JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 20, 733-751 (2004)

Path Selection on VC-Networks within a Distributed Media Server for Concurrent Stream Retrieval

CHUN-CHAO YEH AND JIE-YONG JUANG*
Department of Computer Science
National Taiwan Ocean University
Keelung, 202 Taiwan
**Department of Computer Science and Information Engineering*
Taichung Healthcare and Management University
Taichung, 413 Taiwan

This article addresses path selection problems that arise in a large-scale media server. In particular, the problem on a distributed server with two tiers of server nodes connected by a VC-enabled interconnection network is considered. To serve each incoming request, a pair of server nodes should be selected to handle the request. In addition, a network path in the interconnection network should be arranged to pipe the program stream between the pair of server nodes for the request. The problem is modeled as a network flow problem. A two-pass flow (TPF) algorithm is proposed to solve the flow problem that involves multiple types of flow. The proposed algorithm allocates system resources effectively to maximize the number of admitted requests without violating resource constraints. Unlike some other works, the algorithm does not need to rely on assumptions about any particular configuration of the underlying interconnection network.

Keywords: path selection, bandwidth allocation, admission control, multi-commodity flow network, concurrent stream retrieval

1. INTRODUCTION

The popularity of media-streaming services is increasing the demand for powerful multimedia servers. To provide better cost/performance ratios and to ensure scalability, distributed (or clustered) server architecture is usually chosen to fulfill the requirements. A distributed server provides multiple server nodes interconnected by an interconnection network (or LAN) to serve a large number of requests simultaneously. While most of the systems currently available use simple network architectures (for example, a fast Ethernet switch) to connect server nodes, this study argues that a more complex network system, consisting of multiple routing nodes, is needed to connect more server nodes to form a large-scale server to meet future needs. The demands and related design issues of a large-scale server have been pointed out in [1, 2, 6]. This study addresses path selection problem associated with such a large-scale media server. In particular, such a problem on a server with two-tier architecture is considered since it is considered to be scalable and cost-effective. Typically, in two-tier server architecture, some server nodes are configured as front-end (delivery) nodes, and some are configured as back-end (storage) nodes. Communication between front-end nodes and back-end nodes proceeds via a dedicated

Received July 10, 2002; revised March 6 & July 30, 2003; accepted September 17, 2003.
Communicated by Chu-Sing Yang.

interconnection network. When user requests arrive at a front-end node, the server chooses some server nodes to respond to the requests. In such a media server, the front-end node is responsible for fetching the requested program stream from the back-end server nodes and then forwarding the stream to the user. To ensure seamless stream retrieval, the following problems must be properly addressed. (1) How should a front-end node be selected to handle the request? (2) How should a back-end node be selected to provide the program stream? (3) How should a sequence of network links be selected to pipe the program stream from the back-end node to the front-end node? Briefly, in responding to each request, a network path, including two end points (a front-end node and a back-end node), must be effectively arranged so that the request can be served without violating resource constraints.

In a large media server with a complex interconnection network, the path selection problem is important. Inappropriate path selection promotes network congestion and hot-spot problems. The properties of media stream transmission, such as long transmission duration, high data volume, and real-time constraints, necessarily make the problem more serious. Consider an Internet user who wishes to view an MPEG1-encoded movie from the server. All of the requesting packets can be sent in a few seconds at little cost in terms of bandwidth required for transmission, while delivery of the requested movie can take more than an hour, at a network bandwidth cost as high as 1.5Mbps for an MPEG1-encoded video. Consequently, this study argues that effectively organizing network flow within the distributed server at some computational cost is worth doing. This study proposes an efficient algorithm to allocate system resources effectively. Under the constraints of currently available system resources, the algorithm maximizes the number of admitted requests. Meanwhile, balanced utilization of system resources is also considered.

The path selection problem discussed herein is similar to the processor-memory mapping problems on a multiprocessor system. Previous works on the processor-memory mapping problems have produced extensive results reported in the literature. Design issues such as message routing over the interconnection network and the consideration of blocking, permutation and multicasting have been comprehensively addressed [9]. Additionally, Rathi, Tripathi, and Lipovski [10], Fung and Torng [11] and Marsan and Gerla [12] have studied the performance of resource allocation on various multiprocessor architectures. However, most of these studies were based on some well-defined network configurations, including hyper-cube, mesh and ring configurations, or on some regular MIN (multistage interconnection network) structures. In contrast with these previous works, this paper deals with a general network topology. In addition, Elmallah and Culberson [15] studied general routing problems on various classes of MINs and transformed the problems into (integral) multi-commodity flow problems. However, their work is only applicable to some MIN configurations and not to a general network configuration.

Juang and Wah [13] proposed a transformation scheme for transforming multiprocessor resource-sharing problems into network flow problems, to which existing algorithms can be applied. Their method is applicable to any general loop-free network configuration. They transformed the resource-sharing problem into a *minimum-cost integral flow* problem. Using a similar transforming method, they also demonstrated that similar problems on heterogeneous servers can be transformed into *multi-commodity minimum cost integer flow* (MCMCIF) problems. However, integral solutions to this class of prob-

lems cannot be guaranteed [14]. Compared with their work, similar solution techniques are applied in this study. However, the problem considered here has different characteristics and is modeled differently.

More recent studies based on similar system architectures can be found in [5, 6]. Chang et al. in [5] considered multimedia file allocation problems on a distributed media server in which a VC network is assumed to provide connection between all server nodes. To serve each request, a network path should be determined to pipe the requested media-file from the storage node of the media-file resident to the front-end node at which the request arrives. Giving both file information (including the set of media files to be deployed and their access behaviors) and server information (including the network structure and all link bandwidth constraints), they developed efficient file allocation schemes to minimize transfer time of the bottleneck links on the interconnection network. Comparing with this paper, a similar network model (VC network) and constraints (link bandwidth along the flow path) are considered, but the objective is different. Consequently, the techniques used to solve the problem and the results are different. Berenbrink and Brinkmann [6] considered the distributed path selection scheme in a customized storage network called PRESTO, which consists of a set of intelligent routing switches, called active routers. Similar problems under similar server architecture (two-tier architecture with a VC-enabled network) were considered. As in [13], they modeled the distributed path selection problem as an MCMCIF problem. Although their problem model is quite general, the problem is hard to solve (since it is an NP-complete problem). They proposed different heuristic algorithms to solve the problem under some network topological configurations. Tay and Pang [20] proposed a load-sharing scheme for minimizing the request waiting time. The server node capacity is considered in different queuing models. However, the general interconnection network topologies and individual link states of the network are not accounted for.

Many issues are related to the design of an efficient media server, but this study focuses on path selection inside a distributed media server. These issues include for example, disk scheduling [16], file displacement and retrieval [5], VBR [17]/CBR [18] streaming, distributed media-object sharing/synchronization [19], and system availability [21], among others. Each of them deserves to be discussed, but they are beyond the scope of this study. This study makes no assumptions about above design issues. The results of this study complement the preceding research cited above.

The following section details the target system architecture and the problems considered here. Section 3 depicts formal models of the path selection problem and problem transformation schemes used to transform the problem into a network flow problem. Section 4 presents an efficient algorithm, called TPF (two-pass flow) algorithm, to solve the problems. An example is provided to show how the algorithm operates. The correctness and complexity of the algorithm are briefly addressed as well. Section 5 draws conclusions.

2. TARGET SYSTEM ARCHITECTURE AND PROBLEM ISSUES

2.1 Architecture of Target Systems

Some assumptions are made about the two-tier media server discussed herein. First, the interconnection network supports VC (virtual-circuit) switching. A virtual-circuit

network enables the bandwidth to be reserved throughout the connection. Such a VC-enabled network can be realized using a customized-design network, such as PRESTO [6], or using VC-enabled routing devices, such as ATM switches [3]. This assumption can be relaxed when the underlying interconnection network consists of only one switch, as in several LAN-based cluster servers. Second, a powerful request dispatcher (which physically may include multiple nodes to provide sufficient computing power and fault-tolerant capability) is assumed to work together with the server. The dispatcher represents the media server externally. Requests are directed to the dispatcher accordingly. The dispatcher then forwards each request individually to a designated front-end node, based on the results of the path selection scheme (described later). The request forwarding scheme can be realized by using conventional HTTP redirection techniques. A flow manager (which could be the dispatcher or another system node) is configured to help the server establish a network path between the selected front-end node and back-end node pair for each requested program. The assumption of the existence of a dispatcher is normally made in distributed multimedia servers. Various web servers, including SWEB [7] and RobustWeb [8], employ a similar mechanism to redistribute requests directly or indirectly. Finally, we assume that the video stream provided by the system is compressed with same bit-rate. The network bandwidth consumed in transporting each media program in the interconnection network is assumed to vary little, under a CBR (constant bit rate) transmission scheme. For example, all media programs are encoded using CBR scheme with variable-quality frames, or a video smoothing scheme is applied to enable constant-quality VBR (variable bit-rate) -compressed videos to be transported over a CBR service network [18, 23, 24]. Notably, however, no assumption is made about the topology of the underlying interconnection network. The proposed algorithm does not rely on any assumption about the network topology, making it more realistically applicable to real problems. Meanwhile, the assumption of single bit-rate of video program necessarily makes the problem tractable. When multiple bit-rates of video program are allowed (that is, each program stream might demand different network bandwidth along a single network path), the problem resembles a maximal flow problem with "unsplittablility" constraints, which has been shown to be an NP-hard problem. The property and heuristic algorithms of the unsplittable flow problem can be found in [25].

Fig. 1 presents the system blocks and operations of the two-tier server architecture. Client requests are sent via the Internet to the dispatcher (step 1 in the figure). The dispatcher collects a set of requests that arrive within a specified time period. Based on the request patterns and current system state, the flow manager generates a path selection scheme, in which for each individual admitted request, a pair of server nodes (front-end node and back-end node) is assigned to respond to the request and the network path connecting this pair of nodes is determined. According to the assignment, the dispatcher redirects each admitted request to the front-end node corresponding to the pair of nodes selected for the request (steps 2 and 3). The front-end node handles the (redirected) request and forwards the request to the corresponding server node (step 4). The server node then begins to deliver the request stream to the front-end node along the network path selected for the admitted request (step 5). The front-end node forwards the program stream to the client (step 6). Data delivery continues until the stream has been completely transmitted.

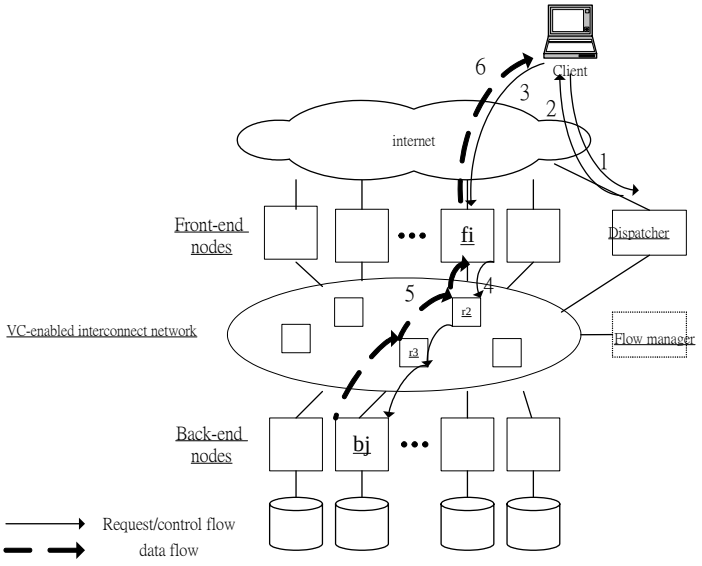

Fig. 1. Two-tier server architecture for the path selection problem.

2.2 The Path Selection Problem

Path selection problems in a two-tier media server arise because of the sharing of system resources, such as the processing power of the server nodes and the link bandwidth of the interconnection network. Transporting a video stream from a back-end node to a front-end node consumes a fraction of the bandwidth on each link along the network path connecting both of the nodes. Thus, only a limited number of streams can be delivered simultaneously. Similarly, each server node can serve only a limited number of streams because of the limitation on processing power or I/O capability. Meanwhile, for each request, the server should properly select a back-end node to serve the request. Only a node capable of providing the requested program and with enough processing power can be selected. Additionally, before a requested program is forwarded to the client, the requested program should be forwarded to the front-end node handling the request. Consequently, for each request, the server should guarantee the existence of an available network path to pipe the requested program stream between the back-end node and front-end node pair selected for the request. For each request, multiple paths may exist for the pair of nodes selected to handle the request. Last but not least, the server should handle multiple requests concurrently. Given all these constraints, an intuitive selection scheme is to randomly choose a feasible path and nodes for each request. However, this scheme is prone to cause blocking on some critical paths, resulting in poor overall performance. Besides, the server status, for example, the utilization and availability of network links and nodes, can change dynamically. A selection scheme should be sufficiently flexible and efficient to adapt to possible changes in server configurations. Finally, an efficient path selection scheme should not only maximize system throughput, but also reduce the system cost, as measured against some criteria, such as resource load balance and priority.

3. PROBLEM MODELS

Inspired by [13], the path selection problem is herein modeled as a network flow problem. A graph $G = (V, E)$ represents the topology of a distributed system, where V represents the set of nodes and E represents the set of all network links. First, a general problem model is considered; this model yields an equivalent *multi-commodity minimum cost Integer flow* (MCMCIF) problem. Although the MCMCIF flow model is sufficiently powerful to model the path selection problem, it is hard to solve. The authors in [13] and [6] experienced similar problems with such a model. A simplified model is presented here to deal with the path selection problem better. Without lose of generality, conventional terms in the flow network are reserved. That is, hereafter, this study refers to *types of services* (in the transformed network flow model) as *programs* (in the path selection problem). A *service flow* (*or flow shortly*) in the network model corresponds to a *program stream* (*or stream shortly*) in the problem; a request to transport a specified type of *service* over the flow network in the model corresponds to a request to fetch a specified program *file* over the interconnection network in the problem.

3.1 A General Model for the Path Selection Problem

First, a general problem paradigm is considered, in which the interconnection network is aware of the variation of program streams. Consider a two-tier media server: multiple requests are made for different program files on different front-end nodes. To serve a request, a back-end node that provides the requested program file should be selected, and a network path for piping the program stream from the back-end node to the front-end node must be available. Since each request consumes a fixed amount of system resources, a feasible assignment of resources should not violate the resource constraints. The problem is similar to the so-called *multi-commodity integer flow* problem. Accordingly, the model of the path selection problem is similar to that of the flow problem. Mathematically, this problem model, called Model 1 hereafter, is as follows:

$$\text{Minimize } \sum_{1\le i\le E}\sum_{1\le j\le T} c_{ij}x_{ij}, \tag{1}$$

Subject to

$$\sum_{i\in Ei(k)} x_{ij} - \sum_{e\in Ee(k)} x_{ej} = 0, \text{ for all intermediate node } k, \text{ and type } j, \tag{2}$$

$$0 \le x_{ij} \le u_{ij}, \text{ for all link } i, \text{ and type } j, \tag{3}$$

$$\sum_{1\le j\le T} x_{ij} \le u_i, \text{ for all link } i, \tag{4}$$

$$\sum_{i\in Ei(t)} x_{ij} \le b_{tj}, \text{ for all back-end node } t, \text{ and type } j, \tag{5}$$

$$\sum_{i\le j\le T}\sum_{i\in Ei(t)} x_{ij} \le b_t, \text{ for all back-end node } t, \tag{6}$$

$$\sum_{e \in Ee(s)} x_{ej} \le \min(d_{sj}, q_{sj}, u_{sj}), \text{ for all front-end node } s, \text{ and type } j, \quad (7)$$

$$\sum_{1 \le j \le T} \sum_{e \in Ee(s)} x_{ej} \le \min(d_{sj}, q_{sj}, u_{sj}), \text{ for all front-end node } s, \quad (8)$$

$$x_{ij} \in I, \text{ for all link } i, \text{ and type } j. \quad (9)$$

In the above formulas, $Ei(x)$ denotes the set of all ingress-links to node x; similarly, $Ee(x)$ denotes the set of all egress-links from node x; Q_s in Eq. (8) equals the sum of all Q_{sj} for all types j; Table 1 summarizes all related notations. As in [13], the cost associated with an unassigned request is assumed to have a large enough value (virtually infinite value). Therefore, the solutions of the above formulas yield the maximal number of admitted requests. Furthermore, if more than one solution yields the same maximal number of admitted requests, the one with the minimal cost is selected. Eq. (2) maintains flow conservation associated with each intermediate node. Eq. (3) sets capacity constraints on individual types of flow associated with each link. Eq. (4) sets constraints on the total flow capacity associated with each link. Similarly, Eqs. (5) and (6) set constraints on the processing capacity associated with each back-end node; and Eqs. (7) and (8) set constraints on the processing capacity associated with each front-end node, in which the flow capacity constraints associated with the out-link of the front-end node and the number of requests available on the front-end node should be considered additionally. Finally, Eq. (9) presents the constraint that the number of admitted requests should be an integer. The above mathematical problem is equivalent to a *multi-commodity minimum cost integer flow* (MCMCIF) problem, which has been demonstrated to be NP-complete [14].

Table 1. Parameters in the general model.

Parameters	
number of links	N_L
number of intermediate nodes	N_I
number of front-end nodes	N_F
number of back-end nodes	N_B
number of service types	T
Service capacity of type j available in back-end node t	b_{tj}
Total Service capacity available in back-end node t	b_t
Service capacity of type j available in front-end node s	d_{sj}
Total Service capacity available in front-end node s	d_s
Link capacity of type j available in link i.	u_{ij}
Total Link capacity available in link i.	u_i
Link capacity of type j available in link i.	u_{sj}
Link capacity available for the out link of front-end node s.	u_s
Unit flow cost for type j passing through link i.	c_{ij}
Number of requests for type j pending at front-end node s.	q_{sj}

3.2 A Simplified Model

Based on the preceding general model, a simplified model is constructed for the path selection problem to fit the problem exactly. Examining the characteristics of the path selection problem described in previous sections reveals that some constraints in the above general model are not necessary for the problem. A modified model, called Model 2 hereafter, of the problem is as follows:

$$\text{Minimize} \sum_{1\le i\le E} \sum_{1\le j\le T} c_{ij} x_{ij}, \tag{10}$$

Subject to

$$\sum_{i\in Ei(k)} x_{ij} - \sum_{e\in Ee(k)} x_{ej} = 0, \text{ for all intermediate node } k, \tag{11}$$

$$\sum_{1\le j\le T} x_{ij} \le u_i, \text{ for all link } i, \tag{12}$$

$$\sum_{i\in Ei(t)} x_{ij} \le b_{tj}, \text{ for all back-end node } t, \text{ and type } j, \tag{13}$$

$$\sum_{1\le j\le T} \sum_{i\in Ei(t)} x_{ij} \le b_t, \text{ for all back-end node } t, \tag{14}$$

$$\sum_{1\le j\le T} \sum_{e\in Ee(s)} x_{ej} \le \min(d_s, u_s), \text{ for all front-end node } s, \tag{15}$$

$$x_{ij} \in I, \text{ for all link } i, \text{ and type } j, \tag{16}$$

$$c_{i1} = c_{i2} \ldots c_{iT} == c_i. \tag{17}$$

Table 2 summarizes the parameters of the modified model. The modified model eliminates (a) the constraints on the capacity of each link flow for each individual flow type (Eq. (3)) and (b) the constraints on the capacity of each front end node and on the number of requests available on the front-end node for each flow type (Eq. (7)). A partial modification is made in Eq. (8). The constraint on the number of available requests is eliminated, yielding Eq. (15). Also, the cost is independent of the type in the modified model, yielding Eq. (17). The modifications are based on the following reasons. First, the types of flow over a network link need not be differentiated. In the path selection problem considered in this study, different types of requests correspond to different programs of media-files. The interconnection network does nothing other than forward program streams. None of the links on the network are aware of the contents of the program stream being carried. Only the data rate is meaningfully related to the consumption of bandwidth associated with the links. Consequently, the cost term C_{ij} associated with each type of service sent over link i can be treated equally on the link. That is, $C_{i1} = C_{i2} = \ldots =$

Table 2. Parameters in the modified model.

Parameters	
number of links	N_L
number of intermediate nodes	N_I
number of front-end nodes	N_F
number of back-end nodes	N_B
number of service types	T
Service capacity of type j available in back-end node t	b_{tj}
Total Service capacity available in back-end node t	b_t
Total Service capacity available in front-end node s.	d_s
Total Link capacity available in link i.	u_i
Link capacity available for the out link of front-end node s.	u_s
Unit flow cost passing through link i.	c_i
Number of requests for type j pending at the dispatcher	q_j

$C_{iT} = C_i$ (Eq. (17)), and the constraints on the capacity of the network links for each type of services are unnecessary (Eq. (3)). Similarly, the constraints on the server capacity for individual types in front-end nodes (Eq. (7)) need not be considered because front-end nodes do not need to process the program stream (service) but need only forward the stream to the client. Finally, the constraints on the numbers of requests available on each front-end node are eliminated (Eqs. (7), and (8)) since in the target system, requests are sent to the dispatcher, not directly to front-end nodes.

3.3 Problem Transformation

A path selection problem mathematically modeled as Model 2 can be transformed into a special case of the MCMCIF problem, in which no explicit type of differentiation is imposed on the network links. That is, no capacity constraint on an individual type of flow is associated with network links, and individual unit flow costs for different types of flow associated with a link are all the same for the link. The transformation scheme is as follows.

1. All physical nodes (including the dispatcher) and network links in the server have corresponding nodes and links in the flow network.
2. A by-pass node p and a terminal node t are added to the flow network.
3. Assign link capacity u_i and unit flow cost c_i to the link that corresponds to network link i, $i = 1 \ldots N_L$. Assign the link capacity u_{si} and the unit flow cost (default = 0) to the link between node ss (corresponding to the dispatcher) and node s_i (corresponding to front-end node i, $i = 1 \ldots N_F$). Assign a link capacity vector (B_{i1}, B_{i2}, …, B_{iT}) and a cost vector (default = 0) to the link that connects terminal node t to node t_i (corresponding to back-end server node i, $i = 1 \ldots N_B$).
4. Expand each node s_i in the flow network (corresponding to front-end node i, $i = 1 \ldots N_F$) into two nodes, s_i and s_i, and connect the two nodes with a new link, as depicted in Fig. 2 (a). Associate the new link with the front-end node capacity and its unit flow cost.

5. Expand each node t_i in the flow network, (corresponding to back-end server node i, i = 1 … N_B) into two nodes, t_i and t_i', connected to each one by a link, as shown in Fig. 2 (b). Associate the new link with the back-end node capacity and its unit flow cost.

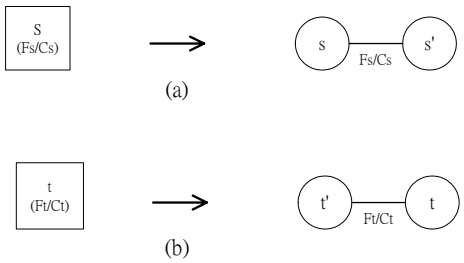

Fig. 2. Problem transformation for Step 4 (a): split each front-end node s into two nodes s and s'; that for Step 5 (b): split each back-end node t into two nodes, t' and t.

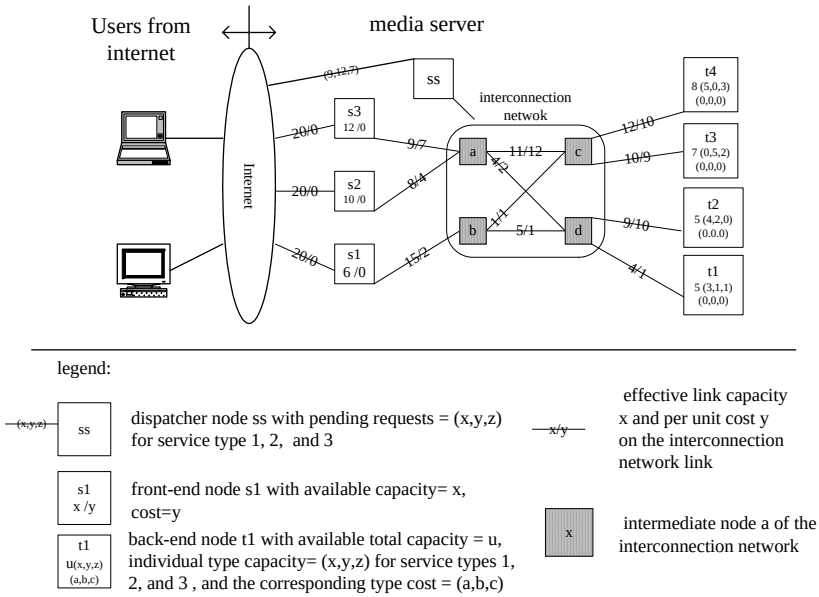

Fig. 3. An example of the path selection problem in a two-tier media server.

Fig. 3 presents an example of a two-tier multimedia server, which includes three front-end nodes and four back-end nodes interconnected by an interconnection network consisting of four routing nodes. The example shows the current system status in terms of incoming requests and available resources. Twenty-eight requests are available on the dispatcher: nine for type-1, 12 for type-2, and seven for type-3 service. The available capacities of the three front-end nodes ($s1$, $s2$, and $s3$) are assumed to be six, 10, and 12, respectively. The four back-end nodes ($t1$, $t2$, $t3$, and $t4$) are assumed to be able to serve

different types of requests with different capacities. Similarly, the link capacity and the unit flow cost are presented for each network link in the example. Fig. 4 shows the result of applying the above transforming scheme to the example.

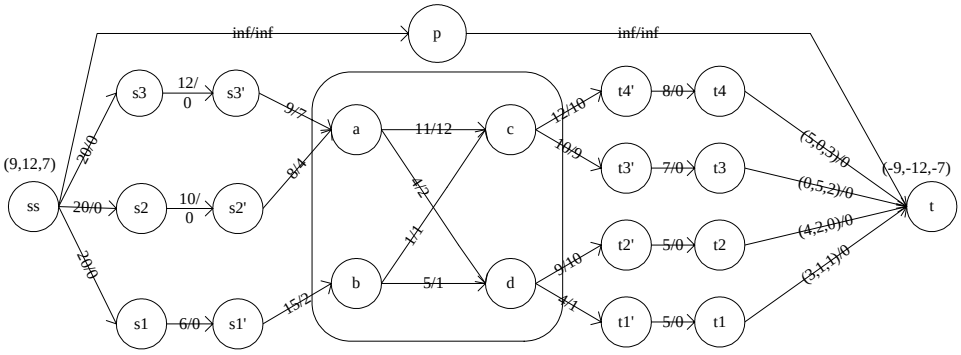

Fig. 4. An instance of the flow network G_1 transformed from the problem example shown in Fig. 3.

3.4 Parameter Setting

Most of the parameters in the problem model are functions of the system structure and device characteristics. The setting of capacity limits on individual server nodes and network links is a matter of policy. The upper bounds of these values can be obtained by means of off-line measurement. The cost terms can be set more flexibly. No physical constraint is associated with the cost terms. The setting depends totally on the resource allocation policy. For example, one way to balance the utilization of system resources is to assign a cost to a resource (server node or network link) in proportion to the inverse of the capacity available on the resource. Therefore, when two resources can be used to serve a request, the one with the lighter load (that is, with more capacity) is chosen.

4. TWO-PASS FLOW ALGORITHM

This section presents an algorithm, called the TPF (two-pass flow) algorithm, to solve the path selection problem described above. The two-pass flow algorithm, first constructs a flow network to represent the problem. Then, two passes of flow algorithms are applied to yield optimal solutions.

4.1 TPF (Two-Pass Flow) Algorithm

The two-pass flow algorithm is summarized below and illustrated using an example. Figs. 4 to 10 present the results of applying the algorithm step by step to the problem example depicted in Fig. 3.

Algorithm TPF

1. Generate a flow network, G_1, for the given problem. Transform a given path selection

problem into a network flow problem by using the transformation scheme presented in the previous section (section 3.3). The result of the transformation yields a flow network, named G_1 hereafter. Fig. 4 presents an instance of the flow network, G_1, obtained by transforming the path selection problem depicted in Fig. 3, as an example.

2. Reconstruct a flow network G_2 from G_1. Reconstruct the flow network G_1 by inserting T new nodes (denoted as q_{t1}, q_{t2}, …, q_{tT}) between terminal node t and nodes t_i, i = 1 … N_B. Replace the old links (t_i, t) with new links (t_i, q_{tj}) between nodes t_i and q_{tj} and new links (t, q_{ti}) between nodes t and q_{ti}, j = 1 … T, i = 1 … N_B. For each link (t_i, q_{tj}), assign a capacity vector (b_{i1}, b_{i2}, …, b_{iT}), where $b_{ik} = B_{ij}$ if $k == j$, else $b_{ik} = 0$. Similarly, for each link (t, q_{tj}), assign a capacity vector (q_1, q_2, …, q_T), where $q_k = Q_j$ if $k == j$, else $q_k = 0$. The result of reconstruction yields a new flow network, named G_2 hereafter. Following the above example, Fig. 5 presents an instance of the flow network G_2 reconstructed from the one depicted in Fig. 4.

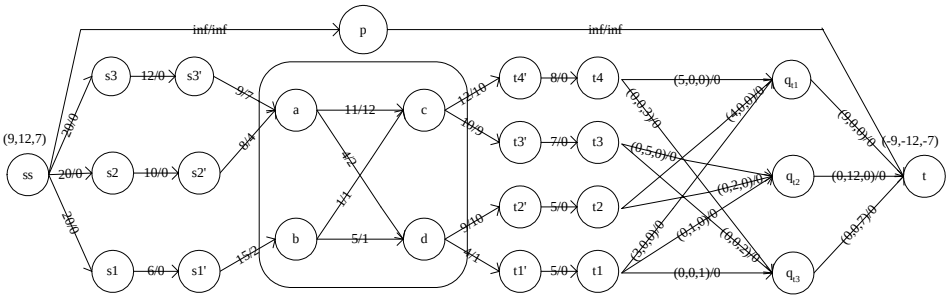

Fig. 5. The results for the example after Step 2 in the TPF algorithm.

3. Apply conventional flow algorithm to a single-type flow network obtained from G_2. (First pass) Treat the flow network G_2, determined in Step 2, as a single-commodity flow network by considering all types of flow as a single flow type. That is, replace the capacity constraint vector (u_1, u_2, …, u_T) for T types of flow with the capacity constraint value $u = u_1 + u_2 + \ldots + u_T$. Also, replace the request vector (Q_1, Q_2 , …, Q_T) with the number of total requests $Q = Q_1 + Q_2 + \ldots + Q_T$. Then, apply conventional minimal cost flow algorithm [14] to this single-type flow network, yielding the optimal flow assignment, denoted as $A = \{a_i \mid i = 1, 2, \ldots, |E_2|\}$ hereafter, where $|E_2|$ is the number of links in G_2. The value of a_i indicates the amount of flow that will pass through link i when maximal flow (at minimal cost) is achieved in the flow network. Fig. 6 presents the results of applying the flow algorithm to the example flow network depicted in Fig. 5.
4. Reconstruct a flow network G_3 from G_2. Reverse the flow direction of the network G_2 obtained in Step 2 (such that the flow direction is from node t to node *ss*). For each link i in the flow network, replace the value of the capacity constraint associated with the link with the value a_i obtained in Step 3. Based on the result shown in Fig. 6, Fig. 7 presents an instance of the flow network G_3 reconstructed from the one depicted in Fig. 5, as an example.

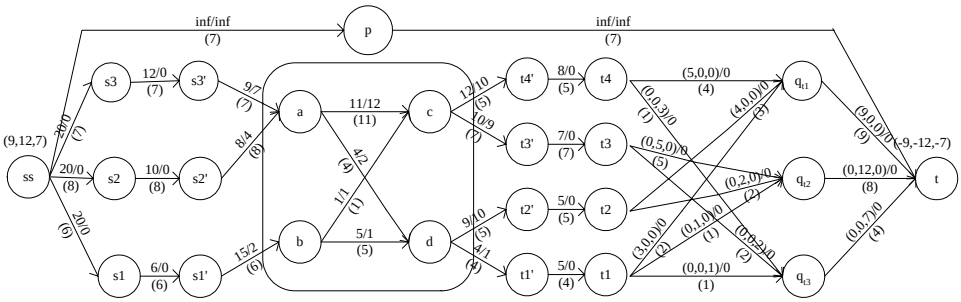

Fig. 6. The results for the example after Step 3 in the TPF algorithm. (The value inside the parentheses below each link indicates the result of flow assignment.)

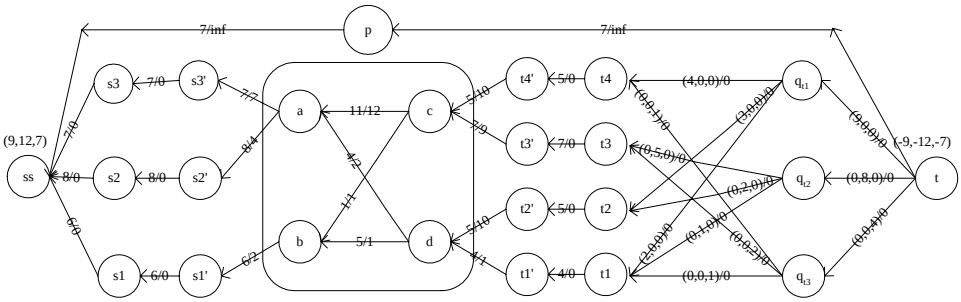

Fig. 7. The results for the example after Step 4 in the TPF algorithm.

5. Apply flow algorithm to G_3. (Second pass) For each type of flow, say type k, on the flow network G_3, obtained in Step 4, perform the following. (Assume each link i initially has a capacity constraint u_i.)
   (a) Apply conventional minimal cost flow algorithm to the type-k flow on flow network G_3 by turning off all the links (t, q_{tj}) for all $i \neq k$. , thus obtaining an optimal flow assignment, say $F^k = \{F^k_i \mid i = 1, 2, \ldots, |E_3|\}$ hereafter, for the flow network G_3 , where $|E_3|$ is the number of links in G_3.
   (b) For each link i in the network, update the capacity constraint by decreasing the value by F^k_i. That is, $u_i = u_i - F^k_i$.

Figs. 8 to 10 show the results of optimal flow assignment for the instance of the flow network G_3 shown in Fig. 7, which was obtained from the example problem depicted in Fig. 4.

6. Obtain the paths selected for each admitted request from $\{F^k \mid k = 1, \ldots, T\}$. The flow assignments $\{F^k \mid k = 1, \ldots, T\}$, obtained in Step 5, are the optimal flow assignments to the flow network. The assigned flow on the by-pass links represents the requests that cannot be admitted for the moment. Each flow path for type-k service in the flow network corresponds to a path selected for a request of program k in the server. That is, if a type-k flow f passes through nodes t_i' and s_j' in the flow network, then the

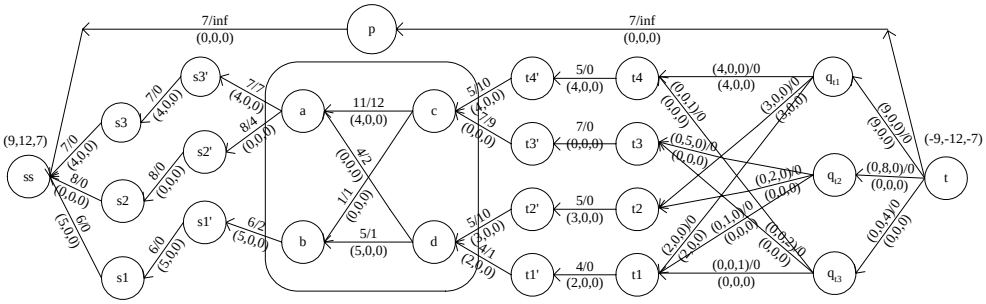

Fig. 8. The results for the example after type-1 requests are processed in Step 5 in the TPF algorithm. (The vector below each link indicates the result of flow assignment.)

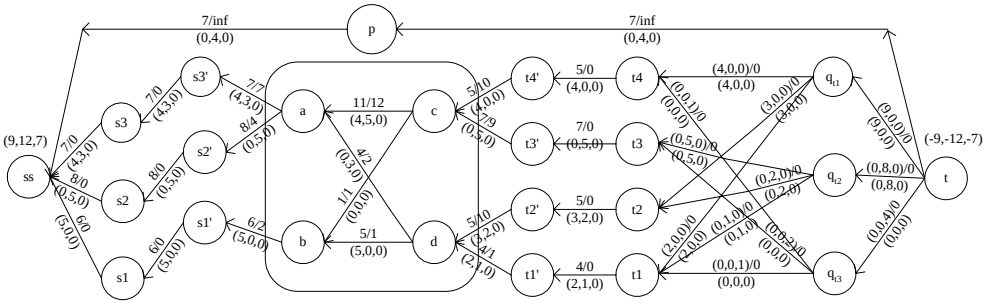

Fig. 9. The results for the example after type-1/type-2 requests are processed in Step 5 in the TPF algorithm.

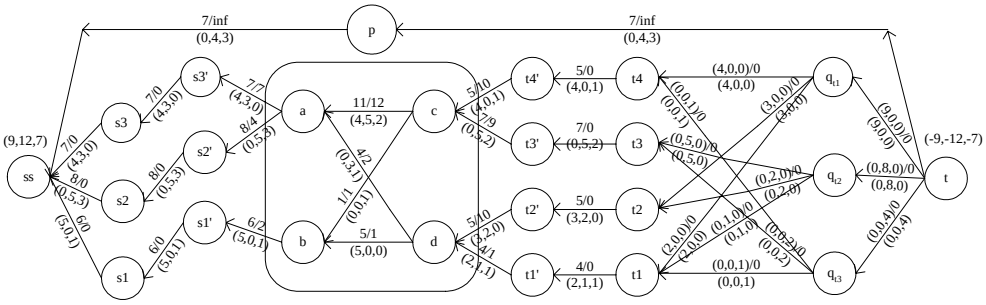

Fig. 10. The results for the example after all requests are processed in Step 5 in the TPF algorithm. (The vector below each link indicates the result of flow assignment.)

server selects back-end node *i* and front-end node *j* to handle a request for program *k*. Moreover, the path between the pair of server nodes (*i*, *j*) for the request can be determined by referring to the path of flow *f* in the flow network.

Given a path selection problem with all of the related system parameters shown in Table 2, applying the above two-pass flow algorithm yields an optimal path selection scheme that maximizes the number of admitted requests at minimal cost. The flow as-

signments $\{F^k \mid k = 1, \ldots, T\}$ derived in Step 5 of the TPF algorithm indicate the amount of resource units being consumed on individual network links and server nodes in serving the admitted requests. A flow assignment $f = (f_1, f_2, \ldots, f_T)$ to the link (s_i, s_i') implies that the dispatcher should forward individually f_k program-k requests to the front-end node i, $k = 1, \ldots, T$. A flow assignment $f = (f_1, f_2, \ldots, f_T)$ to the link (t_i', t_i) implies that back-end server node i will be assigned to serve f_k program-k requests. The flow assignment $f = (f_1, f_2, \ldots, f_T)$ to a link (i, j) that corresponds to the network link connecting nodes i and j in the server implies that the server will direct f_k program-k streams over the network link. Restated, the server will direct a total of $f_1 + f_2 + \ldots + f_T$ programs over the network link.

For example, Fig. 10 shows an optimal path selection for the example media server shown in Fig. 3. The results imply that the server can serve up to 21 requests (nine program-1, eight program-2, and four program-3 requests) at a minimal cost of 406 (by summing all the flow costs contributed by all the links). The flow assignment (4, 3, 0) to the link (s_3, s_3') implies that among all the pending requests (nine program-1, 12 program-2 and seven program-2 requests), the dispatcher should forward four program-1 requests and three program-2 requests to front-end server node 3. The flow assignment (4, 0, 0) to the link (t_4', t_4) implies that back-end server node 4 will be assigned to serve four program-1 requests. The flow assignment (4, 0, 1) to the link (c, t_4') implies that the server will direct five program streams (four program-1 and one program-3 streams) over the network link that connects intermediate node c to back-end server node 4.

4.2 Complexity and Correctness

The above TPF algorithm requires, at worst, $T + 1$ passes of network flow computation, where T is the number of service types supported by the server. One pass is associated with the first flow-pass (forward pass) in Step 3; T passes are associated with the second flow-pass (backward pass) in Step 5, each one yielding an optimal assignment for one type of flow. In practice, only $k + 1$ passes are required if only k different types of requests are made during the scheduling period. The minimal cost flow can be efficiently computed in polynomial time [14, 22]. Consequently, the proposed TPF algorithm can be implemented efficiently.

The following paragraphs briefly discuss correctness of the TPF algorithm. First, the problem transformation from the path selection problem to the flow network problem is correct. Similar techniques can be found in [13]. The reconstructed flow-network G_2 in Step 2 can be easily shown to be equivalent to the flow network G_1 built in Step 1. That is, any feasible flow assignment to G_1 is also feasible to G_2, and vice versa. The link (t_i, t) in G_1 is replaced with T links (t_i, q_{tk}), $k = 1, \ldots, T$ in G_2. Each link (t_i, q_{tk}) is associated with a capacity constraint $(0, \ldots, 0, B_{ik}, 0, \ldots, 0)$, where B_{ik} is the available capacity for type-k flow on the link (t_i, t). From the perspective of node t_i, the capacity constraints are equivalent. Meanwhile, in G_2, all of the flow from node t_i will be forwarded to node t through a set of T nodes q_{tk}, $k = 1, \ldots, T$. Node q_{tk} added to G_2 in Step 2 plays the role of a concentrator of type-k flow to sink all possible type-k flow that emerges from all the nodes t_i, $i = 1 \ldots N_B$. Note that in G_1, the amount of type-k flows available on source node ss is Q_k, and in G_2, the constraint of capacity vector $(0, \ldots, 0, Q_k, 0, \ldots, 0)$ imposed on the link (q_{tk}, t) allows at most the same amount (Q_k) of type-k flow to pass through the

link. Consequently, the constraints are equivalent between G_1 and G_2. Then, in Step 3, the flow type constraints are relaxed and, consequently, the available flow can be pushed as much as possible from source node *ss* to terminal node *t* via all possible routes. Apparently, the cost of using such a strategy yields a low bound of the minimal cost for the same flow network with type-differentiation. Accordingly, if a feasible flow assignment can be found under type-specific constraints, and if it yields the same cost for the same flow network with relaxation of the constraints associated with individual types, then the typed flow assignment is optimal. The flow network G_3 constructed in Step 4 is a special kind of network, called a capacity-conservative flow network [4], in which for each node except the source node and the terminal node, the total capacity of all the links that emerge from the node is the same as the total capacity of all the links that sink to the node. Our previous work [4] showed that if a flow network is capacity-conservative, then all the link capacity can be fully utilized. Therefore, the TPF algorithm up to Step 6 yields a multiple type flow assignment $\{F^k \mid k = 1, \ldots, T\}$, in which to each link on the flow network, the number of units of flow assigned is equal to the capacity available on the link. That is, the assignment will lead to the same cost as in Step 3 (the first pass). Consequently, the flow assignment $\{F^k \mid k = 1, \ldots, T\}$ in Step 6 is optimal for the flow network G_1 built in Step 1.

5. CONCLUSIONS

This study has investigated path selection problems in a media server with a two-tier architecture, in which a VC-enabled network interconnects all front-end nodes and back-end nodes. System resources must be effectively allocated to enable as many requests as possible to be served. The process-power limitations on the server nodes and bandwidth constraints on the network links have been simultaneously considered. With proper problem transformation, the path selection problem can be transformed into a special form of a minimal-cost multi-commodity integer flow problem, and an efficient algorithm has been proposed to solve the problem. The proposed TPF algorithm involves, at worst, $T + 1$ passes of minimal cost flow algorithms. In practice, only $k + 1$ passes are required if only k different types of requests are made during the scheduling period. The proposed path selection scheme is intended to be used with a large media server, in which two-tier server architecture is utilized. In particular, the underlying interconnection network of the server is VC-enabled and subject to change over time. No topological constraint is imposed on the interconnection network.

Many problems have characteristics similar to those of (integer) network flow problems. A general multi-commodity integral flow problem is hard to solve. However, some degenerated forms of the problems can be simple but still sufficiently powerful to model real-world problems. The path selection problem is an example. The corresponding network flow problem solved here involved multiple flow types. We believe the solution techniques presented in this paper are applicable to other problems that can be modeled with mathematical forms similar to those discussed herein.

REFERENCES

1. L. Golubchik, R. R. Muntz, C. F. Chou, and S. Berson, "Design of fault-tolerant large-scale VOD servers: with emphasis on high-performance and low-cost," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, 2001, pp. 363-386.
2. D. M. Dias, W. Kish, R. Mukherjee, and R. Tewai, "A scalable and highly available web server," in *Proceeding of 41st IEEE International Computer Society Conference* (*COMPCOM*), 1996, pp. 85-92.
3. S. Baqai, M. Woo, and A. Ghafoor, "Network resource management for enterprise-wide multimedia services," *IEEE Communications Magazine*, 1996, pp. 78-85.
4. C. C. Yeh, "Design of a multiprocessor data server", Ph.D. dissertation, Department of Computer Since and Information Engineering, National Taiwan University, Taiwan, 1998.
5. P. Y. Chang, D. J. Chen, and K. M. Kavi, "Multimedia file allocation on VC networks using multipath routing," *IEEE Transactions on Computers*, Vol. 49, 2000, pp. 971-977.
6. P. Berenbrink and A. Brinkmann, "Distributed path selection for storage networks," in *Proceedings of International Conference on Parallel and Distributed Processing Techniques*, 2000, pp. 1097-1105.
7. D. Andresen et al., "SWEB: towards a scalable world wide web server on multicomputers," in *Proccedings of the 10th International Parallel Processing Symposium*, 1996, pp. 850-856.
8. B. Narendran, S. Rangarajan, and S. Yajnik, "Data distribution algorithms for load balanced fault-tolerant web access," in *Proceedings of the 16th Symposium on IEEE Reliable Distributed Systems*, 1997, pp. 97-106.
9. H. J. Siegel, *Interconnection Network for Large-Scale Parallel Processing*, McGraw-Hill Series in Computer Organization and Architecture, McGraw-Hill, New York, 1990.
10. B. D. Rathi, A. R. Tripathi, and G. J. Lipovski, "Hardwired resource allocators for reconfigurable architectures," in *Proceedings of International Conference on Parallel Processing*, 1980, pp. 109-117.
11. F. Fung and H. Torng, "On the analysis of memory conflicts and bus contentions in a multiple-microprocessor system," *IEEE Transactions on Computers*, Vol. 28, 1979, pp. 28-37.
12. M. A. Marsan and M. Gerla, "Markov models for multiple bus multiprocessor systems," *IEEE Transactions on Computers*, Vol. 31, 1982, pp. 239-248.
13. J. Y. Juang and B. W. Wah, "Resource sharing interconnection networks in multiprocessors," *IEEE Transactions on Computers*, Vol. 38, 1989, pp. 115-128.
14. S. Even, *Graph Algorithms*, Computer Science Press, 1979.
15. E. S. Elmallah and J. C. Culberson, "Multicommodity flows in simple multistage networks," *Networks*, Vol. 25, 1995, pp. 19-30.
16. T. P. J. To and B. Hamidzadeh, "Run-time optimization of heterogeneous media access in a multimedia server," *IEEE Transactions on Multimedia*, Vol. 2, 2000, pp. 49-61.
17. D. Makaroff, G. Neufeld, and N. Hutchinson, "Design and implementation of a VBR continuous media file server," *IEEE Transactions on Software Engineering*, Vol. 27,

2001, pp. 13-28.
18. M. Y. Wu and W. Shu, "Efficient support for interactive browsing operations in clustered CBR video servers," *IEEE Transaction on Multimedia*, Vol. 4, 2002. pp. 48-58.
19. L. Golubchik, V. S. Subrahmanian, S. Marcus, and J. Biskup, "Sync classes: a framework for optimal scheduling of requests in multimedia storage servers," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, 2000, pp. 60-77.
20. Y. C. Tay and H. H. Pang, "Load sharing in distributed multimedia-on-demand systems," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, 2000, pp. 410-428.
21. R. Tewari, D. M. Dias, R. Mukherjee, and H. M. Vin, "High-availability in clustered multimedia servers," in *Proceeding of IEEE International Conference on Data Engineering*, 1996, pp. 645-654.
22. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of ACM*, Vol. 19, 1972, pp. 248-264.
23. J. Lauderdale and D. H. K. Tsang, "A new techniques for transmission of pre-encoded MPEG VBR video using CBR service," in *Proceeding of International Conference on Communication*, 1996, pp. 1416-1420.
24. Z. L. Zhang, J. Kurose, J. Salehi, and D. Towsley, "Smoothing, statistical multiplexing, and call admission control for stored video," *IEEE Journal on Selected Areas in Communications*, Vol. 15, 1997, pp. 1148-1166.
25. Y. Dinitz, N. Garg, and M. X. Goemans, "On the single-source unsplittable flow problem," in *Proceeding of IEEE Symposium on Foundations of Computer Science*, 1998, pp. 290-299.

Chun-Chao Yeh (葉春超) received his Ph.D. and M.S. degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1998 and 1991, respectively, and his B.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 1989. While he was working toward his Ph.D. degree at National Taiwan University, he got a DAAD/NSC scholarship to be a visiting researcher in GMD, Germany, from July 1996 to April 1997. He has been an Assistant Processor of Computer Science Department, National Taiwan Ocean University, Keelung, Taiwan, since 2000. His research interests include computer networks, real-time and embedded systems, and web servers.

Jie-Yong Juang (莊志洋) received the B.S. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 1976, the M.S. degree in Computer Science from University of Nebraska, Lincoln, in 1981, and the Ph.D. degree in Electrical Engineering from Purdue University, West Lafayette, IN, in 1985. He is a Professor of the Department of Computer Science and Information Engineering of the Taichung Healthcare and Management University, Taichung, Taiwan. From February 1991 to July 2000, he was a Professor of the Department of Computer Science and Information Engineering of National Taiwan University, Taipei, Taiwan. He was with the Department of Electrical Engineering and Computer Science of Northwestern University, Evanston, IL, from January 1985 to January 1991. His areas of research include Multimedia and Communication, Embedded OS, and Knowledgement.