PROPOSAL ALGORITHMS TO SOLVE DELAY-CONSTRAINED ROUTING PROBLEM

Deya J. Kadhim
Electrical Engineering Department
University of Baghdad

ABSTRACT

One of the Quality of Service (QoS) routing problems is link-constrained, path constrained routing, and example of such problem is delay and bandwidth-constrained routing. Here, the proposed solution is presented through our proposal routing algorithms DCUR-AB (Delay Constraint Unicast Routing at Available Bandwidth) and ADCUR-AB (Adaptive Delay Constraint Unicast Routing at Available Bandwidth).

DCUR-AB and ADCUR-AB routing algorithms are proposed to solve the most popular QoS problem (Delay-constrained routing problem). Our proposed algorithms solve this problem in addition to take into account one of the important network resource (bandwidth).

From experimental results for DCUR-AB it is found that this algorithm has a limitation searching to find the optimal path with taking into account the bandwidth constrained, therefore; it is proposed here another routing algorithm called ADCUR-AB that has a wide search to find the optimal path, then a practical comparison between the two proposed algorithms results is applied and it is found that the second algorithm is better than the first one.

الخلاصة

أحدى مشاكل تحديد المسار في نوعية الخدمة (QoS) هي مشكلة تحديد الاتصال-تحديد المسار ومثال على هكذا مشاكل هي مشكلة تحديد التأخير(Delay Constrained)-تحديد الحزمة (Bandwidth Constrained). هنا تم أقتراح خوارزميتين لحل هذه المشكلة وهما خوارزمية تحديد المسار الاحادي مع تحديد التأخير بوجود الحزمة(DCUR-AB) وخوارزمية تحديد المسار الاحادي مع تحديد التأخير بوجود الحزمة المطورة (ADCUR-AB).

هاتان الخوارزميتان اقترحتا لحل مشكلة عامة ومنتشرة في نوعية الخدمة وهي مشكلة تحديد التأخير للمسار وقد تم الأخذ بنظر الاعتبار تحديد الحزمة التي تعتبر من الموارد المهمة في الشبكة.

من النتائج المختبرية لخوارزمية(DCUR-AB) وجد أن لها حدود بسيطة في عملية البحث لايجاد المسار الافضل عند الأخذ بنظر الاعتبار تحديد الحزمة لذلك تم أقتراح الخوارزمية الأخرى (ADCUR-AB) التي تملك مجال بحث واسع لايجاد المسار الافضل. ثم بعد ذلك تم عمل مقارنة عملية فوجد أنه الخوارزمية الثانية هي الافضل بالأداء من الأولى.

KEY WORDS
Routing Algorithms, QoS Routing, QoS constraints, link-constrained and path constrained routing problem.

INTRODUCTION

The establishment of a computer network requires the development of efficient route selection algorithms that are designed to take the QoS constraints into account [A. S. Tanenbaum 2003].
In this paper, it will be concentrated on the problem of designing efficient route selection algorithms to meet the delay-constraint requirements of networked applications. This problem will be addressed from the point of view of the Unicast model.
So it will be categorized the problem into two concepts; link-constrained and path constrained routing problem. The algorithms proposed in this thesis will discuss the two concepts deeply showing the effect of appending one of the important network resource (bandwidth) in the existing delay constrained Unicast routing algorithm (DCUR). So the proposed algorithm called Delay constrained Unicast routing algorithm at available bandwidth (DCUR-AB).
This matter led us to derive another routing algorithm depending on the concept of distance-vector routing approach, since it performs an extensive search of the various possible routes taking also into account availability of bandwidth and compare the results between two proposed algorithms after apply them on static network. The second proposed routing algorithm called Adaptive Delay constrained Unicast routing algorithm at available bandwidth (ADCUR-AB).

This algorithm has the following properties that make it the best to apply in computer network:

1- They must be able to maximize the overall performance of the network without sacrificing the requirements of any particular call.
2- They must be designed to enable resource reservation to be built into the routing strategy.
3- The algorithms must be able to function with as little global state information as possible.
4- They must also be adaptive to changes in link parameters such as link delay, available bandwidth, etc.
5- They must be able to optimize on multiple constraints, which is required in the QoS routing.

CONSTRAINED-OPTIMIZATION ROUTING PROBLEMS

Path selection within routing is regarded here as a shortest path optimization problem. The objective function for optimization will selected to be any one of a variety of parameters such as number of hops, delay, cost, etc. However, in the context of real-time networks, with many channel-establishment requests being simultaneously active, each specifying diverse QoS requirements, algorithms become increasingly complex as constraints are introduced into these optimization problems. Typically, this makes the problem intractable [G.Feng 2001].
A number of heuristic routing algorithms for such constrained optimization problems will be proposed here. In a flooding based approach, a packet is forwarded to all (or some) of the neighbors of a given node, except the node from which the packet was received. In a preferred neighbor based approach, a packet is forwarded to a preferred neighbor [R.Widyono 1994]. The advantage with the flooding based approach is that it performs an extensive search of the various possible routes, enjoys smaller setup times, and is more adaptive to dynamic link parameter variations than the other approaches. However, it suffers from excessive resource reservation, which may result in lower call acceptance rates. The preferred neighbor approaches will be used to overcome this problem at the cost of overhead for table maintenance. However, these solutions reduce the efficiency of these algorithms in the case of dynamic networks [Stallings 1998].

Problem Formulation

The Unicast channel-establishment request (also referred to as a call) is modeled in a network G = (V; E) where V is a set of nodes and E is a set of links in any network, as a 5-tuple:

R = (id, s, d, B, Δ), where
id : is the call-request identification number,
s ∈ V :is the source node for the call,
d ∈ V :is the destination node for the call,
B: Is the bandwidth requirement, and
Δ: is the delay constraint to be satisfied
Let Psd denote the set of all paths of the form P = (s = v_0, v_1, v_2,.., v_n = d) between source s and destination d that satisfy the following three conditions:

- AB (e) > B, e = (v_i, v_i+1), 0 ≤ i ≤ n-1 (Bandwidth Test).
- D (P) < Δ (Delay Test).
- Any node is not duplicated in P (Loop Test).

THE PROPOSED (DCUR-AB) ALGORITHM

One of the QoS requirements is to reduce the end-to-end time. In existing DCUR algorithm this requirement is achieved without taking into consideration one of the important network resource (Bandwidth).
At the proposed Delay Constrained Unicast Routing with Available Bandwidth algorithm (DCUR-AB) in order to find the optimal path between any two nodes, since each node makes a choice only between next node on the least cost path and the next node on the least delay path to the destination. DCUR-AB routing algorithm programmed with (Visual C++) language and this algorithm is described in the following steps:

Step 0: Start.

Step 1: Enter the number(s) of call-request, let be denoted as N.

Step 2: Assign a call-request identification number (id) for each call request.
Each id is labeled with ordered variables (s, d, B, Δ), where:
s is the source node for the call,
d is the destination node for the call,
B is the Bandwidth requirement, and
Δ is the delay constrained to be satisfied.

Step 3: (Iterative step), for id = 1, …, N

Step 4: The source *s* becomes the current active node, denoted *active-node*, (at all time there is only one active node in the network).
The variable *delay-so-far* is denoted to previous delay (is set to 0 at the beginning of process).
The variable *previous-active node* is denoted to the address of the previous active node (is set to null at the beginning of process).

Step 5: The *active node* reads the address of the next hop node on LC (least cost) path toward destination from its cost vector, denoted as *LCNHOP*. Before forwarding the packet along this link try to satisfy the *Bandwidth test*.

Step 6: The *active node* sends Query message to LCNHOP, requesting the LD (least delay) value for LCNHOP to destination.

Step 7: *LCNHOP* looks up the requested value, LDELAY (LCNHOP, d) from its delay vector, and sends a response message back to active-node with this information.

Step 8: After active-node receive the response message it check if:
Delay-so-far + D(active-node,LCNHOP) + LDELAY(LCNHOP,d)≤ Δ
If the inequality is satisfied, then there exist delay-constrained path from active node to destination, which use the link (active-node, LCNHOP), and active-node selects the direction of LC path towards destination after success the Bandwidth test, and the call is *accepted*.

Step 9: If the inequality in Step 8 is not satisfied, then the active-node reads the next hop node of LD (least delay) path towards destination from it's delay vector, depend as *LDNHOP*. Then calculate the Delay Value if,
Delay so far + D(active node, LDNHOP) + D(LDNHOP,d) $\leq \Delta$
If the inequality is satisfied, then their exist delay constrained path from active node to destination, then a packet if forwarded along this path after success the Bandwidth test (*), and the call is *accepted.*

Step 10: If either condition in Step 8 or in Step 9 is not satisfied then check the value of least Delay value from source to destination if it's less then delay constrained then the call is *accepted.*

Step 11: If all attempts in Step (8-9-10) are not satisfied then the call is *rejected.*

Step 12: Go to the Step 3 to begin a new test for new call.

Step 13: Print the result.

Step 14: End.

(*) Bandwidth test: for each link between two nodes verify that available Bandwidth of that link is greater then the Bandwidth requirement of the call request.

THE PROPOSED (ADCUR-AB) ALGORITHM

The preferred neighbor approach to distributed route selection is a general framework for the construction of routing algorithms. This framework was used here along with heuristics such as shortest path first (SPF) to establish real-time channels. The modification of this approach called the preferred-link approach. The routing decisions are taken with reference to link parameters rather than using the properties of neighboring nodes. This modification was based on the following two arguments:

1- Since the properties of a link can be continuously monitored by the nodes that are adjacent to it, this gives the approach the ability to easily adapt to changes in link parameters.

2- Many of the QoS metrics such as delay-bounds and bandwidth are link parameters. It therefore makes more sense to decide routing decisions based on their parameters rather than be guided by properties of the neighboring nodes (which may be outdated).

To illustrate this approach, the propose Adaptive Delay Constrained Unicast Routing At Available Bandwidth algorithm (ADCUR-AB) will be employed for constructing delay-constrained least cost paths that used to decide on the ordering of neighboring links of a node.
In this section, the propose heuristic will be used in conjunction with the preferred-link table (PLT) approach for constructing delay-constrained paths. These heuristics are used to load the PLT tables at each node in the network. The following notation will be used:

- LDELAY (x; d) = the least delay from node x to node d in the network.
- LCOST (x; d) = the cost of the least cost path from node x to node d.
- LDNHOP (x; d) = the first link on the least delay path from x to d.
- LCNHOP (x; d) = the first link on the least cost path from x to d.

These values are assumed to be available at each node as a result of executing a distributed distance vector algorithm like the Bellman-Ford algorithm.
Let a call setup packet P belonging to the call-request R = (id; s; d; B; Δ) arrive at node v. For each link l = (v; x) at v, ADCUR-AB (l; R) denote the value of the heuristic for link l corresponding to the call request R. Then, we define:

$$\textbf{ADCUR-AB (l, R)} = \frac{C(l)}{\Delta - Pdelay - D(l) - LDELAY(x, d)} \qquad (1)$$

Where C (l) and D (l) respectively denote the cost and delay of link l. If, in the calculation of the function, a particular link l produces a negative denominator, then that link is not included in the preferred list. The links are arranged in increasing order of their ADCUR-AB values so that the links with lower ADCUR-AB values are given greater preference. The intuitive idea underlying this function is to maximize the residual delay (i.e., the delay available for setting up the rest of the path) at the same time minimizing the cost of the link chosen.

ADCUR-AB routing algorithm programmed with (Visual C++) language and this algorithm is described in the following steps:

Step 0: Start.

Step 1: Enter numbers(s) of calls-request

Step 2: Assign a call-request identification number (id) for each call request. Each id is labeled with ordered variables (s, d, B, Δ), where s is the source node for the call, d is the destination node for the call, B is the bandwidth requirement, and Δ is the delay constrained to be satisfied.

Step 3: (Iterative step), for id = 1, …., N

Step 4: The source *s* becomes the current active node, denote active-node (at all times there is only one active node).

Step 5: Let Psd denotes to the path which is represented as a finite sequence of non-repeated nodes for source to destination,
Psd = (s = V_0, V_1, ….., V_n = d), (Psd is set to null at beginning).

Step 6: Before forwarding the packet along all neighbors, the following tests will be taken into account:

- Bandwidth test: verify that available bandwidth for each link between active node and each neighbor nodes greater than the bandwidth requirement for the id call request.
- Loop test: verify that each neighbor nodes is not a node in Psd.

Step 7: Select the node(s) that satisfy the condition in step 6, then calculate the value of the function.

$$\textbf{ADCUR-AB (l, R)} = \frac{C(l)}{\Delta - Pdelay - D(l) - LDELAY(x, d)}$$

Step 8: If the result of the function in step 7 f (or a particular link) produce a negative denominator, then that link is skipped (not in preferred path Psd), otherwise, the links are arranged in ascending order of their ADCUR-AB values so that the links with lower ADCUR-AB values are given greater preference, and put this node to preferred path (Psd).

Step 9: Repeat step (4-8) until the path is completed (reach to the destination node), then the call is *accepted.*

Step 10: If any test in (step 6 or step 7) is failed then each node sends a *reject* packet to the node from which it received (active-node) this is formulated as backtracking based route selection method.

Step 11: If all attempts failed then the call is *rejected.*

Step 12: Go to Step 3 to begin a new test for new call.

Step 13: Print the results.

Step 14: End.

DEMONSTRATED EXAMPLE

As a demonstrated example a set of five call-requests in a small 9-node network are handled by the proposed (DCUR-AB) algorithm. Then the same requests are handled by the second proposed algorithm (ADCUR-AB). The 9-node network is shown in **Fig. (1).**

Each edge is labeled with an ordered pair representing the (cost, delay) values of the link. Each edge is assumed to have a total bandwidth of 30 units. The set of 5 call-requests given in Table 1 are assumed to occur one after another in the specific order. **Table (2)** specifies the least-cost and least-delay paths between every pair of vertices in the network

Table (1) Set of Call-requests

Call Id	Source	Destination	Bandwidth	Delay Constrained
1	1	3	10	6
2	2	4	10	7
3	2	4	10	6
4	2	4	10	6
5	2	4	10	7

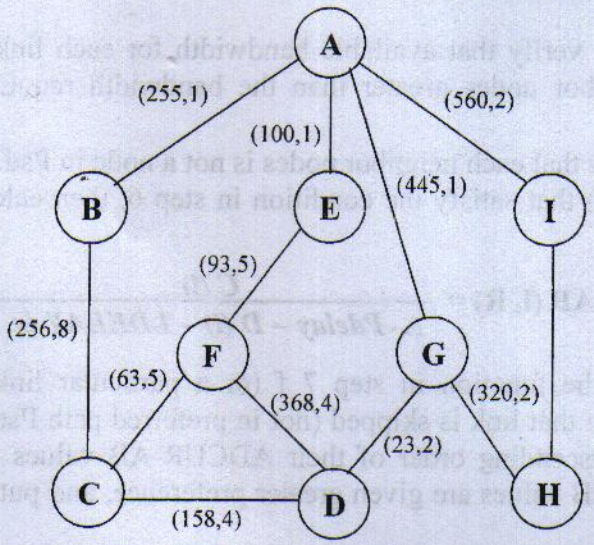

Fig. (1) Network used for the example

Table (2) Least-Cost and least-delay paths

Vertex Pair	Least Cost path	Least Cost	Least Delay path	Least delay
A , B	A → B	225	A → B	1
A , C	A → E → F → C	256	A → G → D	7
A , D	A → E → F → C → D	414	A → G → D	3
A , E	A → E	100	A → E	1
A , F	A → E → F	193	A → E → F	6
A , G	A → E → F → C → D → G	437	A → G	1
A , H	A → E → F → C → D → G → H	757	A → G → H	3
A , I	A → I	560	A → I	2
B , C	B → C	256	B → C	8
B , D	B → C → D	414	B → A → E → D	6
B , E	B → A → E	325	B → A → E	2
B , F	B → C → F	319	B → A → E → F	7
B , G	B → C → D → G	437	B → A → G	2
B , H	B → C → D → G → H	757	B → A → G → H	4
B , I	B → A → I	785	B → A → I	3
C , D	C → D	158	C → D	4
C , E	C → F → E	156	C → D → E	8
C , F	C → F	63	C → F	5
C , G	C → D → G	181	C → D → G	6
C , H	C → D → G → H	501	C → D → G → H	8
C , I	C → F → E → A → I	816	C → D → G → A → I	9
D , E	D → C → F → E →	314	D → E	4
D , F	D → C → F	221	D → C → F	9
D , G	D → G	23	D → G	2
D , H	D → G → H	343	D → G → H	4
D , I	D → G → H → I	600	D → G → A → I	5
E , F	E → F	93	E → F	5
E , G	E → F → C → D → G	337	E → A → G	2
E , H	E → F → C → D → G → H	657	E → A → G → H	4
E , I	E → A → I	660	E → A → I	3
F , G	F → C → D → G	244	F → E → A → G	7
F , H	F → C → D → G → H	564	F → E → A → G → H	9
F , I	F → E → A → I	753	F → E → A → I	7
G , H	G → H	320	G → H	2
G , I	G → H → I	577	G → A → I	3
H , I	H → I	257	H → I	4

Routes Chosen By The DCUR-AB Algorithm

Call 1: src = A , Dest. = C , Δ = 12 :

at node A :

LCNHOP (A,C) = A → E

D (A → E) + LDELAY (E,C) = 1+8 = 9 < Δ

at node E :

LCNHOP (E, C) = E → F

P. delay + D (E → F) + LDELAY (F, C) = 1 + 5 + 5 = 11

at node F :

LCNHOP (F , C) = F → C

P. delay + D (F → C) = 6 + 5 = 11

∴ The selected route is A → E → F → C; COST = 256.

Call 2: source = I , Dest. = C $\Delta = 10$

at node I :

LCNHOP (I, C) = I → A

D (I → A) + LNELAY (A, C) = 2+7 = 9 < 10

at node A :

LCNHOP (A, C) = A → E

P.delay + D (A → E) + LDEIAY (E, C) = 2 + 1 + 8 = 11 > 10 then

Node A chooses LDNHOP (A, C) = A → G

P.delay + D (A → G) + LDEIAY (G, C) = 2 + 1 + 6 = 9 < 10

At node G:

LCNHOP (G, C) = C → D

P. delay + D (G → D) + LPELAY (D, C) = 3 + 2 + 4 = 9 < 10

At node D:

LCNHOP (D, C) = C → D

P. delay + D (C → D) = 5 + 4 = 9 < 10

∴ the selected rout is I → A → G → D → C ; Cost = 1186

Call 3: src = E , dest. = H , $\Delta = 9$

At node E:

LCNHOP (E,H) = E → F

D (E → F) + LDELAY (F,H) = 5 + 9 = 13 > 9 then node E

Chooses LDNHOP (E,H) = E → A

D (E → A) + LDELAY (A,H) = 1 + 3 = 4 < 9

At node A:

LCNHOP (A, H) = A → E, skipped because it fuils loop test.

∴ Chooses LDNHOP (A, H) = A → G

P. delay +D (A → G) + LDELAY (G,H) = 1 + 1 + 2 = 4 < 9

At node G:

LCNHOP (G, H) = G → H

P. delay + D (G → H) = 2 + 2 = 4

∴ The selected route is E → A → G → H; cost = 865.

Call 4: Src = F, dest. = H, $\Delta = 9$

At node F:

LCNHOP (F, H) = F → C

D (F → C) + LDELAY (C, H) = 5 + 8 = 13 > 9, LNELAY (F, H) = 9

Then chooses LDELAY (F, H) = F → E

D (F → E) + LDELAY (E, H) = 5 + 4 = 9 = Δ

At node E:

LCNHOP (E, H) = E → F → it fails

Then take LDNHOP (E, H) = E → A

P. delay + D (E → A) + LDELAY (A, H) = 5 + 1 + 3 = 9 = Δ

At node A:

LCNHOP (A, H) = A → E; it fails

Then take LDELAY (A, H) = A → G

P. delay + D (A → G) + LDELAY (G → H) = 6 + 1 + 2 = 9 = Δ

At node G:

LCNHOO (G, H) = G → H

P. delay + D (G → H) 7 + 2 = 9 = Δ

∴ The selected route is F → E → A → G → H; cost = 1021.

Call 5: Src = B; Dest = H, Δ = 10

At node B:

LCNHOP (B, H) = B → C

D (B → C) + LDELAY (C, H) = 8 + 8 = 16 > Δ

Then node B chooses LCNHOP (B, H) = B → A

D (B → A) + LDELAY (A, H) = 1 + 3 = 4 < 10

At node A:

LCNHOP (A,H) = A → E . It fails because

AB = 30 then this cali can't pass through A → E because

It's full; node A chooses LDNHOP (A, H) = A → G

Fails because the link A → G it's full because AB = 30

Then call cann't pass through A → G then the call 5 is rejected.

Routes Chosen by the ADCUR-AB Algorithm

Call 1: Src = A , Dest. = C , Δ = 12

At node A:

- for link (A → B) ; 225 / (12-0-1-8) = 75
- for link (A → E) ; 100 / (12-0-1-8) = 33.333
- FOR LINK (A → G) ; 445/ (12- 0-1-2) = 49.444
- FOR LINK (A → I) ; 5601 / (12-0-2-4) = 93.333

The lowest value is for link (A → E) → for worded to node E

At node E :

- for link (E → A) ; skipped because it fails loop test.
- For link (E → F) ; 93/ (12-1-5-5) = 93
- for link (E → D) ; 368 / (12-1-4-4) = 122.666

The lowest values for link (E → F) → forward to F

At node F :

- for link (F → C) ; 63/ (12-6-5-0) = 63
- for link (F → E) ; skipped because it fails loop test .

Then the path chose is A → E → F → C; cost = 256.

Call 2: Scr = I , Dest. C, Δ = 10

At node I : ,

- for link (I → A); 560 / (10-0-2-7) = 560
- for link (I → H) ; 257 / (10-0-4-8) < 0;skiped.

Forward to node A

At node A :

- for link (A → B); 225 / (10-2-1-8) < 0; skipped.
- For link (A → E); 100 / (10-2-1-8) < 0;skipped.
- For link (A → G); 445 /(10-2-1-6) = 445

Forwarded to node G

At node G:

- for link (G → D); 23/(10-3-2-4) =23.

- for link (G → H); 320/(10-3-2-8) < 0 ; skipped.

At node D :

- For link (D →E); 386 / (10-5-4-4)< 0 ; skipped.
- For link (D→C) ; 158 / (10-5-4-0) = 158
- For link (D→G) ; skipped because it fails loop test .

Then the path chose is I → A→G → C ; cost = 1186.

Call 3: Src = E . Dest. = H, Δ = 9

At node E : -

- for link (E → A) ; 100 / (9-0-1-3) = 20
- for link (E → F) ; 93/ (9-0-5-9) = < 0 ; skipped.
- for link (E → D) ; 368/ (4-0-4-4) = 368.

The lowest value is E → A

Forwarded to node A

At node A : -

- for link (A → B) ; 225/ (9-1-1-4) = 75
- for link (A → G) ; 445/ (9-1-1-2) = 89
- for link (A→ I) ; 560/ (9-1-2-4) = 280.

The lowest value is A → B

Forwarded to node B

At node B : -

- for link (B → A) ; skipped because it fails loop test .
- for link (B → C) ; 225/ (9-2-8-8) < 0 ; skipped.

Then we will back to node A and choose link A → G

Forwarded to node G

At node G : -

- for link (G → D) ; 23 / (9-2-1-4) = 11.5
- for link (G → H) ; 320/ (9-2-1-0) = 53.333
- for link (G → A) ; skipped because it fails loop test .

choose link G → D

Forwarded to node D

At node D: -

- For link (D → E); skipped because it fails loop test.
- For link (D → C); 158/ (9-4-4-8) < 0; Skipped

Then we will back to node G and choose link G → H

Then the path chose is E → A→G → H; cost = 865.

Call 4: Src = F . Dest. = H, Δ = 9

At node F : -

- for link (F → E) ; 93 / (9-0-5-4) = ∞ ; skipped
- for link (F → C) ; 63/ (9-0-5-8) < 0 ; skipped

Then the call will rejected

Call 5: Src = B . Dest. = H, Δ = 10

At node B : -

- for link (B → A) ; 225 / (10-0-1-3)=37.5
- for link (B → C) ; 256/ (10-0-8-8) < 0 ; skipped

At node A : -

- for link (A → I) ; 560/ (10-1-2-4) = 186.666
- for link (A → G) ; 445/ (10-1-1-2) = 74.1666
- for link (A → E) ; 100/ (10-1-1-4) = 25.

At node E : -

- for link (E → F) ; 93/ (10-2-5-9) < 0; Skipped
- for link (E → D) ; 368/ (10-2-4-4) = ∞ ; Skipped

we will choose at node A the link (A → G)

At node G : -

- for link (G → D) ; 23/ (10-2-2-2) = 5.75
- for link (G → H) ; 320/ (10-2-2-0) = 53.333

At node D : -

- for link (D → E) ; 368/ (10-4-4-4) < 0; Skipped
- for link (D → C) ; 158/ (10-4-4-8) < 0; Skipped

Back to node G and choose link (G → H)

After that the path is chosen is B → A → G → H; cost = 990.

CONCLUSIONS

In the case of calls 1 and 2, the DCUR-AB algorithm, at each node, attempted to forward the packet via the least cost route. However, since the delay constraint was not satisfied it finally chose only the least delay route between the source and destination. The ADCUR-AB algorithm however was able to find a route that was neither the least cost nor the least delay route, but which satisfied the delay constraint without excessive cost. Calls 3, 4, and 5 illustrate that because of it's ability to search for alternate paths, the ADCUR-AB algorithm is able to distribute the hot-pair communication load between nodes 2 and 4 among two different routes, thus providing greater call acceptance.

REFERENCES

A.S. Tanenbaum, (2003), Computer Networks, Third edition, Prentice-Hall,.

G. Feng, (2001), Neural network and algorithmic methods for solving routing problems in high speed network, Ph.D. thesis, University of Miami, Department of Electronic and Communications,.

R. Widyono, (1994), The Design and Evaluations of Routing Algorithms for Real Time Channels, TR-94-024, University of California at Berkeley, International Computer Science Institutes, June.

W. Stallings, (1998), High-Speed Networks, TCP/IP and ATM design and principles, New Jersey: Prentice-Hall,.