Computer Communications 21 (1998) 1655–1669

Preferred link based delay-constrained least-cost routing in wide area networks

R. Sriram, G. Manimaran, C. Siva Ram Murthy*

Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, India

Received 31 October 1997; received in revised form 21 May 1998; accepted 22 May 1998

Abstract

Multimedia applications involving digital audio and/or digital video transmissions require strict QoS constraints (end-to-end delay bound, bandwidth availability, packet loss rate, etc.) to be met by the network. To guarantee the real-time delivery of packets satisfying these constraints, a *real-time channel* (D. Ferrari and D.C. Verma, A scheme for real-time channel establishment in wide-area networks. IEEE JSAC, 8(3), 368–379, 1990) needs to be established before the transmission of packets of a connection can begin. The establishment of such channels requires the development of efficient route selection algorithms that are designed to take into account the QoS constraints.

The general problem of determining a least-cost delay-constrained route in a given communication network has been proved to be NP-hard (M.R. Garey and D.S. Johnson, Computers and Intractability: a guide to the theory of NP-completeness, W.H. Freeman, 1979). In this paper, we describe a preferred link approach to distributed delay-constrained least-cost routing in order to establish real-time channels. The approach attempts to combine the benefits of probing and backtracking based algorithms (better adaptiveness and wider search) with the advantages of distance-vector type algorithms (lower setup time). The scheme is flexible in that a variety of heuristics can be employed to order the neighbouring links of any given node. Three heuristics are proposed and their performance is studied through simulation experiments. The simulation results indicate that the proposed heuristics provide better performance than other preferred neighbour methods, in terms of increased call acceptance rate and lower average route cost. The heuristics are also shown to adapt much better to dynamic variations in network and link characteristics. © 1998 Elsevier Science B.V. All rights reserved

Keywords: QoS routing; Wide area networks; Preferred neighbour approach; Constrained optimization; Heuristics

1. Introduction

Packet switched networks are increasingly being used to transmit multimedia traffic such as video and audio streams besides supporting traditional data communication applications. These multimedia applications require stringent quality of service (QoS) constraints to be met by the underlying network in terms of end-to-end delay bound, delay jitter, bandwidth availability, packet loss rate, etc. For such a network to provide performance guarantees, it is necessary that efficient route selection strategies are employed to determine routes between sender and destination nodes in the network [1].

In traditional computer networks, routing algorithms attempted to optimize a particular metric, such as message delay or routing distance, for a single connection. However, the overall performance of the network is enhanced only if global metrics such as average call acceptance rate, average call setup time, and average route distance are optimized [2]. At the same time the algorithms must attempt to ensure that each accepted call is assured of the agreed-upon (by call admission control) QoS. From the point of view of overall network efficiency and efficient management of the network resources, it is important to model the utilization of the network by each call, in terms of a cost for the call to use the chosen route. Routing algorithms must therefore also attempt to minimize the cost of using a particular route to connect the source and destination nodes.

Routing algorithms are expected to satisfy certain additional constraints to make them suitable for actual practical implementation on wide area networks. Typically the routing algorithms must attempt to minimize the extent to which they rely on global state information. The algorithms must also scale well to larger networks, by minimizing the call setup overhead and call setup time. Since transmission of state information across wide area networks takes a fair amount of time, routing algorithms must also be designed to be adaptive to changes in network characteristics and must be capable of working with out-of-date information.

In this paper, a new distributed route selection method, which employs the idea of preferred neighbouring links at

* Corresponding author. Fax: +91 44 2350509; e-mail: murthy@iitm.ernet.in

each network node, is described. The rest of the paper is organized as follows. In Section 2, we discuss the existing routing strategies and the motivation for our work. The proposed routing method is presented in Section 3. In Section 4, we present a simple example that illustrates the algorithm. In Section 5, we compare the performance of our algorithm with that of an existing algorithm, and also present and discuss the simulation results. Section 6 concludes the paper, highlighting the advantages of the proposed approach.

2. Background and motivation

2.1. Constrained-optimization routing problems

Traditionally, path selection within routing is formulated as a cost-optimization problem. The objective function for optimization could be any one of a variety of parameters, such as number of hops, delay, and cost. 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 [3]. Wang and Crowcroft [1] have studied the QoS routing problem, and have classified metrics into additive, multiplicative, and concave metrics. They have shown that the problem of finding a path subject to constraints on two or more additive and multiplicative metrics in any possible combination is NP-complete [3]. As a result, heuristic methods need to be employed to attempt to achieve performance close to optimal.

A number of heuristic routing algorithms for such constrained optimization problems have been proposed. In a flooding based approach, a packet is forwarded to all (or some) of the neighbours of a given node, except the node from which the packet was received. A distributed route selection scheme, based on flooding, that tries to bound the number of messages used to establish a call, is discussed in [4]. In a preferred neighbour based approach, a packet is forwarded to a preferred neighbour that is chosen based on certain heuristics. Such an approach has been proposed in [2], where heuristics such as shortest path first (SPF), lightly loaded link first (LLF), and two-level shortest path first (TSPF) have been analysed. 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 results in lower call acceptance rates. The preferred neighbour approaches overcome this problem at the cost of overhead for table maintenance. However, the dependence of these algorithms on the accuracy of the tables reduces their efficiency in the case of dynamic networks.

One of the problems studied in this class of constrained optimization problems is the least-cost delay-constrained routing problem [5]. Delay constraint is a very common requirement of many multimedia applications. Cost minimization captures the need to distribute the network resources efficiently amongst the various calls. Cost of a link is intended as an abstraction which could, in practice, be mapped to a variety of link parameters such as the reciprocal of available bandwidth and the number of calls using the link.

Widyono [6] has proposed an optimal centralized delay-constrained least-cost routing algorithm known as the constrained Bellman–Ford algorithm, which performs a breadth-first search to determine the optimal path. However, because of its optimality, the worst-case running time of the algorithm grows exponentially with network size. Jaffe [7] studied a variation of the problem, in which both cost and delay were specified as constraints, and proposed pseudo-polynomial-time and polynomial-time heuristics for solving the problem. A recent method, proposed in [5], that addresses the least-cost delay-constrained routing problem, uses entries in the cost and distance vector tables maintained at each node to decide on the next node to which the routing packet is to be passed. Every node initially attempts to forward the packet to the next node along the least cost path to the destination. However, if the least delay from the next node to the destination is such that the delay constraint is violated, then the node attempts to forward the packet to the next hop along the least delay path to the destination. Each node therefore makes a choice only between the next node on the least cost path and the next node on the least delay path to the destination. The algorithm, by restricting the choice to these two nodes, fails to consider links that could potentially offer a better overall cost–delay performance. In addition, because of its reliance on cost and distance vector tables, the algorithm is dependent on the accuracy of these tables. For dynamic networks, whose link parameters vary frequently, this accuracy cannot be guaranteed.

2.2. Motivation for our work

It is clear from the above discussion that though a number of algorithms for delay-constrained least-cost routing (and other QoS routing problems) have been developed, they have generally tended to concentrate purely on the optimization aspects of routing. For an algorithm to actually perform well in practice, it is necessary to also take into account factors such as overall network performance, possibility of out-of-date information in the routing tables, frequent changes in link parameters and resource reservation during channel establishment.

We believe that routing algorithms that are intended to be used as route selection mechanisms for real-time channel establishment in wide area networks must possess the following characteristics:

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 [8].
3. The algorithms must be able to function with as little global state information as possible.
4. They must be adaptive to changes in link parameters such as link delay and available bandwidth.
5. They must be able to optimize on multiple constraints, which is required in the case of QoS routing [1].

As discussed in Section 2.1, flooding is better suited for achieving properties 3 and 4, whereas preferred-neighbour- or distance-vector-based algorithms are suitable for satisfying properties 1, 2, and 5. As an attempt to satisfy all the objectives set forth above, in this paper we propose a flexible preferred link based approach to distributed path selection for setting up cost-minimized delay-constrained paths.

3. The proposed routing approach

3.1. Network model

In this paper, the network is modelled as an undirected graph $G = (V,E)$, where V is the set of nodes and E is the set of interconnecting links. We associate the following four functions with each link $e \in E$.

Delay function	D	:	$E \to R^+$
Cost function	C	:	$E \to R^+$
Total Bandwidth function	TB	:	$E \to R^+$
Available Bandwidth function	AB	:	$E \to R^+$

A path $P = (v_0,v_1,v_2,\ldots,v_n)$ in this network, has two associated characteristics:

$$\text{Cost } \mathrm{C}(P) = \sum_{i=0}^{n-1} \mathrm{C}(v_i, v_{i+1})$$

$$\text{Delay } \mathrm{D}(P) = \sum_{i=0}^{n-1} \mathrm{D}(v_i, v_{i+1})$$

In the case of the **static network model,** we assume that for each $e \in E$, C(e), D(e) and TB(e) are fixed, though AB(e) varies depending on the usage of the link. In the dynamic network model, C(e) and D(e) are also allowed to vary. When the parameters C(e) and D(e) of a particular link, e, change, we assume that this change is known to the nodes attached to e immediately, even though update of tables in remote nodes (to reflect these changes) may be delayed. This is a reasonable assumption to make, since nodes can be expected to monitor the state of their adjacent links and register changes in the link parameters immediately. Propagation of this information either directly or indirectly (by executing distributed Bellman–Ford Algorithm [9]) to other nodes will generally be delayed.

3.2. Problem formulation

We model a channel-establishment request (also referred to as a call) in the network described above, as a 5-tuple:

$R = (id, s, d, B, \Delta)$,

where: id is the call-request identification number; $s \in V$ is the source node for the call; $d \in V$ is the destination node for the call; B is the bandwidth requirement; and Δ is the delay constraint to be satisfied.

Let P_{sd} denote the set of all paths of the form $P = (s = v_0,v_1,v_2,\ldots,v_n = d)$ between source s and destination d that satisfy the following two conditions:

- $\mathrm{AB}(e) \geq B, \forall\ e = (v_i,v_{i+1}), 0 \leq i \leq n-1$
- $\mathrm{D}(P) \leq \Delta$.

The delay-constrained least-cost routing problem can now be formulated as

$$\text{Find } P' \in P_{sd} \text{ such that } \mathrm{C}(P') = \min\{\mathrm{C}(P){:}P \in P_{sd}\}$$

3.3. The routing strategy

The preferred neighbour approach to distributed route selection is a general framework for the construction of routing algorithms. This framework was used in [2], along with heuristics such as SPF and LLF, to establish real-time channels. In this paper, we propose to adapt this framework for constructing delay-constrained least-cost paths. For this, three new heuristics are described, which are used to decide on the ordering of neighbouring links of a node. In the following sections, we will first informally describe the preferred-link routing framework. We will then present the three heuristics to be used in conjunction with this framework and also describe the data structures to be present at each node in order to implement the routing heuristics. Finally, we will formalize the algorithm.

3.3.1. The preferred link routing framework

The preferred link routing framework is fundamentally a backtracking-based route selection method. This framework describes a set of actions to be performed by each node whenever it receives a call setup or a call reject packet. When a node v receives a call setup packet, it forwards it along the first preferred link. If a reject packet is received from the node at the other end of this link, then node v attempts to forward the packet along the next preferred link and so on, until a specified number of links have been tried out. If all such attempts result in failure, then v sends back a reject packet to the node from which it received the call setup packet. If the call setup packet reaches the destination, then the call is successfully setup. If the source gets rejected on all attempts, the call is rejected.

3.4. Data structures at each node

To implement the proposed heuristics in conjunction with the preferred link routing framework, each node in the network is equipped with two data structures, namely, a **history buffer** and a **preferred link table**.

3.4.1. History buffer

The history buffer (HB) at each node, v, contains one entry for every call for which v has received a call setup packet. Each entry contains a pair of elements (*packet*, *tried*) where *packet* is the call setup packet received by this node and *tried is* the number of preferred neighbour links on which v has tried to forward the request. Therefore, the HB at node v contains the complete status information for every call that was handled by v. The entry corresponding to a call is removed when the call is either accepted or rejected.

3.4.2. Preferred link table

The structure of the preferred link table (PLT) to be maintained at each node depends on the nature of the heuristic function that is employed to construct the table. For describing the structure of the PLT, we classify all heuristic functions into two major categories, namely *destination-specific heuristics* and *call-specific heuristics*.

Destination-specific heuristics are those, whose computation is specific to each destination. Therefore if the destination nodes of two different call-requests arriving at a given node are the same, then the two calls will share an identical list of preferred links. Each node, v, in the network is equipped with a PLT that contains one row for every destination. Each row contains the preferred links for that particular destination ordered in terms of decreasing preference. The maximum number of entries per row is denoted by κ. Obviously κ is upper-bounded by the maximum degree of any node in the network. The preference for a link will be decided based on the value of a heuristic function that is computed for each (link, destination) pair.

Call-specific heuristics are those whose computation depends on the particular parameters carried by a call setup packet arriving at the node. In such cases, the list of preferred links is individually computed for each call request. As a result, the ordering of the links will be call-specific instead of being just destination specific. For such heuristic functions, the number of rows in the PLT table will vary dynamically, depending on the number of calls currently being handled by the node. The table entries corresponding to a particular call are removed when the call is accepted or rejected.

3.4.3. Tests before forwarding

Before forwarding any packet along a link, each node conducts three tests on the link parameters. The link is used for forwarding the packet only if all the tests are successful. The tests are described below.

Let $R = (id,s,d,B,\Delta)$ be a call request, and let P be a call-request packet arriving at a node v. Let *P.path* denote the path taken by the packet up to this point, and let *P.delay* denote the cumulative delay along this path. Before forwarding the packet along link $l = (v,v')$, node v conducts the following three tests.

Bandwidth Test	:	Verify that $AB(l) \geq B$
Delay Test	:	Verify that $P.delay + D(l) \leq \Delta$
Loop Test	:	Verify that v' is not a node in *P.path*

3.5. Proposed heuristic functions

In this section, we propose heuristics that are used to load the PLT tables at each node in the network. We will also describe the computation to be performed and the intuitive reason behind the choice of each heuristic. For the description of the heuristics, we will use the following notation:

- 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 [9].

1. **Residual delay maximizing (RDM) heuristic**: this heuristic is a call-specific heuristic. Let a call setup packet P belonging to the call-request $R = (id,s,d,B,\Delta)$ arrive at node v. For each link $l = (v,x)$ at v, let RDM(l,R) denote the value of the heuristic for link l corresponding to the call request R. Then, we define

•

$$\mathrm{RDM}(l,R) = \frac{\mathrm{C}(l)}{\Delta - P.delay - \mathrm{D}(l) - \mathrm{LDELAY}(x,d)}$$

- 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 RDM values, so that the links with lower RDM 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) while at the same time minimizing the cost of the link chosen. A similar idea of residual delay has also been used by Kompella et al. [10] in their multicast routing algorithm.

2. **Cost delay product (CDP) heuristic**: this is a destination-specific heuristic. We define the cost–delay product, corresponding to the destination, d, of a link $l = (v,x)$ to be

•

$$\mathrm{CDP}(l) = \mathrm{C}(l) * (\mathrm{D}(l) + \mathrm{LDELAY}(x, d))$$

- where $\mathrm{C}(l)$ is the cost of the link and $\mathrm{D}(l)$ is its delay. To load the PLT entries corresponding to the destination d, the following steps are performed:
- The links adjacent to v are arranged in increasing order of their CDP values and first κ links are chosen.
- If this chosen set does not contain LCNHOP(v,d), then LCNHOP(v,d) is placed as the first preferred link and the last link in the originally chosen set is dropped.
- If now the set does not contain LDNHOP(v,d), then LDNHOP(v,d) is used to replace the last preferred link in the chosen set.
- This final set of links is used to populate the PLT entry for destination d.

3. **Partition-based ordering (PBO) function**: this heuristic is a destination-independent and call-independent heuristic. Let avg(v) denote the average cost of all the links adjacent to v. The links adjacent to a node v are partitioned into two sets *below* and *above*, where:

$$\mathrm{below}(v) = \{l : \mathrm{C}(l) \le \mathrm{avg}(v)\}$$

$$\mathrm{above}(v) = \{l : \mathrm{C}(l) > \mathrm{avg}(v)\}$$

The links in the two sets are then separately sorted in increasing order of their delay values. A new list is then created containing the sorted *below* set followed by the sorted *above* set. The first κ links from this new list are chosen and used to populate the table (which in this case reduces to a single-row table).

In all cases, ties between two links are resolved by giving preference to the link with larger available bandwidth.

3.5.1. Formal algorithm description

The algorithm is described as a pair of procedures, action-on-reject and action-on-setup, which outline the steps to be taken by a node on receiving a call reject and call setup packet, respectively.

Notation

- We will use the notation HB(v,I) to specify a function that accesses the history buffer of node v and returns the buffer entry corresponding to a call-request with identifier I. Each such entry will contain a tuple (*packet*,*tried*) as defined earlier.
- In the case of destination-specific heuristics, we will use the notation PLT(v,i,d) to denote a function that accesses PLT and return the ith preferred link at node v for routing a packet to a destination node d.
- For call-specific heuristics, PLT(v,i,j) will denote a function that accesses the PLT and returns the ith preferred link at node v for routing a packet belonging to a call with call-id j.
- To represent the **Bandwidth, Loop, and Delay** tests conducted on a link l, we will use three functions, Bandwidth(l), Loop(l) and Delay(l), each of which will return *true* if l passes the test and *false* otherwise.
- For a packet P, *P.callid* will denote the identifier of the call to which P belongs and *P.prev* will denote the penultimate node in the current path travelled by P.

```
Action-on-reject(v,P) /* reject packet P arrives at node v */
begin
BufferEntry Q = HB(v,P.callid);
Boolean sent = false;
while ((Q.tried < κ) and not(sent))
begin
Q.tried = Q.tried + 1;
Link l = PLT(v,Q.tried,x)
/* x = destination node of the call if destination-specific heuristic
x = P.callid if call-specific heuristic */
if (Bandwidth(l) and Loop(l) and Delay(l)) then
begin
Forward Q.packet along link l;
sent = true;
end;
end;
if not(sent) then
begin
if (v = source node for the call) then call is rejected;
else send reject packet to Q.packet.prev;
end;
end;

Action-on-setup(v,P) /* call setup packet P arrives at v */
begin
If (v = destination for the call) then call is accepted
else begin
Add a new entry to HB containing the pair (P,0);
Let Q be this new entry;
```

```
If (call-specific heuristic is being used) then
begin
Create a new PLT entry corresponding to this call;
Evaluate heuristic for each link and populate this new entry;
Boolean sent = false;
end;
repeat
Q.tried = Q.tried + 1;
Link l = PLT(v,Q.tried,x)
/* x = destination node of the call if destination-specific heuristic
x = P.callid if call-specific heuristic */
If (Bandwidth(l) and Loop(l) and Delay(l)) then
begin
Forward Q.packet along link l;
sent = true;
end;
until ((Q.tried > κ) or (sent = true));
if not(sent) then begin
if (v = source node for the call) then call is rejected;
else send reject packet to Q.packet.prev;
end;
end;
end;
```

4. Example

In this section, we will describe how a set of five call-requests in a small five-node network are handled by the delay-constrained unicast routing (DCUR) algorithm [5]. We will then illustrate how the same requests are handled by our proposed algorithm using the RDM heuristic. The 5-node network is shown in the above figure. Each edge is labelled with an ordered pair representing the (cost, delay) values of the link. Each edge is assumed to have a total bandwidth of 30 units. Consider the following five call-requests, which occur one after another in the specified order.

Call Id	Source	Dest.	B/W	Δ
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

Table 1 contains the least-cost and least-delay paths between every pair of vertices.

4.1. Route selection by the DCUR and RDM algorithms

In the following description of the steps executed by both the algorithms, we will use the notations LDNHOP(x,d), LCNHOP(x,d), LDELAY(x,d), and LCOST(x,d) defined previously. We will also use D($a \rightarrow b$) and C($a \rightarrow b$) to denote the delay and cost of the link connecting vertices a and b.

4.1.1. Routes chosen by DCUR

1. **Call 1**: source = 1; destination = 3; $\Delta = 6$;

   At node 1:

- Attempt along the least cost path using the link *LCNHOP*(1,3) = ($1 \rightarrow 2$)
- Calculate minimum possible delay if ($1 \rightarrow 2$) is chosen. D($1 \rightarrow 2$) + LDELAY(2,3) = 3 + 4 = 7

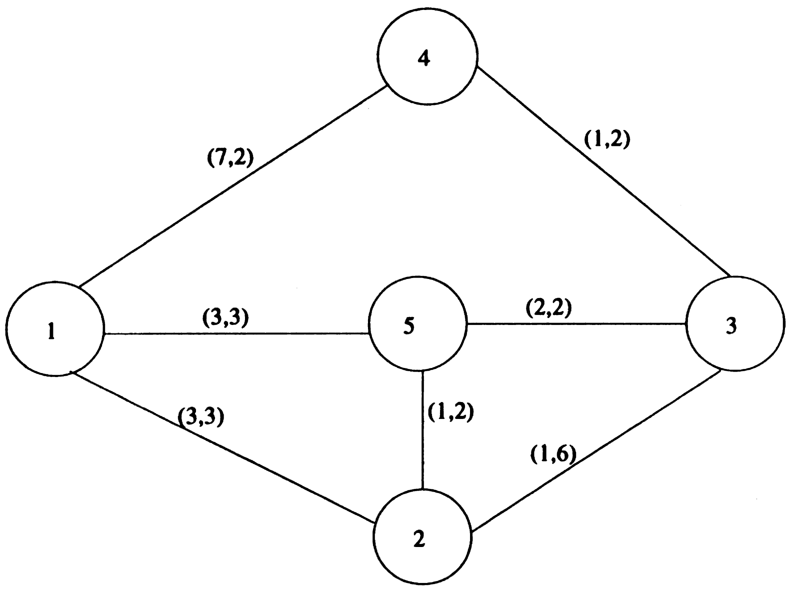


Table 1
Least-cost and least-delay values

Vertex Pair	Least-cost path	Least cost	Least-delay path	Least delay
(1,2)	$1 \to 2$	3	$1 \to 2$	3
(1,3)	$1 \to 2 \to 3$	4	$1 \to 4 \to 3$	4
(1,4)	$1 \to 2 \to 3 \to 4$	5	$1 \to 4$	2
(1,5)	$1 \to 5$	3	$1 \to 5$	3
(2,3)	$2 \to 3$	1	$2 \to 5 \to 3$	4
(2,4)	$2 \to 3 \to 4$	2	$2 \to 1 \to 4$	5
(2,5)	$2 \to 5$	1	$2 \to 5$	2
(3,4)	$3 \to 4$	1	$3 \to 4$	2
(3,5)	$3 \to 5$	2	$3 \to 5$	2
(4,5)	$4 \to 3 \to 5$	3	$4 \to 3 \to 5$	4

- Since $7 > \Delta$ and LDELAY(1,3) $< \Delta$, node 1 chooses LDNHOP(1,3) $= (1 \to 4)$.
- Forward packet to node 4.

At node 4:

- Receive a packet P from node 1 with $P.delay = D(1 \to 4) = 2$.
- Attempt along least cost path. LCNHOP(4,3) $= (4 \to 3)$.
- Since $P.delay + D(4 \to 3) < \Delta$ forward packet to node 3 which is also the destination.

Hence the selected route is **1 → 4 → 3**; **Cost = 8**

2. **Call 2:** source = 2; destination = 4; $\Delta = 7$;

   The algorithm proceeds as in the previous call.
   The route that is chosen is **2 → 1 → 4**; **Cost = 10**

3. **Calls 3,4, and 5**: all use the pair of vertices (2,4) as the source–destination pair. These set of calls were chosen to illustrate how the algorithms deal with **hot-pair communication (HPC)** wherein a large number of calls are generated for a given (source,destination) pair. The DCUR algorithm will choose the path $2 \to 1 \to 4$ for each of these calls, as the delay constraint for each of them is less than 8 (which is the delay along the least-cost path between 2 and 4). However, after call 3 is accepted, link (1,4) will have no available bandwidth as it is supporting calls 1, 2, and 3, each of which requires 10 units. Therefore, calls 4 and 5 will be rejected by the DCUR algorithm.

4.1.2. Routes chosen by RDM heuristic

1. **Call 1:** Source = 1; Destination = 3; $\Delta = 6$;

   At node 1: The RDM value is computed for each of the three links adjacent to node 1.

- For link $(1 \to 2)$: RDM$(1 \to 2) = 3/(6 - 0 - 3 - 4) < 0$; skipped as it fails delay test.
- For link $(1 \to 5)$: RDM$(1 \to 5) = 3/(6 - 0 - 3 - 2) = 3$
- For link $(1 \to 4)$: RDM$(1 \to 4)) = 7/(6 - 0 - 2 - 2) = 3.5$

The lowest value is for link $(1 \to 5)$. Hence packet is forwarded to node 5.
At node 5: Here again the RDM values are computed for the three links adjacent to node 5.

- For link $(5 \to 1)$: skipped because of the loop test.
- For link $(5 \to 2)$: RDM$(5 \to 2)) = 1/6 - 3 - 2 - 4 < 0$; skipped
- For link $(5 \to 3)$: RDM$(5 \to 3) = 2/6 - 3 - 2 - 0 = 2$;

Hence the packet is forwarded via link $(5 \to 3)$.
The selected route is **1 → 5 → 3**; **Cost = 5**.

2. **Call 2:** source = 2; destination = 4; $\Delta = 7$;

   At node 2:

- For link $(2 \to 1)$: RDM$(2 \to 1) = 3/(7 - 0 - 3 - 2) = 1.5$
- For link $(2 \to 5)$: RDM$(2 \to 5) = 1/(7 - 0 - 2 - 4) = 1$
- For link $(2 \to 3)$: RDM$(2 \to 3) = 1/(7 - 0 - 6 - 2) < 0$; skipped

The packet is forwarded to node 5 via link $(2 \to 5)$.
At node 5:

- For link $(5 \to 2)$: RDM$(5 \to 2) = 2/(7 - 2 - 2 - 5) < 0$; skipped
- For link $(5 \to 1)$: RDM$(5 \to 1) = 3/(7 - 2 - 3 - 2) = \infty$
- For link $(5 \to 3)$: RDM$(5 \to 3) = 2/(7 - 2 - 2 - 2) = 2$

The packet is forwarded to node 3 via link $(5 \to 3)$.
At node 3:

- For link $(3 \to 2)$: skipped because it fails loop test
- For link $(3 \to 5)$: skipped because it fails loop test
- For link $(3 \to 4)$: RDM$(3 \to 4) = 1/(7 - 4 - 2 - 0) = 1$

The packet is forwarded to node 4 which is the destination.
The chosen route is **2 → 5 → 3 → 4**; **Cost = 4**

3. **Calls 3,4, and 5:** RDM will route call 3 along the path $2 \to 5 \to 3 \to 4$ path similar to call 3. After this, link (5,3) will now be saturated as it is used by calls 1,2, and 3. When call 4 arrives, RDM will choose (2,5) and forward the packet to node 5. At 5, link (5,3) will fail the

bandwidth test and link (5,1) will fail the delay test. Therefore backtrack to node 2. The link (2,1) will be the next preferred link at node 1 and the path 2 → 1 → 4 will be selected. The same path will be chosen for call 5 also. Hence all the calls will be accepted. In fact, RDM will be able to accomodate another call between (2,4) with the same bandwidth requirement.

4.2. Comments

The following table summarizes the performance of the two algorithms in the above example.

Call number	DCUR performance	RDM performance
1	Accepted; Cost = 8	Accepted; Cost = 5
2	Accepted; Cost = 10	Accepted; Cost = 4
3	Accepted; Cost = 10	Accepted; Cost = 4
4	Rejected	Accepted; Cost = 10
5	Rejected	Accepted; Cost = 10

The above example clearly illustrates where our proposed approach using the RDM heuristic scores over the DCUR algorithm. In the case of calls 1 and 2, the DCUR 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 RDM 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 its ability to search for alternate paths, the RDM algorithm is able to distribute the hot-pair communication load between nodes 2 and 4 among two different routes, thus providing greater call acceptance.

5. Experimental results

In this section, we present the results of the simulation experiments that were conducted to analyse and compare the performance of the proposed algorithms with the DCUR algorithm of Salama et al. [5]. We will first define the performance metrics, then describe the simulation model and finally present and discuss the results.

5.1. Performance metrics

For an accepted call-request 'R', let us define the functions:

- accepted(R) = 1.
- cost(R) = cost of the path chosen for R.
- setup(R) = number of vertices visited by the call setup packet.
- diet(R) = length of the path (in terms of hop-count) chosen for R.

For a call request, R, that is rejected, all the functions return a value of 0. Let 'N' be the total number of call requests generated. The following metrics were used to analyse the performance of the routing algorithms.

- **Average call acceptance rate (ACAR):** the average probability of accepting a real-time channel establishment request.

$$ACAR = \frac{\sum_{i=1}^{N} \text{accepted}(R)}{N}$$

- **Average cost (AC):** the average cost of the established channels.

$$AC = \frac{\sum_{i=1}^{N} \text{cost}(R)}{\sum_{i=1}^{N} \text{accepted}(R)}$$

- **Average call setup time (ACST):** the average time required to setup a real-time channel measured in terms of number of vertices visited by the call setup packet.

$$ACST = \frac{\sum_{i=1}^{N} \text{setup}(R)}{\sum_{i=1}^{N} \text{accepted}(R)}$$

- **Average routing distance (ARD):** the average hop-count of the established channels.

$$ARD = \frac{\sum_{i=1}^{N} \text{dist}(R)}{\sum_{i=1}^{N} \text{accepted}(R)}$$

The first metric is important, as it is a measure of network throughput. The second metric is also important, because cost minimization is one of the stated goals. Metric 3 is important in the context of real-time multimedia applications that require a call to put through quickly. Metric 4 is also important in the sense that a shorter route will in general consume less network resources and will therefore contribute towards improving network throughput and lowering cost.

5.2. Simulation model

To conduct the simulation studies, we have used randomly generated networks on which the algorithms were executed. The reason for using random networks instead of using existing real networks was to make the results independent of the characteristics of any particular network topology. Using randomly generated network topologies also provided the necessary flexibility to tune the network

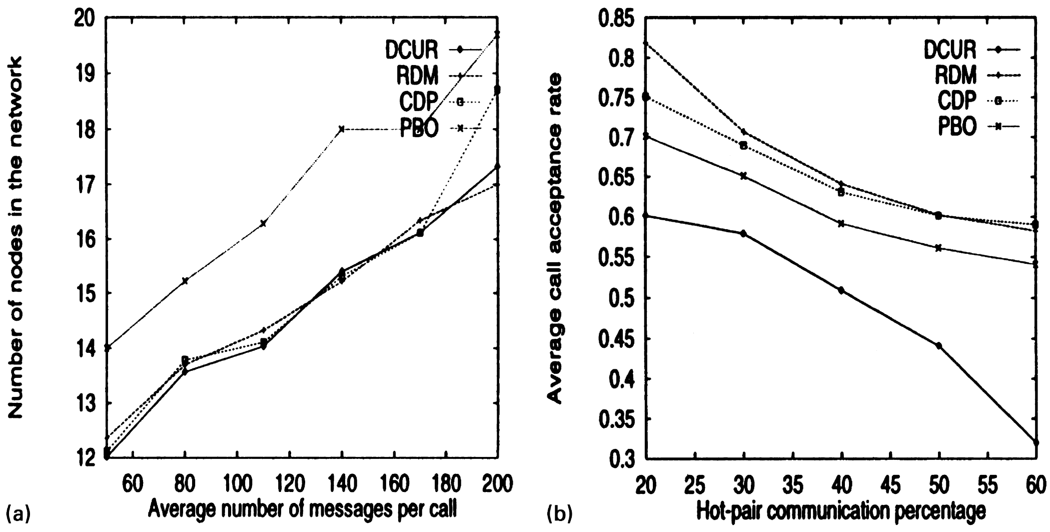


Fig. 1. (a) Effect of network size on average number of messages. (b) Effect of HCP percentage on ACAR.

parameters such as average degree, number of nodes, and number of edges, and to study the effect of these parameters on the performance of the algorithms.

5.2.1. Random graph generation

In generating random graphs, we have adopted the method used in [11], where vertices are placed randomly in a rectangular coordinate grid by generating uniformly distributed values for their x and y coordinates. The graphs' connectivity is ensured by first constructing a random spanning tree. This tree is generated by iteratively considering a random edge between nodes and accepting those edges that connect distinct components. The remaining edges of the graph are chosen by examining each possible edge (u,v) and generating a random number $0 \leq r < 1$. If r is less than a probability function $P(u,v)$ based on the edge distance between u and v, then the edge is included in the graph. The distance for each edge is the Euclidean distance (denoted as $d(u,v)$) between the nodes that form the end-points of the edge. We used the probability function

$$P(u,v) = \beta e^{\frac{-d(u,v)}{2\alpha n}}$$

where α and β are tunable parameters, and n is the number of nodes in the graph.

5.2.2. Simulation parameters

Except in the case of Fig. 1, all the networks used for simulation had 60 vertices. The parameters α and β were tuned to produce networks with average node degree 4. Random edge costs were generated uniformly from the set [1,10]. Edge delays were made proportional to the Euclidean distance of the edges in the coordinate plane. Each link in the network was assigned a total bandwidth of 100 units. Every simulation run consisted of a batch of 5000 call requests. Each point in every plot is the average over the values generated by 20 random networks with the above specified characteristics. Each plot compares the performance of the proposed algorithm using heuristics RDM, CDP, and PBO with the DCUR algorithm described in [5].

The call requests were generated with the following parameters.

- Source and destination nodes were chosen uniformly from the node set, except in the case of the hot-pair communication plot in Fig. 1b. In this plot, a specified p% of the calls always used one of three specifically chosen source-destination pairs, whereas the rest of the calls had randomly generated source and destination vertices. These specifically chosen vertex pairs therefore acted as hot-pair vertices.
- Call duration, bandwidth requirement, and delay constraint were uniformly distributed between their respective maximum and minimum values.
- The inter-arrival time of call establishment requests followed exponential distribution with mean $1/\lambda$.

The parameters used for simulation are summarized in the Table 2. Each entry represents the default values used for the specified parameter (i.e. when that parameter is not being used as the x-axis parameter).

5.3. Discussion of results

The performance of the proposed heuristics (RDM, CDP, and PBO) and the DCUR algorithm [5] were studied under two different network models, static and dynamic.

5.3.1. Static model

In this model, of the four link parameters (cost, delay, available bandwidth and total bandwidth) only the available bandwidth value changes as calls dynamically reserve

Table 2
Parameters and default values

Parameter	Default values
Inter-arrival time of call requests	Exponential distribution with mean $1/\lambda = 2$
Delay constraint	Uniformly distributed in (20,30)
Bandwidth requirement	Uniformly distributed in (4,8)
Call duration	Uniformly distributed in (100,200)
Number of preferred links(κ)	3
Table Update periodicity (T_{update})	Every 300 calls
Perturbation periodicity (T_{perturb})	Every 300 calls

resources during setup and release them when they are torn down.

1. **Effect of bandwidth requirement:** Fig. 2a–d represents the effect of bandwidth requirement of the calls on the performance metrics. The parameter that is varied on the x-axis is the maximum bandwidth requirement.

- *Effect on ACAR*: as the bandwidth requirement increases, the ACAR decreases for all the algorithms as it becomes increasingly tough to find links with the required available bandwidth. The PBO, RDM and CDP algorithms perform better than the DCUR algorithm, because they probe for alternative paths much better than DCUR does. In the case of DCUR, there is atmost a two-way probe. As multiple calls between a given pair of nodes are generated, after a while, the least delay path and least cost path between these two nodes become saturated, thus preventing further calls from being accepted. The PBO, RDM, and CDP algorithms however probe for alternate paths and thereby distribute the load more uniformly.
- *Effect on AC*: as the bandwidth requirement gets tighter, higher cost edges might need to be chosen in favour of lower cost edges, because the latter might already be saturated. Hence the general trend is an increase in AC

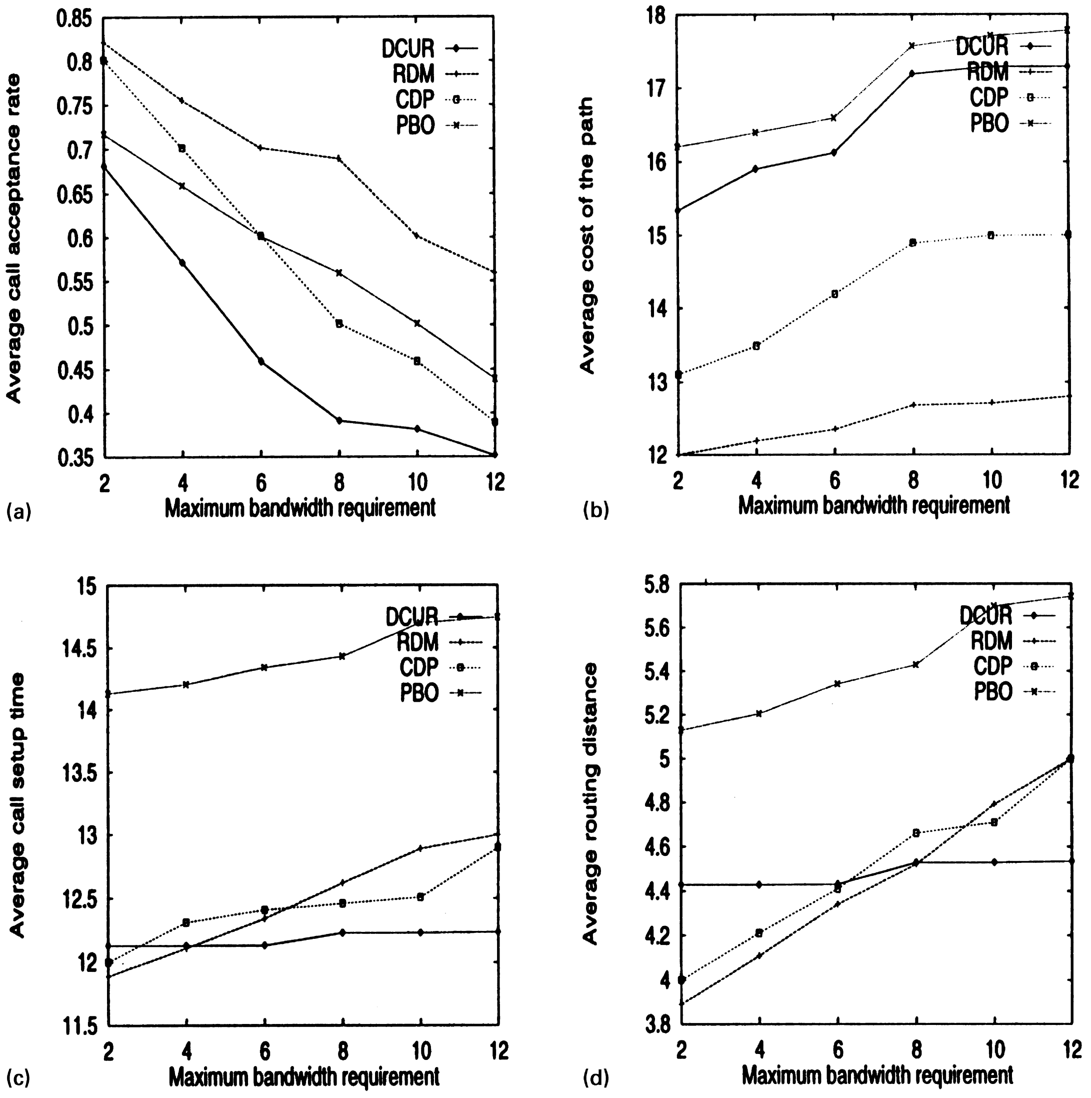


Fig. 2. (a) Effect of max b/w on ACAR. (b) Effect of max b/w on AC. (c) Effect of max b/w on ACST. (d) Effect of max b/w on ARD.

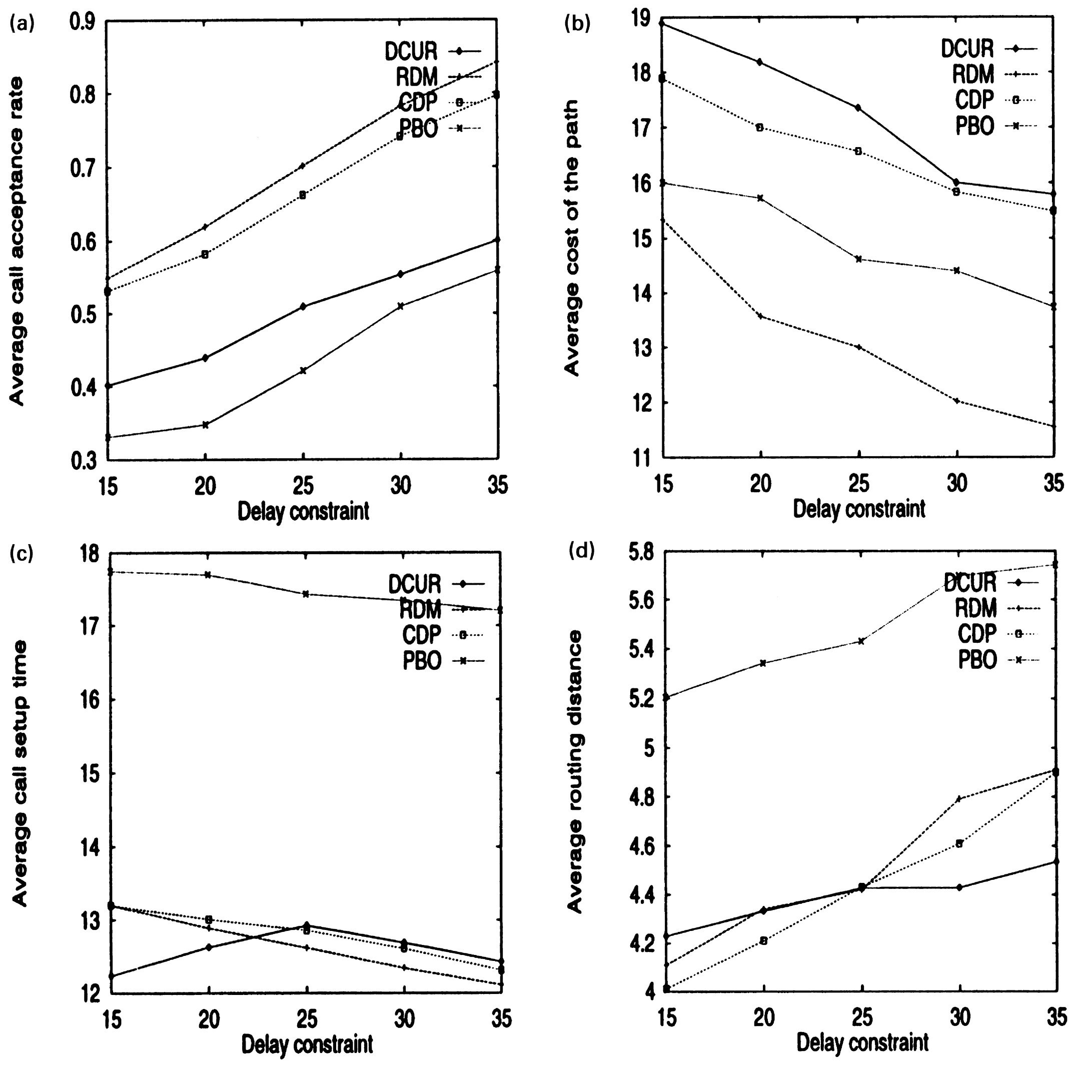


Fig. 3. (a) Effect of delay constraint on ACAR. (b) Effect of delay constraint on AC. (c) Effect of delay constraint on ACST. (d) Effect of delay constraint on ARD.

with increase in bandwidth requirement. However, the more extensive search of alternate paths by the proposed algorithms accounts for the improved performance over DCUR.

- *Effect on ACST and ARD*: similar to the case of AC, shorter routes might not be available with the required bandwidth and hence ACST and ARD increase. Heuristics RDM and CDP provide performance that is close to the DCUR algorithm which, because of its restricted search, provides lowest ACST values.

2. **Effect of delay constraint:** the plots in Fig. 3a–d present the effect of increasing delay-constraint.

- *Effect on ACAR*: as the delay constraint becomes less tighter the acceptance rate increases.
- *Effect on AC*: as the delay constraint becomes less tighter, the algorithms are able to choose low-cost edges even if the they have a higher delay value. Therefore AC decreases for all algorithms.

3. **Effect of call arrival rate:** the plots in Fig. 4a–d present the effect of increasing the call-arrival rate. As call arrival rate increases there is a drop in ACAR and increase in AC, ACST, and ARD. This is due to more calls competing for the network resources. The proposed heuristics are able to manage the resources more efficiently than DCUR and hence exhibit higher ACAR and lower AC.
4. **Effect of κ:** the plots in Fig. 5a–d present the effect of κ on the performance parameters. The plots in Fig. 5a and b show that as the maximum number of preferred links increase, there is a general increase in ACAR and a drop in AC. However, this trend continues only up to a value of $\kappa = 4$. For $\kappa > 4$, the plots flatten out. This observation is consistent with the intuitive reasoning that choosing κ much larger than average degree ($= 4$ in this case) will not yield too much improvement. As κ increases, there is scope for a larger number of links to be attempted at each node. This could result in a larger setup time, as indicated

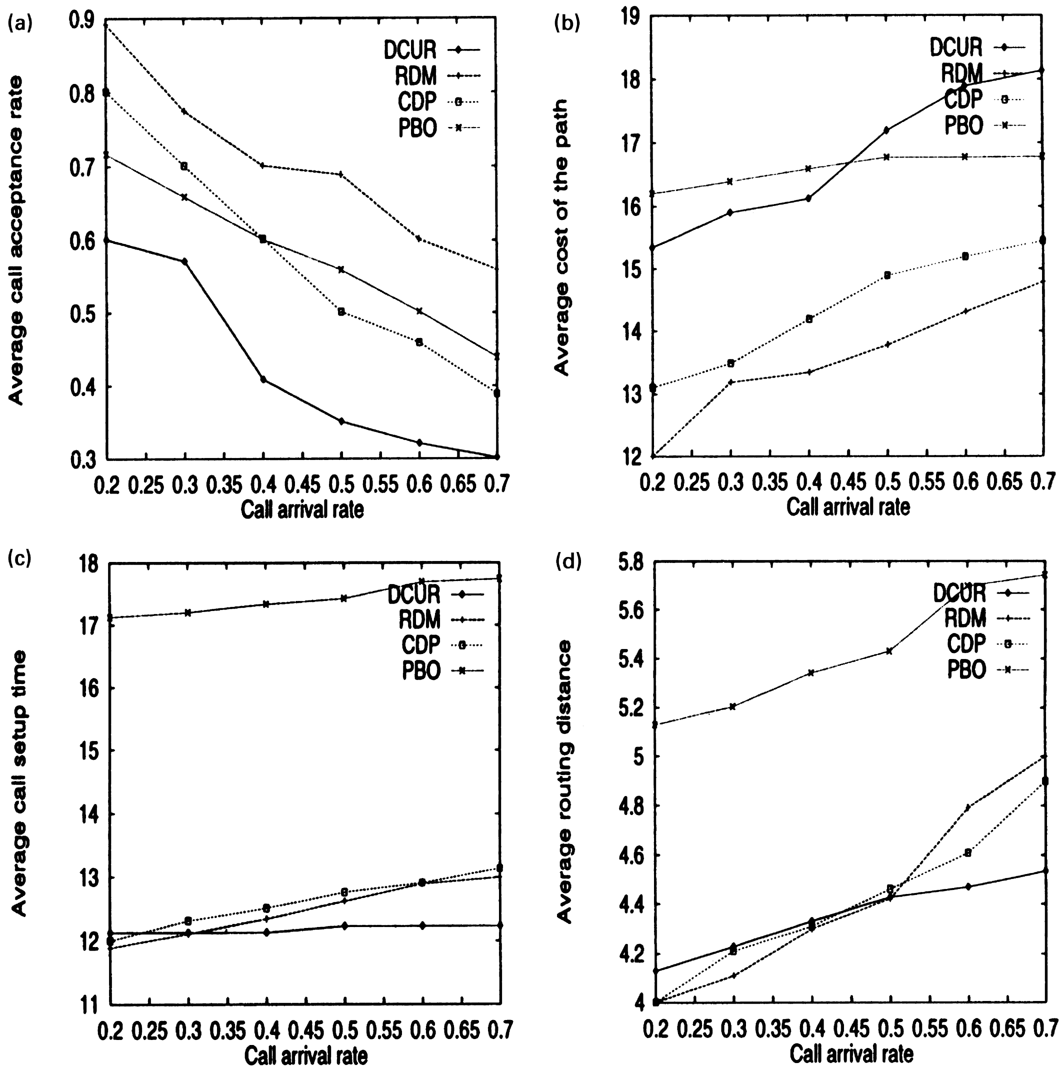


Fig. 4. (a) Effect of call arrival rate on ACAR. (b) Effect of call arrival rate on AC. (c) Effect of call arrival rate on ACST. (d) Effect of call arrival rate on ARD.

in Fig. 5c. Fig. 5d illustrates the fact that a destination-independent heuristic like PBO produces longer routes (larger ARD) when compared with destination-dependent heuristics like RDM and CDP. It also indicates that, as expected, κ does not influence the ARD metric as significantly as it influences the other three metrics.

5. **Effect of average degree:** the effect of average node degree on the ACAR metric is plotted in Fig. 6. As the average degree of the nodes in the network increases, the RDM, PBO, and CDP heuristics utilize the greater connectivity in the network much better than DCUR does. Hence they exhibit much higher ACAR compared with the DCUR algorithm with the performance gap widening as the degree increases.
6. **Effect of network size on number of messages:** the plot in Fig. 1a indicates that the three heuristics do not generate any message explosions and scale well to larger networks. Heuristics RDM and CDP are powerful enough to keep the search directed towards the destination and provide performance comparable to DCUR (which, because of its restricted search, is expected to provide the best performance). PBO, which is destination- and call-independent, provides the least impressive performance with regard to number of messages per call.
7. **Effect of hot-pair communication:** the plot in Fig. 1b portrays the influence of hot-pair communication on the call acceptance rate of the algorithms. As the hot-pair communication percentage increases, there is a general decline in the acceptance rates of all the algorithms, because of saturation of the links connecting the two hot-pair vertices. However, it is seen that the proposed heuristics, because of their ability to adapt and search alternative paths, perform better than the DCUR algorithm.

5.3.2. Dynamic model

In the dynamic model of the network, besides the variation in the available bandwidth at each link, the cost

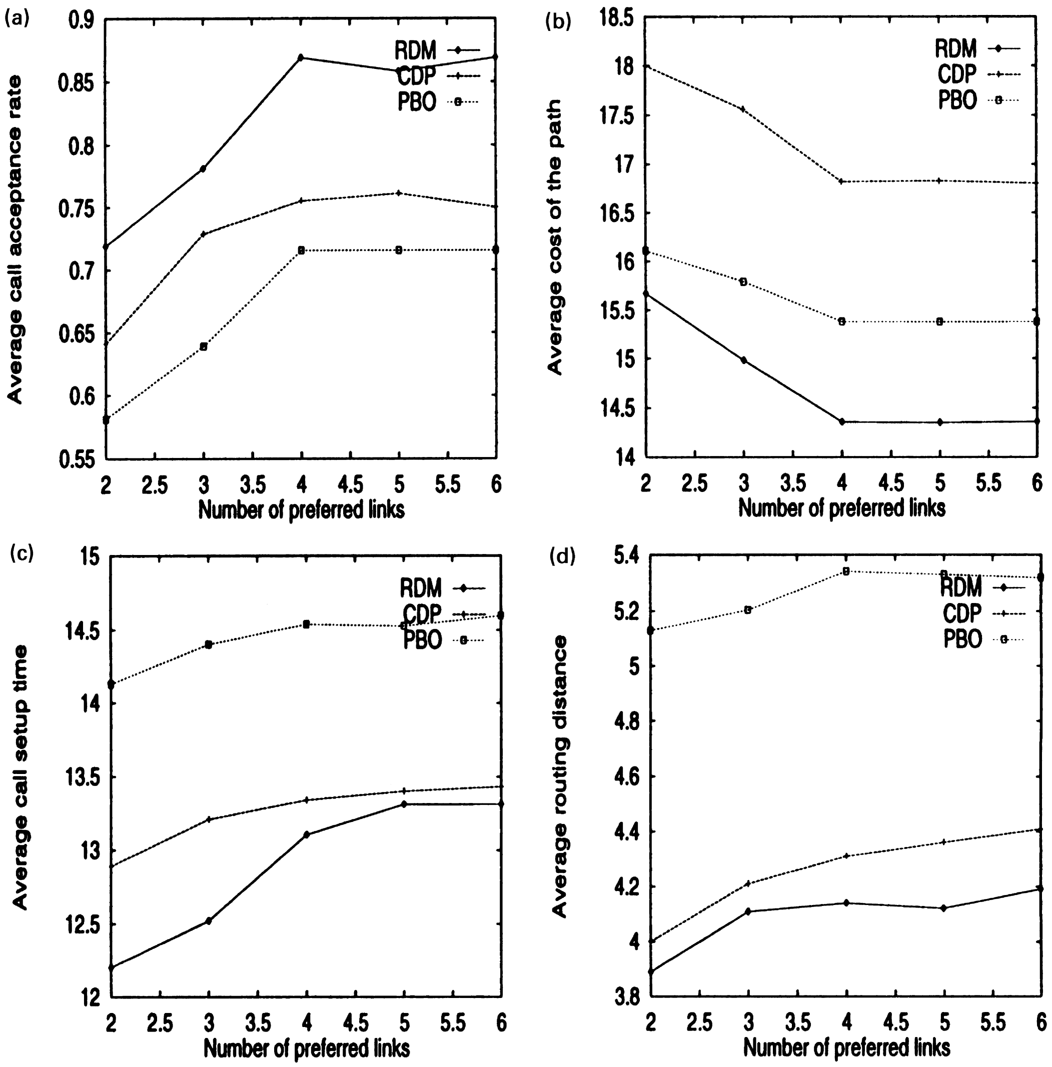


Fig. 5. (a) Effect of κ on ACAR. (b) Effect of κ on AC. (c) Effect of κ on ACST. (d) Effect of κ on ARD.

and delay values are also allowed to vary with time. This model attempts to capture the variations in link parameters with time, either because of physical reasons or because of the network traffic characteristics that result in certain areas of the network getting congested. We assume that in order to maintain routing tables consistently, there is an underlying protocol which executes to exchange and disseminate information about link changes to all nodes in the network. In order to quantify the dynamic nature of the network and the periodicity of information exchange, two new parameters T_{perturb} and T_{update} were introduced. Since call arrivals are distributed with fixed arrival rate, these parameters were defined in terms of number of calls rather than in terms of timing parameters. For all T_{perturb} calls, the cost and delay values of various links in the network were changed. For this, an edge was chosen at random and its cost was arbitrarily increased or decreased by a fixed percentage. A similar process was done for the delay values of the links. The percentage of edges to be perturbed and the extent of perturbation were chosen (after some experimentation) to be 20% and 35%, respectively. Every T_{update} calls, the entries in the distance vector tables (namely the *LDNHOP, LCNHOP, LDELAY,* and *LCOST* functions defined in Section 3.5) of all the nodes were updated to reflect the changed link parameters.

1. **Effect of periodicity of perturbation:** the plots in Fig. 7a and b represent the effect of T_{perturb} on acceptance rate and average cost. The performance of the DCUR algorithm shows a marked improvement as the frequency of perturbation is decreased (i.e. as T_{perturb} is increased). The proposed heuristics show much less dependence on this parameter, and adapt to link state changes better than DCUR. The reason for this lies in the fact that DCUR uses the cost and distance vector tables to decide the next node to which the routing packet is to be forwarded. Hence the final path produced by DCUR is restricted to being an interleaving of the least-delay and least-cost paths. When these paths become out-of-date, the resultant path chosen by DCUR is also poor. The PBO

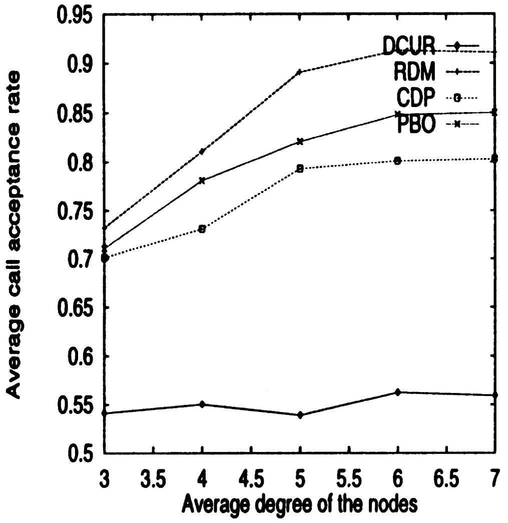


Fig. 6. Effect of average degree on ACAR.

heuristic, which is destination- and call-independent shows a flat response as expected. The RDM heuristic uses only the *LDELAY* values to guide the routing decision and does not directly use the *LDNHOP* values as used by DCUR. Therefore it does not suffer as much as DCUR if the tables are out-of-date. Of the three heuristics, only CDP uses the *LDNHOP* and *LCNHOP* functions and hence provides the least impressive performance even though it is still better than DCUR.

2. **Effect of table updates:** the explanation for the trends exhibited in plots in Fig. 8a and b is identical to the explanation for the plots in Fig. 7a and b.

5.3.3. Overall comparison with DCUR

DCUR works by essentially restricting its search to least-cost and least-delay neighbours (i.e. at node v, the route is extended via either LCNHOP(v) or LDNHOP(v)). The main requirement for the execution of DCUR is the need to store two distance–vector tables, one each for the delay and cost metrics. Our algorithm, on the other hand, requires the maintenance of a PLT, which is constructed using any of our proposed heuristics. Besides the information required to construct the DCUR tables, our heuristics only require knowledge of local link properties (such as the cost and delay of links adjacent to a given node) which can be easily and accurately monitored. Hence, without additional overhead, our heuristics provide the following benefits: higher acceptance rates; lower costs; lower routing distances; better utilization of network resources; facility to achieve a trade-off between optimality and setup time (using κ), and better adaptation to variations in link parameters. We have also shown, through simulation, that the RDM and CDP heuristics provide setup times that are comparable to those of DCUR. This indicates that the two heuristics provide the above-mentioned benefits by only visiting, on average, as many nodes as DCUR visits. The PBO heuristic is useful when the network is highly dynamic, but is generally quite poor with regard to ACST and ARD, as its search is unrestricted and independent of the destination.

6. Conclusion

In this paper, we adapted the preferred link routing approach to delay-constrained least-cost routing for real-time channel establishment and presented a set of heuristics that could be employed with this approach. We also presented simulation results that have shown that the suggested heuristic functions are able to provide increased netwok throughput, better adaptiveness, and lower average cost than DCUR [5], a recently proposed algorithm. Our simulation studies have revealed the following advantages of our proposed algorithm:

- Since the route search is essentially by probing and there is no fixed precalculation, as is the case with distance

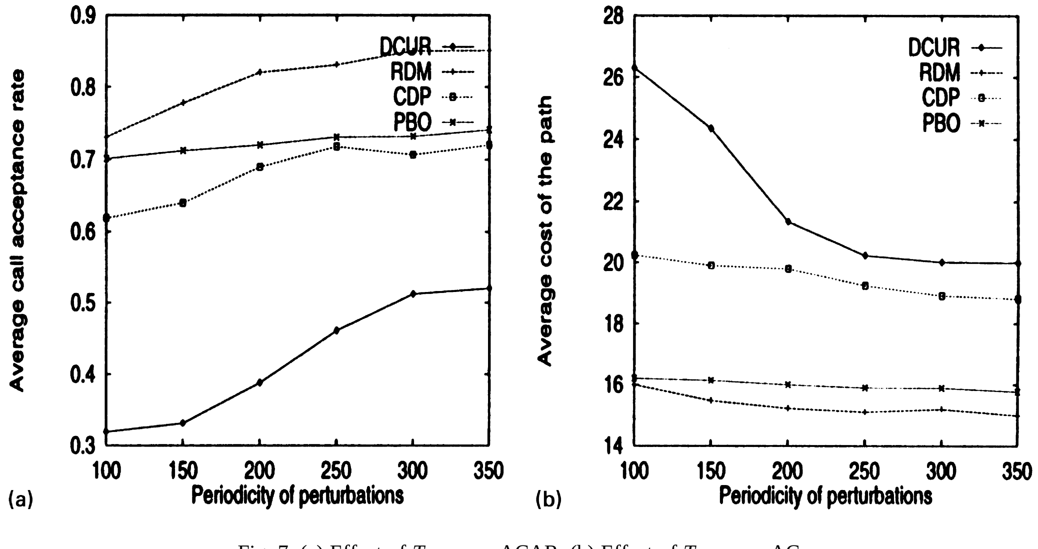
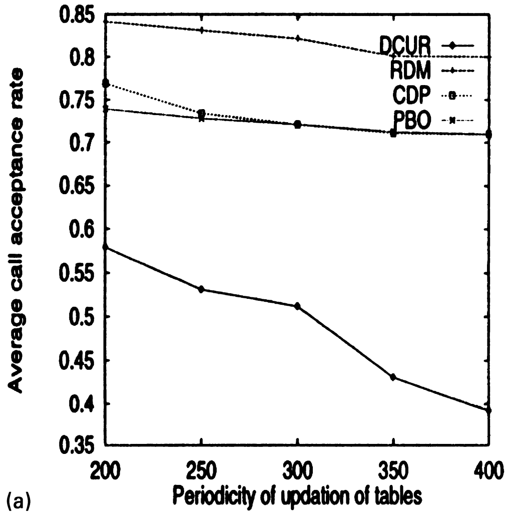


Fig. 7. (a) Effect of T_{perturb} on ACAR. (b) Effect of T_{perturb} on AC.



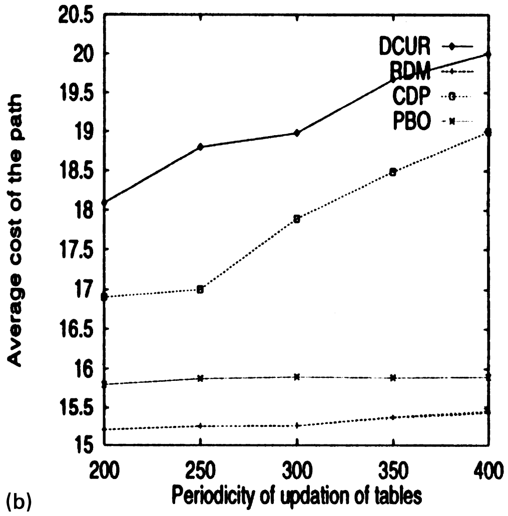


Fig. 8. (a) Effect of T_{update} on ACAR. (b) Effect of T_{update} on AC.

vector based algorithms, the proposed approach is more responsive to network changes.

- The heuristic functions use a minimum of global information, basing most of their decisions on local information at each node. Therefore there is less overhead required to communicate link state changes to the rest of the network.
- The approach combines resource reservation with probing thus avoiding a separate reservation phase.
- The algorithm provides for trade-off between lower setup time and optimality of the route by suitably selecting the maximum number of preferred links to be used at each node.

Areas for future research include development of improved heuristic functions that exploit the possibility of adaptively using different heuristics at different nodes along a route. We are also currently investigating the extension of this approach to constrained multicast routing.

References

[1] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, IEEE JSAC 14 (7) (1996) 1228–1234.
[2] N. Huang, C. Wu, Y. Wu, Some routing problems in broadband ISDN, Computer Networks and ISDN Systems 27 (1994) 101–116.
[3] M.R. Garey, D.S. Johnson, Computers and Intractability: A guide to the theory of NP-completeness, W.H. Freeman, 1979.
[4] K.G. Shin, C. Chou, A distributed route-selection scheme for establishing realtime channels, High Performance Networking (1995) 319–330.
[5] H.F. Salama, D.S. Reeves, Y. Viniotis, A distributed algorithm for delay-constrained unicast routing. IEEE INFOCOM, 1997.
[6] R. Widyono, The design and analysis of routing algorithms for real-time channels. Tech. Rep. ICSI TR94-024, University of California at Berkeley, International Computer Science Institute, June 1994.
[7] J. Jaffe, Algorithms for finding paths with multiple constraints, Networks 14 (1) (1984) 95–116.
[8] S. Shenker, L. Breslau, Two issues in reservation establishment. ACM SIGCOMM, 1995.
[9] D. Bertsekas, R. Gallager, Data Networks, 2nd edn. Prentice-Hall International, 1992.
[10] V.P. Kompella, J.C. Pasquale, G.C. Polyzos, Multicast routing for multimedia communication, IEEE/ACM Trans. on Networking 1 (1993) 286–292.
[11] B.M. Waxman, Routing of multipoint connections, IEEE JSAC 6 (1988) 1617–1622.
[12] C.M. Aras, J.F. Kurose, D.S. Reeves, H. Schulzrine, Real-time communication in packet-switched networks, Proc. IEEE 2 (1) (1994) 122–139.

R. Sriram obtained the B.Tech. degree in Computer Science and Engineering from Indian Institute of Technology, Madras, in 1998. He is a recipient of President of India Gold Medal for academic excellence in the B.Tech programme. Currently, he is a research student at the Computer Science Department, Stanford University, USA. His research areas of interests are routing, multicasting, and fault-tolerance in real-time networks.

G. Manimaran obtained the B.E. degree in Computer Science and Engineering from Bharathidasan University, Thiruchirappalli, in 1989, M.Tech. in Computer Technology from the Indian Institute of Technology, Delhi, in 1993, and Ph.D in Computer Science and Engineering from the Indian Institute of Technology, Madras, in 1998. His research interests are resource management in parallel and distributed real-time systems, real-time networks, and fault-tolerant computing.

C. Siva Ram Murthy obtained the B.Tech. degree in electronics and communications engineering from Regional Engineering College, Warangal, in 1982, M.Tech. in computer engineering from Indian Institute of Technology (IIT), Kharagpur, in 1984, and Ph.D. in computer science from Indian Institute of Science (IISc), Bangalore, in 1988. From March 1988 to September 1988 he worked as a Scientific Officer in the Supercomputer Education and Research Centre at IISc. He subsequently joined IIT Madras as a Lecturer of Computer Science and Engineering. He became an Assistant Professor in August 1989 and is currently an Associate Professor at the same place. He has held visiting positions at German National Research Centre for Information Technology (GMD), Sankt Augustin, Germany, University of Washington, Seattle, USA, and University of Stuttgart, Germany. He is a recipient of the Seshagiri Kaikini Medal for the best Ph.D. thesis and also of the Indian National Science Academy Medal for Young Scientists.