March 1999

Multi-cost Routing in Max-Min Fair Share Networks[1]

by

Francisco Javier Gutierrez, Emmanouel A. Varvarigos[2], and Stamatis Vassiliadis[3]

Abstract

Traditional routing algorithms use a single cost measure for each network link, and choose the optimum path according to a least cost criterion. We present a multi-cost routing algorithm that uses several cost parameters for each link. A set of candidate paths, called non-dominated paths, is obtained, and the optimum path is chosen according to an optimization function. The only limitation on the optimization function is that it is assumed to be monotonic with respect to each of the cost parameters. We compare the multi-cost routing algorithm for several choices of the optimization function with two traditional routing algorithms: the algorithm that chooses the widest-shortest path and the algorithm that chooses the shortest-widest path. The performance of the three algorithms is compared for a max-min fair share network. This network operation has been identified by the ATM Forum as one of the main design goals for ABR traffic management algorithms.

[1] Research supported by DARPA under the MOST project.

[2] University of California, Department of Electrical and Computer Engineering, Santa Barbara, CA 93106.
[3] T.U. Delft, Department of Electrical Engineering, Delft, the Netherlands.

1. INTRODUCTION

The traditional routing approach is to assign to each link a single cost measure, which may in general be a function of several network parameters, and execute an algorithm to obtain the minimum cost path [ScS80]. Routing algorithms of this kind limit the type of cost criteria that can be used for selecting a path, and do not permit different sessions to be treated in different ways, depending on the quality of service they require. Moreover, a single path is usually obtained for each source-destination pair, which leads to non-uniform distribution of traffic and the potential of oscillation problems [BeG87].

In this paper we examine the more general approach, where a vector of cost parameters is assigned to each link. A cost vector is then found for each path, based on the cost vectors of its links. Given a pair of nodes and the set of paths between them, an optimization function is applied to the cost vector of each path to select the optimal path. We will refer to this approach as *multi-cost routing* and to the more traditional approach as *single-cost routing*. Although the two approaches may look similar, they are actually rather different. Single-cost routing combines the different parameters for a link in a single cost parameter for that link, often called the length of the link, and then obtains the minimum length path. Multi-cost routing considers the network parameters separately until the very end, when a function of them is optimized. The function that is optimized can be different for each session, depending on the needs of the application. In spite of its potential advantages, multi-cost routing has been rarely studied. General multi-cost routing with two cost parameters has been considered in [Han80], [Han82]. Particular cases of the bicriterion path problem have been considered in [Mar84], [Ahu88], [ChH96]. Also a great deal of work has been done on the quickest path problem; see, for example, [ChC90], [RSX91], [HuC91].

A multi-cost routing algorithm consists of two phases. In the first phase, an enumeration of candidate paths for a given source-destination pair is performed. In the second phase, the optimum path is chosen from this set according to some optimization function. If the optimization function is arbitrary, all the paths between a pair of nodes may need to be considered in order to choose the optimum one. This would involve a much bigger effort in computation and communication than in single-cost routing. Fortunately, when the function that is optimized is monotonic in each of its parameters, as is the case with most cost functions of interest, the number of paths that are considered is greatly reduced by discarding some of them at intermediate phases of the routing algorithm. A set of non-dominated paths is obtained for each source-destination pair, and an optimization function (that may be different for different connections) is applied to select one of them. By *non-dominated path*, we mean that it is not possible to find another path that is better with respect to one of the cost criteria without being worse with respect to some other criterion. The monotonicity conditions that the optimization function has to satisfy are not very restrictive, allowing us to consider the routing problem with enough generality.

The choice of the optimization function used to select the optimum path from the set of non-dominated paths can be thought of as a "political" problem. In Section 5 we consider several choices of the optimization function and compare the performance obtained by each of them for max-min fair share networks. In max-min fair share networks all the connections using a link obtain a "fair share" of the link capacity. The concept of max-min fair share has been identified by the ATM forum as one of the main design goals for ABR traffic management algorithms. The nature of bandwidth sharing in such networks makes the notion of "available capacity" on a link different than what we are used to. We examine both regular and irregular topologies, under a variety of assumptions on the link capacities and the size of the data carried by each session. For the networks examined, we find that the optimization function $f(d,b,r,h) = (d+b/r)\cdot h^{0.5}$, where d is the propagation delay, r is the available (in the max-min fair share sense) capacity, b is the amount of data to be transfered and h is the number of hops, outperforms the widest-shortest and the shortest-widest routing algorithms as well as other multi-cost routing algorithms, by taking into account both the interests of the network and the interests of the user. We also investigate the effect that long propagation delays and outdated link utilization information have on the performance of the routing algorithms.

The remainder of the paper is organized as follows. In Section 2 we give some definitions and obtain some preliminary results. In Section 3 we present some properties of the set of non-dominated paths and give a formal description of a Dijkstra-based algorithm to obtain this set. In Section 4, we consider the implementation of the multi-cost routing algorithm. Finally, in Section 5 we compare multi-cost routing for several choices of the optimization function to two traditional single-cost routing algorithms for max-min fair share networks.

2. DEFINITIONS AND PRELIMINARY RESULTS

A *network* is represented by a *directed graph* $G = (N, A)$, where N is a set of nodes and A is a set of ordered pairs of nodes. We assume that there are no self-loops or multiple arcs with the same direction between two nodes. A *path* in the network is defined as a sequence of distinct nodes $(n_1, n_2, \ldots, n_l)$. Each link j of the network is assigned a k-dimensional cost vector

$$\mathbf{v_j} = (v_{1j}, v_{2j}, \ldots, v_{kj}),$$

where $v_{ij} \geq 0$ for all i. Each path is assigned a cost vector

$$\mathbf{V} = (V_1, V_2, \ldots, V_k)$$

whose components are obtained from the link cost components according to some *monotonic associative operator* $\oplus$, that is, $V_i = \oplus_{j=1}^{l} v_{ij}$. By monotonic we mean that either $a \oplus b \geq a$ for all $a, b \geq 0$, or that $a \oplus b \leq a$ for all $a, b \geq 0$ (but not both). The *cost of a path* is given by a function $f(V_1, V_2, \ldots, V_k)$ of its cost vector. The objective of the routing algorithm is to choose the path that has the the minimum cost.

Each component of the path cost vector is characterized by the way it is obtained from the link cost vector components (that is, by the associative operator $\oplus$), and also by the criterion that has to be applied to it (that is, minimization or maximization). For example, if $f(V_1, V_2, \ldots V_k)$ is an increasing (or decreasing) function of V_i and it has to be minimized, then we have a minimization (or maximization, respectively) problem for V_i. We are particularly interested in the case where each cost component V_i of a path made up of l links is obtained from the link cost components in one of the following two ways:

- $V_i = \sum_{j=1}^{l} v_{ij}$, $v_{ij} \geq 0$, and f is monotonically increasing in V_i (so our objective is to minimize V_i), or
- $V_i = \min_{j=1,\ldots,l}\{v_{ij}\}$, $v_{ij} \geq 0$, and f is monotonically decreasing in V_i (so our objective is to maximize V_i).

Cost parameters of the first type will be referred to as *additive cumulative*, while cost components of the second type will be referred to as *restrictive.*

Additive cumulative parameters include several important cost measures used in practice. For example, if v_{ij} is the delay on link j, then V_i represents the delay of the path, which in most practical situations has to be minimized. If $v_{ij} = 1$ for all links j, then V_i corresponds to the number of hops on the path. Since paths that use a small number of links are more economical in terms of resource utilization, it is natural to assume that the cost function f is an increasing function of V_i. Another interesting case arises when $v_{ij} \in [0, 1]$ represents the probability that link j is operational, and $V_i = \prod_{j=1}^{l} v_{ij}$ is the probability that all links on a path are operational (assuming links fail independently of each other). For the routing algorithm to favor reliable paths, the cost function f should be a decreasing function in V_i. This problem can be reduced to a problem involving cumulative additive components by defining new cost components $v'_{i1} = -\ln v_{i1}, \ldots, v'_{il} = -\ln v_{il}$, where $v'_{ij} \geq 0$. Then maximizing the reliability V_i of a path is equivalent to minimizing $V'_i = \sum_{j=1}^{l} v'_{ij}$.

Restrictive cost parameters appear in routing problems when capacities or transmission rates are considered. In particular, if v_{ij} is the available capacity on link j, then $V_i = \min_{j=1,\ldots,l}\{v_{ij}\}$ represents the capacity of a path, which is defined as the minimum of the capacities available on the links of the path. For the routing algorithm to favor less congested paths, the cost function f should be a decreasing function of V_i.

Other choices for the monotonic associative operator $\oplus$ used to obtain a path cost component from the corresponding link cost components are also useful in routing problems. For example, if the parameter v_{ij} is Boolean (representing some property of link j, such as, that the link is operational, or reliable, or congested, or that its end-nodes can handle a certain protocol), then the OR, AND, and other Boolean operators (but not the exclusive OR operator, which is not monotonic) could also be used to calculate an appropriate cost parameter for the path. We will not discuss explicitly Boolean operators in this paper, but the algorithms that we will give can be applied to such cases as well.

We denote by P the *set of all the possible paths* between a given pair of nodes. For a given cost function $f : R^k \to R$ we define the *set* $P_{opt} \subset P$ *of optimum paths* between the nodes, as the set of all paths that minimize f. It is also useful to define a *domination relation* among the paths in P as follows. Let p and p' be two paths in P, and $\mathbf{V} = (V_1, V_2, \ldots, V_k)$ and $\mathbf{V}' = (V_1', V_2', \ldots, V_k')$ be their associated cost vectors, respectively. We assume, without loss of generality, that the first s components of these vectors are additive cumulative and the $k - s$ following components are restrictive; f is then assumed to be an increasing function of the first s components and a decreasing function of the last $k - s$ components. We say that path p dominates path p' if

$$V_1 \le V_1', \ldots, V_s \le V_s',$$

$$V_{s+1} \ge V_{s+1}', \ldots, V_k \ge V_k',$$

and the inequality holds at least once. It is clear that if p dominates p', the cost of p will be smaller than that of p'.

We define the *set* Q *of non-dominated paths* between two nodes as the subset of P that is obtained once all the dominated paths have been eliminated. In other words, it is not possible to find in Q a pair of paths that verify the domination relation previously described.

The following proposition allows us to obtain the optimum path from a reduced set of paths.

Lemma 1: The set P_{opt} of optimum paths between two nodes is contained in the set Q of non-dominated paths, that is, $P_{opt} \subset Q$.

Proof: We assume for the sake of contradiction, that there exists a path $p \in P_{opt}$ for which $p \notin Q$. Then there is a path $p' \in Q$ that dominates p. Let $\mathbf{V} = (V_1, V_2, \ldots, V_k)$ and $\mathbf{V}' = (V_1', V_2', \ldots, V_k')$ be the cost vectors of p and p', respectively. We assume that the first s components of these vectors are additive cumulative and the $k-s$ following components are restrictive. Since p' dominates p, we have $V_1' \le V_1, \ldots, V_s' \le V_s$, $V_{s+1}' \ge V_{s+1}, \ldots, V_k' \ge V_k$, and the inequality holds at least once. Since the cost function f is an increasing function of the first s components and a decreasing function of the remaining $k-s$ components, we have $f(\mathbf{V}') < f(\mathbf{V})$. This means that p is not an optimum path, which contradicts our assumption. **Q.E.D.**

Since all the optimum paths are included in the set Q of non-dominated paths, the multi-cost routing algorithm can restrict the search for optimum paths to this set.

3. OBTAINING THE SET OF NON-DOMINATED PATHS

In this section we present an algorithm for obtaining the set of non-dominated paths from a given source node to all destination nodes. Before doing so, it is useful to describe some properties of the set Q of non-dominated paths.

3.1 Some properties of the set of non-dominated paths

We let P be the set of all the paths between two particular nodes, and $Q \subset P$ be the set of all the non-dominated paths in P. We limit ourselves to the case where the cost vector $\mathbf{V} = (d, c)$ consists of only $k = 2$ components, where d is additive cumulative, while c is restrictive. We represent the path cost vector $\mathbf{V} = (d, c)$ as a point in the 2-dimensional space, and the set P as a set of such points, as shown in Fig. 1a. Recall that a path p with cost vector $\mathbf{V} = (d, c)$ dominates a path p' with cost vector $\mathbf{V}' = (d', c')$ if $d \leq d'$ and $c \geq c'$ and the inequality holds at least once. Therefore, if we imagine the point representing a path as the center of a four quadrants graph, this path dominates all the paths in the lower right quadrant (Fig. 1b), which can then be eliminated. The set Q of non-dominated paths can be obtained graphically by successively using this rule, as shown in Fig. 1b.

Lemma 2: Let P_{dmin} be the set of paths in P that minimize the first component d of the cost vector, and $p_{dmin} \in P_{dmin}$ be the path that maximizes the second component. Let also P_{cmax} be the set of paths in P that maximizes the second component c, and $p_{cmax} \in P_{cmax}$ be the path that minimizes the first component. Then, $p_{dmin} \in Q$ and $p_{cmax} \in Q$.

Figure 1 shows graphically that both p_{dmin} and p_{cmax} belong to Q. The following lemma describes the transitive property of the domination relation.

Lemma 3: If path p dominates p' and p' dominates p'', then p dominates p''.

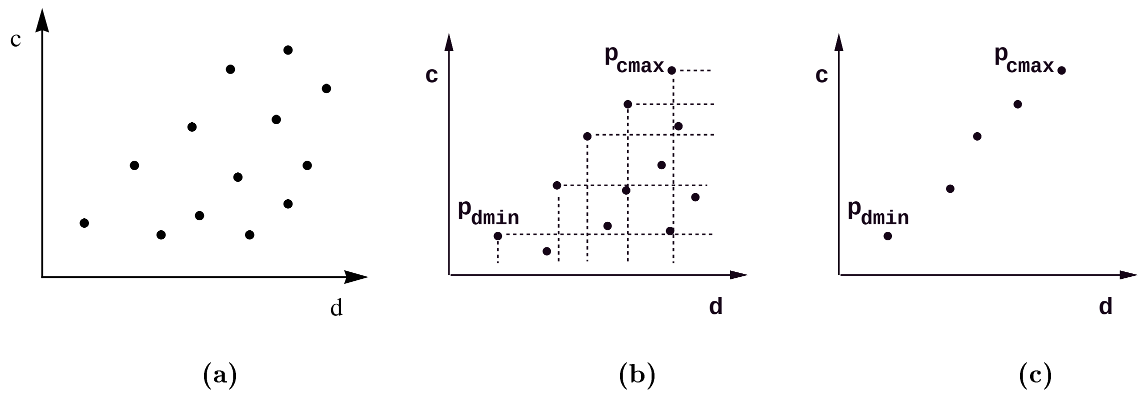


Figure 1: Graphic representation of the paths: (a) The set P of all the paths. (b) Obtaining the set Q. (c) The set Q of non-dominated paths. Paths p_{dmin} and p_{cmax} are contained in it.

Note that the path in Q that minimizes the additive component is also the one that minimizes the restrictive component, and the path that maximizes the restrictive component is also the path that maximizes the additive component. Also, the paths in Q form an increasing succession in both components. These properties can be generalized to the case where $k > 2$, and will be the basis for the algorithm given in the next section for obtaining the set Q of non-dominated paths.

3.2. Algorithm for computing the set of non-dominated paths

In this section we present an algorithm for obtaining the set of non-dominated paths from a given node to all other nodes of the network. The algorithm can be viewed as a generalization of Dijkstra's algorithm [Dij59], which only considers networks with single costs.

Dijkstra's algorithm requires all arc lengths to be nonnegative. Our algorithm inherits this requirement, which is consistent with the assumptions made in Section 2 for the cost vector components. The general idea of Dijkstra's algorithm is to find the shortest paths in order of increasing length, beginning with the path to the neighbor that is closest to the origin node and progressively finding the shortest paths to more distant nodes. Our algorithm uses a similar approach to obtain the set of non-dominated paths. It first obtains a non-dominated path between the origin and a direct neighbor; then a non-dominated path between the origin and one of its neighbors or between the origin and one of the neighbors of the previously considered neighbor and so on. The basic difference with Dijkstra's algorithm is that a set of non-dominated paths between the origin and a destination node is obtained, instead of a single path, and a destination for which a path has already been found may have to be considered again later.

To clarify the operation of the algorithm we first describe the case where the cost vectors consist of only $k = 2$ components: the delay of the link (which is additive cumulative) and the capacity of the link (which is restrictive). The algorithm starts by considering the neighbors of the origin node, and choosing the one with the smallest link delay; in case of a tie, the one with the largest capacity is chosen. We have already obtained the first non-dominated path between the origin and that neighbor. Clearly, this path is non-dominated because the delay is an additive cumulative parameter and no other path can have a smaller delay. The algorithm then considers the neighbors of these two nodes and chooses the one that has the smallest delay to the origin and so on.

Before giving a formal description of the algorithm, we present the operation of the algorithm for the graph of Fig. 2. The evolution of the algorithm for origin node A is shown in the table of Fig. 3. Each row of the table corresponds to a destination node and the columns represent the steps of the algorithm. The first column shows the initial situation. Nodes B and C are directly connected to node A with links of finite length and nonzero capacity. For the nodes D and E that are not directly connected to node A, we use the convention that the delay from node A to these nodes is initially infinite, and the capacity is zero. Node C has the smallest delay to node A, and is chosen as the base node for the next step of the algorithm. In the second column the neighbors of node C obtain the cost vector to go to node A through node C, and compare it to the cost vectors they had previously. The sign "+" used in the calculations does not represent addition, but the way the cost of the concatenation of a path and a link is obtained, with each component of the cost vector calculated according to the corresponding monotonic associative operator. The cost vector is compared to the previously obtained vectors for that node using the domination relation and any dominated paths are discarded. The cost vectors of the non-dominated paths are shown in the third column, where the first node on the way to the origin is also illustrated. Again, the node with the smallest delay is chosen, which in this case is node B with cost vector $(3, 2, C)$. In the fourth column the neighbors of each node calculate their cost vectors to the origin node A through the node B. In the fifth column the non-dominated path are shown. There are now two vectors with the same delay, $(4, 3, A)$ corresponding to destination B and $(4, 2, B)$ corresponding to destination D. The first one is chosen because it has the largest capacity. The algorithm continues in this way and finishes when all the non-dominated paths have been chosen. The last column shows the final set of non dominated paths for every destination node.

Columns 1,3,5,7... in Fig. 3 contain the temporary sets of non-dominated paths. Only the underlined paths are guaranteed to be non-dominated at the end of the algorithm. The other paths may be dominated and discarded in the following steps of the algorithm. The same principle and similar data structures can be used to obtain algorithms for the case where k-dimensional cost vectors are considered.

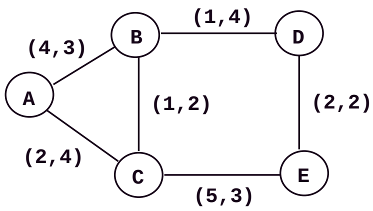


Figure 2: Graph associated to the network. Each edge is characterized by a 2-dimensional cost vector, the first component of which is additive cumulative, while the second is restrictive. An edge represents a pair of arcs with opposite directions and the same cost vector.

B	(4,3,A)	(1,2)+(2,4) (3,2,C)	(3,2,C) (4,3,A)		(3,2,C) (4,3,A)		(3,2,C) (4,3,A)	
C	(2,4,A)		(2,4,A)		(2,4,A)	(1,2)+(4,3) (5,2,B)	(2,4,A)	
D	(∞,0,-)		(∞,0,-)	(1,4)+(3,2) (4,2,B)	(4,2,B)	(1,4)+(4,3) (5,3,B)	(4,2,B) (5,3,B)	
E	(∞,0,-)	(5,3)+(2,4) (7,3,C)	(7,3,C)		(7,3,C)		(7,3,C)	(2,2)+(4,2) (6,2,D)

B	(3,2,C) (4,3,A)		(3,2,C) (4,3,A)		(3,2,C) (4,3,A)		(3,2,C) (4,3,A)
C	(2,4,A)		(2,4,A)	(5,3)+(6,2) (11,2,E)	(2,4,A)		(2,4,A)
D	(4,2,B) (5,3,B)		(4,2,B) (5,3,B)		(4,2,B) (5,3,B)	(2,2)+(7,3) (9,2,E)	(4,2,B) (5,3,B)
E	(6,2,D) (7,3,C)	(2,2)+(5,3) (7,3,B)	(6,2,D) (7,3,C)		(6,2,D) (7,3,C)		(6,2,D) (7,3,C)

Figure 3: Evolution of the algorithm to obtain the set of non-dominated paths.

3.3. Formal description of the algorithm

The algorithm to obtain the set of non-dominated paths between an origin node and all other nodes of the network is now formally described. We assume that the link cost vectors have k components, the first s of which are additive cumulative and have to be minimized, and the remaining $k-s$ are restrictive and have to be maximized. We denote by e_{ij} the cost vector of link (i,j). Every path is represented by a label that includes the cost vector associated with it and the first hop to the origin using that path. The origin is taken to be node 1.

We let W_i be the set of labels of destination node i, and $W = \cup_{i \neq 1} W_i$ be the set of all labels. Initially, every node has a single label corresponding to the link that connects it directly to the origin node. If a node has no link connecting it to the origin, its cost vector has the first s components equal to infinity and the remaining $k-s$ components equal to zero. In each the following steps, the algorithm marks labels (equivalently, paths) from the the set W as *final*. We let $W^d \subset W$ be the subset of all final labels for all the destinations, and $W_i^d \subset W_i$ be the set of final labels for destination node i. We also let $T \subset N - \{1\}$ be the set of destination nodes with at least one final label. The algorithm can now be described as follows:

- Step 0 (**Initialization**): $W = \{2, 3, \ldots, N\}$, $W^d = \emptyset$, $T = \emptyset$.
- Step 1 (**Choosing the optimum label**): The label whose cost vector minimizes the first component is chosen. In case of a tie, the second components are compared and the vector with the minimum second component is chosen. In case there is still a tie, the third component is compared and so on. For the first s components, the smallest values are chosen; for the remaining $k-s$ components, the largest values are preferred. If $\mathbf{V}$ is the cost vector of the chosen label and i is the corresponding node, then the following updates are performed:

$$W_i^d := W_i^d \cup \{\mathbf{V}\}, \quad W^d := W^d \cup \{\mathbf{V}\}, \quad T := T \cup \{i\}.$$

- Step 2 (**Obtaining the new labels**): The neighbors of node i, which may or may not belong to the set T, are now considered and are given new labels (except for the origin node and the node specified as the previous node in the label). The new label for a neighbor j of node i is then computed in the following way. The new cost vector is updated according to $\mathbf{U} = \mathbf{V} + e_{ij}$, where "+" represents the operation required to compute the new cost vector, which is componentwise addition for for the first s (additive) components, and componentwise minimization for the remaining $k-s$ (restrictive) components. The previous node on the path corresponding to vector $\mathbf{U}$ is set to be node i.
- Step 3 (**Discarding dominated paths**). Every neighbor considered in step 3 compares its new label with its previous labels using the domination relation. Let j be one of the neighbors of node i, $\mathbf{U}$ the new cost vector and W_j be the set of labels for this node. The new label has to be compared with the labels in W_j (both final and non-final). If any cost vector in W_j dominates $\mathbf{U}$, then $\mathbf{U}$ is discarded and W_j does not change. If the new cost vector $\mathbf{U}$ is not dominated by any of the vectors in W_j, then $\mathbf{U}$ is added to the set W_j and W, so that $W_j := W_j \cup \{\mathbf{U}\}$ and $W := W \cup \{\mathbf{U}\}$. If the new vector dominates one of the vectors in W_j, then W_j and W are updated eliminating the dominated vectors and adding the new vector $\mathbf{U}$. Note that it is not possible for the new vector to dominate an existing vector and be dominated by another one at the same time.
- Step 4. (**Termination**). If $W^d = W$, the algorithm is completed. Otherwise, (that is, when there are still some labels to be chosen) the algorithm goes back to Step 1.

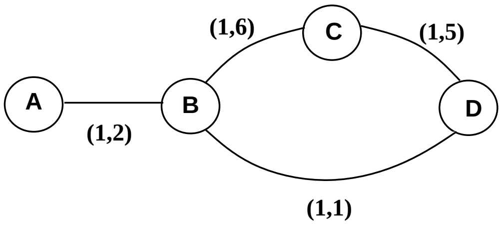
4. IMPLEMENTATION OF THE MULTI-COST ROUTING ALGORITHM

We distinguish between two kinds of networks: networks that use distributed routing and networks that use source routing. In distributed routing every switch in the path participates in the routing decision (in virtual circuit switching the routing decision is made once, when the set up packet arrives, while in datagram switching a routing decision is made for each packet) and the switches use the packet destination and a local routing table to decide the next hop to the destination. In source routing, each packet carries a description of the entire sequence of output ports that have to be traversed. An intermediate switch upon reception of a packet on an input port, reads the corresponding output port number in the header and transmits the packet on that output.

Distributed routing algorithms are usually based on the shortest path idea, where a single cost is assigned to each link. One of the reasons that make the combination of distributed routing and shortest path routing possible is that the shortest paths satisfy a *property of inclusion*: every subpath of a shortest path is a shortest path. In other words, if a shortest path from node A to node E traverses nodes A, B, C, D, E, then a shortest path from node B to node E traverses nodes B, C, D, E. As a result, a packet sent from node A to node E only has to carry information about the destination node, and can use the forwarding tables at intermediate nodes (which are consistent with each other because of the inclusion property, assuming no changes in the link costs occur while the packet is in transit) to find the next hop.

Multi-cost routing is not always compatible with distributed routing, because for some choices of the cost function the optimal paths do not have the inclusion property that shortest paths have. To see this, consider the example of Fig. 4, where the path cost vector is $\mathbf{V} = (h, c)$. The first link cost component is equal to 1, is additive cumulative, and will account for the number of links on a path. The second component represents the link capacity, and is restrictive. We assume that each node has already computed the set of non-dominated paths for each of the other nodes, and node A is about to establish a connection to destination node D. The table of Fig. 4 shows the set of non-dominated paths from node A to node D and from node B to node D.

Consider first the optimization function $f_1 = h/c$, where h is the total number of hops and c is the minimum of the capacities on the path. This function favors paths that have a small number of links and large capacity. For this function, the optimum path from node A to node D is (A, B, C, D), while the optimum path from node A to node B is (B, C, D). Therefore, in this case the optimum path from node B to node D is contained in the optimum path from node A to node D, and the packets sent by node A to node D through node B only have to have to carry with them information about the destination node D, and not a full description of the path. Consider now the optimization function $f_2 = h^2/c$ which penalizes long paths more severely. For this function, the optimum path from node A to node D is (A, B, D) while the optimum path from node B to node D is (B, C, D). Node A can send packets to node D through node B expecting them to be routed directly to node D, but node B will route these packets through node C because that is its optimum path to node D.



Non dominated paths	Function $f = \frac{h}{c}$	Function $f = \frac{h^2}{c}$
A→D	**A→D**	**A→D**
$(A, B, C, D), (3, 2)$	$f(3, 2) = 1.5$ *	$f(3, 2) = 4.5$
$(A, B, D), (2, 1)$	$f(2, 1) = 2$	$f(2, 1) = 4$*
B→D	**B→D**	**B→D**
$(B, C, D), (2, 5)$	$f(2, 5) = 0.4$*	$f(2, 5) = 0.8$*
$(B, D), (1, 1)$	$f(1, 1) = 1$	$f(1, 1) = 1$

Figure 4: The table of non-dominated paths for source nodes A and B and destination node D for two different cost functions. Optimum paths are marked with the symbol "*".

This simple example reveals that a distributed implementation of a multi-cost routing algorithms is not always possible. Depending on the optimization function used, some intermediate nodes may not choose the paths originally intended by the source node, leading to a certain loss of optimality. This problem does not arise when source routing is used. The source node records the complete path to be followed on the header of each packet, and intermediate nodes just have to follow the instructions. With source routing, however, we lose the benefits that may be obtained through the more recent link utilization and topology information available at intermediate nodes.

5. PERFORMANCE RESULTS

Multi-cost routing widens the horizons of the routing problem by allowing the consideration of any set of network parameters, and the use of the most convenient optimization criteria for each application. In this section we use simulations to compare the performance of several multi-cost routing algorithms with that of two traditional single cost routing algorithms:

- The *widest-shortest path* routing algorithm, where the path with the minimum number of links is chosen. In case of a tie, the one with the maximum capacity is selected.
- The *shortest-widest path*, where the path with the maximum available capacity is chosen. In case of a tie, the path with the minimum number of links is selected.

5.1. General description of the simulation

The routing algorithms will be compared for networks of varying topology and link capacities, under the assumption that sessions share the links according to a max-min fair share scheme. This scheme has been identified by the ATM Forum as one of main design goals for ABR traffic management algorithms. We assume throughout this section that source routing is used.

5.1.1. ABR service parameters and traffic model

ABR service exploits excess network bandwidth and uses flow control to prevent cell loss, maximize network utilization, and provide fair access to all users. In our simulations we assume that closed-loop rate-based flow control is used. Intermediate switches inform the end-stations about the congestion in the network, which then adjust their transmission rates accordingly (see [HuH95], [Ric95], [BoF95]). When an application requests an ABR connection, it negotiates with the network a PCR (Peak Cell Rate) and a MCR (Minimum Cell Rate). The ABR user agrees not to transmit at rates above the PCR, and the network agrees to always provide at least the minimum cell rate MCR, which is typically the minimum required to keep the application running. An application requesting ABR service is not required to specify peak and minimum cell rates, but it may simply accept the network default settings. In such cases, the PCR defaults to the access rate and the MCR defaults to zero.

We assume that connection requests (sessions) are generated at each node according to a Poisson random process of rate λ, and session destinations are uniformly distributed over all nodes, except for the origin. We assume well behaved traffic sources: When the network report changes in the link utilization, the sources adapt their transmission rates exactly and instantaneously. In the intervals between such changes, we assume that the sources transmit packets at a constant rate. These assumptions guarantee that the network switches do not reveive more data than they can send to their outgoing links, so that little storage is needed and the queueing delays of packets at intermediate nodes can be ignored. The assumption of instantaneous feedback is clearly unrealistic, but we believe that it does not affect the comparisons regarding the relative merits of the routing algorithms we will examine. In Section 5.3.6 we discuss the effects delayed feedback has on routing performance.

A connection request is refused if the network cannot provide the requested value for MCR. In that case, a new Poisson process is generated for the session retries at a node, whose mean interarrival times are several times larger than the mean interarrival times of the original Poisson process. Then the retrial process can be considered independent from the process corresponding to new arrivals, and the combined process can be approximated by a Poisson process. A session that has been blocked retries until the connection is accepted.

Sessions in our simulations are divided in two categories depending on the amount of data to be transmitted: a) medium-size sessions, where the amount of data sent is uniformly distributed in the interval (50Kytes, 5Mbytes), and b) large-size sessions, where the amount of data sent is uniformly distributed in the interval (5Mbytes, 500Mbytes).

5.1.2. Parameters for the multi-cost routing algorithms and optimization functions considered

The cost parameters that we consider in our simulations are the delay, the number of links, and the available capacity of the path. The delay of the path is equal to the sum of the link propagation delays, since the queueing delays in our model are negligible (the propagation delay per km of the fiber is 5μsec/km.) In reservation-based networks, the capacity of a link is usually taken to be the residual link bandwidth (unreserved bandwidth), and the capacity of a path is the minimum of the capacities of the links in the path. In the max-min fair share networks that we consider, however, the residual bandwidth cannot be used as a measure of the available capacity since the nature of bandwidth sharing in this kind of networks is different. Instead, an estimate of the max-min fair rate that would be obtained by the new connection if admitted is used as the available capacity, as will be explained in detail in Section 5.2.

We consider several cost functions in our simulations. The first function is

$$f(h,r) = \frac{h}{r}$$

where h represents the number of links in the path and r represents the available capacity of the path. This function tries to minimize h and maximize r simultaneously, and represents a trade-off between the interests of the network and the interests of the connection. An individual connection wants to get the maximum capacity possible, no matter how many links are used to get it. The network has to prevent the uncontrolled use of the resources, so that it is fair to connections that may arrive later. This objective can be achieved by making the cost function an increasing function of the number of links.

Another function we consider is

$$f(d, b, r) = d + \frac{b}{r}$$

where d represents the path propagation delay, b represents the amount of data to be transmitted and r is, again, the available capacity. This function tries to minimize the total connection time, which consists of the propagation delay and the transmission time. It chooses the quickest path to transmit the data.

A combination of the two previous functions is

$$f(d, b, r, h) = \left(d + \frac{b}{r}\right) * h.$$

It can be thought of as a way to obtain a quick path while penalizing at the same time paths that have an excessive number of links. It is a trade-off between the time requirements of the connection and the network general interests. It is also possible to adjust the relative importance of these two considerations by adding a parameter n. A function we used in our simulations is

$$f(d, b, r, h, n) = \left(d + \frac{b}{r}\right) * h^n.$$

Note that functions of the kind given above could not be used in single-cost routing algorithms.

We have developed an event-driven simulator that allows the following options:

- Choosing the routing algorithm (widest-shortest routing, shortest-widest routing, or multicost routing that uses a variety of network parameters and cost functions, such as the ones described previously).
- Selecting the network topology, and the link capacities.
- Adjusting the offered load through the session arrival rate at a node λ, and the proportion of medium size and large-size sessions.
- Determining the type of updating for the routing information (instantaneous or periodic updating).

5.2. Simulation of the max-min fair share algorithm

The concept of max-min fair share was first proposed by Jaffe [Jaf81] to address the issue of fair allocation of resources in networks using virtual circuit switching, and has received considerable attention recently, since it was identified by the ATM Forum as one of the main design goals for ABR traffic management algorithms (see [BoF95], [ZuC94], [TsW96]).

In a max-min fair share network, the connections using a particular link get a "fair share" of the link bandwidth. If a connection cannot use up its fair share bandwidth, because it has a lower source rate or it is assigned a lower bandwidth on another link, the excess bandwidth is split "fairly" among all other connections (see [BeG87], [MSZ96]). There are several definitions of what is meant by "fair share". In the basic definition, each of the N connections competing for the link or excess bandwidth gets $1/N$ of the bandwidth. Other definitions such as weighted fair sharing and multi-class fair sharing, require pricing or administrative policies to assign a weight or class for each connection. Since these policies are still an active area of research, the basic max-min fair share model will be the first one to be supported by commercial ATM networks, and it is the one we use in this paper.

There are two ways of implementing a max-min fair share network. One option is for all switches to use a Fair Queueing Scheduler [DKS89]. The other option is to have switches explicitly compute the max-min fair rate for each connection and inform the corresponding source nodes of this rate; sources are required to transmit at no more than their max-min fair rate (see [CCJ95], [CRL96], [ChR96]). While the first approach requires per-connection queueing in the switch, the second one does not, which simplifies the switch design. This is one of the reasons the ATM Forum selected the second implementation that uses explicit rate calculations. Another advantage of the second approach is that switches obtain rate information for all connections, which can be used to perform load sensitive routing.

The algorithms proposed in the literature for the rate calculations are usually distributed and asynchronous, and it takes some time for the sources to converge to the fair transmission rates. In our simulations, we assume that a synchronous centralized assignment mechanism is used and the sources react to the network requests instantaneously. Since there are some real implementations (see [CCJ95]) that have achieved very small convergence times, we believe that the assumption of a centralized algorithm for computing the rates is adequate for simulating the behavior of the network and comparing the different routing approaches. In Section 5.3.6 we examine the effects periodic updating has on the performance of the routing algorithms.

A simple centralized algorithm for calculating the estimates for the max-min fair rates obtained by a connection if accepted is described next (this algorithm is essentially the same with the one given in [MSZ96]). Given a set S of connections, we let $L(S)$ be the set of all links in the network with at least one connection in S using them. We also let C_l be the capacity of the link l. A connection is called saturated if it has reached its desired source rate or a link on the path traversed by the connection is saturated. A link is called saturated if all of its bandwidth has been allocated to connections sharing the link. We let CN be the set of all connections in the network, CN_l be the set of connections that use link l, and SAT and $UNSAT$ be the set of saturated and unsaturated connections, respectively. We also let $SAT_l = CN_l \cap SAT$ and $UNSAT_l = CN_l \cap UNSAT$. We denote by r_i^{est} the transmission rate of connection i and by PCR_i its maximum peak cell rate. Me also let R_l be the transmission rate obtained on link l. We assume that the new connection is not limited by its maximum rate (i.e., $PCR_c = \max\{C_l\}$). A centralized algorithm for estimating the max-min fair rates obtained if the new connection c was routed through a particular path can then be described as follows:

- For every link in the network,
- **Step 0** (Initialization): $SAT_l = \emptyset$, $UNSAT_l = CN_l \cup \{c\}$, $r_l^{\mathrm{est}} = 0$, $r_i = 0$ for all $i \in UNSAT_l$.
- While $UNSAT_l \neq \emptyset$:
- **Step 1**: Calculate
$$R_l = \frac{C_l - \sum_{i \in SAT_l} r_i}{|UNSAT_l|}.$$
- **Step 2**: Update the transmission rate for all the connections in $UNSAT_l$ according to $r_i = \min\{PCR_i, R_l\}$ for all $i \in UNSAT_l$.
- **Step 3**: Move new saturated connections in $UNSAT_l$ to SAT_l. If $UNSAT_l \neq \emptyset$ go back to Step 1. Otherwise, the calculations for link l have finished, and the expected rate on this link is the last value assigned to R_l, that is, $r_l^{\mathrm{est}} = R_l$. Go back to Step 0 for the next link.

5.3. Simulation results

When a new connection request is generated, it specifies the destination node and the amount of data to be transmitted. The node that receives the request obtains the path to the destination. Then, the max-min algorithm is executed so that a transmission rate is assigned to the new connection and the transmission rates of the other connections in the network are updated. When a connection is terminated, the max-min algorithm is executed and the remaining connections in the network obtain their new transmission rates.

5.3.1. Different network topologies

We compare the following routing algorithms:

- Widest-shortest algorithm [abbreviated min(h))].
- Shortest-widest algorithm [abbreviated max(r))].
- Multi-cost routing algorithm using the function $f = (d + b/r) * h$, where d is the path propagation delay, b is the amount of data to be transmitted, r is the available capacity and h is the number of links in the path.

In the simulations, unless stated otherwise, the value for the available capacity is the estimated rate for the new connection obtained by running the algorithm of Section 5.2. The value of PCR is assumed to be always equal to the maximum link capacity in the network. The link lengths are uniformly distributed in the interval between 50Km and 100Km, corresponding to propagation delays in the interval (0.25, 0.5)msec. Connections transmit data in the intervals (50Kbytes, 5Mbytes) and (5Mbytes, 500Mbytes), with 50% of the data belonging to each traffic category.

Connections that specify a value for MCR may not be accepted in the network. In such a case, they retry later as many times as necessary until they are accepted. We first consider the case where the connections do not specify a value for MCR, in which case MCR defaults to zero, and all connection requests are accepted. The efficiency of the routing algorithms can then be evaluated based on the average connection, which is the sum of the propagation delay d and the transmission time b/r.

We consider two different topologies: (a) a 4x4 mesh network (see Fig. 5a), with $N = 16$ nodes and $L = 24$ links, and (b) a random network with $N = 10$ nodes and 3 links per node (see Fig. 5b). All link capacities in the random network are taken to be equal to 155Mbps, while links in the mesh network have capacity equal to 155Mbps or 622Mbps. Our intention in considering the random network is to see how the irregularities in the topology affect the performance of the routing algorithms. We consider 3 different mesh networks that differ on the total number of links with capacity 622Mbps: 1) a mesh with only 2 links of capacity 622Mbps, 2) a mesh with 6 links of capacity 622Mbps, 3) a mesh with 10 links of capacity 622Mbps. Our intention in considering these three cases is to see how the behavior of the routing algorithms is affected by the lack of regularity in the capacity assignment. For the topologies considered and the values assumed for the propagation delays, link capacities and amounts of transmitted data, transmission times dominate the total connection times. In Section 5.3.7 we discuss the case of very high speed networks, where the propagation delays are not negligible.

5.3.2. Results for the mesh network

Figure 6 illustrates the connection time versus the offered load for the Mesh No. 1 network, which is very regular both in the distribution of the links and also in the values of link capacities. For

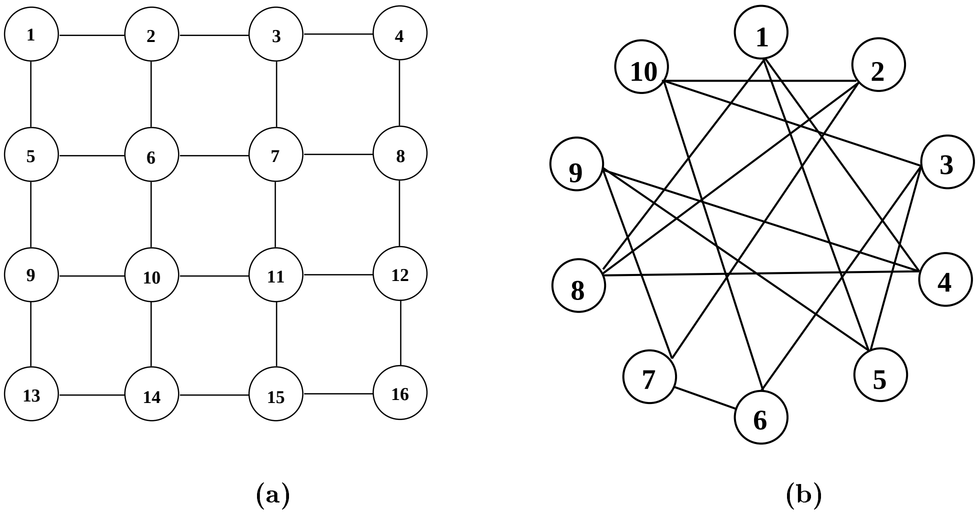


Figure 5: (a) A 4x4 mesh. (b) The random network.

light load all three algorithms have similar behavior. As the load increases, the multi-cost routing algorithm slightly outperforms the widest-shortest algorithm, because it also takes into account the parameter r, which captures the differences in the distribution of the traffic. The shortest-widest algorithm presents the worst results for heavy load. This is because it always selects the path with the maximum capacity, independently of the number of links it uses, which leads to inefficient use of the resources. By permiting new connections to be greedy in the amount of resources they use, this algorithm reduces the amount of resources available for future connections that will consequently have larger connection times. Moreover, the longer the path is, the larger is the probability that the connection's share of the capacity will be reduced as a result of fair sharing with connections that are added later.

Figure 7 shows the connection time as a function of the offered load for Mesh No. 2. This mesh has a larger number of 622Mbps links than Mesh No. 1, and the differences in capacity among the links are more pronounced than in Mesh No. 1. For light load, all three algorithms give similar results, with the shortest-widest algorithm giving a slightly smaller average connection time. This is because in trying to maximize exclusively the capacity a session can take advantage of the high capacity of some of the links, improving the connection times. In other words, under light load conditions a session can be greedy without punishing other connections. As the offered load increases, the differences become smaller, and at heavy load the shortest-widest algorithm becomes again the worst performer.

Figure 8 illustrates the connection time versus the offered load for Mesh No. 3. In this network the percentage of links with 622Mbps is even larger than in Mesh No. 2. The shortest-widest algorithm gives the best results under light and medium load conditions. The multi-cost routing algorithm

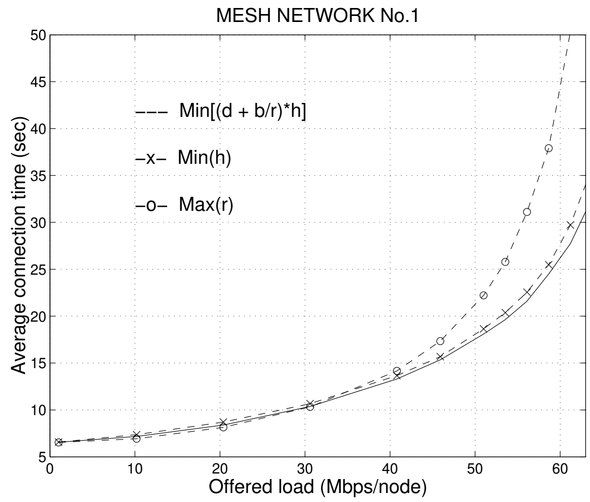


Figure 6: Average connection time in the mesh network No. 1.

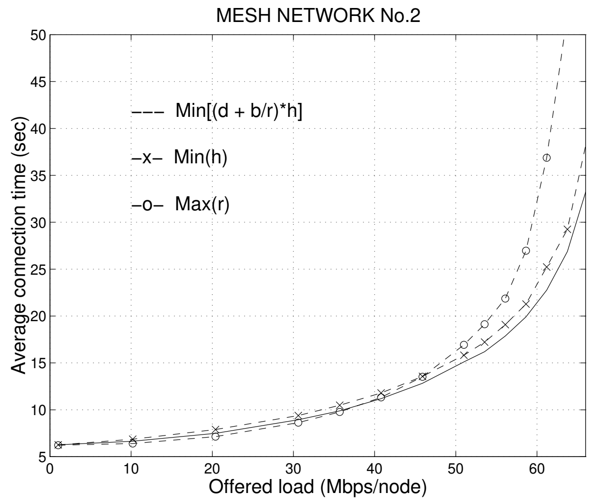
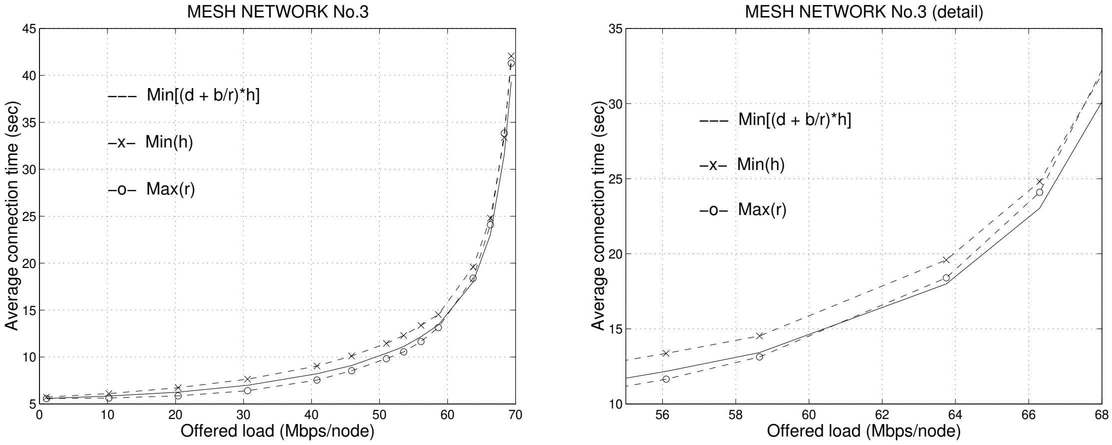


Figure 7: Average connection time in the mesh network No. 2.

gives the second best results, and the widest-shortest algorithm gives the worst results, because it is almost insensitive to the remarkable differences in capacity existent in the network. Under heavy load conditions, the shortest-widest algorithm gives again the worst results (see Fig. 8b).

These first results for the mesh network bring some interesting conclusions:

- The widest-shortest algorithm captures the regularity of the network and brings out acceptable results. It is essentially a non-adaptive routing algorithm that takes always the same decisions independently of the offered load.
- The multi-cost routing algorithm obtains always better results than the widest-shortest algorithm.



(a) **(b)**

Figure 8: (a) Average connection time for the mesh network No. 3. (b) Detail of the curves at heavy load.

This is because it captures not only the regularity of the network but also its dynamics as given by the available capacity.

- The efficiency of the shortest-widest algorithm depends strongly on the offered load. For light load conditions it performs well, but for heavy load it always gives the worst results. When the network is congested, the capacity that a new connection can capture will be small and will be similar for all the links on the path. Therefore, what is important is to achieve a good distribution of the traffic using as little resources as possible.

Figure 9 illustrates the connection time versus the offered load to the network for the random network. Recall that the random network has $N = 10$ nodes each of which is connected to 3 other nodes, and all link capacities are equal to $C = 155Mbps$. The widest-shortest algorithm presents the worst performance. The lack of strict regularity in the network penalizes this algorithm, because it does not take into account the irregular distribution of traffic. Despite its randomness, however, the network still has a certain degree of regularity (it guarantees 3 links per node), and this is why the performance of the widest-shortest algorithm is not even worse. The shortest-widest algorithm performs slightly better for light load, but its performance gets worse as the network load increases. The multi-cost routing algorithm has good performance for all network loads, because it captures both the slight regularity in the network and the differences in the distribution of traffic.

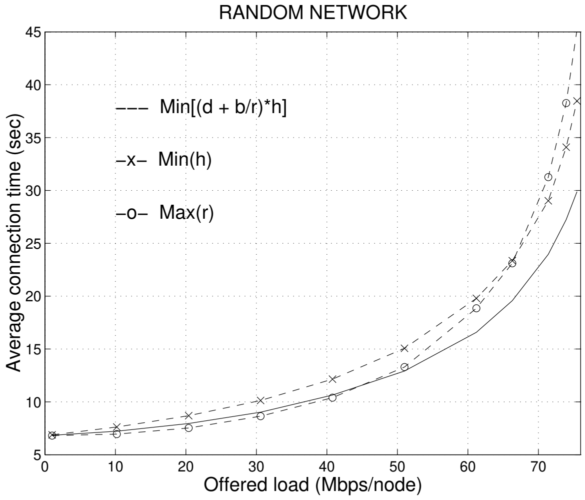


Figure 9: Average connection time in the random network.

5.3.3. Parametric optimization functions

We have just seen that when the load is low and the network is not very regular, the shortest-widest algorithm can achieve a slightly better performance than the multi-cost routing algorithm. The multi-cost routing algorithm of the preceding section used the optimization function

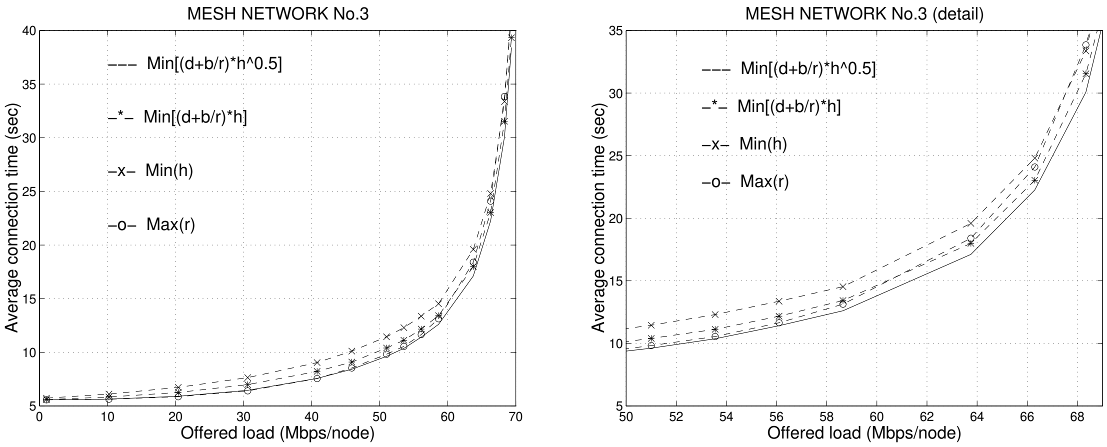
$$f = \left(d + \frac{b}{r}\right) * h,$$

where d is the propagation delay, b is the amount of data to be transmitted, r is the available capacity and h is the number of links in the path. If we want the multi-cost routing algorithm to perform well even for irregular networks, we need to give more relative importance to the capacity rather than to the number of links. This can be achieved using a parametric function

$$f = \left(d + \frac{b}{r}\right) * h^n$$

where n is a real number in the interval (0,1) that reduces the importance of the number of hops. The value for n should be chosen based on the network topology; the smaller the degree of regularity in the network, the bigger the value of n should be.

Figure 10 plots the connection time versus the offered load for Mesh No. 3. We compare the widest-shortest, the shortest-widest and the multi-cost routing with $n = 1$ and $n = 0.5$. The multi-cost routing algorithm with $n = 0.5$ clearly gives the best performance. Under light load conditions, the performance of the algorithm is similar to that of the shortest-widest algorithm, while for heavy load its performance is significantly better (see Fig. 10b). Although the new optimization function

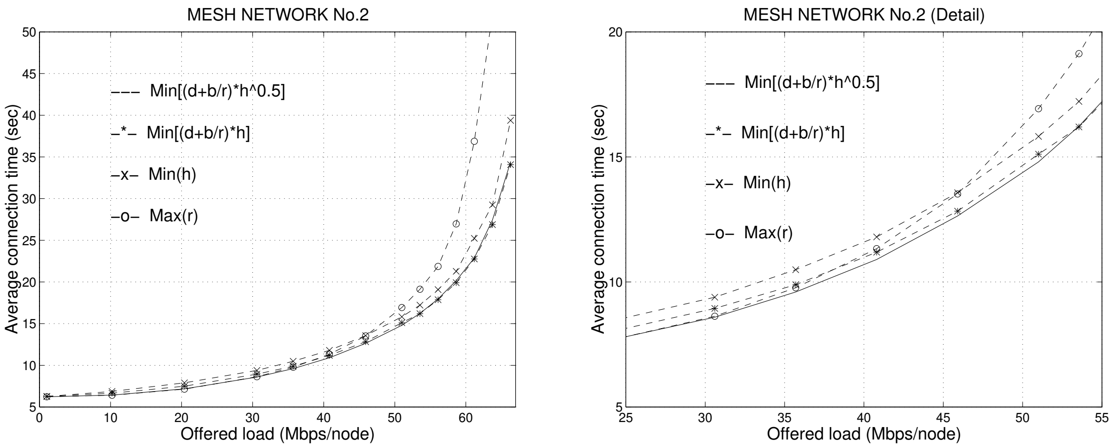


(a) **(b)**

Figure 10: (a) Average connection time in the mesh network No. 3. (b) Detail of the curves.

gives more importance to the available capacity r, it still considers the number of hops, and that is why its performance is good even under heavy load.

Similar improvements are achieved by the new multi-cost routing algorithm in the mesh network No. 2 and the random network as shown in Figs. 7 and 12. The differences between the shortest-widest and the multi-cost routing algorithm under light load conditions disappear, and the multi-cost routing algorithms continues to outperform the other schemes under heavy load.



(a) **(b)**

Figure 11: (a) Average connection time in the mesh network No. 2. (b) Detail of the curves.

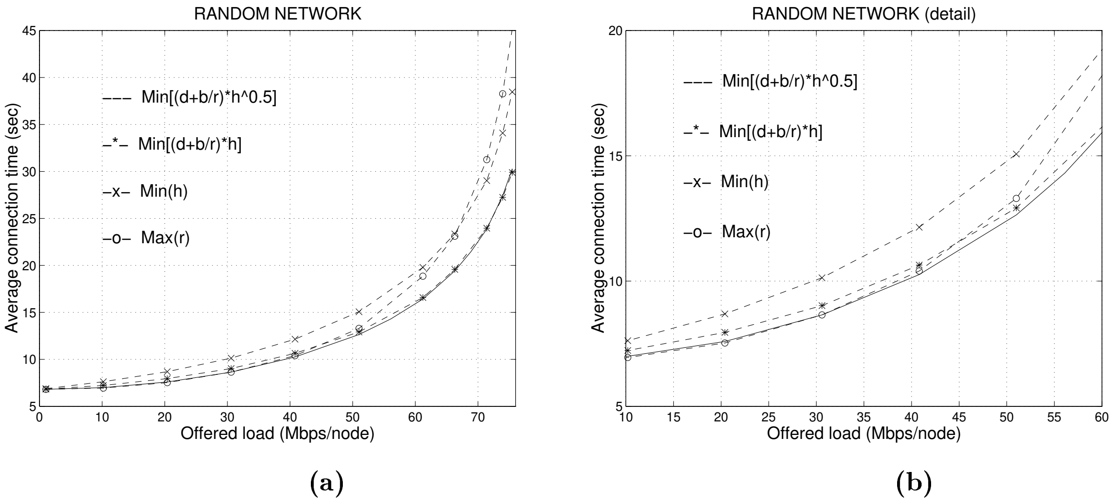


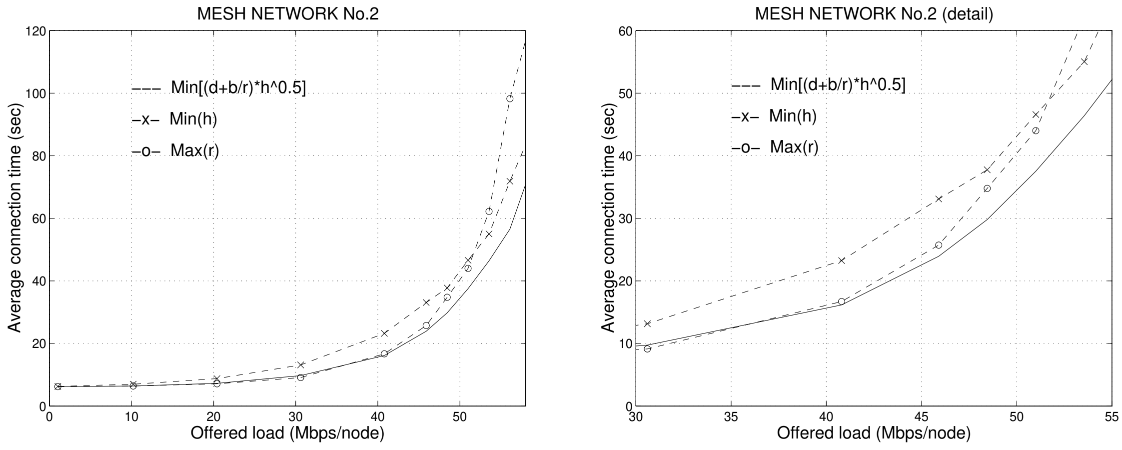
Figure 12: (a) Average connection time in the random network. (b) Detail of the curves.

5.3.4. Specification of a value for MCR

When a nonzero value for MCR is specified, a connection may not be accepted in the network. The efficiency of the routing algorithms can then be evaluated based on two measures: (a) the average connection time, which consists of the propagation delay, the transmission delay, and the total waiting time till the connection is accepted, and (b) the probability of success for a new connection. The routing algorithm affects directly the operation of the network. Good routing distributes the traffic evenly in the network and tries to obtain the smallest connection time, keeping in this way the blocking probability for new connections at low levels.

We present simulation results for the mesh network No. 2 and the random network. We compare the widest-shortest, the shortest-widest and the multi-cost routing using the general minimization function with parameter $n = 0.5$. For the mesh No. 2 we use $MCR = 50$Mbps for the traffic in the interval (5Mbytes, 500Mbytes), and $MCR = 5$Mbps for the traffic in the interval (50Kbytes, 5Mbytes). For the random network, we use $MCR = 40$Mbps for the traffic in the interval (5Mbytes, 500Mbytes), and $MCR = 4$Mbps for the traffic in the interval (50Kbytes, 5Mbytes). The ABR service was designed to use the excess bandwidth in the network, and the values for MCR are generally small. The values for MCR used here are bigger than usual, because we assume that the network carries only ABR traffic and we want the blocking phenomenon to be accentuated.

Figure 13 illustrates the connection time as a function of the offered load for Mesh No. 2, while Fig. 14 plots the success probability versus the offered load for the same network. Note that the multi-cost routing algorithm with $n = 0.5$ presents the best results for both the average connection time and the success probability. The shortest-widest algorithm achieves good performance under light and medium load, but when the load increases, the average connection time suffers a considerable increase. (Both the figures for the connection time and the success probability show this decrease in efficiency around 40Mbps). Connections that are not accepted in the network, wait an interval of time and then retry. This waiting time increases the average connection time.



(a) **(b)**

Figure 13: (a) Average connection time in the mesh network No. 2. (b) Detail of the curves for medium and heavy load.

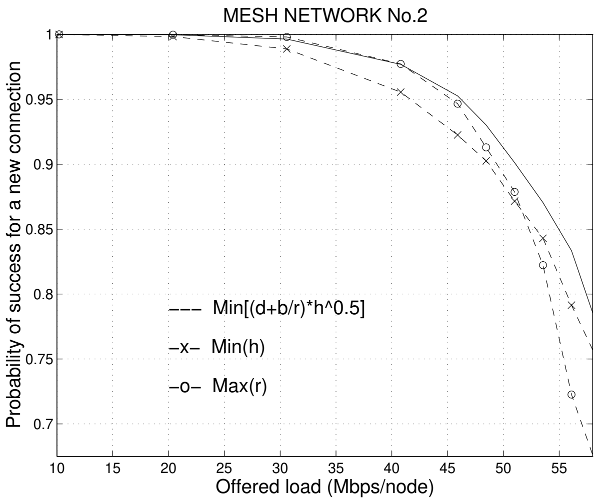
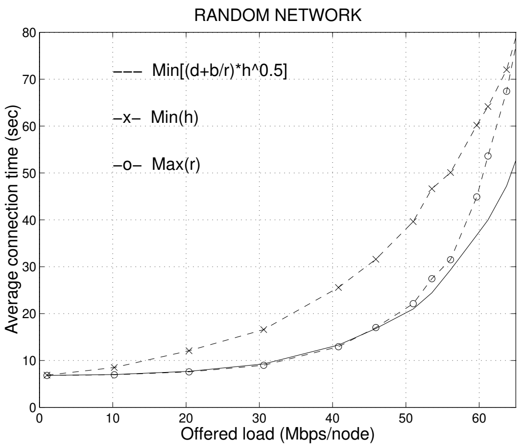


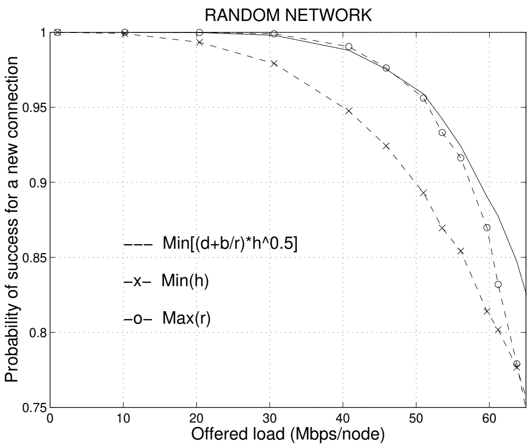
Figure 14: Curves of the success probability for a new connection in the mesh network No. 2.

The widest-shortest algorithm gives acceptable results and a graceful performance degradation when the network load increases. For the mesh network, this routing method distributes well the traffic and keeps the number of rejected connections increasing gradually, not steeply. The success probability is the first one to suffer a decrease; this is because of the static nature of the algorithm that cannot avoid the early congestion points in the network when the load increases. The multi-cost routing algorithm has the best performance, combining the advantages of the two other algorithms: it consumes resources when the load is low and conserves resources when the load is high.

In Figs. 15a and 15b we illustrate the connection time and the success probability as a function of the offered load for the random network. The widest-shortest algorithm exhibits the worst results. This is because of the lack of regularity of the network that makes this routing strategy inappropriate. The shortest-widest algorithm and the multi-cost routing algorithm present similar results under light and medium load conditions. Further increases in the load, however, lead to a steep degradation in the performance of the shortest-widest algorithm.



(a)



(b)

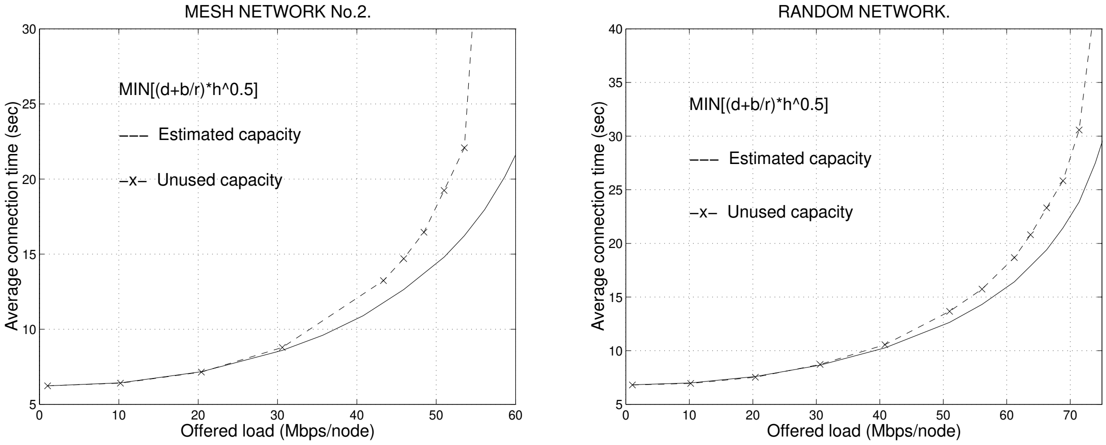
Figure 15: (a) Average connection time in the random network. (b) Curves of the success probability for a new connection in the random network.

5.3.5. Influence of the way the "available capacity" is defined

In this section we examine the way the definition of the "available capacity" affects the performance of the routing algorithm. We compare two ways of defining the available capacity:

- As the estimate of the max-min rate that would be assigned to a new connection if accepted; this is the value that has been used in all the simulations.
- As the unused capacity on each link; this definition does not reflect the nature of bandwidth sharing in max-min fair networks but it is a traditional way of defining the available capacity.

We compare the performance of the multi-cost routing algorithm with $n = 0.5$ using both definitions for the available capacity r. Figure 16a shows the results obtained for mesh No. 2, while Fig. 16b shows the results obtained for the random network. For light load, both definitions give similar results, because the paths with the most unused capacity are generally the paths that also give the biggest estimates for the max-min rate for a new connection. When the load increases, however, the definition that identifies available capacity with unused capacity leads to longer connection times, because it does not correctly reflects the nature of bandwidth sharing in max-min fair share networks.



(a) (b)

Figure 16: (a) Average connection time in the mesh No. 2. (b) Average connection time in the random network.

5.3.6. The effect of outdated link utilization information

The simulation results in previous sections assume that the most updated link utilization information is available and the estimated rates for a new connection are obtained every time the cost of a path is calculated. These assumptions are clearly unrealistic. A more practical model would be to assume periodic updates of the link utilization information and the estimated rates. This would reduce the burden on the switches at the expense of a certain loss of efficiency in the routing.

In Fig. 17 we compare the the average connection time of the multi-cost routing algorithm with $n = 0.5$ for mesh No. 2 for two types of updating: 1) ideal instantaneous updating, and 2) periodic updating, with a period of $P = 10$, 50, or 100 seconds. Naturally, the algorithm that assumes ideal updating gives the best performance, and the larger is the updating period the worse are the results.

Also, the larger the offered load is (that is, the more connection requests arrive per second), the more severe is the effect of outdated information. The results obtained with $P = 10sec$ are very close to those obtained when ideal updating is assumed, which makes this choice of the period a practical alternative to ideal updating for the data sizes and link capacities we have assumed.

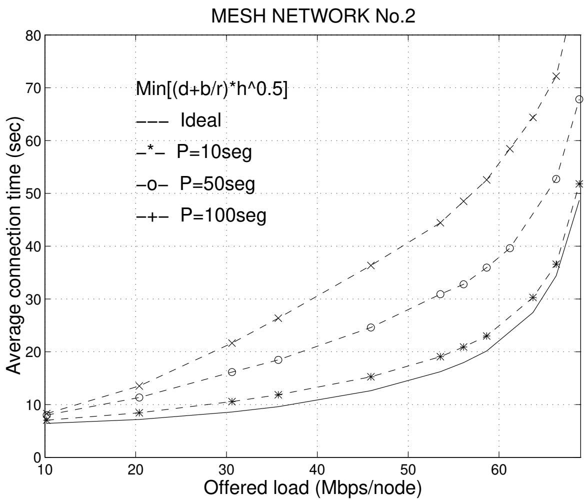


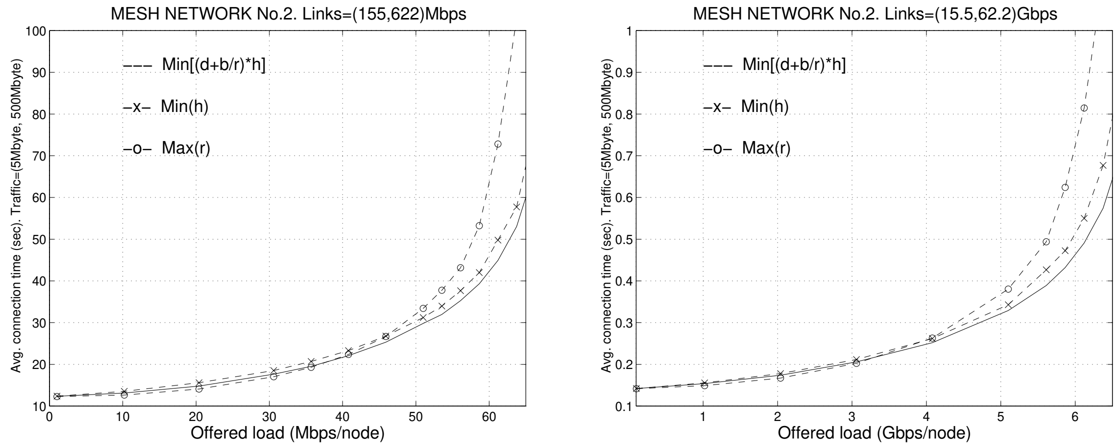
Figure 17: Average connection time for the multi-cost routing algorithm using instantaneous and periodic updating of the routing information.

5.3.7. Importance of the propagation delay

In the preceding simulations, the link capacities allowed transmission times that were several orders of magnitude larger than the propagation delays. The situation is very different for very high speed networks (with capacities of the order of Gbps), where the transmission times may be comparable to the propagation delays.

To study the effects of the propagation delay we consider again the mesh network No. 2, but assume link capacities that are 100 times larger, that is, 15.5 Gbps and 62.2 Gbps instead of 155Mbps and 622Mbps. We also increase the link lengths to distances in the interval (500Km, 300Km), so that the propagation delays become more significant. The amounts of data transported by a session are again uniformly distributed in the intervals (50Kbytes, 5Mbytes) and (5Mbytes, 500Mbytes).

We compare the widest-shortest and the shortest-widest algorithms with the multi-cost routing algorithm that has minimization function $f = (d + \frac{b}{r}) * h$. We obtain the curves for the connection time separately for each category of traffic. Figure 18 illustrates the results obtained for the traffic in the interval (5Mbytes, 500Mbytes). Figure 18a, shows the results for the original mesh No. 2, and Figure 18b for the high speed mesh No. 2. We observe that the differences among the algorithms are proportionally the same in both networks: the transmission times are 100 times smaller in the high speed network, but this affects equally all the algorithms. The amount of data to be transmitted is still large enough to require, on the average, transmission times that are much larger than the propagation delays.



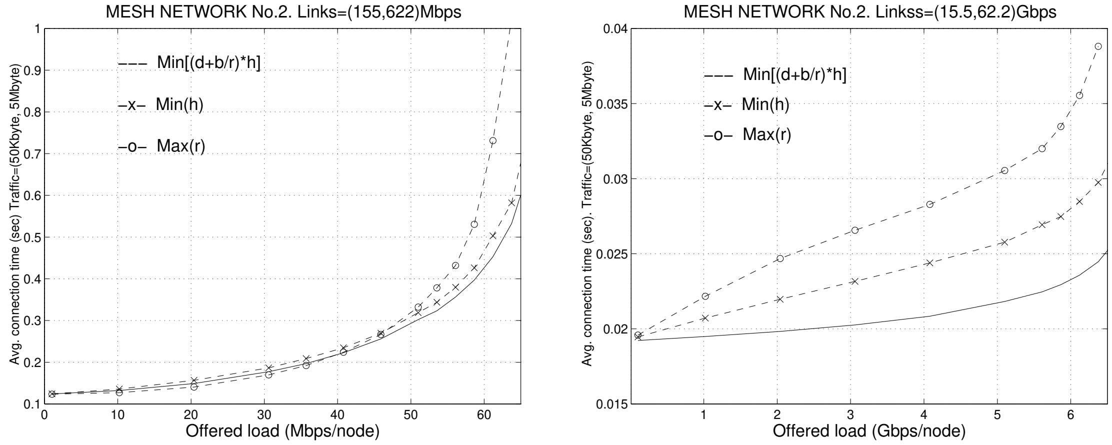
(a) **(b)**

Figure 18: (a) Average connection time for the traffic in the interval (5Mbytes, 500Mbytes) in the original mesh No. 2. (b) Average connection time for the traffic in the interval (5Mbytes, 500Mbytes) in the high speed mesh No. 2.

Figure 19 shows the results obtained for the traffic in the interval (50Kbytes, 5Mbytes). Figure 19a, illustrates the results for the original mesh No. 2, and Figure 19b for the high speed mesh No. 2. In the high speed network, the propagation delays are now comparable to the transmission timee, and the differences in the performance are significant. The widest-shortest algorithm performs rather well, because by minimizing the number of hops, it also tends to minimize indirectly the total propagation delay. The multicost routing algorithm gives again the best performance among the three algorithms. Therefore, that even for high-speed networks, minimizing exclusively the propagation delay does not seem to be the right objective of the routing algorithm. The right approach to the routing problem requires taking in account all the network parameters globally, as the multi-cost routing algorithm does.

REFERENCES

[Ahu88] Ahuja, R .K., "Minimum Cost-Reliability Ratio Path Problem," Computer Operations Research, Vol. 15, 1988, pp. 83-89.



(a) **(b)**

Figure 19: (a) Average connection time for the traffic in the interval (50Kbytes, 5Mbytes) in the original mesh No. 2.(b) Average connection time for the traffic in the interval (50Kbytes, 5Mbytes) in the high speed mesh No. 2.

[BeG87] Bertsekas D. P., and Gallager, R. G., Data Networks, Prentice-Hall, Edit. 1, New Jersey, 1987, p. 486.

[BoF95] Bonomi, F., and Fendick, K. W., "The Rate-based Flow Control Framework for Available Bit Rate ATM Service," IEEE Network, 1995, March/April 1995, pp. 25-39.

[CCJ95] Charny, A., Clark, D., and Jain, R., "Congestion Control with Explicit Rate Indication," IEEE Intl. Confe. on Communications, Vol. 3, 1995, pp. 1954-1963.

[ChC90] Chen, Y. L., Chin,Y. H., "The Quickest Path Problem," Computers Operations Research, Vol. 17, 1990, pp. 157-163.

[ChH96] Chang, Y.-C., and Hsu, L.-H., "On Minimal Cost-Reliability Ratio Spanning Trees and Related Problems," Operations Research Letters, Vol. 19, 1996, pp. 65-69.

[ChR96] Charny, A., and Ramakrishnan, K. K., "Time Scale Analysis of Explicit Rate Allocation in ATM Networks," IEEE Infocom, Vol. 3, 1996, pp. 1182-1189.

[CRL96] Charny, A., Ramakrishnan, K. K., and Lauck, A. G., "Scalability Issues for Distributed Explicit Rate Allocation in ATM Networks," IEEE Infocom, Vol. 3, 1996, pp. 1198-1205.

[Dij59] Dijkstra, E. W., "A Note in two problems in connection with Graphs," Numerische Mathematik, Vol. 1, 1959, pp. 269-271.

[DKS89] Demers, A., Keshav, S. and Shenker, S., "Analysis and Simulation of a Fair Queueing Algorithm," ACM SIGCOMM 89, Vol. 4, 1989, pp. 1-12.

[DPZ95] Djidjev, H. N., Pantziou, G. E., and Zaroliagis, C. D., "On-line and Dynamic Algorithms for Shortest Path Problems," Algorithms and Data Structures, 1995, pp. 193-204.

[Elw95] Elwalid, A. I., "Analysis of Adaptive Rate-Based Congestion Control for High-Speed Wide Area Networks," IEEE Intl Conference on Communications, Vol. 3, 1995, pp. 1948-1953.

[Fis73] Fishman, G. S., Concepts and Methods in Discrete Event Digital Simulation, John Wiley & Sons, Edit. 1, 1973, pp. 385.

[Fuj96] Fujimora, K., "Path Planning with Multiple Objectives," IEEE Robotics and Automation Magazine, 1996, pp. 33-38.

[GaB84] Gafni, E.M. and Bertsekas, D.P., "Dynamic Control of Session Input Rates in Communication Networks," IEEE Transactions Automatic Control, Vol. AC-29, 1984, pp. 1009-1016.

[Hay81] Hayden, H., "Voice Flow Control in Integrated Packet Networks, Cambridge, MA: MIT Laboratory for Information and Decision Systems, Report LIDS-TH-1152, 1981.

[HuC91] Hung, Y. C., Chen, G. H., "The Quickest Path Problem," Proc. ICCI 91, LNCS 497, Springer-Verlag, 1991, pp. 44-46.

[Han80] Hansen, P., "Bicriterion Path Problems," Multiple Criteria Decision Making: Theory and Applications. Lecture notes in Economics and Mathematical Systems, 1980, pp. 109-127.

[Han82] Hansen, P., "A Bicriterion Shortest Path Algorithm," European Journal of Operational Research, Vol. 11, 1982, pp. 399-404.

[HuH95] Hughues D., and Hooshmand, K., "ABR stretches ATM network resources," Data Communications, April 1995, pp. 123-128.

[Jaf81] Jaffe, J. M. "Bottleneck Flow Control," IEEE Trans. on Communications, Vol. COM-29, No. 7, July 1981, pp. 954-62.

[KPZ95] Kagaris, D., Pantziou, G. E., and Zaroliagis, C. D., "On the Computation of Fast Data Transmissions in Networks with Capacities and Delays, Algorithms and Data Structures, 1995, pp. 291-302.

[Mar84] Ernesto Queiros Vieira Martins, "An Algorithm to Determine a Path with Minimal Cost / Capacity Ratio," Discrete Applied Mathematics, Vol. 8, 1984, pp. 189-194.

[MSZ96] Ma, Q., Steenkiste, P., Zang, H., "Routing High-Bandwidth Traffic in Max-Min Fair Share Networks," ACM SIGCOMM, 1996, pp. 206-217.

[PeD96] Peterson, L. L., Davie, B. S. Computer Networks. A Systems Approach, Morgan Kaufmann Inc, Edit. 1, San Francisco, CA, 1996.

[Ric95] Rickard, N., "ABR: Realizing the Promise of ATM," Telecommunications, April 1995, pp. 27-33.

[RSX91] Rosen, J. B., Sun, S. Z., . Xue, G. L., “Algorithms for the Quickest Path Problem and the Enumeration of Quickest Paths,” Computers Operations Research, Vol. 6, 1991, pp. 579-584.

[ScS80] Schwartz, M., and Stern, T. E., “Routing Techniques used in Computer Communication Networks,” IEEE Trans. on Communications, Vol. 4, April 1980, pp. 539-552.

[SiT94] Siu, K.-Y., and Tzeng, H.-Y., “Intelligent Congestion Control for ABR Service in ATM Networks,” Comp. Communications Review, 1994, Vol. 24, pp. 81-106.

[TsW96] Tsang, D. H. K., and Wong, W. K. F., “A New Rate-Based Switch Algorithm for ABR Traffic to Achieve Max-Min Fairness with Analytical Approximation and Delay Adjustment,” IEEE INFOCOM 96, 1996, Vol. 3, pp. 1174-1181.

[ZuC94] Zukerman, M., and Chan, S., “Congestion Control by Maintaining Fairness in High Speed Data Networks,” IEEE Globecom, Vol. 3, 1994, pp. 1576-1580.