A New Distributed Routing Algorithm for Supporting Delay-Sensitive Applications

Quan Sun   Horst Langendörfer

Institute of Operating Systems and Computer Networks

Technical University of Braunschweig

Bültenweg 74/75, 38106 Braunschweig, Germany

Abstract

We propose a new distributed delay-constrained unicast routing algorithm which can always find a delay-constrained path with small message complexity if such a path exists. At each network node, limited information about the network state is needed and only a small amount of computation is required. Simulation results show that the proposed algorithm has much better cost performance than the least delay path algorithm.

Keywords: Routing, Distributed Algorithm, Delay Constraint, Cost

1 Introduction

The appearance of many new real-time network applications, e.g. multimedia applications, has great influences on the network routing algorithms. These new applications usually have stringent end-to-end delay requirement and consume a lot of network resources. The routing function in traditional computer networks like finding paths with least end-to-end delay or hop count can not meet the new situation. At current stage of network evolution, it may be appropriate and important to design a simple, scalable routing algorithm that satisfies a given end-to-end delay bound *and* manages the network resources efficiently, particularly if the designed algorithm can be easily integrated into the current Internet routing protocols. The routing algorithms used in current Internet routing protocols are mainly based on shortest path algorithms [1]. Shortest path algorithm can only find a least delay path or find a path with most available resources. Recently, Wang and Crowcroft [3] proposed a shortest-widest path algorithm which chooses bandwidth and propagation delay as the routing metrics and finds the smallest propagation delay path among all widest paths. Unfortunately, the paths computed by this algorithm may not satisfy a given end-to-end delay constraint (see section 4). The existing delay-constrained routing algorithms are designed for multicast traffic (e.g. [4] [5] [6] [7] [8]) and many of them are very complex. Most recently, Salama *et al* proposed a particularly interesting algorithm - the distributed delay-constrained unicast routing algorithm (DCUR) [10]. DCUR is relatively simple and efficient. However, DCUR has a very large worse case communication complexity (the message complexity of DCUR is $O(|V|^3)$ where $|V|$ is the number of nodes in the network).

In this paper, we propose a new efficient distributed unicast routing algorithm which can easily compute a loop-free delay-constrained path with only $O(|V|)$ message complexity and it requires only a small amount of computation at any node. The proposed algorithm can also easily be integrated into the distance-vector based routing protocol. The paper is organized as follows. The problem definition is given in section 2. Section 3 gives a detailed description of the proposed algorithm. The simulation results are presented in section 4. Section 5 gives a short summary and our future work.

2 Network Model and Problem Definition

Our network model and problem definition is similar like that described in [10]. A network is modeled as a connected, directed graph *G=(V,E)* where V is the set of network nodes and E is the set of links. Two non-negative real value functions are associated with each link e ($e \in E$): delay *D(e)* and cost *C(e)*. The link delay *D(e)* is the delay a data packet experiences on the corresponding link, and the link cost *C(e)* is a measure of the utilization of the corresponding link's resources. Links are asymmetrical, namely the cost and delay for the link $e = (v_i, v_j)$ (from node v_i to node v_j) and the link $e' = (v_j, v_i)$ (from node v_j to node v_i) may not be the same.

Given a source node s and a destination node d, a set of links $e_1 = (s, v_2)$,$e_2 = (v_2, v_3)$, ..., $e_k = (v_k, d)$ constitute a directed path *P(s,d)* from s to d. The cost of the path *P(s,d)* is defined as follows:

$$C(P(s,d)) = \sum_{e \in P(s,d)} C(e) \qquad (1)$$

Similarly, the end-to-end delay of the path *P(s,d)* is defined as follows:

$$D(P(s,d)) = \sum_{e \in P(s,d)} D(e) \qquad (2)$$

We study the Delay-Constrained Least-Cost path problem (DCLC). This problem can be formally expressed as follows.

Problem (DCLC) For a directed network *G=(V,E)*, given a link delay function $D : E \rightarrow \Re^{+}$, a link cost function $C : E \rightarrow \Re^{+}$, a source node s, a destination node d and a delay constraint $\triangle$, find a directed path *P(s,d)*, such that

$$D(P(s,d)) < \triangle \qquad (3)$$

and $C(P(s,d))$ is minimized.

The DCLC problem is *NP*-complete [3]. An optimal solution for this problem was proposed by Widyono [6], the proposed algorithm (called constrained Bellman-Ford (CBF) algorithm) is a centralized one requiring complete information about the network topology at every network node. The running time of CBF grows exponentially with the network size.

Obviously, we have chosen *cost* and *delay* as the routing metrics, both metrics are additive. The delay metric is essential for applications with stringent end-to-end delay requirement, and the cost metric is usually used for efficiently managing the network resources. The cost metric has been widely studied in the past, especially for connection-oriented networks. For example, in [11] each link in the network is assigned a cost that is exponential in its utilization, the work in [12] shows that network resources can be efficiently managed using such link cost function. In [6], the link cost is chosen to be a function of residual bandwidth, residual buffer space and estimated delay bound of the link, and it is shown that a link cost of being inversely proportional to the link residual bandwidth can lead to good performance regarding the number of destinations successfully established in the multicast simulation environment in [6].

3 The Proposed Algorithm

3.1 Routing information

We first discuss the routing information which needs to be available for the proposed algorithm to compute the delay-constrained path. Each node in the network should know the delays of all outgoing links and must maintain the following information: a delay vector and a cost vector.

The delay vector at node $v_i \in V$ consists of $|V| - 1$ entries, one entry for each other network node. The entry for node $v_j \in V$ $(v_j \neq v_i)$ contains the following items:

- The destination node identity: v_j

- The end-to-end delay of the least delay path $P_{ld}(v_i, v_j)$ from v_i to v_j: $D(P_{ld}(v_i, v_j))$
- The next hop node on the least delay path $P_{ld}(v_i, v_j)$ from v_i to v_j: $id(P_{ld}(v_i, v_j))$

The cost vector at node $v_i \in V$ also consists of $|V| - 1$ entries, one entry for each other network node. The entry for node $v_j \in V$ ($v_j \neq v_i$) contains the following items:

- The destination node identity: v_j
- The end-to-end delay of the least cost path $P_{lc}(v_i, v_j)$ from v_i to v_j: $D(P_{lc}(v_i, v_j))$
- The next hop node on the least cost path $P_{lc}(v_i, v_j)$ from v_i to v_j: $id(P_{lc}(v_i, v_j))$

The above delay vectors and cost vectors[1] are both similar to the distance vectors of the existing distance vector based routing protocol [2]. The same procedures for maintaining the distance vectors can be used for maintaining the delay vectors and the cost vectors[2]. Because our focus is on the routing algorithm, we will not discuss these procedures in this paper. In the following, we assume that the contents of the delay vectors and the cost vectors do not change during the execution of the routing algorithm.

3.2 The proposed distributed algorithm

Our proposed distributed delay-constrained routing algorithm, called DCR, constructs the path one node at a time, from the source node s to the destination node d. Figure 1 shows the procedure executed by the source node s. Initially, the source node s checks

$$D(P_{ld}(s, d)) < \triangle \qquad (4)$$

If (4) is not satisfied, there exists no path that can satisfy the given delay constraint from s to d and DCR stops; otherwise the source node s further checks

$$D(P_{lc}(s, d)) < \triangle \qquad (5)$$

[1] In [8] and [10], similar routing information is also available at each network node.

[2] Note that the procedures for maintaining the cost vectors should give the end-to-end delay (instead of the cost) of the least cost path.

```
if D(P_ld(s,d)) < △
    if D(P_lc(s,d)) < △
        next_node ← id(P_lc(s,d));
        path_direction ←LC;
    else
        next_node ← id(P_ld(s,d));
        path_direction ← LD;
        delay_so_far ← D(s, id(P_ld(s,d)));
    end if
    if path_direction = LC
        Path_construction ← (d, path_direction);
    else
        Path_construction ← (d, path_direction, △, delay_so_far);
    end if
    send the Path_construction message to next_node;
else
    exit with the message "No delay-constrained path";
end if
```

Figure 1: The procedure executed by the source node s

if (5) is satisfied, s reads the next hop node on the least cost path towards d, namely $id(P_{lc}(s,d))$, from its cost vector and sets:

- $next\_node = id(P_{lc}(s,d))$
- $path\_direction = LC$

otherwise, s reads the next hop node on the least delay path towards d, namely $id(P_{ld}(s,d))$, from its delay vector and sets:

- $next\_node = id(P_{ld}(s,d))$
- $path\_direction = LD$

if $v \neq d$

if *path_direction* in the received *Path_construction* message = LC or

delay_so_far in the received *Path_construction* message $+ D(P_{lc}(v,d)) < \triangle$

$next\_node \leftarrow id(P_{lc}(v,d))$;

$path\_direction \leftarrow LC$;

else

$next\_node \leftarrow id(P_{ld}(v,d))$;

$path\_direction \leftarrow LD$;

$delay\_so\_far \leftarrow delay\_so\_far + D(v, id(P_{ld}(v,d)))$;

end if

if $path\_direction = LC$

$Path\_construction \leftarrow (d, path\_direction)$;

else

$Path\_construction \leftarrow (d, path\_direction, \triangle, delay\_so\_far)$;

end if

send the *Path_construction* message to *next_node*;

end if

Figure 2: The procedure executed by node v having received a *Path_construction* message

- *delay_so_far* = $\mathrm{D}(s, id(P_{ld}(s,d)))$

where $\mathrm{D}(s, id(P_{ld}(s,d)))$ is the delay of the link $(s, id(P_{ld}(s,d)))$.

The source node s then constructs a *Path_construction* message and sends it to *next_node*. The *Path_construction* message contains the destination node d and the *path_direction*. If *path_direction* = *LD*, the *Path_Construction* message contains additionally the given end-to-end delay constraint $\triangle$ and *delay_so_far*, which represents the delay of the already constructed path from s to *next_node*.

When a node $v \neq d$ receives a *Path_construct* message, it executes the procedure shown in figure 2. Node v first looks up the *path_direction* in the received message. If *path_direction* is LC, v reads the next hop node on the least cost path towards d, namely $id(P_{lc}(v,d))$, from its cost vector and sets:

- $next\_node = \ id(P_{lc}(v,d))$
- $path\_direction = LC$

otherwise, v further checks

$$delay\_so\_far + D(P_{lc}(v,d)) < \triangle \qquad (6)$$

if (6) is satisfied, v sets

- $next\_node = id(P_{lc}(v,d))$
- $path\_direction = LC$

otherwise v sets

- $next\_node = id(P_{ld}(v,d))$
- $path\_direction = LD$
- $delay\_so\_far = delay\_so\_far + D(v, id(P_{ld}(v,d)))$

Node v then constructs a *Path_construction* message and sends it to *next_node*. Similarly, the *Path_construction* message contains destination node d and *path_direction*. If *path_direction* $= LD$, the *Path_Construction* message contains also the delay constraint $\triangle$ and *delay_so_far*.

If the destination node d receives the *Path_Construction*, it means that the delay-constrained path has been successfully constructed. The destination node d may send an acknowledgment back to the source s.

Figure 3 shows an example of how DCR constructs the path from the source node A to the destination node E with $\triangle = 5$. The least cost path from A to E is $P_{lc}(A,E) = \{(A,B), (B,D), (D,E)\}$ with the delay $D(P_{lc}(A,E)) = 2+1+2 = 5$. The least delay path from A to E is $P_{ld}(A,E) = \{(A,C), (C,E)\}$ with the delay $D(P_{ld}(A,E)) = 1+1 = 2 < \triangle$. Because $D(P_{lc}(A,E)) < \triangle$ is not satisfied, the source node A chooses the direction of

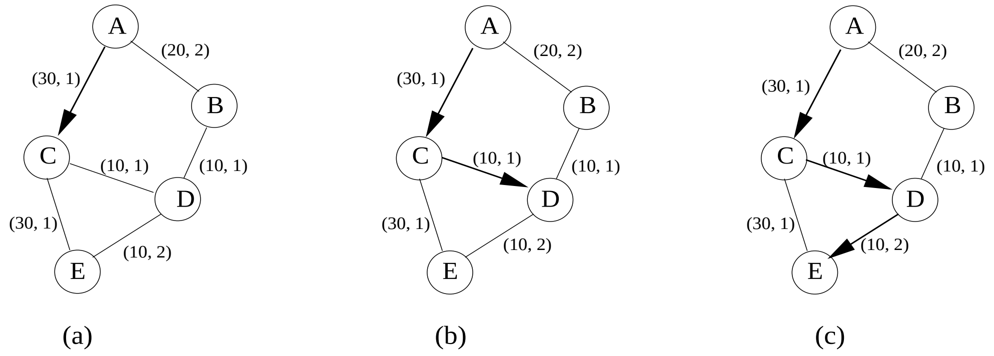


Figure 3: An example of the construction of the path from node A to node E with a delay bound of 5 using DCR. Figures along links are (cost, delay) for both directions.

the least delay path from A to E and sends a *Path_Construction* message to node C. The *Path_Construction* message contains destination node E, delay bound $\triangle = 5$, *delay_so_far* *D(A,C)=1* and *path_direction* = LD. The partially constructed path contains the link *(A,C)* (see Fig.3(a)). When the *Path_Construction* message reaches node C, because the *path_direction*=LD, node C first looks up the delay value of the least cost path from C to E from its cost vector. The least cost path from C to E is $P_{lc}(C,E) = \{(C,D)),(D,E)\}$ with the delay $D(P_{lc}(C,E)) = 1 + 2 = 3$. Because $D(P_{lc}(C,E))$ + *delay_so_far* = 3 + 1 =4 $< \triangle$ (namely (6) is satisfied), node C chooses the direction of least cost path from C to E. It sends a *Path_Construction* message to node D (the next hop node on the least cost path from C to E). The *Path_Construction* contains only destination node E and *path_direction*= LC. For now, the constructed path consists of link *(A,C)* and link *(C,D)* (see Fig.3(b)). When the *Path_Construction* message reaches node D, because *path_direction*= LC, node D simply chooses the direction of least cost path from D to the destination node E and sends a *Path_Construction* message to the next hop node on the least cost path from D to E, namely node E. The *Path_Construction* message contains destination node E and *path_direction*= LC. When node E receives the *Path_Construction* message, the path construction process ends successfully. The final constructed path from A to E is shown in Fig.3(c).

The shortest path algorithm in the existing distance-vector routing protocol [2] guarantees to find a loop-free path if the contents of the distance vectors at all nodes are up-to-date and when the network is stable. Under similar conditions, namely if the contents of the delay vectors and the cost vectors at all nodes are up-to-date and do not change during the path construction process, we have also

Theorem 1: The paths constructed by DCR are loop-free.

Proof: For a given source node s, destination node d and a delay constraint $\triangle$, if the end-to-end delay of the least cost path $P_{lc}(s,d)$ is less than $\triangle$, the path construction process will begin with the least cost path from s to d and $P_{lc}(s,d)$ will be the final constructed path, which is clearly loop-free. If the end-to-end delay of the least delay path $P_{ld}(s,d)$ is equal to or larger than $\triangle$, then no path satisfying the given delay constraint exists; otherwise the path construction process will begin with the least delay path from s to d. If all the last nodes of the already (partially) constructed paths choose the direction of least delay path, the final constructed path will be the least delay path from s to d, which is also loop-free. However, once the last node of a partially constructed path chooses the direction of the least cost path, the direction of the least cost path will be always followed until the constructed path reaches the destination node d. Fig.4 shows a scenario of this situation.

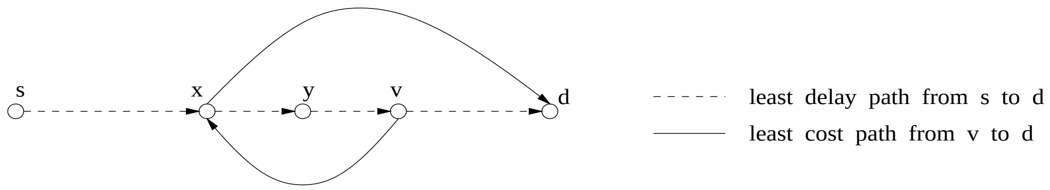


Figure 4: A loop involving node v and node x

In Fig.4 the already constructed path P_e including nodes s, x, y and v is a part of the least delay path from s to d and v is the last node of this constructed path. If node v now follows the direction of least cost path from v to d, then this least cost path will be always followed until d is reached. A loop could exist only if at least one of the nodes on the least cost path from v to d is also a node on the already constructed path P_e. Without loss of generality, let x be such a node (see Fig.4). Because x is a node on the path P_e and it has chosen the direction of the least delay path from x to d, this means

$$D(P(s,x)) + D(P_{lc}(x,d)) \geq \triangle \quad (7)$$

where $D(P(s,x))$ is the delay of the path from s to x along the path P_e and $D(P_{lc}(x,d))$ is the delay of the least cost path from x to d. Because x is also a node along the least cost path from v to d, we have

$$D(P_e) + D(P_{lc}(v,x)) + D(P_{lc}(x,d)) < \triangle \quad (8)$$

where $D(P_e)$ is the delay of the path P_e and $D(P_{lc}(v,x))$ is the delay from v to x along the least cost path from v to d. Because $D(P(s,x)) < D(P_e)$, (7) and (8) contradict each other. Thus, P_e and $P_{lc}(v,d)$ have no nodes in common. The proof completes. □

Theorem 2: DCR can always find a path from a source node s to a destination node d satisfying a given delay constraint if such a path exists.

Proof: If there exists a path from s to d satisfying the given end-to-end delay constraint, the source node s will send a *Path_Construction* message to *next_node*, where *next_node* is the next hop node either on the least cost path from s to d or on the least delay path from s to d. Because we assume that the routing information maintained at all nodes does not change during the path construction process, the *next_node* must itself be able to send out a *Path_Construction* message while assuring that the partially constructed path satisfies the given delay constraint. The same process will continue until reaching the destination node d. □

Theorem 3: The worst case message complexity of DCR is O($|V|$), where $|V|$ is the number of network nodes.

Proof: Because DCR guarantees to find a loop-free path and the longest possible loop-free path in the network consists of $|V|$ nodes and constructing such a path requires only

$|V| - 1$ *Path_Construction* messages, thus the worst case message complexity of DCR is $O(|V|)$. $\Box$

It is also easy to see that a node only requires a small fixed amount of computation to process a *Path_Construction* message, thus the computational complexity of DCR at any node is O(1) time.

Theorem 4: If $C(e) = D(e)$ for all $e \in E$ (see section 2), the paths computed by DCR are least cost paths.

Proof If $C(e) = D(e)$ for all $e \in E$, the content of the cost vector and the content of the delay vector at each network node are the same. In this case, if a delay-constrained path between two given nodes exists, the path construction process will always follow the direction of the least cost path (see Fig.1 and Fig.2). $\Box$

Theorem 4 shows that DCR essentially becomes a shortest path algorithm if $C(e) = D(e)$ for all $e \in E$.

4 Cost Performance Studies

We used simulation experiments to evaluate the cost performance of DCR. The multicast routing simulation environment described in [9] was modified to evaluate and compare the performances of DCR, the least delay path algorithm (called LDP), the shortest-widest path algorithm (called SWP) [3] and the centralized optimal delay-constrained algorithm (CBF) [6]. In the simulation, full duplex ATM networks with homogeneous link capacities of 155 Mbps (OC3) were used. The positions of the network nodes were kept in a rectangle of size 3000 * 2400 Km^2, roughly the area of the USA. A random graph generator based on Waxman's generator [13] was used to create networks. The generator first creates a list of nodes (at random locations) and then creates links between these nodes. The probability of a link to exist between two nodes u and v is given by

$$P(u, v) = \beta exp^{\frac{-l(u,v)}{L\alpha}} \qquad (9)$$

where $l(u,v)$ is the distance between u and v, and L is the maximum distance between any two nodes, α and β are parameters between (0,1]. A large value of α increases the ratio of the number of long links to short links, and a large β increases the average node degree of the network. The simulated networks were always connected and had an average node degree of 4 through adjusting the parameters.

Each node in the created network represented a non-blocking ATM switch. Each link was assumed to have a small output buffer. The propagation speed through the links was taken to be two thirds the speed of light. The propagation delay was dominant under these conditions, and the queuing component was neglected when calculating the link delay $D(e)$. In the simulation, the link cost $C(e)$ was set to be the total currently reserved bandwidth on the link, which is a suitable measure of the utilization of both the link's bandwidth and its buffer space. For more information about the simulation environment, please see [9].

For each run of the experiment we generated a random network, a random source node and a random destination node. Random background traffic for each link was also generated. The equivalent bandwidth of each link's background traffic is a random variable uniformly distributed between 5 Mbps and 125 Mbps. The routing algorithm was applied to generate a path from the source node to the destination node satisfying a given end-to-end delay constraint. The experiment was run repeatedly until confidence intervals of less than 5% using 95% confidence level were achieved, unless otherwise specified, for all the following measures: the average path cost, the average end-to-end path delay and the success rate of the algorithm (namely the frequency of successfully finding a path satisfying the given delay bound).

We have integrated both DCR and a centralized version of the shortest-widest path algorithm [3] into the simulation environment mentioned above. The original simulation environment contains the least delay path algorithm (LDP) and the optimal delay-constrained routing algorithm (CBF). The delay-constrained unicast routing algorithm (DCUR) proposed by Salama *et al* [10] was not included in our simulation experiments, because Salama *et al* have used the similar simulation environment to evaluate their proposed algorithm and the simulation results are available in [10]. The cost performance of DCUR is up to 10% worse than that of the optimal CBF algorithm for networks of 200 nodes [10]. We will see in the following that DCR has a similar cost performance like DCUR.

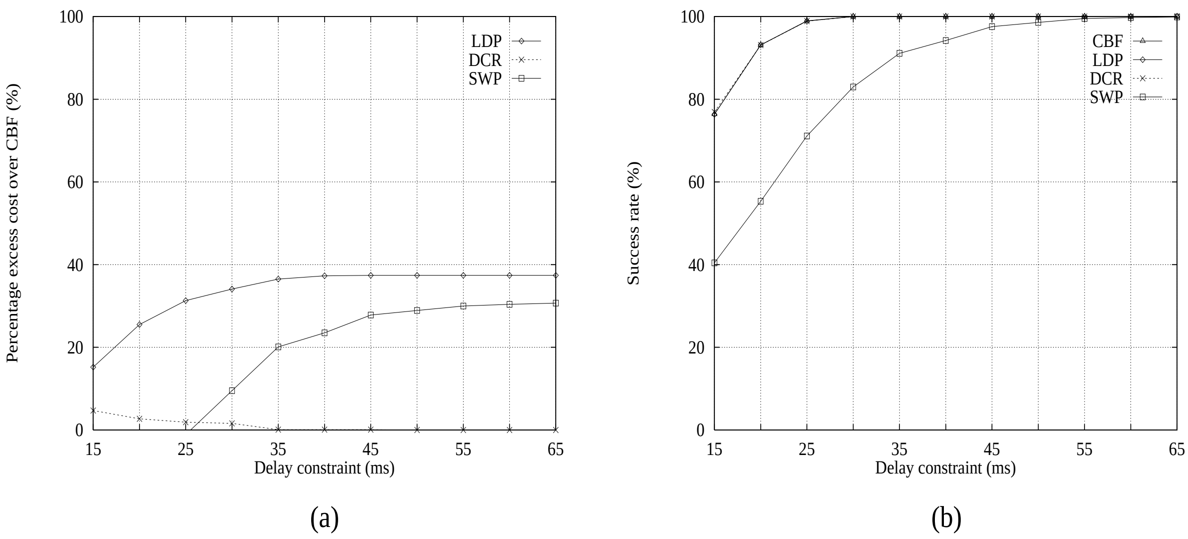
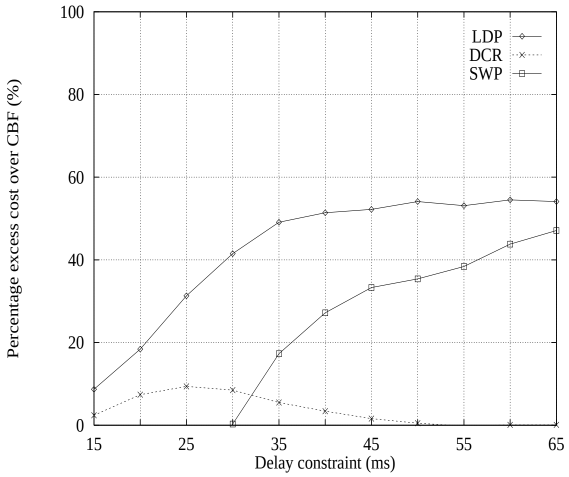
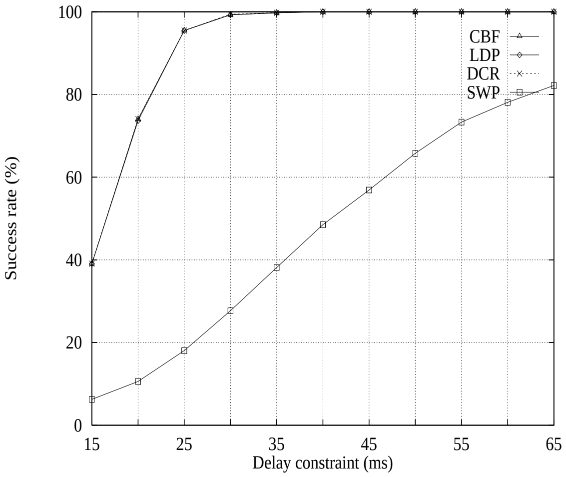


Figure 5: (a) percentage excess cost over CBF and (b) success rate vs. delay constraint for 20-node networks with average node degree of 4

Figure 5 and Figure 6 show the percentage excess (path) cost over CBF and success rate versus the delay constraint for networks of 20 nodes and 200 nodes respectively. Figure 7 shows the average end-to-end delay versus the delay constraint for networks with 200 nodes. All networks have an average node degree of 4. Note that in Fig.6(b) the confidence interval for the success rate of SWP is less than 10% using 95% confidence level when the delay constraint is less than 35 ms (over 4,000 networks have been simulated). From the simulation results we can draw the following conclusions. For the small network size of 20 nodes (see Fig.5), the cost performance of DCR is close to that of the optimal algorithm



(a)



(b)

Figure 6: (a) percentage excess cost over CBF and (b) success rate vs. delay constraint for 200-node networks with average node degree of 4

CBF, while both LDP and SWP have a much worse cost performance than CBF (over 25% worse than CBF when the delay constraint is larger than 45 ms). For the large network size of 200 nodes (see Fig.6), the path cost of DCR is always within 10% larger than that of the optimal CBF, while LDP and SWP are up to 54% and 47% worse than CBF respectively. Because CBF, LDP and DCR are all capable of constructing a path satisfying the given delay constraint if one exists, all of these three algorithms have the same success rate and they achieve 100% success rate if the delay constraint is greater than 30 ms in Fig.5 and Fig.6. The SWP algorithm, however, has a much smaller success rate than CBF, LDP and DCR, especially for large networks. For example, the success rate of SWP in Fig.6 is less than 40% for the delay constraint of 35 ms. This observation shows that SWP may not be suitable for use in large networks and for applications with stringent end-to-end delay requirement. As expected, the average end-to-end delay of LDP is smaller than both DCR and CBF, while SWP leads to the largest average end-to-end delay (see Fig.7).

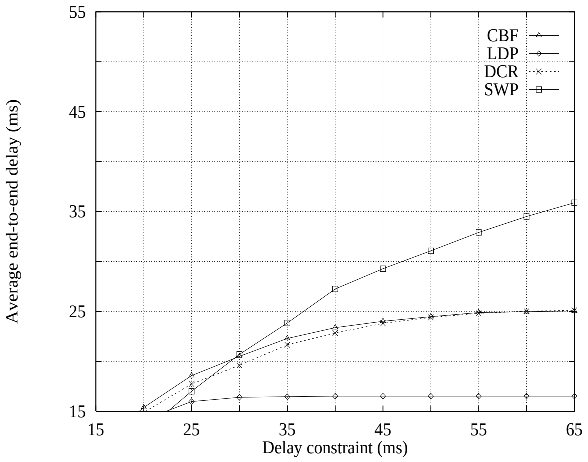


Figure 7: Average end-to-end delay vs. delay constraint for 200-node networks with average node degree of 4

5 Summary and Future Work

In this paper, we have presented a distributed delay-constrained unicast routing algorithm. The route computation is based on the delay vector and the cost vector maintained at each network node. The proposed algorithm can always find a loop-free delay-constrained path with small message complexity if such a path exists. The amount of computation needed at any node is also very small. Simulation results show that the cost performance of this algorithm is very satisfactory.

There are a few areas for further study:

(1) How to choose the link cost function in datagram networks such as Internet?

(2) In section 3, we have assumed that the contents of the delay vectors and cost vectors do not change during the execution of the routing algorithm. Future work should study mechanisms that enable the proposed algorithm to cope with dynamic network environment where the contents of the delay vectors and cost vectors maintained at different nodes are not consistent.

(3) A simple, scalable and efficient delay-constrained unicast routing algorithm is useful not only for delay-sensitive applications involving two participants. In fact, multicast routing algorithms for delay-sensitive applications involving dynamic multiple participants could benefit from it. We are also currently designing a distributed, dynamic multicast routing algorithm for real-time traffic based on the proposed delay-constrained unicast routing algorithm.

References

[1] Bertsekas, D. and Gallager, R., Data Networks, 2nd ed., Englewood Cliffs: Prentice Hall, 1992

[2] Hedrick, C, Routing Information Protocol, Internet RFC 1058, June 1988

[3] Wang, Z. and Crowcroft, J., Quality-of-Service Routing for Supporting Multimedia Applications, *IEEE Journal on Selected Areas in Communications*, 14(7), 1996

[4] Kompella, V.P., et al., Multicast Routing for Multimedia Communication, *IEEE/ACM Transactions on Networking*, 1(3),pp.286-292, 1993

[5] Kompella, V.P., et al., Two Distributed Algorithms for the Constrained Steiner Tree Problem", in *Proceedings of the Second International Conference on Computer Communications and Networking*, pp.343-349, 1993,

[6] Widyono, R., The Design and Evaluation of Routing Algorithms for Real-Time Channels. Technical Report TR-94-024, Tenet Group, Department of EECS, University of California at Berkeley, 1994

[7] Zhu, Q., et al, A Source-Based Algorithm for Near-Optimum Delay-Constrained Multicasting, *IEEE INFOCOM'95*, pp.377-385, 1995

[8] Wi, S. and Choi, Y., A Delay Constrained Distributed Multicast Routing Algorithm, *12th International Conference on Computer Communication, ICCC'95*, pp.833-838, 1995

[9] Salama, H.F., et al, Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks, in *High Performance Networking VI, IFIP 6th International Conference on High Performance Networking*, pp.27-42, 1995

[10] Salama, H.F., Reeves, D.S. and Viniotis, Y., A Distributed Algorithm for Delay-Constrained Unicast Routing, *IEEE INFOCOM'97*, pp.84-91, 1997

[11] Plotkin, S., Competitive Routing of Virtual Circuits in ATM Networks, *IEEE Journal of Selected Areas in Communications*, 13(6), pp.1128-1136, 1995

[12] Gawlick, R, et al, Routing and Admission Control in General Topology Networks, Technical Report STAN-CS-TR-95-1548, Stanford University, Stanford, CA, 1995

[13] Waxman, B.M., Routing of Multipoint Connections, *IEEE Journal on Selected Areas in Communications*, 6(9),pp.1617-1622, 1988