Proceedings COMPSAC, November, 1984

OPTIMAL SCHEDULING ALGORITHMS FOR MULTISTAGE RESOURCE SHARING INTERCONNECTION NETWORKS

Jie-Yong Juang and Benjamin W. Wah
School of Electrical Engineering
Purdue University
W. Lafayette, IN 47907

ABSTRACT

In this paper resource scheduling for a circuit-switched multistage interconnection network, named Multistage Resource Sharing Interconnection Network, is studied. The resource scheduling problem entails the efficient search of a mapping from requesting processors to free resources so that circuit blockages in the network are minimized and free resources are maximally utilized. The optimization is achieved by transforming the scheduling problems into various network-flow problems for which existing algorithms can be applied.

KEYWORDS AND PHRASES: Circuit switching, interconnection network, linear programming, maximum flow, minimum-cost flow, multicommodity network flow, resource sharing, scheduling, Simplex method.

1. INTRODUCTION

In resource sharing there is a pool of computing resources to be shared by multiple requesting agents. A *resource* is a processing unit that implements a designated function and may be a general purpose processor, a special functional unit, a VLSI systolic array, an input/output device or a communication channel. A *request* is directed to any one of the free resources that is capable of executing the designated task. A logical representation of the architecture as depicted in Figure 1 consists of a set of processors that generate requests, a set of resources that are capable of executing tasks specified by requests, and an interconnection network that connects the processors and resources. A processor and a resource need not be physically distinct, and a processor may sometimes act as a resource to provide service to others. Examples of resource sharing systems include data-flow machines [TRE82], PUMPS for image analysis and pictorial database management [BRI82] and systems with pools of VLSI systolic arrays [KUN81]. Realization of such a system relies heavily on the interconnection network and the underlying resource scheduling algorithm.

Static allocation is the simplest way to manage multiple functionally identical resources. In this case each resource is assigned to a given set of processors. Since resource sharing is only achieved within the given set of processors, there may be unbalanced loads resulting in low utilization and poor response time [CHO70].

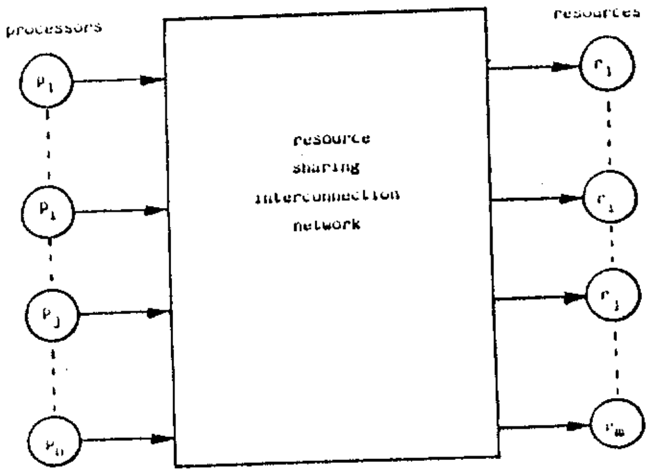


Figure 1. A logical model of a multiprocessor system with resource sharing (p_i - i'th host processor for generating requests; r_j - j'th resource for executing a designated task).

The resource sharing capability of static allocation can be enhanced by *load balancing* [HWA81] that engages communication facilities to support remote job execution. A request can migrate from a heavily loaded resource to a lightly loaded resource if the completion time can be reduced. This is coordinated by cooperating local schedulers that decide if a migration is needed. Migration policies may be classified as probabilistic and deterministic [NI81,CHO70]. Probabilistic strategies dispatch requests randomly by choosing a resource with a fixed probability regardless of the current status of the system. This is incapable of adapting to instantaneous load changes and hence results in poor performance. On the other hand, deterministic strategies dispatch requests according to the load status of resources. The major problems lie in the collection and dissemination of status information and the design of the supporting interconnection network.

The collection of status information by a centralized scheduler is usually impractical due to delays and overheads involved. To avoid these overheads, the status information should be disseminated and used by the

This research was partially supported by National Science Foundation Grant ECS80-16580 and by CIDMAC, a research unit of Purdue University, sponsored by Purdue, Cincinnati Milicron Corporation, Control Data Corporation, Cummins Engine Company, Ransburg Corporation and TRW.

scheduler in a distributed fashion. An efficient scheme for distributing status information in load balancing has been studied with respect to CSMA/CD networks [WAH83]. Another approach is to allow requests to be sent without any destination tags, and it is the responsibility of the network to route the maximum number of requests to the free resources. The network is also responsible for distributing the status information of resources to the processors. In this way the scheduling intelligence is distributed in the network. This network is called a *Resource Sharing Interconnection Network* (RSIN) [WAH84].

One important problem in RSINs is the efficient scheduling of resources to avoid network blockage and to maximize resource utilization. In this paper we study optimal resource scheduling algorithms on a *Multistage Resource Sharing Interconnection Network* (MRSIN). In the next section the concepts and assumptions of MRSINs are reviewed briefly, and the issues of scheduling are discussed. In Section 3 we develop a methodology to optimize resource scheduling on a homogeneous MRSIN and discuss transformations leading to simple yet efficient optimal scheduling algorithms. In Section 4 resource scheduling on heterogeneous MRSINs are discussed.

2. MULTISTAGE RESOURCE SHARING INTERCONNECTION NETWORKS

A MRSIN has a similar interconnection structure but a different routing strategy as conventional interconnection networks with address mapping [FEN81]. In a conventional interconnection network the resource address has to be determined before a request enters the network [SIE81]. In a MRSIN a request is directed to any free resource in the pool and does not require the destination address of the resource before entering the network.

Blockage in conventional interconnection networks may be due to blockage in the destination or blockage in the network links. In a MRSIN blockage in a resource can be avoided by searching for an alternate free resource. 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 due to blocked paths. A scheduling algorithm is, therefore, essential. For example, consider an 8-by-8 Omega network [LAW75] (Figure 2a) with interchange boxes that can be set to one of the two possible states: straight or exchange (broadcast connection is not needed since one resource is needed for each request). In this example assume that processors p_1, p_3, p_5 are requesting one resource each, and resources r_1, r_2, r_3 are available. Other processors are not making requests, and other resources are busy. Further, the network is completely free. All the resources will be allocated if the following processor-resource mappings are used: $\{(p_1,r_1), (p_3,r_2), (p_5,r_3)\}$, $\{(p_1,r_2), (p_3,r_1), (p_5,r_3)\}$, $\{(p_1,r_3), (p_3,r_1), (p_5,r_2)\}$ or $\{(p_1,r_3), (p_3,r_2), (p_5,r_1)\}$. But if the following processor-resource mappings are used: $\{(p_1,r_1), (p_3,r_3), (p_5,r_2)\}$ or $\{(p_1,r_2), (p_3,r_3), (p_5,r_1)\}$, then a maximum of two out of three resources can be allocated without blocking. This illustrates that the scheduler must be designed properly to give the maximum resource utilization. Simulations have shown that the average blocking probability can be as low as 2% for a MRSIN with an 8-by-8 cube network [HIC82,WAH82]. (A cube network is an Omega network with a permutation of the output ports [SIE81].) If a heuristic scheduling algorithm is used, the average blocking probability increases to around 20%. Further degradation occurs if the network is not completely free.

The basic assumptions made in this study are:

(a) Circuit switching is assumed for the MRSINs rather than packet switching for the following reasons. First, packet switching is used in conventional address mapping networks because it allows a network path to be shared by more than one requests concurrently. This reduces the waiting time in accessing a resource. The issue is less critical in MRSINs because a request can always search for another available resource provided that the network is free. Further, the overhead of rerouting a packet when a path or resource is blocked is higher than that of rerouting a resource request. Second, due to the characteristics of resources, 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. A MRSIN is classified by the number of types of resources connected. A MRSIN with only one type of resources attached is called to a *homogeneous* MRSIN, while a MRSIN with multiple types of resources is a *heterogeneous* one.

(c) Request priority may be associated with a request to indicate the urgency of the request. Resource preference may be associated with a resource to indicate the desirability of being used for service.

(d) Each request needs one resource only.

3. OPTIMAL RESOURCE SCHEDULING FOR HOMOGENEOUS MRSINS

In this section methods for optimizing resource mapping are discussed. Exhaustive methods that examines all the possible ordered mappings have exponential complexity. In a homogeneous MRSIN suppose x processors are making requests, y resources are available and the network is completely free. The scheduler has to try a maximum of $\begin{bmatrix} x \\ y \end{bmatrix} y!$ (for $x \geq y$) or $\begin{bmatrix} y \\ x \end{bmatrix} x!$ (for $y \geq x$) mappings in order to find the best one [WAH82,HIC82]. Sub-optimal heuristics can be used but is only practical when x and y are small. This problem is more complex when the network is partially occupied.

In this section we transform the optimal resource mapping problem into various network flow problems for which there exists many efficient algorithms [GOL81]. The basic concepts of flow networks are discussed briefly first.

3.1. Flow Networks

A flow network is usually represented by a digraph with link capacity functions. Let $\mathbf{D} = (\mathbf{V},\mathbf{E})$ be a digraph with distinct vertices s (*source*) and t (*sink*). A capacity function, c(e), is defined on arc e of $\mathbf{D}$ such that c(e) be a non-negative real number for all $e \in \mathbf{E}$. A flow function f is an assignment of a real number $f(e)$ to each arc, e, such that the following conditions hold:

(1) *Capacity limitation* : For every arc $e \in \mathbf{E}$,
$$0 \leq f(e) \leq c(e)$$
This condition ensures that a flow is less than the link capacity.

(2) *Flow conservation*: Let $\alpha(v)$ be the set of incoming arcs of vertex v and $\beta(v)$ be the set of outgoing arcs of vertex v. For every $v \in \mathbf{V} - \{s,t\}$,

$$\sum_{e\in\alpha(v)} f(e) = \sum_{e\in\beta(v)} f(e)$$

This constraint ensures that each intermediate node in the flow network does not absorb or create flows.

A *legal flow* is a flow assignment that satisfies capacity limitation and flow conservation constraints. The *network flow problem* is to find the legal flow that optimizes a given objective function. Examples include the maximum-flow, the minimum-cost flow and the transshipment problems [PAP82]. A *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 so that all s-t paths are of equal length is known as *s-t path equalization.* For convenience, all s-t paths may be equalized or de-equalized in the following discussions.

The maximum-flow problem of a flow network, G(V,E,s,t,c), finds the maximum flow, F, from s to t under the capacity and flow conservation constraints. It can be formulated as a linear programming problem:

Maximum-Flow Problem:
Maximize F
subject to:

$$(1)\quad \sum_{e\in\alpha(v)} f(e) - \sum_{e\in\beta(v)} f(e) = \begin{cases} -F & v=s \\ F & v=t \\ 0 & \text{otherwise} \end{cases}$$

$$(2)\quad 0 \le f(e) \le c(e) \qquad \text{for all } e\in E$$

3.2. Optimal Resource Mapping in Homogeneous MRSINs

A switch-box in a MRSIN is a m-by-n crossbar switch, where $m,n \ge 1$. The setting of the switch is equivalent to a legal integral flow assignment in a flow network of unit capacity. The following theorem assures that a valid resource mapping can be obtained by finding a legal integral flow in a corresponding flow network.

Theorem 1: For any MRSIN, there exists a flow network for which a legal integral flow is equivalent to a valid resource mapping.
Proof: Consider a n-by-m switch-box, where n is the number of input links and m is the number of output links. A physically realizable switch setting is one in which an input link is connected to only one output link and vice versa. This switch-box is equivalent to a node in a flow network with $|\alpha(v)| = n$ and $|\beta(v)| = m$. The switch setting is equivalent to a legal integral flow assignment. Since a link in a switch is either allocated or free while a flow assignment assigns a link to either zero or one unit of flow, the capacity constraints are always satisfied if the capacity of links are set to 1. There is a direct correspondence between a switch-box and a node, and a switch setting and a flow assignment. So an equivalent flow can be constructed by a direct one-to-one mapping from a circuit-switched MRSIN. □

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

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

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

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

T2. Add an arc between the source and every node of an associated processor. This set of arcs is denoted by **S**.

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

Add an arc between every node of an associated 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, 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:

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

T3. Define capacity function c as follows:

$$\underset{e\in B}{c(e)} = \begin{cases} 0 & \text{associated link is occupied} \\ 1 & \text{associated link is free} \end{cases}$$

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

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

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

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

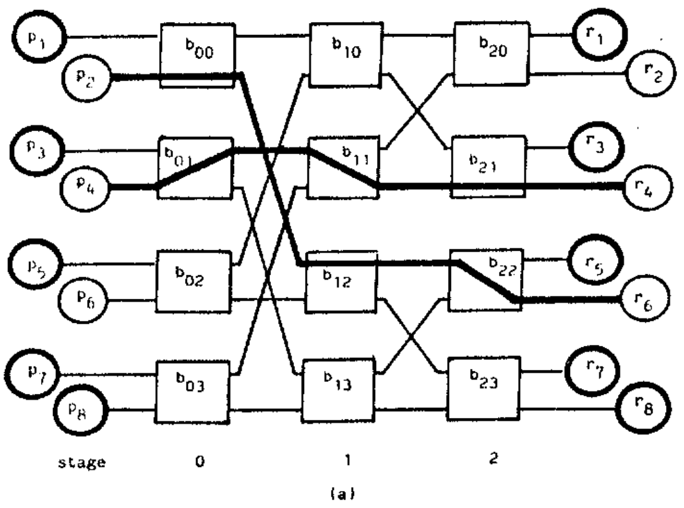
Obtain node-set **V** by deleting nodes that are not reachable from s.

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

An an example, consider a MRSIN embedded in an 8-by-8 Omega network in which two paths are already occupied (Figure 2a). Processors p_1, p_3, p_5, p_7 and p_8 are making requests, and resources r_1, r_3, r_5, r_7 and r_8 are available. By applying Transformation 1, the resulting flow network is shown in Figure 2b.

The correctness of the transformation is proved in the following theorem.

Theorem 2: In a homogeneous MRSIN the number of resources allocated by a mapping is equal to the value of a legal integral flow in the corresponding flow network obtained by Transformation 1.
Proof: Consider a flow network with equalized s-t paths. The integral flow assignment of nodes in stage i is a mapping, g_i, that maps output flows in stage i−1 to input flows in stage i+1. A legal integral flow assignment for the network can be represented by a composite function, $h = g_1g_2...g_L$, where L is the length of the equalized s-t paths. For a flow network of integral capacity as obtained by Transformation 1, it has been proved that a flow assigned to an arc e, $f(e)$, is either 0 or 1 [BON81] and obeys the conservation law so that g_i is one-to-one. It follows that h is also a one-to-one function. For a one-to-one function, the norm of its domain is equal to the



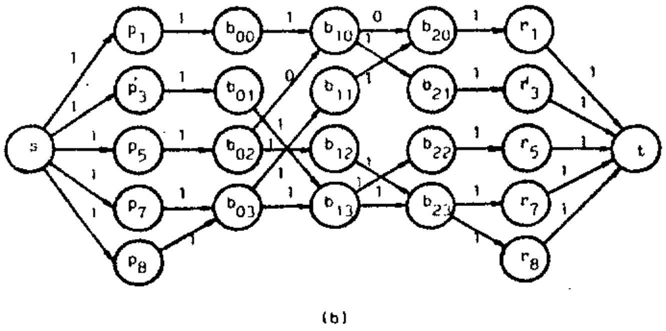


Figure 2

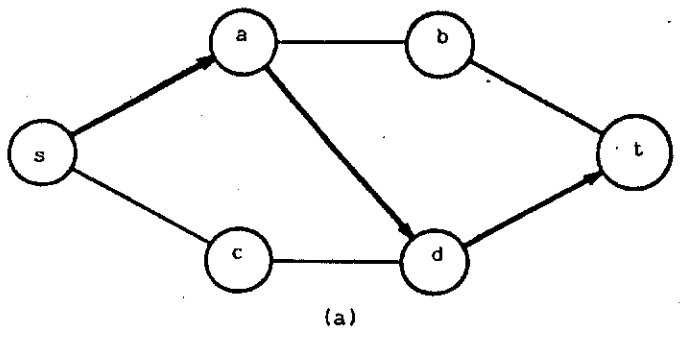
(a) A MRSIN embedded in an 8-by-8 Omega network (dark paths in the network are already occupied; processors p_1, p_3, p_5, p_7, p_8 are making requests; resources r_1, r_3, r_5, r_7, r_8 are available).

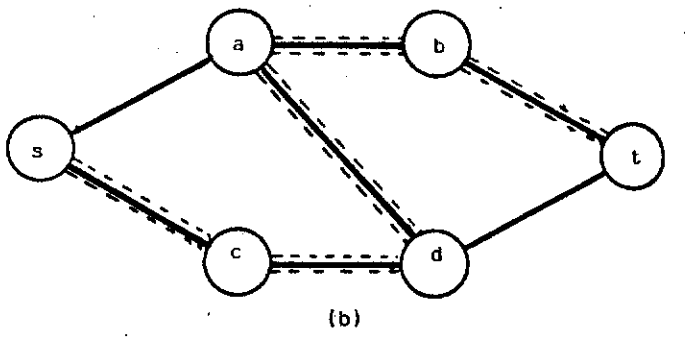
(b) The associated flow network obtained by Transformation 1 (all arcs have capacity 1).

norm of its range. The norms of the composite flow assignment function, h, is equal to the total flow leaving the source and entering the sink. Thus, $|\mathbf{I}| = |\mathbf{O}| = F$, where $\mathbf{I}$ and $\mathbf{O}$ represent the domain and range of g_i and F is the value of the flow. This implies that every legal flow defines a set of F nonoverlapping paths from s to t. In the MRSIN each of these paths joins a pair of requesting processor and free resource so that the number of resources allocated is equal to the value of the corresponding flow in the flow network. □

From Theorem 2 and from a known result that the maximum flow of a network with integral capacity is integral [BON81], we can conclude that an integral maximum flow of the transformed flow network is equivalent to the optimal resource mapping.

The flow assignment can be obtained by applying the maximum-flow algorithm. The first algorithm proposed for solving maximum flow problems is due to Ford and Fulkerson [FOR62]. It is a primal-dual algorithm in which the flow value is increased by iteratively looking for *flow augmenting paths* until a minimum cut-set is saturated. A flow augmenting path is a simple path from source to sink that can be used to increase the flow value





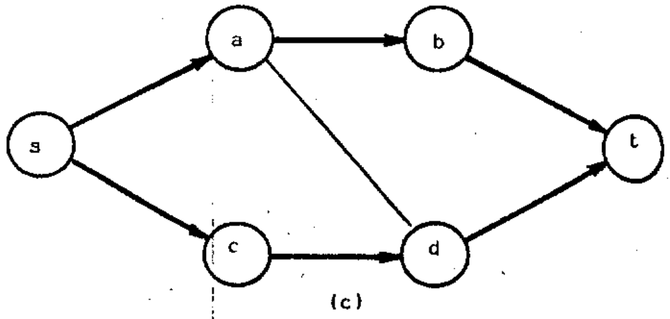


Figure 3

(a) A flow network with an initial flow assigned to path s-a-d-t (all arcs have capacity 1).

(b) An augmenting path s-c-d-a-b-t exists in the network.

(c) Final flow after advancing a unit of flow through the augmenting path.

and does not have to be a directed path. When an arc e on this path points in the direction from s to t, then additional flow may be pushed through e if the current flow assigned to e is less than its capacity, c(e). If arc e points in the opposite direction, then additional flow from s to t may be pushed through it if some of its flow is canceled. Thus, $f(e) > 0$ always holds. For example, in Figure 3a, 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 (Figure 3b). 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 (Figure 3c).

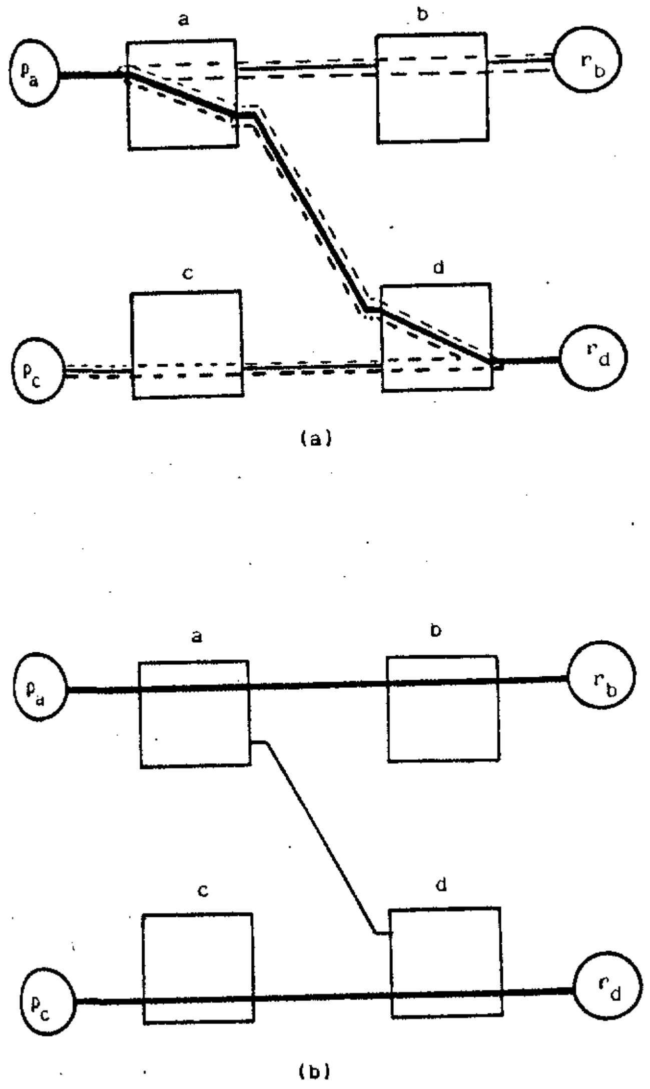


Figure 4

(a) An initial resource allocation.

(b) Two resources are allocated after relocation.

In the MRSIN advancing a flow through an augmenting path is equivalent to a *resource relocation*, i.e., a permutation of resource mapping. Consider a MRSIN (Figure 4a) which is the counterpart of the flow network used in the last example. The original flow f is equivalent to an allocation of resource mapping $\{(p_a, r_d)\}$, while the existence of the flow augmenting path s-c-d-a-b-t implies that there is a blockage due to this mapping. Advancing flow through this path removes the blockage and results in a new allocation $\{(p_a, r_b), (p_c, r_d)\}$ as shown in Figure 4b. As an example, applying the network flow algorithm on the graph in Figure 2b, the following processor-resource mappings are obtained: $\{(p_1, r_3), (p_3, r_5), (p_5, r_8), (p_7, r_1), (p_8, r_7)\}$. There exists mappings that cannot allocate all the free resources but are eliminated by the network-flow algorithm. The flows in the graph are also shown in Figure 2b.

Finding a flow augmenting path from source to sink in a flow network is the central idea in most maximum-flow algorithms. The improvement lies in the efficient search of flow augmenting path [EDM72,DIN70]. 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 a polynominal $O(|E|^3)$ for general networks. For a flow network of unit capacity as in our case, a even better time complexity is expected.

3.3. Implementation of Homogeneous MRSINs

In the implementation of distributed network-flow algorithms on a homogeneous MRSIN, it is desirable that optimal scheduling be achieved at the speed of signal propagations. The scheduling intelligence is distributed in the switching elements of the network. A processor is connected to the network through a request server (RQ), and a resource is monitored by a resource server (RS). A single bus links these components together. The organization is illustrated in Figure 5 by an 8-by-8 Omega network. Autonomous processes in each component communicate with each other by status signals on the bus and tokens via direct links. The signals on the status bus are the logical OR of the status of the processes and are accessible to every one in the system.

The network switching elements are responsible for finding the augmenting paths, while the other components are responsible for synchronization of the scheduling iterations. RQ accepts a request from the processor and broadcasts a request-pending status to the bus. Whenever a resource becomes ready, the associated RS will send a resource token to the network that will eventually arrive at the requesting RQs. A requesting RQ will submit a request token to the network to bid for a resource as soon as it receives a resource token. When a request token is received by a ready RS, the RQ and RS will form a matched pair, and the path connecting them is registered.

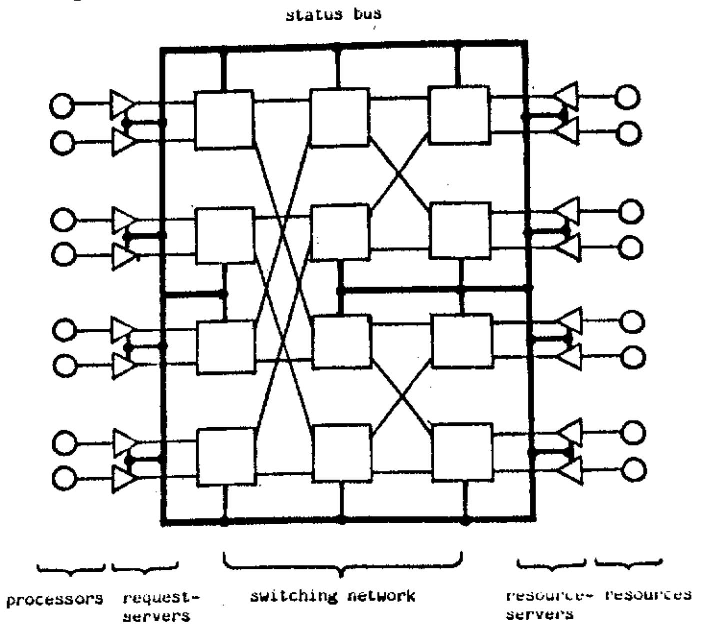


Figure 5. Architecture of a MRSIN embedded in an 8-by-8 Omega network.

To achieve optimal mapping, a switching process has to follow several token-propagation rules according to the status of the path connected. If the path is free, a switching process will deliver any resource token received from the resource side to the processor side. These directions are reversed if the path is registered. A request token is expected from where a resource token was delivered. In case that there are more than one sending paths, a resource token is duplicated, and one token is sent along each path. No duplication is done for the request token, and one of the paths is chosen at random. To avoid chaos in the parallel search of augmenting paths, the scheduling procedure is divided into alternating phases. In each phase only one type of tokens are allowed to propagate. The phase transitions is synchronized by broadcasting the status of each process on the status bus. The scheduling process is terminated when all the tokens are blocked in the resource-token distribution phase.

It can be shown that every process involved can be implemented by a simple sequential machine and is realizable by logic circuits. With this design, there are two factors that may contribute to a significant speedup: (a) the augmenting paths are searched in parallel; and (b) the time complexity is measured in terms of gate delays instead of instruction execution cycles. Therefore, the scheduling algorithm will run at least 100 times faster than a software implementation of the network flow algorithm.

3.4. Homogeneous MRSINS With Request Priority and Resource Preference

In a homogeneous MRSIN with request priority and resource preference, a request is associated with a priority level, and a resource is assigned a preference value. Many application-dependent attributes such as workload, execution speed, utilization and capability can be encoded into the request priorities and resource preferences. The objectives 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 that requests and resources be allocated in order of priorities and preferences. Further, the allocation of a request of higher priority (cf. resource of higher preference) may block other requests of lower priority (cf. resources of lower preference). For this class of MRSINs, the resource mapping problem can be transformed into the minimum-cost flow problem.

Consider a flow network $G(V,E,s,t,c,w)$ in which a cost per unit flow, $w(e)$, is associated with edge $e \in E$. The minimum-cost flow problem finds a legal s-t flow assignment of value F_0 with the minimum cost. The constraints in the minimum-cost flow problem are the same as those for the maximum-flow problem. However, the objective function is to determine the cheapest s-t paths through which a fixed flow F_0 can be pushed. 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:

$$(1)\quad \sum_{e \in \alpha(v)} f(e) - \sum_{e \in \beta(v)} f(e) = \begin{cases} -F_0 & \text{if } v = s \\ F_0 & \text{if } v = t \\ 0 & \text{otherwise} \end{cases}$$

$$(2)\quad 0 \leq f(e) \leq c(e) \qquad \text{for all } e \in E$$

In allocating resources the objective is to find a corresponding flow network whose optimal legal flow leads to an optimal resource mapping. The main idea behind the transformation is to embed priority and preference information into the objective function through the proper weight assignments on links. A possible transformation is given as follows:

Transformation 2: Create a flow network $G(V,E,s,t,c,w)$ from a homogeneous MRSIN with request priority and resource preference.

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

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

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 assigns the cost of carrying one unit of flow through a link as follows:

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

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_s is the preference of resource s.

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

T6. Set the total flow F_0 to the number of requests.

As an example, in the MRSIN shown in Figure 6a, each request is attributed with 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 is obtained by applying Transformation 2 to the MRSIN and is shown in Figure 6b.

The following theorem shows the correctness of Transformation 2.

Theorem 3: The optimal resource mapping on a homogeneous MRSIN with request priority and resource preference is equivalent to the minimum-cost integral flow in the flow network obtained by Transformation 2.

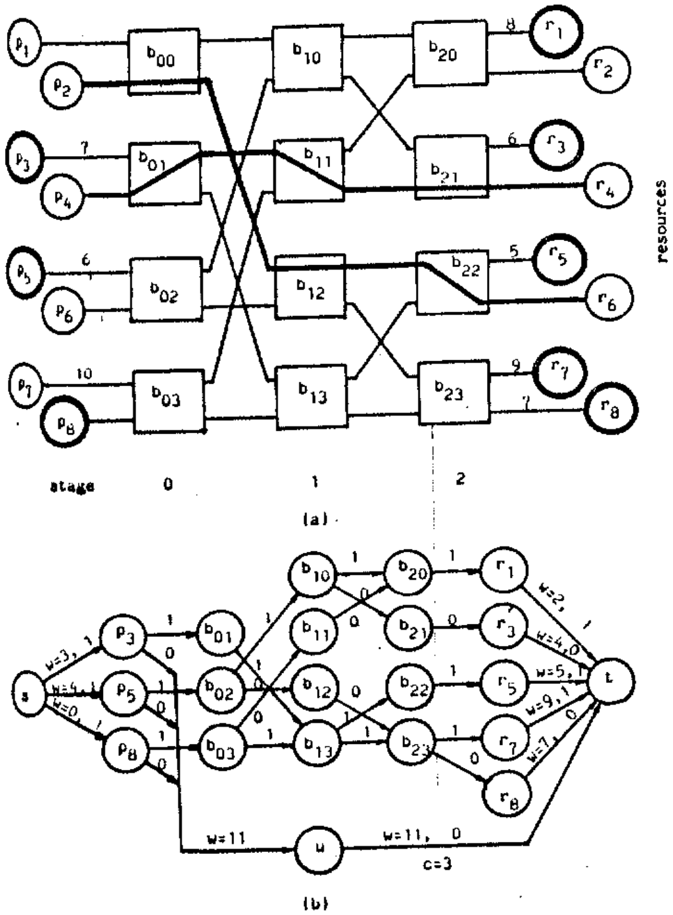


Figure 6

(a) A MRSIN with request priority and resource preference (highest priority=10; highest preference=10; dark paths in the network are already occupied; processors p_1, p_5, p_8 are making requests; resources r_1, r_3, r_5, r_7, r_8 are available).

(b) The associated flow network obtained by Transformation 2 (all arcs have capacity 1; cost of arc is zero except indicated otherwise).

Proof. It is easy to verify that a feasible flow does exist since one can always push the required amount of flow F_0 through the bypass node u. A flow that passes through the bypass node implies that the request is not allocated. Thus, maximizing the resource mapping is equivalent to assigning as much flow as possible to the part of the flow network other than the bypass node and is achieved by minimizing the cost of flow assignment. This can be proved by contradiction. Assume that the minimum-cost flow assignment does not define a maximum allocation, then there exists a s-t path such that the bypass node u is not on this path, and the path is not saturated so that at least one unit of flow can be advanced through it. The remaining flow that could have passed through the s-t path will pass through the bypass node u. According to the cost function w defined in Transformation 2, the cost of advancing flow through such a s-t path is less than that of advancing the same amount of flow through a path passing through node u. The total cost could be reduced if more flow is pushed through the s-t path instead of passing through the bypass node u. The existence of such a path implies that the original assignment is not minimum which contradicts the assumption. □

Edmonds and Karp have shown a scaled out-of-kilter algorithm to solve the minimum-cost flow problem of a general flow network in polynominal time [FUL61,EDM72]. For a flow network of 0-1 capacity, this is bounded by $O(|V||E|^2)$. Furthermore, the minimum-cost flow for a flow network of integral capacity is integral. Thus the optimal resource mapping on homogeneous MRSINs with request priority and resource preference can be obtained efficiently. As an example, applying the minimum-cost flow algorithm on the flow network in Figure 6b results in the following processor-resource mappings: $\{(p_1,r_5), (p_5,r_1), (p_8,r_7)\}$ The flow values of the arcs are also shown in Figure 6b.

A distributed implementation of the out-of-kilter algorithm is complex. The scheduling algorithm can be implemented efficiently in software.

4. OPTIMAL RESOURCE SCHEDULING ON HETEROGENEOUS MRSINS

A heterogeneous MRSIN consists of resources of different types, and a processor may generate a request of any type. A resource allocation problem on a heterogeneous MRSIN is equivalent to the *multicommodity maximum-flow problem*. The transformation is similar to Transformation 1 except that multiple source-sink pairs are introduced.

For k types of resource in the MRSIN, a multicommodity flow network has k source-sink pairs, (s^i,t^i) for i=1 to k. Let F^i be the flow of the i'th commodity. The search for the maximum flow can be formulated as a linear programming problem [AS78]:

Multicommodity Maximum-Flow Problem

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

subject to:

$$(1)\quad \sum_{e\in\alpha(v)} f^i(e) - \sum_{e\in\beta(v)} f^i(e) = \begin{cases} -F^i & v = s^i \\ F^i & v = t^i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for } i = 1, \ldots, k$$

$$(2)\quad 0 \le \sum_{i=1}^{k} f^i(e) \le c(e) \qquad \text{for all } e\in E$$

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 network and can be solved in a similar way as in the maximum-flow problem. The problem of finding the maximum integral flow in a general multicommodity flow network has been shown to be NP-hard [GAR79]. Fortunately, for a certain class of multicommodity flow networks, the maximum-flow values are always integral [EVA78]. Utilizing this property, the integral multicommodity maximum flow may be obtained efficiently by the *Simplex Method* that has been shown empirically to be a linear-time algorithm [MCC82]. Most multistage interconnection networks, including the cube and Omega, have transformations that belong to this class. However, distributed implementation of the algorithm as in the case of homogeneous MRSINs is very difficult, and software implementation has to be used.

The optimal mapping on a heterogeneous MRSIN with request priority and resource preference can also be obtained by transforming the problem to a multicommodity minimum-cost flow problem. Let $w^i(e)$ be the cost per unit flow for the i'th commodity on edge e, and $f^i(e)$ be the corresponding flow. A multicommodity minimum-cost flow problem may be formulated as follows:

Multicommodity Minimum-Cost Flow Problem

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

subject to:

$$(1) \quad \sum_{e \in \alpha(v)} f^i(e) - \sum_{e \in \beta(v)} f^i(e) = \begin{cases} -F_0^i & v = s^i \\ F_0^i & v = t^i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for } i = 1, \ldots, k$$

$$(2) \quad 0 \leq \sum_{i=1}^{k} f^i(e) \leq c(e) \qquad \text{for all } e \in E$$

The equivalent flow network consists k source-sink pairs and k bypass nodes where k is the number of resource types being requested. As for the case with no request priority, the flow network may also be visualized as the superposition of k single-commodity flow networks, and Transformation 2 can be applied to each layer.

5. CONCLUSION

A MRSIN is a suitable interconnection network for supporting resource sharing in a multiprocessor system. Optimal resource mapping in MRSINs is obtained by maximizing the number of communication paths that interconnect pairs of processors and resources. In this paper we have transformed various resource mapping problems into network flow problems for which efficient algorithms exist. In the simple case with homogeneous resources, the scheduler can be implemented efficiently by logic gates. In other cases software implementation of the scheduling algorithm is necessary.

The methodology described here is simple, yet it is not limited to solve resource mapping problems only. It can be applied to solve load balancing problems in multistage interconnection networks and packet routing in MRSINs. The former problem can be transformed to the minimum-cost flow problem by encoding processor loads into request priorities and resource preferences. The latter problem requires a packet to be sent through a route of the shortest delay and arrive at a processor of the minimum response time. This is achievable by encoding buffer delays into link costs in the minimum-cost flow problem.

REFERENCES

[AS78] A.A. Assad, "Multicommodity Network Flows - A Survey," *Networks*, Vol. 8, pp. 37-91, 1978.

[BON81] J.A. Bondy and U.S.R. Murty, *Graph Theory With Applications*, North-Holland, New York, N.Y., 1981

[BRI82] F.A. Briggs, K.S. Fu, K. Hwang and B.W. Wah, "PUMPS Architecture for Pattern Analysis and Image Database Management," *IEEE Trans. on Comp.*, Vol. C-31, No. 10, pp 969-983, Oct. 1982.

[CHO79] Y.C. Chow and W. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," *IEEE Trans. on Comp.*, Vol. C-28, No. 5, pp. 334-361, May 1979.

[DIN70] 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.

[EDM72] J. Edmonds and R.M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. ACM*, Vol. 19, No. 2, pp. 248-264, April 1972.

[EVA78] J.R. Evans and J.J. Jarvis, "Network Topology and Integral Multicommodity Flow Problems," *Networks*, Vol. 18, pp. 107-119, 1978.

[FEN81] T.Y. Feng, "A Survey of Interconnection Networks," *IEEE Computer*, pp.12-27, Dec. 1981.

[FOR62] L.R. Ford and D.R. Fulkerson, *Flow in Networks*, Princeton University Press, Princeton, N.J., 1962.

[FUL64] D.R. Fulkerson, "An Out-of-Kilter Method for Minimum Cost Flow Problems," *SIAM J. of Computing*, Vol. 9, No. 1, pp. 18-27, 1961.

[GAR79] M.R. Garey and D.S. Johnson, *Computer and Intractability: A guide to the theory of NP-completeness*, W.H. Freeman and Company, San Francisco, 1979.

[GOL81] B. Golden, M. Ball and L. Bodin, "Current and Future Research Directions in Network Optimization," *Computer and Operations Research*, Vol. 8, pp. 71-81, 1981.

[HIC82] A. Hicks, *Resource Scheduling on Interconnection Networks*, M.S. Thesis, Purdue University, Aug. 1982.

[HWA81] K. Hwang et al., "A Unix-based Local Computer Network With Load Balancing," *IEEE Computer*, Vol. 15, No. 4, pp. 55-66, April 1982.

[KUN81] H.T. Kung et al., *VLSI Systems and Computations*, Computer Science Press, Inc., Rockville, Maryland, 1981.

[LAW75] D. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. on Computers*, Vol. C-24, No. 12, pp. 215-255, Dec. 1975.

[MCC82] E.H. McCall, "Performance Results of the Simplex Algorithm for a Set of Real-Word Linear Programming Models," *Comm. of ACM*, Vol. 25, No. 3, pp. 207-213, March 1982.

[NI81] L.M. Ni and K. Hwang, "Optimal Load Balancing Strategies for a Multiple processor System," *Proc. of 10th Int'l Conf. on Parallel Processing*, pp. 352-357, Aug. 1981.

[PAP82] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[SIE81] H.J. Siegel and R.J. McMillen, "The Multistage Cube : A Versatile Interconnection Network," *IEEE Computer*, Vol. 14, No. 12, pp. 65-76, Dec. 1981.

[TRE82] P.C. Treleaven, D.R. Brownbridge and R.P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture," *Computing Surveys*, Vol. 14, No. 1, March 1982.

[WAH82] B.W. Wah and A. Hicks, "Distributed Scheduling of Resources on Interconnection Networks," *Proc. National Computer Conference*, AFIPS Press, pp. 697-709, 1982.

[WAH83] B.W. Wah and J.Y. Juang, "Load Balancing on Local Multiaccess Networks," *Proc. of 8th Conference on Local Computer Networks*, pp. 56-66, Oct. 1983.

[WAH84] B.W. Wah, "A Comparative Study of Distributed Resource Sharing on Multiprocessors," *IEEE Trans. on Computers*, Vol. C-33, No. 8, pp. 700-711, Aug. 1984.