A Selection Function Based Distributed Algorithm for Delay-Constraint Least-Cost Unicast Routing

Wei Liu Wenjing Lou Yuguang Fang
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611
Email: {liuw@, wjlou@, fang@ece.}ufl.edu

Abstract - It is well-known that Distributed Delay-Constrained Least-Cost (DCLC) unicast routing problem is NP-complete. In this paper we propose an efficient distributed algorithm, namely, selection function based DCLC (SF-DCLC), based on a novel selection function for the DCLC problem. The proposed SF-DCLC algorithm requires limited network state information at each network node and is always able to find a loop-free path satisfying the delay bound if such paths exist. Simulation study shows that the SF-DCLC is not as sensitive to the delay bound and network size as some other DCLC routing algorithms, and attains very low cost-inefficiency (less than 3% to the optimal one) in various network scenarios we simulate. The most attractive feature of SF-DCLC is that SF-DCLC has very high probability to find the optimal solution or a near-optimal solution in polynomial time with low computational complexity and message complexity.

Keywords: DCLC, QoS, Routing, Unicast routing.

I. INTRODUCTION

The emerging distributed real-time multimedia applications have divergent and stringent service requirements, defined as Quality of Service (QoS) metrics in the Service Level Agreement (SLA) between the service provider and the application user. For example, the delay-sensitive applications such as real-time voice and video require the data stream to be received at the destination within certain time. Much work has been done within the Internet Engineering Task Force (IETF) in order to provide the support to such QoS requirements in the current computer networks. Many service models and mechanisms have been proposed, including the integrated service/Resource Reservation Protocol (RSVP) model, the differentiated services (DS) model, Multi-Protocol Label Switching (MPLS), traffic engineering, and QoS routing [1].

QoS routing is one of the most promising mechanisms. The basic function of QoS routing is to find a feasible path, which has sufficient residual (unused) resources to satisfy the QoS requirements of a connection. Here, the QoS requirement is represented as a set of constraints, which can be link constraints, end-to-end path constraints, or tree constrains for the entire multicast tree. The constraints can also be constraints on bandwidth, delay, delay jitter, loss ratio, and so on. In addition, a QoS routing algorithm should also consider the optimization of resource utilization, which is usually measured by an abstract cost metric. The optimization of QoS routing then is to find the lowest-cost path among all the feasible paths [2].

Many QoS routing algorithms have been proposed with a variety of constraints considered. For unicast routing problem, the path-constrained path-optimization (PCPO) and the multi-path-constrained (MPC) problem are the most notorious ones for their NP-complete property [2]. PCPO routing is to find a path, which satisfies the required path constraint and is optimized on another QoS metric, such as cost. An example of PCPO is the delay-constrained least-cost (DCLC) routing, which is to find a least-cost path with bounded delay. MPC routing is to find a path, which satisfies multiple path constraints. An example of MPC is the delay/delay-jitter-constrained routing, which is to find a path with both bounded delay and bounded delay jitter. The two are related, MPC may be simpler than PCPO because MPC dose not optimize on any metric, instead, it only finds a path that meets all the constraints. Due to their NP-complete property, they cannot be solved in polynomial time. Heuristic algorithms have been proposed to find the near-optimal solution.

Much work has been done to solve the MPC problem. Jaffe [3] proposed a pseudo-polynomial heuristic and a polynomial-time heuristic for the MPC problem when the metrics values are in small range. Based on the Dijkstra's algorithm, Wang et al [4] proposed an algorithm to find a path satisfying the bandwidth and delay constraint. Chen et al [5] mapped the unbounded link metrics into bounded integers, then tried to solve the MPC problem with extended the Bellman-Ford (EBF) algorithm or the extended Dijkstra's algorithm (EDSP). Neve et al [6] proposed a non-linear function of link cost and delay to convert the problem into a much easier single-metric routing problem. Meanwhile, the DCLC problem, one of the most famous PCPO problems, has attracted much research attention. In 1994, Widyono [7] proposed a Constrained Bellman-Ford (CBF) algorithm that can solve the DCLC problem optimally. Unfortunately, the worst case running time of the CBF grows exponentially with the network size. Other researchers tried to map the DCLC problem into the possibly easier MPC problem. Guo et al [8] introduced a cost bound according to the network state and then employed the k-shortest path algorithm with a non-linear function of the path delay and cost to search the path that meet the required delay constraint and cost constraint. In [9][10][11], the authors

This work was supported in part by the Office of Naval Research Young Investigator Award under grant N000140210464 and the Office of Naval Research under grant N000140210554.

proposed similar algorithms based on the Lagrange relaxation technique. The basic idea is to first construct an aggregate weight with linear or non-linear function, and then use Dijkstra's algorithm repeatedly to find a feasible path. In [12][13][14], the authors proposed three distributed algorithms in order to alleviate the centralized computation overheads. However, these heuristics for the DCLC problem are either too complex in terms of computation or communication message, or too costly in terms of execution time, or fail to find the optimal solution with reasonable possibility.

In this paper, we propose a more efficient distributed algorithm, namely, SF-DCLC, for DCLC unicast routing. The proposed algorithm makes the use of two vectors, the least delay path vector and the least cost path vector, similar to the ones used in [12][13]. Our new algorithm uses a novel selection function, which is able to lead the heuristics to a satisfying path closer to the optimal one. This algorithm can easily find a loop-free delay-constrained path with only O(|V|) message complexity in the worst case and has very high probability to find the optimal solution if such solution exists.

II. PROPOSED SF-DCLC ROUTING ALGORITHM

A. Description of the DCLC Routing Problem

A network is modeled as a connected, directed graph $G=(V, E)$, where V is the set o f the network nodes and E is the set of edges representing physical or logical connectivity between nodes. Let R^+ denote the set of non-negative real numbers. Two non-negative functions are defined associated with each link e $(e \in E)$: the delay function $\boldsymbol{delay(e):E \rightarrow R^+}$ and the cost function $\boldsymbol{cost(e):E \rightarrow R^+}$. Each link may be asymmetric, that is, the costs and the delays of the link $e=(v_i, v_j)$ and the link $e'=(v_j, v_i)$ may have different values. We also define the non-negative delay and cost functions for any path p as

$$delay(p) = \sum_{e \in p} delay(e) \quad \text{and} \quad \cos t(p) = \sum_{e \in p} \cos t(e)$$

Given a source node $s \in V$, a destination node $d \in V$, and a positive delay constraint Δ, the DCLC routing problem is to find a path p from s to d such that $min\{cost(p), p \in P_d\}$ is achieved, where P_d is the set of all feasible paths from s to d that satisfy the delay constraint Δ, i.e. $delay(p) \leq \Delta$.

It is well-known that the DCLC problem is NP-complete even for undirected networks.

B. Routing Information – the Vectors

The traditional distance vector routing algorithms operate by having each router maintains a table (i.e, a vector), which gives the best known distance to each destination and which outgoing link to use to get there. While in the DCLC routing algorithm, each node maintains two vectors, the least delay vector and the least cost vector, which provide the best known values on two different metrics, delay and cost. Each vector is indexed by, and containing one entry for, each node in the network. One entry in the least delay vector contains the following information (assume at node v_i):

- $\boldsymbol{v_j}$: the destination node identity
- $\boldsymbol{delay}(\boldsymbol{P_{ld}}(v_i, v_j))$: the delay of the least delay path $\boldsymbol{P_{ld}}(v_i, v_j)$;
- $\boldsymbol{cost}(\boldsymbol{P_{ld}}(v_i, v_j))$: the cost of the least delay path $\boldsymbol{P_{ld}}(v_i, v_j)$;
- $\boldsymbol{nid}(\boldsymbol{P_{ld}}(v_i, v_j))$: the next hop on the least delay path $\boldsymbol{P_{ld}}(v_i, v_j)$;

where the least delay path $\boldsymbol{P_{ld}}(s,d)$ is the path from s to d which satisfies $\boldsymbol{delay}(\boldsymbol{P_{ld}}(s,d))=\min\{\boldsymbol{delay}(p), p \in P(s,d)\}$, where $P(s,d)$ is the set of all possible path from s to d.

```
Step 1: (Initially at source node s)
    if (delay(P_ld(s,d)) ≤ Δ)      (1)
      delaySoFar=0; P_sf={s};
      goto step2;
    else
       "the delay-constrained path does not exist"; stop
Step 2: (Upon receiving a PATH_CONSTRUCT message or
at source node s)
    if (this_node≠d)
      if delay(P_lc(this_node,d)) + delaySoFar ≤ Δ      (2)
        delaySoFar = delaySoFar
                  +delay(this_node, nid(P_lc(this_node,d)));
        P_sf= P_sf+ nid(P_lc(this_node,d));
        v= nid(P_lc(this_node,d));
        Send PATH_CONSTRUCT to v;
      else
        for each neighboring node w and w∉ P_sf
          calculate judge(this_node,w);
        end
        v=select(this_node);
        delaySoFar = delaySoFar +delay(this_node,v);
        P_sf= P_sf+v;
        Send PATH_CONSTRUCT to v;
      end
    else
      "path found, P_sf"; stop
```

Figure 1. Pseudo code for the SF-DCLC algorithm

Similarly, the entry in the least cost vector contains the following information:

- $\boldsymbol{v_j}$: the destination node identity
- $\boldsymbol{delay}(\boldsymbol{P_{lc}}(v_i, v_j))$: the delay of the least cost path $\boldsymbol{P_{lc}}(v_i, v_j)$;
- $\boldsymbol{cost}(\boldsymbol{P_{lc}}(v_i, v_j))$: the cost of the least cost path $\boldsymbol{P_{lc}}(v_i, v_j)$;
- $\boldsymbol{nid}(\boldsymbol{P_{lc}}(v_i, v_j))$: the next hop on the least cost path $\boldsymbol{P_{lc}}(v_i, v_j)$;

where the least cost path $\boldsymbol{P_{lc}}(s,d)$ is the path from s to d, which satisfies $\boldsymbol{cost}(\boldsymbol{P_{lc}})=\min\{\boldsymbol{cost}(p), p \in P(s,d)\}$, where $P(s,d)$ is the set of all possible path from s to d.

The least delay vector and the least cost vector are similar to the vector used in the existing distance vector routing protocols. We assume that each node always knows the delay and cost to all its neighboring nodes. Then, the same procedure used to update and maintain the vector in the existing distance vector routing protocol can be used to update and maintain these two vectors. We further assume that the contents of the vectors are up-to-date and the contents of the two vectors do not change during the route setup period.

C. *Operation of SF-DCLC*

The proposed SF-DCLC algorithm constructs the DCLC path node by node from the source node *s* to the destination node *d*. Each node chooses its subsequent node by evaluating a judge function *judge()* on all its neighbors. A special PATH_CONSTRUCT message is sent by the node to its selected subsequent node that requests the continuing construction of the path, till the destination. The PATH_CONSTRUCT message contains the following information {*d*, Δ, *delaySoFar*, $P_{sf}(s,d)$}, where *d* is the destination node identity, Δ is the delay bound, *delaySoFar* is the accumulated delay till the current node, and $P_{sf}(s,d)$ is the set of nodes indicating the found DCLC path from *s* to *d*.

The operation of the algorithm is summarized in Figure 1. Initially, the source node *s* checks if condition (1) is satisfied. If (1) is not satisfied, there is no path exist that meets the given delay constraint from *s* to *d* and the SF-DCLC stops. Further action could be interactive negotiation with the application for looser constraint, which is out of the scope of this paper. If condition (1) is satisfied, that means there should exist at least one or more feasible paths that satisfy the delay constraint. Then the source node *s* proceeds to check condition (2). If (2) is satisfied, the least cost path $P_{lc}(s,d)$ is the optimal path. A PATH_CONSTRUCT message {*d*, Δ, *delay*(*s*,*nid*($P_{lc}(s,d)$)), {*s*}} is sent to node *nid*($P_{lc}(s,d)$) from its least cost vector. If condition (2) is not satisfied, it will compute functions *judge()* and *select()* and based on which, the subsequent node is chosen. Assume that the current node is v_i, for each neighboring node v_j, the function *judge()* is defined as follows,

$$judge(v_i,v_j)=\begin{cases}\dfrac{\cos t(v_i,v_j)+\cos t'(v_j,d)}{\cos t(P_{lc}(v_i,d))} & \\ \qquad delaySoFar+delay(v_i,v_j)+delay(P_{ld}(v_j,d))\le\Delta \\ +\infty \qquad \text{otherwise} & \end{cases}$$

where

$$\cos t'(v_j,d)=\begin{cases}\cos t(P_{lc}(v_j,d)) & \\ \qquad delaySoFar+delay(v_i,v_j)+delay(P_{lc}(v_j,d))\le\Delta \\ \cos t(P_{ld}(v_j,d)) \qquad \text{otherwise} & \end{cases}$$

Then, the function *select()* is to choose the node, say *w*, whose judge function value *judge*(v_i,*w*) is the minimum among all the neighboring nodes. If more than one nodes have the same minimum value, choose the one with the least *delaySoFar* +*delay*(v_i,v_j) +*delay*($P_{lc}(v_j,d)$).

Once the subsequent node has been chosen, a new PATH_CONSTRUCT message is formed and sent to that node. The new *delaySoFar* contained in the new message equals to the old *delaySoFar* plus the delay of link v_i to *w*. The new $P_{sf}(s,d)$ is the old $P_{sf}(s,d)$ concatenated by node *w*.

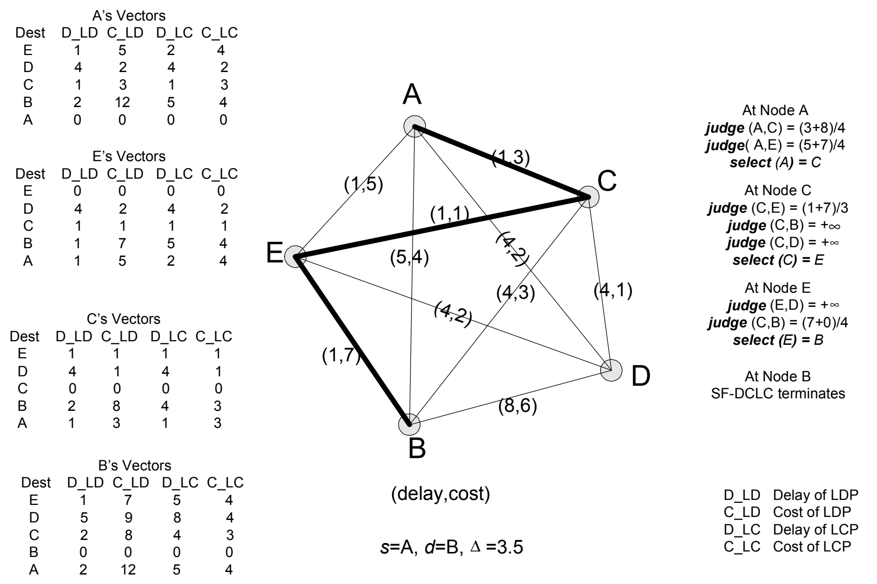
When node different from the destination receives the PATH_CONSTRUCT message, it will repeat the similar procedure in step 2 and send the PATH_CONSTRUCT message to the next hop it selects. When PATH_CONSTRUCT message arrives at the destination d, the algorithm terminates and a feasible path $P_{sf}(s,d)$ has been found. Further action such as resource reservation can be performed.

The messages transmitted during the path finding procedure is one PATH_CONSTRUCT per node (except the destination node). In the worst case, the longest path from the source to the destination contains |*V*| nodes, then the message complexity for the path finding is O(|*V*|), where |*V*| is the number of nodes in the network.

We notice that the proposed SF-DCLC takes advantage of the judge function approach used in [14] by using a better judge function. It is different from [13] in the sense that [13] only uses information from two nodes on the Least-Delay path and Least-Cost path, while our algorithm utilizes all neighboring nodes.

D. *An example*

Figure 2 shows an example of the path constructed by the SF-DCLC algorithm from source *s*=A to destination *d*=B with Δ=3.5. The least delay path from A to B is path (A→E→B), the least cost path from A to B is path (A→ B).The SF-DCLC path P_{sf}(A,B) found is path (A→C→E→B) for Δ=3.5.



A's Vectors

Dest	D_LD	C_LD	D_LC	C_LC
E	1	5	2	4
D	4	2	4	2
C	1	3	1	3
B	2	12	5	4
A	0	0	0	0

E's Vectors

Dest	D_LD	C_LD	D_LC	C_LC
E	0	0	0	0
D	4	2	4	2
C	1	1	1	1
B	1	7	5	4
A	1	5	2	4

C's Vectors

Dest	D_LD	C_LD	D_LC	C_LC
E	1	1	1	1
D	4	1	4	1
C	0	0	0	0
B	2	8	4	3
A	1	3	1	3

B's Vectors

Dest	D_LD	C_LD	D_LC	C_LC
E	1	7	5	4
D	5	9	8	4
C	2	8	4	3
B	0	0	0	0
A	2	12	5	4

At Node A
judge (A,C) = (3+8)/4
judge(A,E) = (5+7)/4
***select** (A)* **=** *C*

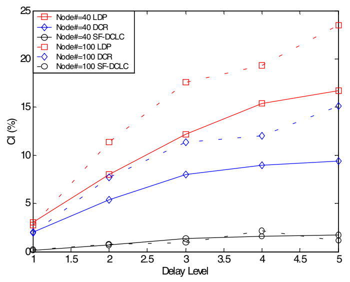
At Node C
judge (C,E) = (1+7)/3
judge (C,B) = +∞
judge (C,D) = +∞
select (C)* = *E

At Node E
judge (E,D) = +∞
judge (C,B) = (7+0)/4
select (E)* = *B

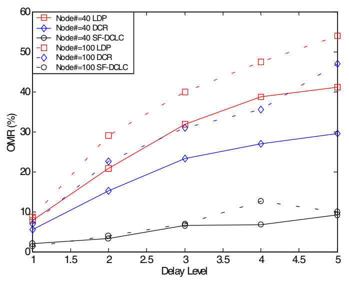
At Node B
SF-DCLC terminates

D_LD	Delay of LDP
C_LD	Cost of LDP
D_LC	Delay of LCP
C_LC	Cost of LCP

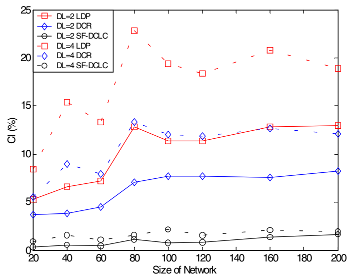
Figure 2. A example of the construction of the path from node A to node B with delay bound of 3.5 using SF-DCLC. Figures along links are (delay, cost) for both direction.



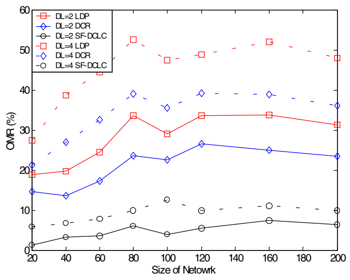
(a) CI vs. delay level
(size of network = 40, 100)



(b) OMR vs. delay level
(size of network = 40, 100)



(c) CI vs. size of network
(delay level (DL) = 2,4)



(d) OMR vs. size of network
(delay level (DL) = 2,4)

Figure 3 Performance comparison

Obviously, our algorithm leads to a better choice: a path with lower cost while meets the delay constraint.

III. PERFORMANCE EVALUATION

A. *Simulation Environment*

In this section we evaluate the performance of the proposed SF-DCLC algorithm by simulation. We develop a unicast routing simulator to carry out the simulation. A random graph generator based on Waxman's generator [15] is implemented to create the networks. In our simulation, the generator can always generate connected graph with average node degree of 4, which is close to the average node degree of the current Internet. Each node has at least 2 outgoing links. The cost value of the link varies from 1 to 8 with uniform distribution. As to the delay in our simulation, we only consider propagation delay, which depends on the distance between the communication nodes. In order to capture the delay characteristics of the national wide network, the delay values of the links are selected from three ranges. 75 percent of the delay values are selected from the first range (1-5 ms), which represents the short local links. 20 percent are selected from (5-8 ms), which represents the long local links. The remaining 5 percent are from (20-30ms), representing the continental links. In our simulation we assume the links are undirected. The simulation is repeated with network size ranging from 20 nodes up to 200 nodes. Δ is randomly selected from a range corresponding to the *delay level* ranging from 1 to 5, where the *delay level* is a new comparison index we introduce in this paper and will be described in the next subsection. For a specific network size, 5 network instances are used, and for each network instance 100 routing requests are generated.

For comparison purpose, we also implement three other algorithms, LDP, CBF, and DCR. The LDP algorithm is used to find the least delay path. As we mentioned in Section 1, when there exists a feasible path, the CBF algorithm can always find the optimal DCLC path from source *s* to destination *d*. The DCR algorithm proposed in [13] is very similar to the DCUR algorithm proposed in [12], while DCR improves the worst-case performance of DCUR by avoiding, instead of detecting and removing, loops. DCR is a well-performed DCLC algorithm, so we also compare the performance of our SF-DCLC algorithm with DCR.

B. *Performance Metrics*

In the literature of the performance comparison of DCLC algorithms [12,13,14], arbitrary Δ value is used in the whole set of networks, regardless the source, destination and the actual delay between them. However, according to the operation of the DCLC operation, when $\Delta < delay(P_{ld}(s,d))$, there is no feasible path. All the algorithms would not be able to find a path satisfying that bound. When $\Delta \geq$ *delay (Plc(s,d))*, CBF, DCR, SF-DCLC should all find the same path $P_{lc}(s,d)$. Only when $delay(P_{ld}(s,d)) \leq \Delta < delay(P_{lc}(s,d))$, there exists one or more feasible paths and it depends on the routing algorithm to find out the optimal feasible path. Thus an arbitrary Δ is not an efficient comparison index because for some network instance, this bound might be too loose or too stringent that fails to reveal the sophistication of the algorithms. Here, we introduce a new comparison index metric, *delay level*, which is related to the actual delay between each source and destination. We divide the range between $[delay(P_{ld}(s,d)), delay(P_{lc}(s,d)))$ into 5 equal length period, each period corresponding to a *delay level* (1,2,3,4,5). Thus the smaller the delay level, the more stringent the bound is. In our simulation, Δ is randomly selected from the period corresponding to the five delay levels. The simulation results do not count the cases where $delay(P_{ld}(s,d)) = delay(P_{lc}(s,d))$.

Since the CBF algorithm can always find the optimal DCLC path from source *s* to destination *d*, the cost of the path found by CBF, $cost(P_{CBF})$, can be viewed as the lower bound of the cost of feasible DCLC paths. On the other hand, the least delay path $P_{ld}(s,d)$ is always a feasible path if the delay bound Δ is appropriately chosen. The cost of $P_{ld}(s,d)$ can be viewed as a sort of upper-bound (although theoretically there should not be a upper-bound) on the cost of feasible DCLC paths if such a path exists. We define the following two performance metrics to compare the proposed algorithm with other algorithms.

- *Cost Inefficiency (CI)*:

$$\delta_A = \frac{\cos t(P_A) - \cos t(P_{CBF})}{\cos t(P_{CBF})}$$

where A represents the algorithm by which the path is found. In our simulation, A could be LDP, DCR or SF-DCLC. This metric is used to evaluate the quality of the paths found.

- *Optimality Miss Ratio (OMR)*: The probability that the path found is not the optimal one, e.g. different from the path found by CBF. This measurement is used to evaluate an algorithm's capability to find the optimal path.

C. *Simulation Results*

First we examine the performance of the routing algorithms on various delay constraint levels. Figure 3(a) shows the cost-inefficiency versus delay level for the cases that the size of the network is 40 and 100 nodes. Figure 3(b) shows the non-optimality versus delay level for the same cases.

We should point out here that all three routing algorithms are capable of finding a feasible path if such a path exists. However, the path found by different algorithms might be different and of different costs. It is observed that, given the network size, the Cost Inefficiency (CI) and the Optimality Miss Ratio (OMR) for P_{LDP} grow faster than that of P_{DCR} and P_{SF} with the increase of delay level. When delay bound is very stringent, the CI and the OMR of the three compared algorithms are very close, because in this case, the number of feasible paths is small, the CBF, DCR, and SF-DCLC are all likely to choose the least delay path P_{ld}, thus the CI and the OMR are very close and low. While when the delay level gets higher, the number of existing feasible paths is larger and the algorithms have better chances to choose different paths, thus have different CI and OMR performance. It is observed that the paths found by the proposed SF-DCLC algorithm is very close to the optimal one at all levels of delay constraints, e.g. less than 3% CI and less than 15% OMR, compared with 15% CI and 47% OMR for DCR algorithm, and 23% CI and 54% OMR for LDP algorithm.

Next we examine the performance of the routing algorithms on networks with various sizes. Figure 3(c) shows the CI versus size of the network for the cases that the delay level (DL) is 2 and 4; Figure 3(d) shows the OMR versus the size of network for the same cases.

It is observed that the performance of all the three algorithms is not sensitive to the network size. The CI and OMR of the proposed SF-DCLC algorithm are relative steady in all sizes of networks, while those of LDP and DCR increase slightly with the increase of the network size. Another observation is that the CI and the OMR of the paths found by SF-DCLC are very low and always lower than the other two algorithms, indicating that the cost of SF-DCLC path is very close to the optimal one and better than paths found by the other two algorithms. Thus the SF-DCLC algorithm has better ability to find better path than the other two.

If we look back figure 2, we will find that the SF-DCLC path found in that example is A$\rightarrow$C$\rightarrow$E$\rightarrow$B with cost 11, while the path found by DCR is A$\rightarrow$E$\rightarrow$B with cost 12.

IV. Conclusions and Future Work

In this paper we study the DCLC problem, which is crucial for the emerging delay sensitive applications. We propose a distributed unicast routing algorithm, namely, SF-DCLC, based on a special selection function. This algorithm is always able to find a loop free path with low computation complexity if such path exists. The message complexity for the path finding is O(|V|), which does not grow exponentially with network size, thus scale well with the increase of the network size. We propose a new comparison index, *delay level*, other than the arbitrary Δ value that is commonly used in other papers on DCLC problem. We also evaluate our algorithm by comparing with DCR, LDP and CBF in terms of path cost and optimality. Our simulation results show that SF-DCLC has much better performance than DCR and LDP. The SF-DCLC is insensitive to network sizes and delay levels. The cost inefficiency of SF-DCLC compared to CBF, the optimal one, is less than 3% with different delay levels and different network sizes. The optimality miss ratio of SF-FCLC is much less than that of DCR and LDP. Thus the most attractive feature of the SF-DCLC algorithm is that, it has very high probability to find the optimal path, as found by CBF.

A possible improvement to SF-DCLC is to modify the selection function to take the delay into consideration. Since our algorithm can always find a delay-constrained path with promising cost, our future work is to extend the algorithm to support multi-path or multicast routing for the delay-sensitive multimedia applications.

Reference

[1] X. Xiao, L. M. Ni, "Internet QoS: A big picture", IEEE Networks, pp. 8-18, March/April 1999

[2] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: problems and solutions", IEEE Networks, pp.64-79, Nov/Dec 1998

[3] J.M. Jaffe, "Algorithms for finding paths with multiple constraints", Networks, 14:95-116, 1984.

[4] Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications", IEEE Journal on Selected Areas in Communications, Vol.14, no.7, pp.1228-1234, September 1996

[5] S. Chen and K. Nahrstedt, "On finding multi-constrained paths", ICC'98, pp.874-879, Atlanta, GA, 1998.

[6] H. De Neve, P. V. Mieghem, "A multiple quality of service routing algorithm for PNNI", Proc. of the ATM Workshop, pp.324-328, May 1998.

[7] R. Widyono, "The design and evaluation of routing algorithms for real-time channels", Technical Report ICSI TR-94-024, International Computer Science Institute, U.C. Berkeley, June 1994.

[8] L. Guo and I. Matta. "Search space reduction in QoS routing", Proc. of 19th International Conference on Distributed Computing Systems (ICDCS'99), June 1999

[9] A. Juttner, B. Szviatovszki, I. Mecs and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem", INFOCOM'2001, Alaska, 2001.

[10] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection", INFOCOM'2001, Alaska, 2001.

[11] G. Feng, C. Doulgeris, K. Makki and N. Pissinou, "Performance evaluation of delay-constrained least-cost routing algorithms based on linear and nonlinear lagrange relaxation", IEEE International Conference on Communications (ICC'2002), pp.2273-2278, New York, April 2002

[12] H. F. Salama, D. S. Reeves, Y. Viniotis, "A distributed algorithm for delay constrained unicast routing", INFOCOM'1997, Kobe, Japan, April 1997.

[13] Q. Sun, H. Langendorfer, "A new distributed routing algorithm for delay-sensitive application", Computer Communications, vol.21, no.6, May 1998

[14] Z. Wang, B. Shi, L. Zou, "A distributed algorithm on delay-constrained least-cost unicast routing", Proc. of International Conference on Communication Technology (ICCT2000), Beijing, China, 2000

[15] B. M. Waxman, "Routing of multipoint connections," IEEE Journal on Selected Areas in Communications, 6(9):1617-1622, Dec. 1988