

# QoS Routing in Hypercube Multicomputers <sup>\*</sup>

Jie Wu and Fei Dai  
Department of Computer Science and Engineering  
Florida Atlantic University  
Boca Raton, FL 33431  
{jie,fdai}@cse.fau.edu

## ABSTRACT

In this paper, we present a coding method to capture QoS information in hypercube multicomputers. The coding method is based on a localized algorithm where each node interacts with its neighbors to gather QoS information. Specifically, each node maintains a QoS vector where the  $k$ -th element represents the guaranteed QoS performance to a destination that is  $k$  hops away. The localized algorithm exhibits desirable properties of self-stabilizing, self-optimizing, and self-healing. Simulation results show this coding method provides a good approximation of the minimum QoS value to a  $k$ -hop destination, and at the same time, uses a relatively small number of packets to propagate a change in link state (QoS value) compared with the classical distributed Bellman-Ford method.

**Keywords:** *Approximation, hypercubes, link state, localized algorithms, QoS.*

## 1. INTRODUCTION

Nowadays, more and more computer applications demand for quality of services (QoS) in communication subsystems [7, 8]. The quality of traffic that flows through a communication subsystem depends on the following four factors: *topology, routing, flow control, and switching*. Recently, a few routers with QoS provisioning have been proposed [2]. Most of the proposals focus on switching and/or flow control based on either the hardware solution [4] or the software solution [1]. In this paper, we focus on QoS routing based on a given regular topology. Although the focus is in the hypercube topology, we believe the result obtained in this paper can potentially be extended to other multicomputers and cluster networks with regular topologies.

QoS in routing typically deals with constraints on the selected path. Assume  $m(u, v)$  is the performance metric for the link  $(u, v)$  connecting host  $u$  to host  $v$  and a path  $(u, u_1, u_2, \dots, u_k, v)$  is a sequence of links connecting  $u$  to  $v$ . Three types of constraints on the path are given in [5]:

1. *Additive constraints:* A constraint is additive if  $m(u, v) = m(u, u_1) + m(u_1, u_2) + \dots + m(u_k, v)$ . For example, the end-to-end *delay* $(u, v)$  is an additive constraint which is equivalent to the summation of delays at each link.
2. *Multiplicative constraints:* A constraint is multiplicative if  $m(u, v) = m(u, u_1) \times m(u_1, u_2) \times \dots \times m(u_k, v)$ . The probability  $prob(u, v)$  for a packet to reach  $v$  from  $u$  is the product of individual link probabilities.
3. *Concave constraints:* A constraint is concave if  $m(u, v) = \min\{m(u, u_1), m(u_1, u_2), \dots, m(u_k, v)\}$ . The bandwidth  $band(u, v)$  available along path from  $u$  to  $v$  is the minimum bandwidth among the links on the path.

We focus on QoS with concave constraints. The objective is to find a path that meets the given QoS requirement. Consider  $m(u, v)$  as the link state of  $(u, v)$ , each node can collect global link state through global distribution. Then each source node can easily determine a path to a given destination that meets the QoS requirement. However, link state is a dynamic element and it needs to be flooded to the whole network whenever there is a change in link state. In this paper, we propose a coded scheme to approximate the minimum QoS value of paths to destinations in  $k$  hops. The approximation is based on a conservative approach by taking the advantage of both the unique topological properties of hypercubes and concave QoS constraints. This approach can also be considered as an extension to Wu's safety level model [6] for fault-tolerant routing in hypercube multicomputers. Note that in fault-tolerant routing, there are two levels of QoS for each link: up (1) and down (0).

The localized algorithm exhibits desirable properties of self-stabilizing, self-optimizing, and self-healing. Simulation results show this coding method provides a good approximation of the minimum QoS value to a  $k$ -hop destination, and at the same time, uses a relatively small number of packets to propagate a change of link state (QoS value) compared with the classical Bellman-Ford method. The latter feature is also called *localized maintenance*.

The rest of this paper is organized as follows: Section 2 presents a coded scheme called *QoS vector* and a distributed

---

<sup>\*</sup>This work was supported in part by NSF grant CCR 9900646 and ANI 0073736.

QoS routing algorithm based on single QoS vectors. Section 3 proposes an extension to the single QoS vector scheme called *double QoS vector* scheme, which achieves higher guaranteed bandwidth through longer paths. Section 4 compares the performance of the single QoS vector algorithm and the distributed Bellman-Ford algorithm with simulation. Section 5 concludes this paper. Although in this paper we focus only on QoS routing, in real applications QoS routing should be accompanied and to cooperate with with QoS traffic control.

## 2. MODEL

Let  $u$  be a node in an  $n$ -dimensional hypercube ( $n$ -cube) and it is represented by  $n$  bits  $u_1u_2\dots u_n$ . There are  $2^n$   $n$ -bit combinations assigned to  $2^n$  nodes in an  $n$ -cube with each node being assigned a distinctive code. Two nodes are connected if and only if their addresses differ in one bit position.  $u^i$  denotes the neighbor of  $u$  along dimension  $i$ . A dimension is called a *preferred dimension* with respect to  $v$  if the neighbor along the dimension is closer to  $u$  than the current node; otherwise, it is called a *spare dimension*.  $B(u) = (b_1, b_2, \dots, b_n)$  is a QoS vector associated with  $u$ , where  $b_k$  is called the  $k$ -hop QoS element of  $u$ .  $b_k^j$  is called the  $k$ -hop QoS element of  $u$ 's neighbor along dimension  $j$ .  $m(u, u^j)$  is the QoS value associated with the adjacent link of  $u$  along dimension  $j$  and such information is called *link state information*. Normally, link state information is discretized into multiple levels correspond to levels of QoS.  $rank_k(B)$  selects the  $k$ -th smallest element from a given vector  $B$ .

*Definition 1.* Elements of  $B(u)$  are defined as follows:

$$b_1 = rank_1\{m(u, u^1), m(u, u^2), \dots, m(u, u^n)\}$$

and

$$b_k = rank_k\{\min\{m(u, u^1), b_{k-1}^1\}, \min\{m(u, u^2), b_{k-1}^2\}, \dots, \min\{m(u, u^n), b_{k-1}^n\}\}$$

for  $k = 2, 3, \dots, n$ .

In  $B(u)$ ,  $b_k$  corresponds to the minimum QoS value to any node in  $k$  hops, assuming only preferred neighbors are eligible (i.e., only optimal paths are considered).  $rank_k$  is used because  $u$  has exactly  $k$  preferred neighbors to any  $k$ -hop destination and it corresponds to the minimum QoS value among any  $k$  selected neighbors.

Knowing the specific location of the destination, coded QoS value can be polished as follows: Let  $op(b_k)$  be the minimum QoS value to a given destination in  $k$  hops, assuming that only preferred neighbors are eligible. Let  $sub\_op(b_k)$  be the minimum QoS value to any node in  $k$  hops, assuming that any neighbor can be selected: preferred or spare.  $sub\_op(b_k)$  corresponds to the minimum QoS value of a sub-optimal path. A sub-optimal path consists of all except one preferred dimensions, and hence, its length is 2 more than the optimal one. The neighbors' status can be determined

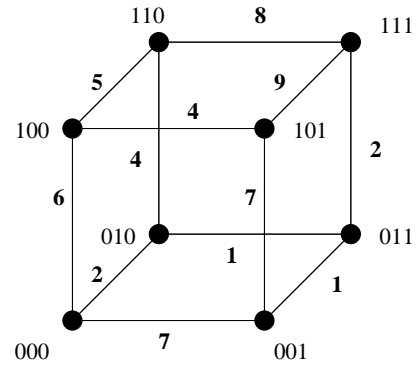


Figure 1: A sample 3-cube.

through the relative location of the source and destination by the exclusive-OR operation on the source and destination addresses. In the result of the exclusive-OR operation, the dimension that has 0 corresponds to a spare dimension and the one that has 1 corresponds to a preferred dimension. Assume that  $PD$  is the set of preferred dimensions and  $SD$  is the set of spare dimensions.

$$op(b_k) = \max\{\min\{m(u, v^i), b_{k-1}^i\}, i \in PD\}$$

If  $\max\{m(u, u^i), b_{k-1}^i\} < \max\{m(u, u^j), b_{k+1}^j\}$ , where  $i \in PD$  and  $j \in SD$ , then the sub-optimal path has a higher QoS value:

$$sub\_op(b_k) = \max\{\min\{m(u, u^j), b_{k+1}^j\}, j \in SD\}$$

Let  $op\_dim(b_k)$  and  $sub\_op\_dim(b_k)$  be the corresponding dimensions. The route is selected as follows:

---

### Routing process

(at the source node):

- The guaranteed QoS value of a minimal path is  $op(b_k)$  and the corresponding dimension  $op\_dim(b_k)$  is selected as the forwarding dimension if  $op\_dim(b_k)$  meets the given QoS requirement; Otherwise,
- if  $sub\_op\_dim(b_k)$  meets the requirement, the guaranteed QoS value of a sub-minimal path is  $sub\_op(b_k)$  and the corresponding dimension  $sub\_op\_dim(b_k)$  is selected as the forwarding dimension.

(at an intermediate node  $u$ ):

- Always select a preferred dimension with the highest  $\min\{m(u, u^i), b_{k-1}^i\}$ , where  $i \in PD$ .
- 

The approximation can be further enhanced if each node has 2-hop information. In this case, both  $b_1$  and  $b_2$  have accurate QoS information. In order to calculate  $b_i$  ( $i > 2$ ),

	000	001	010	011	100	101	110	111
$b_1$	2	1	1	1	4	4	4	2
$b_2$	1	2	2	1	4	2	2	4
$b_3$	4	2	2	2	2	4	4	2

**Table 1: QoS estimates (1-hop information).**

	000	001	010	011	100	101	110	111
$b_1$	-	-	-	-	-	-	-	-
$b_2$	1	2	2	1	4	2	4	4
$b_3$	4	2	4	2	4	4	4	4

**Table 2: QoS estimates (2-hop information).**

the approximation starts at  $b_2$ .  $L$  bandwidth levels are used to represent QoS values. Figure 1 shows a sample 3-cube where QoS values are represented by 10 levels (from integer 0 to 9). The QoS associated with links are as follows:  $00^* : 7$ ,  $01^* : 1$ ,  $10^* : 4$ ,  $11^* : 8$ ,  $0^*0 : 2$ ,  $0^*1 : 1$ ,  $1^*0 : 5$ ,  $1^*1 : 9$ ,  $*00 : 6$ ,  $*01 : 7$ ,  $*10 : 4$ , and  $*11 : 2$ . Table 1 shows the QoS estimates based on 1-hop neighborhood information while Table 2 shows the ones based on 2-hop neighborhood information.

Note that when  $L = 2$ , the corresponding systems represent hypercubes with faulty links with 0 representing a faulty link and 1 as a healthy link.

### 3. EXTENSIONS

In the previous section, when the given QoS requirement cannot be satisfied by a minimal path, a sub-minimal path may be used to meet the QoS requirement, with the constraints that (1) there is only one detour, and (2) only the source node can select a spare dimension. While constraint (1) is important to avoid loop, constraint (2) can be relaxed to allow a detour at an intermediate step; rather than just the first step from the source. The following extension uses an extra QoS vector  $D(u) = (d_1, d_2, \dots, d_n)$  for each node  $u$ , where  $d_k$  is called the  $k$ -hop QoS element with detour. Similarly,  $d_k^j$  is called the  $k$ -hop QoS element with detour of  $u$ 's neighbor along dimension  $j$ .

*Definition 2.* Elements of  $D(u)$  are defined as follows: For  $k = 1, 2, 3, \dots, n - 1$ , the QoS element with an *immediate detour* is

$$imm_k = \max\{\min\{m(u, u^1), b_{k+1}^1\}, \min\{m(u, u^2), b_{k+1}^2\}, \dots, \min\{m(u, u^n), b_{k+1}^n\}\}$$

For  $k = 2, 3, \dots, n$ , the QoS element with an *deferred detour* is

$$def_k = \text{rank}_k(\min\{m(u, u^1), d_{k+1}^1\}, \min\{m(u, u^2), d_{k+1}^2\}, \dots, \min\{m(u, u^n), d_{k+1}^n\})$$

For  $k = 1, 2, 3, \dots, n$ , the QoS element with one detour is

$$d_k = \max\{b_k, imm_k, def_k\}$$

	000	001	010	011	100	101	110	111
$d_1$	4	2	2	4	4	4	4	2
$d_2$	2	4	4	2	4	2	2	4
$d_3$	4	4	2	2	2	4	4	2

**Table 3: Extended QoS estimates (1-hop information).**

In  $D(u)$ ,  $d_k$  corresponds to the minimum QoS value to any node in  $k$  hops, assuming at most one detour is allowed in the entire path.  $imm_k$  and  $def_k$  are intermediate results and not visible to  $u$ 's neighbors. Given the specific location of the destination, sets of preferred dimensions  $PD$  and set of spare dimensions  $SD$ , we can compute

$$\begin{aligned} op(d_k) &= \max\{\min\{m(v, v^i), b_{k-1}^i\}, i \in PD\} \\ imm\_sub\_op(d_k) &= \max\{\min\{m(u, u^j), b_{k+1}^j\}, j \in SD\} \\ def\_sub\_op(d_k) &= \max\{\min\{m(v, v^i), d_{k-1}^i\}, i \in PD\} \end{aligned}$$

Let  $op\_dim(d_k)$ ,  $imm\_sub\_op\_dim(d_k)$  and  $def\_sub\_op\_dim(d_k)$  be the corresponding dimensions. Note that both  $def\_sub\_op(d_k)$  and  $imm\_sub\_op(d_k)$  correspond to  $sub\_op(d_k)$ . The difference is that  $imm\_sub\_op(d_k)$  takes the detour immediately at the source while  $def\_sub\_op(d_k)$  takes a detour at a later step. In the following extended routing process, each routing request carries one bit *detour flag* which is initially set to be *true*:

---

#### Extended routing process

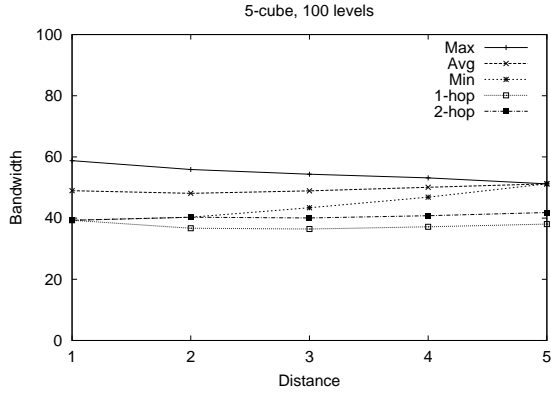
(when detour flag is *false*):

- If  $op\_dim(d_k)$  meets the given QoS requirement, The guaranteed QoS value of a minimal path is  $op(d_k)$  and  $op\_dim(d_k)$  is selected as the forwarding dimension; Otherwise,
- if  $def\_op\_dim(d_k)$  meets the requirement, the guaranteed QoS value of a minimal or sub-minimal path is  $def\_op(d_k)$  and  $sub\_op\_dim(d_k)$  is selected as the forwarding dimension; Otherwise,
- if  $imm\_op\_dim(d_k)$  meets the requirement, the guaranteed QoS value of a sub-minimal path is  $imm\_op(d_k)$ ,  $imm\_op\_dim(d_k)$  is selected as the forwarding dimension, and the detour flag is set to *true*.

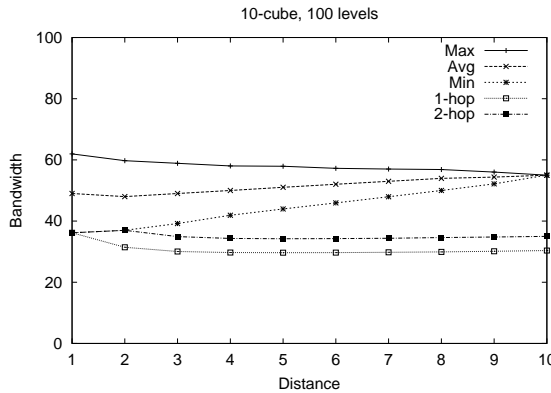
(when the detour flag is *true*):

- Always select a preferred dimension with the highest  $\min\{m(u, u^i), b_{k-1}^i\}$ , where  $i \in PD$ .
- 

Table 3 shows the extended QoS estimates with at most one detour based on 1-hop neighborhood information of Figure 1.



(a)



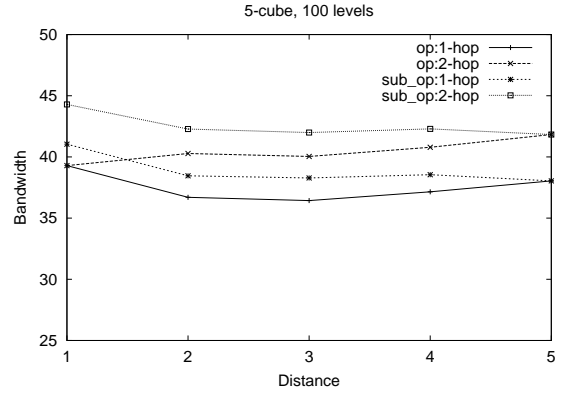
(b)

**Figure 2: The bandwidth guaranteed by the QoS vector scheme and the distributed Bellman-Ford scheme in (a) 5-cubes and (b) 10-cubes.**

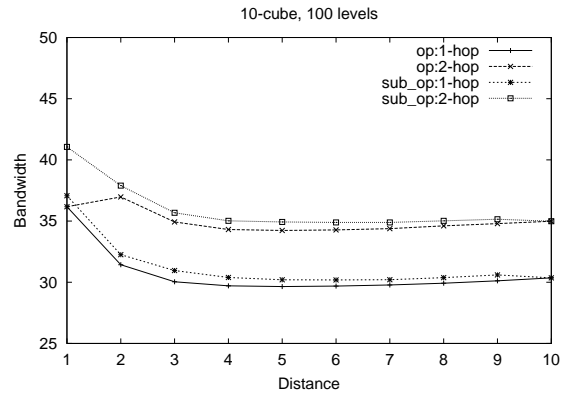
#### 4. SIMULATION

Simulations are conducted to compare the proposed QoS vector scheme with the classical distributed Bellman-Ford scheme, in terms of performance and overhead. The distributed Bellman-Ford scheme maintains at each node a complete routing table with every destination, its maximum bandwidth, and the corresponding next hop information. The performance of a QoS routing scheme is its ability to satisfy a QoS routing request. The overhead includes the memory to store the QoS information and the control packets used to propagate such information over the network. Currently only the basic single QoS vector scheme with optimal and sub-optimal path routing is simulated. The extended double QoS vector scheme that keep two QoS vectors  $B(u)$  and  $D(u)$  will be simulated as part of future work.

The hypercube networks used in the simulation are generated with the following bandwidth model: For a given hypercube dimension  $n$ , we build hypercubes with  $2^n$  nodes and  $2^{n-1}n$  bidirectional links. The bandwidth level of each link is an integer number between 0 and 99 (i.e.,  $L=100$ ). The residual (available) bandwidth of each link is equal to the



(a)



(b)

**Figure 3: The bandwidth guaranteed for the minimal paths and the sub-minimal paths by the QoS vector scheme in (a) 5-cubes and (b) 10-cubes.**

maximum bandwidth minus the reserved bandwidth, which is the total bandwidth reserved by a set of independent QoS flows that use this link. We assume the *central limit theorem* applies in this case and treat each bandwidth level as a random variable with Gaussian (normal) distribution, which is roughly emulated by computing the average value of 12 random variables with uniform distribution (the *convolution* method [3]). Each simulation is repeated long enough to achieve a 90% confident interval of  $\pm 10\%$ .

The first set of simulations evaluate the performance of the single QoS vector scheme; that is, the maximum bandwidth can be guaranteed to a QoS routing request. For the single QoS vector scheme, we compute for each node the QoS vector containing  $b_k$  ( $k = 1, 2, \dots, n$ ), the estimated minimum bandwidth to any node in  $k$  hops based on 1-hop and 2-hop neighborhood information. For the distributed Bellman-Ford scheme, we compute the maximum bandwidth ( $Max$ ), minimum bandwidth ( $Min$ ), and average bandwidth ( $Avg$ ) to any node in  $k$  hops. Figure 2 shows the results on 5-cubes and 10-cubes. For the destinations in  $k$  hops, the difference among  $Max_k$ ,  $Min_k$ , and  $Avg_k$  is relatively small and get

smaller as  $k$  increases. There is only one node that is  $n$  hops away, and therefore,  $Max_n = Min_n = Avg_n$ .  $Min$  is the upper bound for each QoS element,  $b_1 = Min_1$  when 1-hop information is used;  $b_2 = Min_2$  when 2-hop information is used.  $b_k$  is usually smaller than  $Min_k$  because QoS information becomes inaccurate during propagation. As  $k$  increases, the difference is larger because the QoS information is less accurate for distant destinations. The situation is worse for 10-cubes, where more details are omitted by the QoS information coding scheme. However, this difference in performance is relatively small, and using 2-hop neighborhood information improves the situation.

Figure 3 shows the improvement achieved by allowing a detour at the source node. The average bandwidth guaranteed by a sub-minimal path ( $sub\_op : 1-hop$  and  $sub\_op : 1-hop$ ) is only slightly better than that guaranteed by a minimal path ( $op : 1-hop$  and  $op : 1-hop$ ). However, this improvement can be combined with the 2-hop information method, and is achieved without extra communication cost.

Compared with the distributed Bellman-Ford scheme (each routing table contains  $2^n$  entries), the single QoS vector scheme uses much less memory (each QoS vector contains  $n$  entries). Table 5 shows the same ratio for the communication overhead. Here we compare the average number of control packets caused by a single link state information change for different schemes. After the computation of the bandwidth vectors (i.e., routing table in the distributed Bellman-Ford scheme and QoS vector in the single QoS vector scheme) converges in an  $n$ -cube, a link  $(v_1, v_2)$  is randomly selected to change its bandwidth level and the bandwidth vectors of hosts  $v_1$  and  $v_2$  are re-computed. If the new bandwidth vector is different from the old one in any host  $v_i$ , its neighbors are notified with  $n$  control packets containing the changes in the bandwidth vector. Any node receiving the control packet also need to update its bandwidth vector and may send more control packets. This process repeats until the computation converges again. As the number of dimensions increases from 3 to 10, the average number of control packets caused by a single link state information update increases exponentially in the distributed Bellman-Ford scheme; while the same metric increases linearly in the single QoS vector scheme. That is, the single QoS vector scheme has by far the least routing overhead. We expect the same amount of control packets for the double QoS vector scheme. However, its effectiveness still needs to be thoroughly simulated.

## 5. CONCLUSION

In this paper, we have proposed an approximation scheme to capture the minimum QoS value for support routing in hypercubes. The approximation method takes the advantage of the unique topological properties of hypercubes and concave QoS constraints. In addition, the calculation of the approximation is localized. This approximation also exhibits the desirable property of localized maintenance when the link state (QoS value) changes. Preliminary simulation

Dimensions	Bellman-Ford	QoS Vector
3	12.9	11.1
4	26.7	19.5
5	54.9	29.2
6	111.3	42.0
7	228.4	53.6
8	460.4	69.8
9	929.3	78.1
10	1867.2	81.7

**Table 4: Average number of control packets used to propagate a change in link state.**

results are promising and show desirable cost-effectiveness compared with the classical distributed Bellman-Ford method. In-depth simulation, under double QoS vector scheme and/or more sophisticated QoS constraints, will be our future work.

## 6. REFERENCES

- [1] A. A. Chien and J. H. Kim. Approaches to quality of service in high-performance networks. In *Proc. of Parallel Computer Routing and Communications Workshop*, pages 98–108. Lecture Notes in Computer Science, Springer-Verlag, July 1997.
- [2] H. Eberle and E. Qertli. Switzerland: A QoS communication architecture for workstation clusters. In *Proc. of Intl. Symp. Comp. Arch.*, pages 98–108, June 1998.
- [3] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, Inc, 1991.
- [4] Myricom Inc. M3M-PCI64B network interface card. <http://www.myri.com/myrinet/PCI64/m3m-pci64b.html>.
- [5] B. Wang and C.-J. Hou. A survey on multicast routing and its QoS extension: Problems, algorithms, and protocols. *IEEE Network Magazine*, 14(1), Jan./Feb. 2000.
- [6] J. Wu. Reliable unicasting in faulty hypercubes using safety levels. *IEEE Transactions on Computers*, 46(2):241–247, Feb. 1997.
- [7] K. H. Yum, E. J. Kim, and C. R. Das. QoS provisioning in clusters: An investigation of router and NIC design. In *Proceedings of the International Symposium on Computer Architecture*, pages 120–129, June 2001.
- [8] K. H. Yum, A. S. Vaidya, C. R. Das, and A. Sivasubramaniam. Investigating QoS support for traffic mixes with the MediaWorm router. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture*, pages 97–106, Jan. 2000.