RESOURCE SHARING INTERCONNECTION NETWORKS IN MULTIPROCESSORS

Jie-Yong Juang and Benjamin W. Wah[*]

ABSTRACT

In this paper circuit-switched interconnection networks for resource sharing in multiprocessors, named *resource sharing interconnection networks*, are studied. Resource scheduling in systems with such an interconnection network entails the efficient search of a mapping from requesting processors to free resources such that circuit blockages in the network are minimized and resources are maximally used. The optimal mapping is obtained by transforming the scheduling problems into various network-flow problems for which existing algorithms can be applied. A distributed architecture to realize a maximum-flow algorithm using token propagations is also described. The proposed method is applicable to any general network configuration modeled as a digraph in which the requesting processors and free resources can be partitioned into two disjoint subsets.

1. INTRODUCTION

In this paper we investigate the problem on the sharing of computing resources in multiprocessors and the distributed scheduling of shared resources in a circuit-switched interconnection network.

A *resource* is a processing element to carry out a designated function. Examples include a general purpose processor, a special functional unit, a VLSI systolic array, an input/output device, and a communication channel. A resource is accessible by any processor via an interconnection network. A *request* generated by a processor can be directed to any one of a pool of free resources that are capable of executing the designated task. An interconnection network is an essential element of these systems as it interconnects processors and resources. Its function is to route requests initiated from one point to another point connected on the network. The network topology is dynamic, and the links can be reconfigured by setting the network's active switching elements. The notable characteristic of these networks is that they operate with address mapping. That is, a request is initiated with a specific destination or a set of destinations, and routing is done by examining the address bits. Routing of requests is usually done in parallel. As classified by Feng [8], these networks include the single or multistage networks and the crossbar switch. Examples are the banyan, indirect binary n-cube, cube, perfect shuffle, flip, Omega, data manipulator, augmented data manipulator, delta, baseline, Benes, and Clos. Examples of systems designed with interconnection networks are Trac, Staran, C.mmp, Illiac IV, Pluribus, Numerical Aerodynamic Simulation Facility (NASF), the Ballistic Missile Defense testbed, MPP, and Connection Machine. The performance of resource sharing systems under address mapping has been studied by Rathi, Tripathi and Lipovski [21], Fung and Torng [10], and Marsan, et al [17].

Wah proposed a network with distributed scheduling intelligence, called *resource sharing interconnection network* (*RSIN*) [23, 22]. Instead of using an address-mapping scheme, which requires a centralized scheduler to seek and give the address of a free resource to a request before it enters the network, the request is sent into the network without any destination tags. It is the responsibility of the network to route the maximum number of requests to the free resources. In this way the scheduling intelligence is distributed in the network. Distributed resource scheduling avoids the bottleneck of a centralized scheduler [21]. The objective of a good scheduling scheme is to avoid network blockages and to maximize resource utilization, which requires an efficient algorithm at each switching node to collect the minimum amount of status information.

The PUMPS architecture (see Figure 1) for image analysis and pictorial database management [3] is a typical example of resource sharing multiprocessors in which VLSI systolic arrays, each realizing an image processing function, are organized into a pool of resources. Most dataflow architectures can also be considered as resource sharing systems. The processing units are the pool of resources, and a RSIN connects them to memory cells. In a resource sharing system with load balancing, processors are considered as resources, thus requests generated are queued at the processors as well as the resources. An imbalance of workload at the processors will be balanced by a load balancing scheme.

The design of a RSIN with optimal resource scheduling is studied in this paper. The results are derived with respect to multistage interconnection networks, called *multistage resource sharing interconnection networks* (*MRSIN*), and are applicable to any general network configuration modeled as a digraph in which the requesting processors and free resources are partitioned into two disjoint subsets. Central to the design of such an

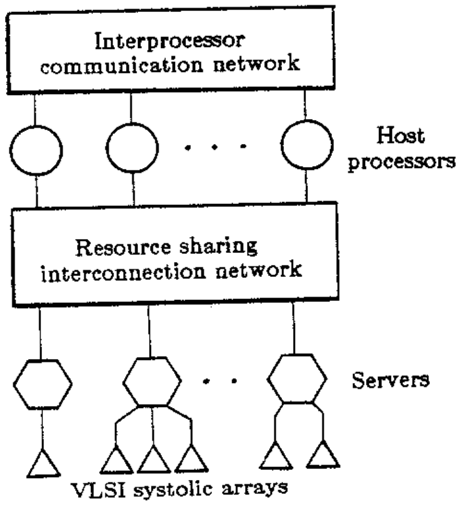

Interprocessor communication network
Host processors
Resource sharing interconnection network
Servers
VLSI systolic arrays

Figure 1. PUMPS: An example of a multiprocessor with shared systolic arrays.

* This research was supported by National Science Foundation Grant DMC 85-19649.

J. Y. Juang is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60201. B. W. Wah is with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

interconnection network is the development of an efficient distributed algorithm to disseminate status information through the complex interconnection structure. The algorithm to be presented is simple, efficient, and independent of the interconnection topology.

2. RESOURCE SHARING INTERCONNECTION NETWORKS

The assumptions of the RSIN used in this study are summaried as follows.

(a) Circuit switching is assumed rather than packet switching for the following reasons. First, packet switching is used in conventional networks with address mapping because it allows a network path to be shared by more than one request concurrently. Reducing the packet delay by balancing loads among alternate paths is less critical in a RSIN because a request can always search for another available resource if a path is blocked. Moreover, the overhead of rerouting a packet is higher than that of rerouting a resource request [16]. Second, owing to the resource characteristics, a task cannot be processed until it is completely received. The extra delay in breaking a task into multiple packets may decrease the utilization of resources, and hence increase the response time of the system.

(b) One or more types of resources may exist in the system. A RSIN connecting only one type of resources is called a *homogeneous* RSIN, while a RSIN connecting multiple types of resources is a *heterogeneous* one.

(c) A priority level may be associated with a request to show the urgency of the request. A preference value may be associated with a resource to show the desirability of being used for service. The costs of allocation are inversely related to the priorities and preferences.

(d) Each request needs one resource only.

(e) A processor can transmit one task at a time to the resources. Other tasks arriving during the task transmission time are queued. The circuit between a processor and a resource can be released once the request has been transmitted. The processor can continue to make other requests, while the resource will be busy until the task is completed.

We have not investigated the problem on the selection of the number of resources in each type and their placements in the output ports. This problem has been studied by Briggs et al., who have considered the problem of choosing the number of resources in each type in which one resource is connected to each output port and one resource is requested each time [2]. We have not considered the case in which more than one resource or multiple types of resources are requested by one request. Here, the scheduling algorithm is dependent on the number of resources in each type, the way that resources are distributed to the output ports, and the network characteristics. Further, deadlocks may occur, and distributed resolution of deadlock may have a high overhead.

The goal of the scheduling algorithm is to find a request-resource mapping such that the total cost is minimized. In the special case in which all requests are of equal priorities and all resources have equal preferences, the scheduling problem becomes the mapping of the maximum number of requests to the free resources.

The maximal request-resource mapping may be hampered by blockages in the system. In a conventional address-mapped interconnection network, blockages may be caused by conflicts in either the same resource being requested by more than one request or a network link requested by two circuits. In a RSIN, a resource conflict can be resolved by rerouting all but one request to other free resources. However, this may not always lead to better resource utilization because the allocation of one request to a resource may block one or more other requests from accessing free resources. A scheduling algorithm that schedules requests according to the state of the network and resources is, therefore, essential. As an example, consider an 8-by-8 Omega network** in Figure 2a with switch-boxes that can be individually set to either a straight or an exchange connection. Processors p_1, p_3, p_5, p_7 and p_8 are requesting one resource each, and resources r_1, r_3, r_5, r_7 and r_8 are available. The circuits between p_2 and r_6 and p_4 and r_4 have been established previously. p_6 is not making request, and r_2 is busy. All free resources will be allocated if one of the following request-resource mappings is used: $\{(p_1, r_3), (p_3, r_5), (p_5, r_7), (p_7, r_1), (p_8, r_8)\}$ or $\{(p_1, r_3), (p_3, r_8), (p_5, r_7), (p_7, r_1), (p_8, r_5)\}$. But if the mapping $\{(p_1, r_1), (p_3, r_5), (p_5, r_3), (p_7, r_7), (p_8, r_8)\}$ is used, a maximum of four out of five resources can be allocated, since the path from p_8 to r_8 is blocked. Simulation results showed that the average blocking probability can be as low as 2% for a MRSIN embedded in an 8-by-8 cube network [22, 12]. If a heuristic routing algorithm is used, the average blocking probability increases to 20%. Further degradation occurs if the network is not completely free.

3. OPTIMAL RESOURCE SCHEDULING IN MRSIN

To bind a request to a resource in the system, a RSIN determines a mapping from pending requests to free resources, and provides connections to as many request-resource pairs as possible. In this section, methods to optimize request-resource mappings are discussed. Exhaustive methods that examine all possible ordered mappings have exponential complexity. In a homogeneous MRSIN, suppose x processors are making requests, y

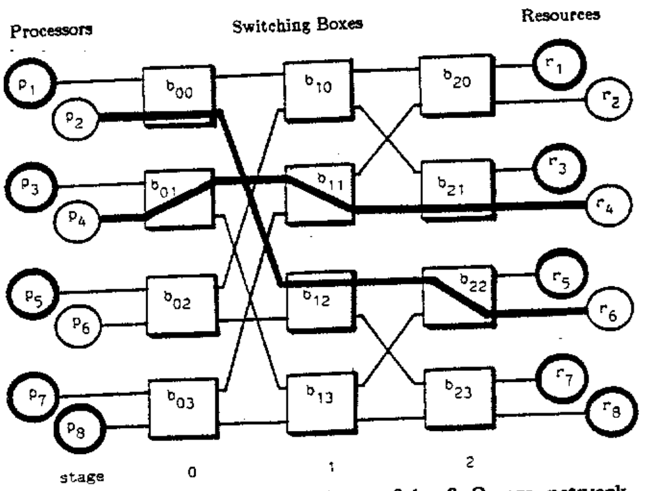


Figure 2a. A MRSIN embedded in an 8-by-8 Omega network (dark paths in the network show circuits that are already occupied; processors p_1, p_3, p_5, p_7 and p_8 are making requests; resources r_1, r_3, r_5, r_7 and r_8 are available).

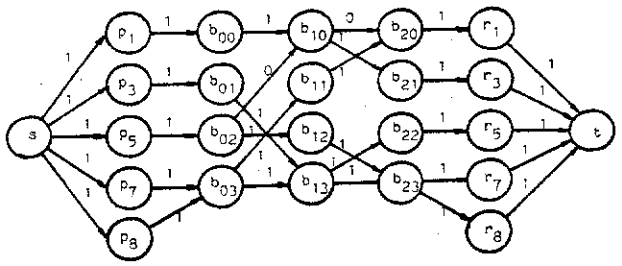


Figure 2b. The flow network transformed from the MRSIN in Figure 2a using Transformation 1 (the number associated with each arc is the amount of flow assigned to it by the maximum-flow algorithm; all arcs have unit capacity).

resources are available, and the network is completely free. The scheduler has to try a maximum of $C_y^x \cdot y!$ (for $x \geqslant y$) or $C_x^y \cdot x!$ (for $y \geqslant x$) mappings to find the best one, where C_i^j is the number of combinations of choosing i objects out of j objects [22, 12]. Suboptimal heuristics can be used but is only practical when x and y are small.

In this section we transform the optimal request-resource mapping problem into various network-flow problems for which many efficient algorithms exists [11, 13]. The basic concepts of flow networks are briefly reviewed first.

3.1. Flow Networks

A flow network is usually represented by a digraph in which each arc is associated with a capacity and possibly a cost. Let $D = (V, E)$ be a digraph with two distinct nodes s (*source*) and t (*sink*). A capacity function, $c(e)$, is defined on every arc of the graph, where $c(e)$ is a non-negative real number for all $e \in E$. A flow function f assigns a real number $f(e)$ to arc e such that the following conditions hold.

(1) *Capacity limitation*:

$$0 \leqslant f(e) \leqslant c(e) \quad \text{for every arc } e \in E \tag{1}$$

(2) *Flow conservation*: Let $\alpha(v)$ (resp. $\beta(v)$) be the set of incoming (resp. outgoing) arcs of vertex v. For every $v \in V$,

$$\sum_{e \in \alpha(v)} f(e) - \sum_{e \in \beta(v)} f(e) = \begin{cases} -F & v = s \\ F & v = t \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The capacity constraint restricts the amount of flow that can be assigned to a link, and flow conservation implies that an intermediate node in the network does not absorb or create flows.

A *legal flow* is a flow assignment that satisfies the capacity and flow-conservation constraints. In a network flow problem, it is necessary to find the legal flow that optimizes a given objective function. For example, in the maximum-flow problem, it is necessary to find the maximum amount of flow that can be advanced from source to sink in a flow network $G(V, E, s, t, c)$ under the capacity and flow-conservation constraints. The problem can be formulated as a linear program.

Maximum-Flow Problem

Maximize F
subject to:
(1) Flow conservation (Eq. (1));
(2) Capacity limitation (Eq. (2)).

Many other examples of network flow problems, including the minimum-cost flow and the trans-shipment problems, can be found in the literature [11].

An *s-t path* is a directed path from s to t. Insertion of a dummy node in a path will increase the path length but will not affect the flow assignment. Increasing the length of s-t paths in this way such that all s-t paths are of equal length is called *s-t path equalization* in this paper. All s-t paths are assumed to be equalized when the network is loop-free.

3.2. Optimal Resource Mapping in Homogeneous MRSIN

A switch-box in a MRSIN is a crossbar switch without broadcast connections. The following theorem shows that the setting of a non-broadcasting switch is equivalent to a legal integral flow assignment in a flow network of unit capacity. Note that an integral flow is a flow assignment in which the amount of flow assigned to each link is of integral value.

Theorem 1***: For any MRSIN, there exists a flow network for which a legal integral flow is equivalent to a valid request-resource mapping.

** The input ports are numbered in a way different from Lawrie's Omega network [15] because all resources are homogeneous, and the permutation of requesting processors will not affect the resource utilization. Broadcast connection is not needed in the switch-boxes since each request needs one resource.

To use existing algorithms to solve a flow problem, a MRSIN has to be transformed into a flow network such that the optimization of request-resource mappings is equivalent to the optimization of the corresponding objective function in the flow network. To this end, additional nodes may be introduced, the capacity of a link may be greater than one, and a cost may be associated with a link.

The following transformation produces a flow network such that the optimal request-resource mapping can be derived from its maximum flow.

Transformation 1: Generate a flow network $G(V, E, s, t, c)$ from a homogeneous MRSIN.

(T1) Create three node-sets **P**, **X** and **R** for processors, switch-boxes and resources, respectively. Introduce two additional nodes: source s and sink t. Let

$$V' = \{s, t\} \cup \mathbf{P} \cup \mathbf{X} \cup \mathbf{R}$$

(T2) Add an arc from the source to every node associated with a processor. Denote this set of arcs by **S**.

$$\mathbf{S} = \{ (s,v) \mid v \in \mathbf{P} \}$$

Add an arc between every node associated with a resource and the sink. This set of arcs is called **T**.

$$\mathbf{T} = \{ (v,t) \mid v \in \mathbf{R} \}$$

For each link in the MRSIN that connects two switch-boxes, or a processor to a switch-box, or a switch-box to a resource, add an arc between the corresponding nodes in the flow graph. Denote this set of arcs by **B**.

$$\mathbf{B} = \{ (v,w) \mid v \in \mathbf{P} \cup \mathbf{X},\ w \in \mathbf{X} \cup \mathbf{R} \}$$

Define $E' = \mathbf{S} \cup \mathbf{T} \cup \mathbf{B}$

(T3) Assign link capacities according to the following function.

$$\underset{e \in \mathbf{B}}{c(e)} = \begin{cases} 0 & \text{associated link is occupied} \\ & \quad \text{or non-existent in the MRSIN} \\ 1 & \text{associated link is free} \end{cases}$$

$$\underset{e \in \mathbf{S}}{c(e)} = \begin{cases} 0 & \text{associated processor does not generate request} \\ 1 & \text{associated processor generates request} \end{cases}$$

$$\underset{e \in \mathbf{T}}{c(e)} = \begin{cases} 0 & \text{associated resource is unavailable} \\ 1 & \text{associated resource is available} \end{cases}$$

(T4) Obtain arc-set E by removing those arcs with zero capacity.

$$E = E' - \{e \mid e \in E',\ c(e) = 0\}$$

Obtain node-set V by deleting those nodes that are not reachable from s.

$$V = V' - \{v \mid \alpha(v) \cup \beta(v) = \emptyset,\ v \in V'\}$$

Applying the above transformation to the MRSIN in Figure 2a results in the flow network in Figure 2b. The following theorem shows that Transformation 1 can be used to find the optimal request-resource mapping.

Theorem 2: In a homogeneous MRSIN, the number of resources allocated by a mapping is equal to the amount of flow that can be advanced from the source to the sink in the flow network obtained by Transformation 1.

From Theorem 2 and a known result that the maximum flow of a network with integral capacity is integral [11], we conclude that the optimal mapping can be derived from the maximum flow in the transformed flow network.

Many algorithms have been developed to obtain the maximum flow in a flow network. The algorithm by Ford and Fulkerson [9] is a primal-dual algorithm in which the flow value is increased by iteratively searching for *flow augmenting paths* until the minimum cut-set of the network is saturated. At this

*** All theorems are stated without proof due to space limitation.

point, no more flow can be advanced since the minimum cut-set is the bottleneck. A flow augmenting path is an s-t path through which additional flow can be advanced from the source to the sink. When arc e on the s-t path points in the direction as the s-t path, additional flow may be advanced through e if the current flow assigned to e is less than c(e). In contrast, if arc e points in the opposite direction, then additional flow may be pushed through the s-t path by canceling its current flow. Advancing flow through an augmenting path in this way will always increase the total amount of flow, and the flow-conservation and capacity limitations will not be violated. For example, in Figure 3, an original flow f is assigned along the path s-a-d-t. Then path s-c-d-a-b-t is a flow augmenting path. Advancing one unit of flow through this augmenting path results in a new flow assignment f'. Two units of flow are pushed through two separate paths s-a-b-t and s-c-d-t according to this assignment.

In the MRSIN, advancing flow through an augmenting path is equivalent to a *resource re-allocation*, i.e., a permutation of the possible request-resource mappings. Consider the MRSIN in Figure 4, which is the counterpart of the flow network in Figure 3.**** The original flow f is equivalent to the request-resource mapping $\{(p_a, r_d), (p_c, r_b)\}$. The allocation of resource r_b to request p_c is blocked according to this mapping. The existence of the flow augmenting path s-c-d-a-b-t shows that this blockage can be removed. Advancing flow through this augmenting path results in a new mapping $\{(p_a, r_b), (p_c, r_d)\}$ and the allocation of both resources. As another example, applying the maximum-flow algorithm to the flow network in Figure 2b, the flow

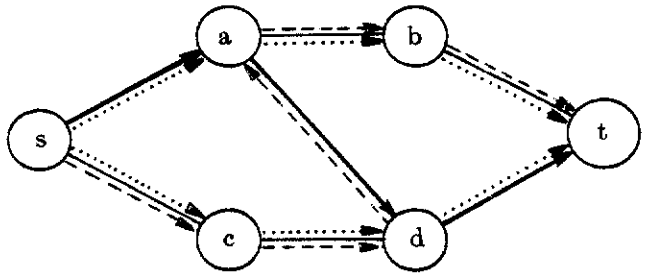


Figure 3. An illustration of advancing flow through a flow augmenting path. (All arcs have unit capacity. The initial flow is assigned to path s-a-d-t. The flow augmenting path s-c-d-a-b-t is indicated as dashed lines. The final flow assignment is obtained after advancing a unit of flow through the flow augmenting path and is indicated as dotted lines.)

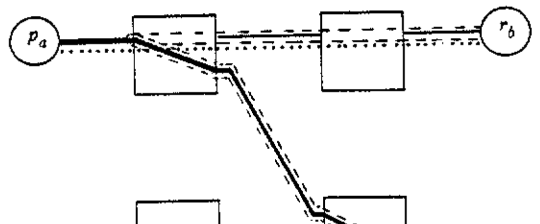


Figure 4. Resource re-allocation corresponding to flow augmentation in Figure 3. (Initially, one resource is allocated—dark lines. The flow augmenting path is indicated as dotted lines. Two resources are allocated after re-allocation—dotted lines.)

**** The switch-boxes are combined with the processors or resources in the flow network in Figure 3, but will not affect the discussion.

assignment as shown in the figure is obtained, and the request-resource mapping $\{(p_1, r_3), (p_3, r_5), (p_5, r_8), (p_7, r_1), (p_8, r_7)\}$ is derived. Note that the optimal mapping may not be unique, but suboptimal mappings are eliminated.

Finding a flow augmenting path from the source to the sink in a flow network is the central idea in most maximum-flow algorithms. The improvement lies in the efficient search of the flow augmenting paths [5, 4]. For example, in Dinic's algorithm, the shortest augmenting path is always advanced first with the aid of an auxiliary layered network, and hence the computational complexity is bounded by $O(|E|^3)$ for general networks. In our case, the links have unit capacity, and the time complexity is reduced to $O(|V|^{2/3} \cdot |E|)$ [11].

3.3. Homogeneous MRSIN with Request Priority and Resource Preference

In a homogeneous MRSIN with request priority and resource preference, each request is associated with a priority level, and each resource is assigned a preference value. Many application-dependent attributes, such as workload, execution speed, utilization and capability, can be encoded into request priorities and resource preferences. The objective of resource scheduling here is to maximize the number of resources allocated, while allowing requests of higher priority to be allocated and resources of higher preference to be chosen. However, it is not necessary for requests and resources to be allocated in order of their priorities and preferences. The allocation of a resource to a request may be blocked by requests of higher priority, and the resource may be allocated to a request of lower priority. A similar argument applies to resources.

With respect to a flow network, the request priority can be considered as the cost of carrying a flow through the path associated with this request. The resource preference can be considered similarly. As a result, the request-resource mapping problem in this class of MRSIN can be transformed into finding a flow assignment to minimize the total cost of flows in the network.

Consider a flow network G(V, E, s, t, c, w) in which w(e), the cost per unit flow, is associated with arc $e \in E$. In the minimum-cost flow problem, a legal s-t flow assignment is sought that allows a given amount of flow F to be circulated from source to sink with the minimum cost. The objective is to determine the set of least expensive s-t paths through which the fixed flow F can be advanced. The constraints in this problem are the same as those in the maximum-flow problem. The problem may be defined in a linear programming formulation.

Minimum-Cost Flow Problem

Minimize $\sum_{e \in E} w(e) f(e)$

subject to:

(1) Flow conservation (Eq. (1));
(2) Capacity limitation (Eq. (2)).

In allocating resources, the objective is to find a corresponding flow network whose optimal flow leads to an optimal request-resource mapping. The main idea behind the transformation is to embed priority and preference information into the objective function by proper cost assignments on links. The amount of flow to be circulated can be considered as the number of requests pending for allocation. However, this amount may exceed the capacity of the flow network or the number of available resources, and additional paths have to be introduced to prevent overflow. A possible transformation is given as follows.

Transformation 2: Generate a flow network G(V, E, s, t, c, w) from a homogeneous MRSIN with request priorities and resource preferences.

(T1) Create node-sets **P**, **X** and **R** for processors, switch-boxes and resources, respectively, and introduce special nodes: source, s, sink, t, and a *bypass node*, u. Let

$$V' = \{s, t, u\} \cup \mathbf{P} \cup \mathbf{X} \cup \mathbf{R}$$

(T2) Create arc-sets S, T and B as in Step (T2) of Transformation 1. Add an arc from the node associated with a processor to the bypass node, and connect the bypass node to the sink. This set of arcs is denoted as L.

$$L = \{(v, u) \mid v \in P\} \cup \{(u, t)\}$$

Define $E' = S \cup T \cup B \cup L$

(T3) Define capacity function c as in Step (T3) of Transformation 1. In addition, define

$$\underset{e \in L}{c(e)} = \begin{cases} 1 & e \neq (u,t) \\ \alpha(u) & e = (u,t) \end{cases}$$

(T4) Define cost function w that represents the cost of advancing one unit of flow through a link as follows.

$$w(e) = \begin{cases} 0 & \text{for } e \in B \\ \max(y_{max}+1, q_{max}+1) & \text{for } e \in L \\ y_{max} - y_p & \text{for } e \in S,\ p \in P \\ q_{max} - q_w & \text{for } e \in T,\ w \in R \end{cases}$$

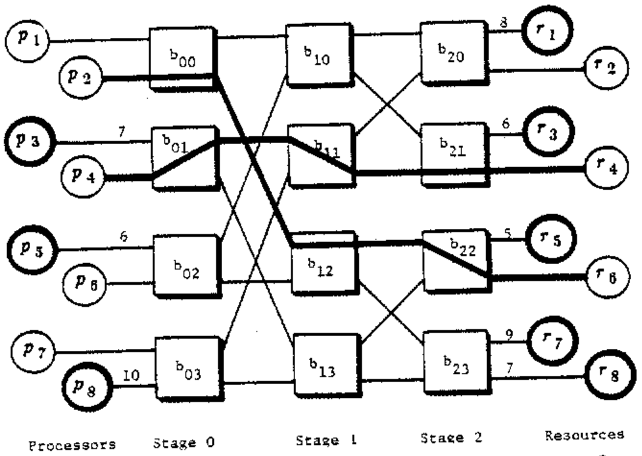


Figure 5a. A MRSIN with request priority and resource preference (highest priority is 10; highest preference is 10; dark paths in the network are already occupied; processors p_3, p_5 and p_8 are making requests; resources r_1, r_3, r_5, r_7 and r_8 are available).

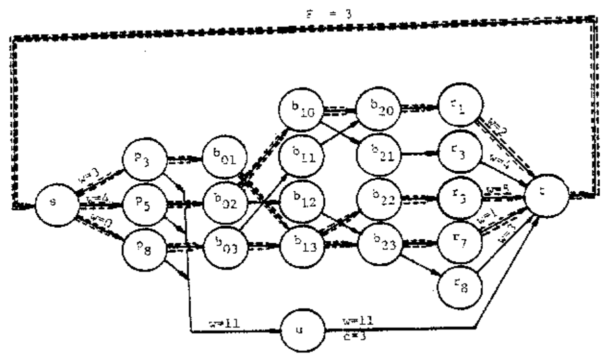


Figure 5b. The flow network transformed from the MRSIN in Figure 5a using Transformation 2 (non-zero flows assigned by the out-of-kilter algorithm are shown as dashed lines in the figure; return flow from t to s is added by the out-of-kilter algorithm; all arcs have unit capacity; cost of arc is zero except where indicated).

where y_{max} is the highest priority level, y_p is the priority of request from processor p, q_{max} is the highest preference level, and q_w is the preference of resource w. Note that any cost function that is inversely related to priorities and preferences can be used.

(T5) Create arc-set E and node-set V as in Step (T4) of Transformation 1.

(T6) Set the total flow F to the number of requests.

As an example, in the MRSIN in Figure 5a, each request is attributed a priority level, and an available resource is given a preference value. The preference and priority levels range from 1 to 10. A minimum-cost flow network obtained from Transformation 2 is shown in Figure 5b. The following theorem shows the correctness of Transformation 2.

Theorem 3: The optimal request-resource mapping on a homogeneous MRSIN with request priority and resource preference can be derived from the minimum-cost integral flow of the flow network, which is obtained by Transformation 2.

Edmonds and Karp have developed a scaled out-of-kilter algorithm to obtain the minimum-cost flow of a general flow network in polynominal time [5]. For a flow network of 0-1 capacity, the time complexity is bounded by $O(|V| \cdot |E|^2)$. Furthermore, in the minimum-cost flow assignment obtained, the flow assigned to a link is integral if the links have integral capacities. As an example, applying the minimum-cost flow algorithm on the flow network in Figure 5b results in the request-resource mapping $\{(p_3, r_5), (p_5, r_1), (p_8, r_7)\}$. The selected paths are shown as dashed lines in Figure 5b. Note that the minimum-cost flow obtained may not be unique, and alternate minimum-cost flows are possible. However, alternative mappings will not improve the cost of allocation.

3.4. Optimal Resource Scheduling in Heterogeneous MRSIN

A heterogeneous MRSIN consists of multiple types of resources, and a processor may generate a request of a given type of resource. Such a MRSIN is equivalent to a flow network carrying different types of commodities. A multicommodity flow network has multiple source-sink pairs, each of which is associated with one type of commodity. A flow coming out of a source of a given commodity can only be absorbed by the sink of the same type of commodity. Flows of different commodities may share a link as long as the total flow does not exceed the capacity of the link.

For a flow network with k types of commodities, there are k source-sink pairs, (s^i, t^i), for i=1 to k. Let F^i be the total flow of the i'th commodity and $f^i(e)$ be the flow of the i'th commodity on edge e. The search for the maximum flow can be formulated as follows [1].

Multicommodity Maximum-Flow Problem

Maximize $\sum_{i=1}^{k} F^i$

subject to:

(1) Flow conservation: For i = 1, ..., k

$$\sum_{e \in \alpha(v)} f^i(e) - \sum_{e \in \beta(v)} f^i(e) = \begin{cases} -F^i & v = s^i \\ F^i & v = t^i \\ 0 & \text{otherwise} \end{cases}$$

(2) Capacity limitation:

$$0 \leq \sum_{i=1}^{k} f^i(e) \leq c(e) \quad \text{for all } e \in E$$

A multicommodity flow network may be visualized as the superposition of k single-commodity flow networks. Each layer in the superposition represents a single-commodity flow. To obtain the optimal request-resource mapping in a heterogeneous MRSIN without priority and preference, a transformation similar to Transformation 1 can be applied to obtain a layer for each type of resources, and the layers are superposed to form a multicommodity flow network.

The optimal mapping for a heterogeneous MRSIN with request priorities and resource preferences can be obtained by transforming the problem into the multicommodity minimum-cost flow problem. Let $w^i(e)$ be the cost per unit flow for the i'th commodity on edge e. The problem can be formulated as follows.

Multicommodity Minimum-Cost Flow Problem

Minimize $\sum_{i=1}^{k} \sum_{e \in E} w^i(e) f^i(e)$

subject to:

(1) Flow conservation: For $i = 1, \ldots, k$

$$\sum_{e \in \alpha(v)} f^i(e) - \sum_{e \in \beta(v)} f^i(e) = \begin{cases} -F^i & v = s^i \\ F^i & v = t^i \\ 0 & \text{otherwise} \end{cases}$$

(2) Capacity limitation:

$$0 \leq \sum_{i=1}^{k} f^i(e) \leq c(e) \quad \text{for all } e \in E$$

The equivalent flow network consists of k source-sink pairs and k bypass nodes, where k is the number of types of requested resources. As for requests without priority, the flow network can be regarded as the superposition of k single-commodity flow networks, and Transformation 2 can be applied to each of them.

The problem of finding the maximum integral flow in a multicommodity flow network of general topology has been shown to be NP-hard. Fortunately, circuit switched loop-free interconnection networks have transformations that belong to a restricted class of multicommodity flow networks in which the optimal flow values are always integral [6]. For this class of flow networks, the integral multicommodity optimal flows can be obtained efficiently by the *Simplex Method*, which has been shown empirically to be a linear-time algorithm [18].

4. ARCHITECTURE OF MRSIN TO SUPPORT OPTIMAL SCHEDULING

Two architectures to carry out the optimal resource scheduling algorithms have been studied. In the first approach, a dedicated monitor is responsible for resource scheduling (see Figure 6). It maintains the status of the interconnection network and resources. The monitor enters a scheduling cycle when there are pending requests. Requests received during a scheduling cycle will not be processed until the next cycle. In a scheduling

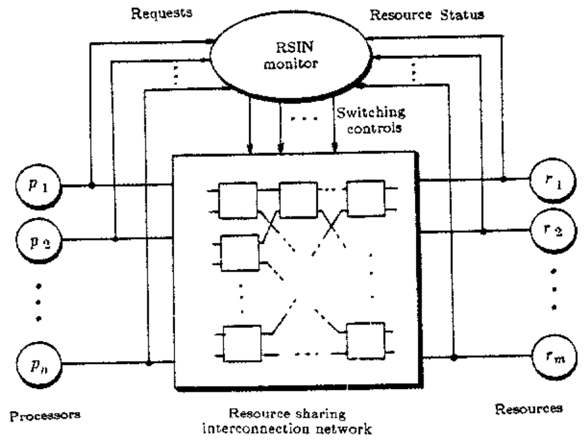


Figure 6. A monitor architecture to carry out optimal resource scheduling in a RSIN.

cycle, a flow network is generated, and the optimal request-resource mapping is derived. Then the monitor sends an acknowledgement to each requesting processor that has been allocated a resource, notifies resources that are allocated, and establishes paths in the network. The implementation is sequential, and the overhead is measured by the number of instructions executed in the algorithm.

A distributed architecture, on the other hand, distributes the scheduling intelligence in the switch-boxes of the interconnection network. Optimal scheduling is achieved through cooperations among processes in the switch-boxes. No transformation to a network-flow problem is necessary because the network-flow algorithm is carried out in a distributed fashion in the switch-boxes. The complexity of the process in each switch-box is central to the design of the distributed architecture. Our previous study shows that the maximum-flow algorithm for homogeneous MRSIN without priority and preference can be efficiently implemented in a distributed fashion [14]. For systems with heterogeneous resources or with priorities and preferences, there is no improvement over a monitor except for reasons such as fault tolerance and modularity.

In the following sections, we describe a distributed realization of Dinic's maximum-flow algorithm to obtain the optimal request-resource mapping.

4.1. Dinic's Maximum-Flow Algorithm

Dinic's algorithm is based on the flow augmentation method described in Section 3.2. It improves over Ford and Fulkerson's algorithm by advancing flow through the shortest augmenting path, which can be found from a *layered network* derived from the original flow graph. A procedure summarizing Dinic's algorithm is as follows.

```
procedure dinic (V, E);
/*  The algorithm has two alternating phases. The algorithm
    alternates between these two phases until no more flow can
    be augmented. */
(1) Initialization: Flow network with an initial flow assignment.
(2) Layered Network Construction Phase:
      (2a) Include s in the first layer;
      (2b) Construct the next layer;
      (2c) If layer is empty, then go to Step (4);
      (2d) If t is not in this layer, then go to Step (2b);
(3) Maximal-Flow Search Phase:
      /* Determine an increment to the flow assignment by finding
         the maximal flow in the layered network. */
      (3a) Search for s-t paths;
      (3b) If an s-t path does not exist, then go to Step (2a);
      (3c) Advance flow through this path; Go to Step (3a);
(4) Stop: Maximum flow assignment has been obtained.
```

In the layered network, nodes of the original flow network are organized into stages. The first stage consists of the source node(s) of the network, and the remaining stages are constructed iteratively. A stage consists of nodes that are not included in the previous stages and have either an unsaturated arc or an arc with non-zero flow originating from nodes in the previous stage. These two types of arcs, called *useful links*, are transformed to arcs in the layered network. Depending on the direction of the associated useful link, its capacity in the layered network can be either the remaining capacity or its current flow. As a result, nodes in a layered network are arranged into disjoint subsets, $V_0, \ldots, V_\ell$, such that no arc points from V_j to V_i ($i \leq j$).

A legal flow in a layered network is said to be maximal if every (s,t)-directed path in the layered network is saturated. Note that a maximal flow in a layered network is not necessarily the maximum flow because flows in opposite directions in the same arc are not considered. Moreover, computing the maximal flow is easier than computing the maximum flow. In Dinic's algorithm, the maximal flow is obtained by a depth-first search.

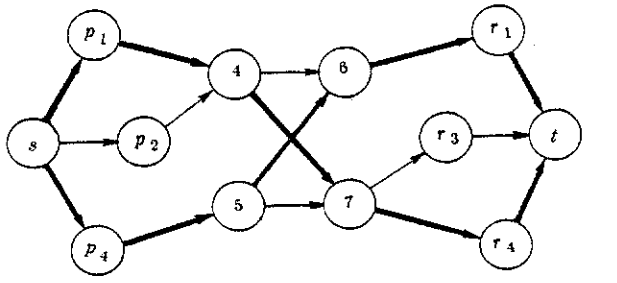
Since the amount of flow that can be advanced through an arc in the layered network is the net increase of flow to the associated arc in the original network, the maximal flow obtained in the layered network is a net increment to the existing flow. Further, the maximum flow of a flow network is finite. Hence, the maximum flow can be obtained in a finite number of iterations in constructing the layered network.

An example illustrating the construction of a layered network is shown in Figure 7. Figure 7a is a flow network associated with a MRSIN in which three processors, p_1, p_2 and p_4, are making requests and three resources, r_1, r_3 and r_4, are available. The flow assignment shown by darkened arcs in Figure 7a results in a mapping such that p_1 is mapped to r_4 and p_4 is mapped to r_1. The request generated by p_2 is blocked. Figure 7b is a layered network constructed from the flow network in Figure 7a. The layered network shows that there is a flow augmenting path from p_2 to r_3. This path includes the arc leading from node 6 to node 5, which is associated with the arc leading from node 5 to node 6 in the original network (see Figure 7a). This flow augmenting path shows that all three resources can be allocated if p_4 is re-allocated to r_3 and p_2 is re-allocated to r_1.

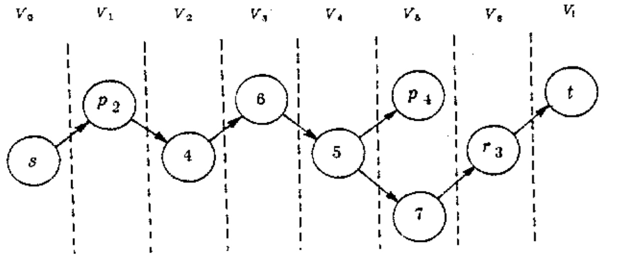
4.2. A Distributed Architecture for Homogeneous MRSIN without priority

A distributed MRSIN embedded in an 8-by-8 Omega network is shown in Figure 8. The scheduling intelligence is distributed in the switch-boxes (NS). A processor is connected to the network through a request server (RQ), and a resource is monitored by a resource server (RS). A common status bus connects these components together. Autonomous process in each component communicates with other processes by passing tokens via direct links, and are synchronized by exchanging status via the status bus.

A scheduling cycle begins when there are pending requests and ready resources. A request generated in the middle of a scheduling cycle has to wait until the next cycle. The procedure indicating the flow of phase transitions in a scheduling cycle is shown as follows.



(a) A flow network (transformed from a 4-by-4 MRSIN) in which flow is advanced through paths s-p_1-4-7-r_4-t and s-p_4-5-6-r_1-t.



(b) The layered network derived from the flow network in (a). The (s,t)-path s-p_2-4-6-5-7-r_3-t is equivalent to an augmenting path in the original flow network.

Figure 7. An illustration of a layered-network construction (all arcs have unit capacity).

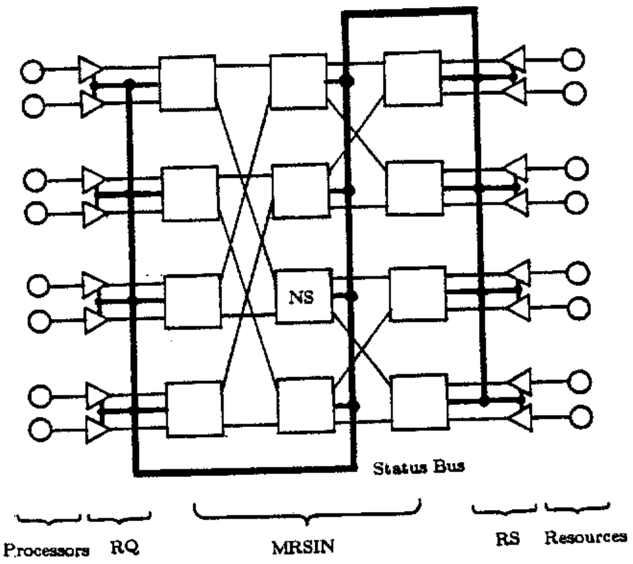


Figure 8. A distributed MRSIN embedded in an 8-by-8 Omega network.

```
procedure mrsin (P, X, R):
/* During a scheduling cycle, the network alternates between
   two phases until all request tokens are blocked. */
(1) Initialization: MRSIN with pending requests and ready
    resources;
(2) Request-Propagation Phase:
    /* In this phase, each requesting RQ sends a request token to
    the network, which will eventually arrive at the resources if
    there are free resources to allocate. */
    (2a) Requesting RQs send tokens;
    (2b) Propagate tokens to next NSs;
    (2c) If all request tokens are blocked, then go to Step (4);
    (2d) If request tokens are not received by the RSs associated
         with free resources, then go to Step (2b);
(3) Resource-Acknowledgment Phase:
    /* The RSs associated with free resources and receiving
    request tokens will send resource tokens to the network to
    acknowledge the acceptance of requests. When a resource
    token is received by a requesting RQ, the RQ and RS will
    form a matched pair, and the path connecting them is
    registered. The network returns to the request-propagation
    phase when all resource tokens are blocked. */
    (3a) Propagate resource tokens to RQs;
    (3b) If all resource tokens are blocked, then go to Step (2a);
    (3c) Register path; Go to Step (3a);
(4) Stop: Optimal request-resource mapping has been obtained.
```

Note that the flow of this procedure is exactly the same as the control flow of Dinic's algorithm.

To carry out Dinic's algorithm by token propagations, a switching element has to follow several token-propagation rules. If a link is free, a switch-box will deliver any request token received from the processor side to the resource side. These directions are reversed if the link is already registered. If there are more than one path to send the request token, the token is duplicated and sent along all paths. A resource token is expected from where a request token was delivered. However, no duplication of resource token is done, and one path is chosen at random. It can be verified that these token-propagation rules correctly implement the layered-network construction process and the search for the maximal flow.

To avoid chaos in token propagations, processes residing in different switch-boxes have to be synchronized on phase transitions. This synchronization can be achieved by broadcasting the status of each process on the status bus. The status of the network can be one of four independent possibilities: request-pending, resource-ready, request-token-propagation, and resource-token-propagation, and can be represented as a Boolean vector (RP, SR, RTP, STP). Each process will synchronize the broadcast of its status on the bus with other processes. The result is a wired-OR of the statuses broadcast by all processes. The occurrence of events that cause a phase transition can be detected by observing the status bus. For example, the transition from a request-propagation phase to a resource-acknowledgement phase is known to every process when each observes a change of the status vector from (1,1,1,0) to either (1,1,1,1) or (1,1,0,1).

We have shown that every process involved can be carried out by a simple sequential machine and is realizable in logic circuits [14]. With this design, there are two factors that contribute to a significant speedup as compared to a monitor architecture: (a) the augmenting paths are searched in parallel, and (b) the time complexity is measured in gate delays instead of instruction execution cycles. As a result, the scheduling algorithm will run at least 100 times faster than a software implementation of the network flow algorithm.

5. CONCLUSIONS

A RSIN is suitable to support resource sharing in multiprocessors. Optimal request-resource mapping in RSIN is obtained by maximizing the number of communication paths that interconnect pairs of processors and resources. In this paper, we have transformed various request-resource mapping problems into network flow problems for which efficient algorithms exist. Table 1 is a summary of the results we have obtained. The proposed method is independent of the interconnection structure and is applicable to all configurations in which the requesting processors and free resources are partitioned into two disjoint subsets. In particular, the method is applicable to networks with multiple paths between source-destination pairs, such as the data manipulator [7], the augmented data manipulator [19], and the Gamma network [20]. The resource utilization, however, will depend on the network configuration, the resources available, the permutation of the various types of resources, and the permutation of the requesting processors.

Scheduling Discipline		*Homogeneous Resources General Topology*		*Heterogeneous Resources*	
		No Priority & Preference	*Priority & Preference*	*Loop-free Topology*	*General Topology*
Optimal Scheduling	Equivalent Flow Problem	Max-Flow	Min-Cost Circulation	Real Multi-Commodity	Integer Multi-Commodity
	Algorithm	Ford-Fulkerson, Dinic	Out-of-Kliter	Linear Programming	NP-hard
Distributed Algorithm		Yes	NA	NA	NA
Implementation		Distributed Architecture	Monitor Architecture		
		Synchronized by Broadcasting	Software		
		Communicate by Token Propagation			

Table 1. Summary of optimal resource scheduling schemes for resource sharing interconnection networks.

REFERENCES

[1] A. A. Assad, "Multicommodity Network Flows—A Survey," *Networks*, vol. 8, pp. 37-91, 1978.

[2] F. A. Briggs, M. Dubios, and K. Hwang, "Throughput Analysis and Configuration Design of a Shared-Resource Multiprocessor System: PUMPS," *Proc. 8th Annual Symposium on Computer Architecture*, pp. 67-79, 1981.

[3] F. A. Briggs, K. S. Fu, K. Hwang, and B. W. Wah, "PUMPS Architecture for Pattern Analysis and Image Database Management," *Trans. on Computers*, vol. C-31, no. 10, pp. 969-983., IEEE, Oct. 1982.

[4] E. A. Dinic, "Algorithm for Solution of a Problem of Maximal Flow in a Network with Power Estimation," *Soviet Math. Dokl.*, vol. 11, pp. 1277-1280, 1970.

[5] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. of the ACM*, vol. 19, no. 2, pp. 248-264, April 1972.

[6] J. R. Evans and J. J. Jarvis, "Network Topology and Integral Multicommodity Flow Problems," *Networks*, vol. 18, pp. 107-119, 1978.

[7] T. Y. Feng, "Data Manipulating Functions in Parallel Processors and Their Implications," *Trans. on Computers*, vol. C-23, no. 3, pp. 309-318, IEEE, March 1974.

[8] T. Y. Feng, "A Survey of Interconnection Networks," *Computer*, pp. 12-27, IEEE, Dec. 1981.

[9] L. R. Ford and D. R. Fulkerson, *Flow in Networks*, Princeton University Press, Princeton, NJ, 1962.

[10] F. Fung and H. Torng, "On the Analysis of Memory Conflicts and Bus Contentions in a Multiple-Microprocessor System," *Trans. on Computers*, vol. C-28, no. 1, pp. 28-37, IEEE, Jan. 1979.

[11] B. Golden, M. Ball, and L. Bodin, "Current and Future Research Directions in Network Optimization," *Computers and Operations Research*, vol. 8, pp. 71-81, 1981.

[12] A. Hicks, *Resource Scheduling on Interconnection Networks*, M.S. Thesis, Purdue University, West Lafayette, IN, Aug. 1982.

[13] J. Y. Juang and B. W. Wah, "Optimal Scheduling Algorithms for Multistage Resource Sharing Interconnection Networks," *Proc. 8'th Int'l Computer Software and Applications Conf.*, pp. 217-225, IEEE, Nov. 1984.

[14] J. Y. Juang, *Resource Allocation in Computer Networks*, Ph.D. Thesis, Purdue University, West Lafayette, IN, Aug. 1985.

[15] D. Lawrie, "Access and Alignment of Data in an Array Processor," *Trans. on Computers*, vol. C-24, no. 12, pp. 215-255, IEEE, Dec. 1975.

[16] M. Lee and C.-L. Wu, "Performance Analysis of Circuit Switching Baseline Interconnection Networks," *Proc. of the 11th Annual Int'l Symp. on Computer Architecture*, pp. 82-90, 1984.

[17] M. A. Marsan and M. Gerla, "Markov Models for Multiple Bus Multiprocessor Systems," *Trans. on Computers*, vol. C-31, no. 3, pp. 239-248, IEEE, March 1982.

[18] E. H. McCall, "Performance Results of the Simplex Algorithm for a Set of Real-Word Linear Programming Models," *Comm. of the ACM*, vol. 25, no. 3, pp. 207-213, March 1982.

[19] R. J. McMillen and H. J. Siegel, "Routing Schemes for the Augmented Data Manipulator Network in an MIMD System," *Trans. on Computers*, vol. C-31, no. 12, pp. 1202-1214, IEEE, Dec. 1982.

[20] D. S. Parker and C. S. Raghavendra, "The Gamma Network: A Multiprocessor Interconnection Network with Redundant Paths," *Proc 9th Annual Symposium on Computer Architecture*, pp. 73-80, 1982.

[21] B. D. Rathi, A. R. Tripathi, and G. J. Lipovski, "Hardwired Resource Allocators for Reconfigurable Architectures," *Proc. Int'l Conference on Parallel Processing*, pp. 109-117, IEEE, Aug. 1980.

[22] B. W. Wah and A. Hicks, "Distributed Scheduling of Resources on Interconnection Networks," *Proc. National Computer Conference*, pp. 697-709, AFIPS Press, 1982.

[23] B. W. Wah, "A Comparative Study of Distributed Resource Sharing on Multiprocessors," *Trans. on Computers*, vol. C-33, no. 8, pp. 700-711, IEEE, Aug. 1984.