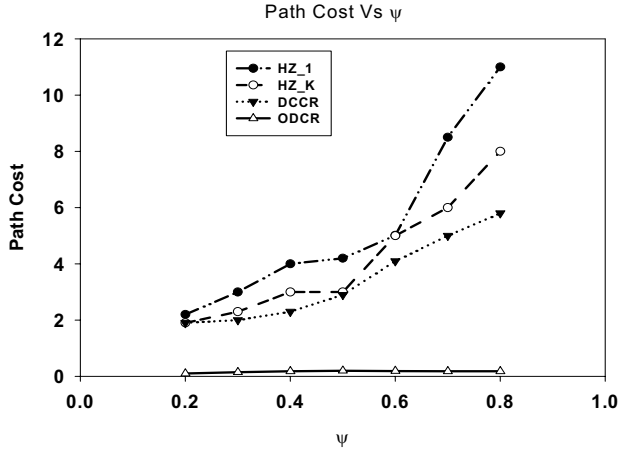

Fig. 4 Percentage of excess cost relative to CBF Vs ψ

VI. Conclusion

We proposed a quick convergence algorithm ODCR is presented in this paper. This algorithm uses a non-linear path weight function to make the path search more accurate and quicker. Results from extensive simulation show that even under the most difficult situation where the link cost and delay are inversely proportional to each other, our improved ODCR algorithm always return very quickly a feasible path whose cost is very close to that of the optimal one, which could only be found using a computationally prohibitive search method.

Appendix

Algorithm.

Routing(G(V,E),s,d,Δ_d, Δ_c, D, C, k)
// Each node has (D, C, W,$\pi_{nd,}$ $\pi_{idx,}$ mark) which is stored in
//ND (u, idx) where π_{nd} points to predecessor node on that path
//and π_{idx} points to the predecessor of that path. A insert
//item has the form (n_id,wgt,idx)
Set $C_{best} \leftarrow \infty$,$P \leftarrow$ nil
InitializeSingleSource(G,s,ND,MH,k)
Insert(MH,(s,0,1))
While MH≠0
  (u, wgt_u,idx_u) ← ExtractMin(MH)
  ND(u, idx_u).mark=EXAMINED
  If u=d
    $C(p) \leftarrow \sum_{l \in p} c(l)$
  $P \leftarrow p \cup P$
  If C(p) < C_{best}
    $C_{best} \leftarrow C(p)$, $p_{best} \leftarrow p$
If u=d and $|P| = k$
Return p_{best}
For each vertex $v \in Adj[u]$
(W(v),D(v),C(v)) ← Compute Weight(u, idx_u,v)
($idx_{u,w}$max) ← FindMax(ND,v)
  if W(v) < w_{max} and path idx_u is not dominated
    ND(v, idx_v) ← (D(v),C(v),W(v),u,id_x, UNEXAMINED)
    i←Search(MH,v, idx_v)
    if i ≠ nil
    Replace(MH, i, (v, W(v), idx_v))
    else Insert (MH, (v, W(v), idx_v))

1) ComputeWeight (u, idx, v)
  D(v) ← ND(u,idx).D +d(u,v),C(v) ← ND(u,idx).C+c(u,v)
  Compute W(v) as in eqn (1)
  Return (W(v), D(v),C(v))
2) FindMax(ND,u)
  Return (idx, ND(u,idx).W)

Acknowledgment

The Authors wish to thank the reviewers whose valuable comments on this paper helped to improve the quality of the paper significantly.

References

[1] Internet traffic Engineering IETF Working Group ftp://ftpext.eng.us.uu.net/tewg
[2] Z.Wang and J Crowcroff "Quality-of-Service routing for supporting Multimedia Applications". IEEE Journal on Select Areas in communication14(7):1188-1234 September 1996.
[3] R.Widyono, "The Design and Evolution of Routing Algorithm for Real-Time channels" Technical Report ICSI, International Computer Science Institute V.C.Berkeley, June 1994.
[4] H.F salama,D.S.Reeves and Y.Viniotis, "A Distributed Algorithm for Delay–Constrained Unicast Routing.In Proc IEEE INFOCOMM '97,April 97.
[5] J.Chen, "New approaches to routing for large scale data networks" ,Ph.D thesis, Department of Computer Science, Rice University,2000.
[6] H.De Nere and Van Mieghan, "A multiple Quality of Service Routing Algorithm for PNNI". IEEE ATM '98, pp 306-314, May 1998.
[7] E.I Chong,S.Maddila and S. Morley " On finding Single source single Destination k shortest paths", International Conference on Computing and information '98. pp 40-47, July 1998.
[8] Y.Lenug, G.Li, Z.B Xu, "A generic algorithm for multiple destination routing problem", IEEE Transaction on Evolutionary Computation 2(4), 2001
[9] Inet topology generator http://topology.eecs.umich.edu.

S. Selvan is serving as Principal at St.Peter's Engineering College, Chennai, India . He has 30 years of teaching experience. He has published more than 75 papers in international, national journals and conference proceedings. His areas of research include, Soft Computing, Computer Networking, Digital Signal Processing.

P. S. Prakash received B.E(EEE) and M.E(Electrical Machines) from P.S.G College of Technology, Coimbatore, India and also received M.E in Computer Science and Engineering from Bharathiar University, India. His research interest include QoS Scheduling, Routing, Network Security and Management. He is currently pursuing Ph.D in Computer Science and Engineering. Presently serving as Asst.Prof at Sri Ramakrishna Engg. College, Coimbatore, India