Scheduling of Load Balancing Across Single-Channel Broadcast Networks

Emile Haddad

TR 93-37

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

December 20, 1993

Scheduling of Load Balancing
Across Single-Channel Broadcast Networks

Emile Haddad
Department of Computer Science
Virginia Polytechnic Institute and State University
2990 Telestar Court, Falls Church, VA 22042
E-mail: haddad@vtopus.cs.vt.edu Tel. (703) 698-6023.

Abstract

The problem of optimizing the balancing of processing load originating at the various sites of a networked set of heterogeneous processors is examined. The interconnection and communication architecture is non-specific, and the network is characterized only by is overall communication capacity . The distributed load is assumed to be arbitrarily divisible with no particular affinity to individual processors. The heterogeneous processors are characterized by their load execution speeds. The objective of load redistribution is to minimize processing completion time using the least allocation of communication bandwidth for load transfer. The optimal amounts of load exchange among the sending and receiving processors are derived. Front-end communication sub-processors implement load transfer across the network concurrently with load execution by the main processors. The load transfer schedule is represented in terms of the time-varying load transfer rates pertaining to each sending and receiving site. Absolute optimality can be achieved only if synchronization delay of receiving processors is eliminated through proper real-time scheduling of communicated load. The necessary and sufficient condition for the absence of synchronization delay is derived. The minimum communication capacity needed for the optimal load exchange is determined. The allocated minimum communication capacity of the network is dynamically partitioned among the individual load transfer rates which are specified as piece-wise constant functions of time that must be precisely adjusted at specific time instants to ensure the desired optimality. Although not unique, the specified optimal load transfer schedule is amenable to simple implementation. Practical implementation of the specified bandwidth partitioning may employ any appropriate scheme of communication multiplexing available on the given network . An example with specific problem parameters is used to illustrate the determination and implementation of the optimal load transfer schedule.

Key words: load balancing, divisible load, load redistribution, dynamic scheduling, execution time minimization, communication delay, synchronization delay, resource allocation, bandwidth partitioning, bandwidth minimization.

1. Introduction

The problem of load distribution and scheduling for performance enhancement in a parallel processing environment has been an important area of active research over the past 15 years. In its most general formulation, this multifaceted problem is highly complex, presenting a large number of modeling, analytical and computational aspects. One dominant issue is the interplay of processing and communication, a persistently recurrent theme of parallel processing. A significant modeling consideration is whether the load is discretely or continuously divisible and the resulting implications on the analytical and computational techniques that can be used. Several key aspects of the problem relate to the architecture of the parallel environment: homogeneous versus heterogeneous processors, shared-memory versus message-passing communication, the topology and protocols of the interconnection subsystem, etc. Other relevant questions pertain to the relationship of the load to the system: do load entities have special affinities to system entities, or where does the load originate in the multiple processor system. The other key issues of the problem are usually formulated by the "problem solver", particularly in relationship to objectives and strategy. What aspects of performance is one seeking to optimize through load distribution and scheduling: processing time, communication cost, system utilization, or combinations thereof. And what approach or scheme is one to allow in implementing the said optimization: is the optimal distribution to be carried out statically, i.e. before processing starts, or dynamically, during load execution.

This paper deals with the problem of optimizing the scheduling of load redistribution for a system of heterogeneous processors linked by an interconnection or communication network. The load is assumed to be homogeneous, arbitrarily divisible, and generated in different amounts at the various processing sites. This formulation is typical of many distributed real-time systems that collect large amounts of repetitive physical data which have no specialized processing requirements. The model can also be used in batch-systems where the load consists of massive data files with small processable elements. The site-generated loads can be processed locally by the host computer or transmitted across the network to be processed by other processors. It is assumed that the communication capacity (bandwidth) of the network is a scarce or costly resource that needs to be conserved. The limited communication bandwidth allocated for load exchange incurs a deterministic transmission delay . Each site has a front-end communication subprocessor that allows the main processor to execute concurrently with load migration across the network. The available communication bandwidth can be dynamically partitioned among the various communication sites. The objective is to minimize the overall processing time by determining the amounts, sources, destinations, bandwidth partitioning, and exact timing of the load redistribution schedule.

Analogous and related types of problems have been investigated in the literature. The earlier research efforts were concerned with discrete or indivisible loads [6]-[13]. More recent investigations have dealt with arbitrarily divisible loads which consist of large amounts of homogeneous data whose elements require identical processing needs. Loads of this nature typically arise in real time applications, such as radar tracking, image processing, and many types of other sensor-driven systems. In [1], a linear set of processors daisy-chained by dedicated communication links share a divisible total load originating at one end for execution in minimum time. In [2], the same problem is examined for a load originating at the root of a tree network of processors with or without front-ends. In [3], homogeneous processors share a load transmitted from one site across a single-channel network. Similar considerations are examined in [4] within the context of the problem of fusion of data from distributed sensors, which has been receiving increasing attention. In [5], static optimization of job execution time is examined for batched divisible loads where the system processors exhibit arbitrary execution time-load characteristics.

2. Problem Formulation

Given a system comprising a set of heterogeneous processor interconnected by a communication network , as shown in the Figure 1. A subset $\{P_i\}$ of the processors, indexed by $i = 1, 2, \ldots n$, is engaged in the generation and processing of shared load:

$$\{P_i\} = \{ P_1, P_2, \ldots, P_n \}$$

Each processor P_i is interfaced with the network via a front-end communication processor C_i. The generated load at each site is accumulated and stored. Periodically, or at certain specified time instants, the processing of theaccumulated load is initiated by pre-empting the processors $\{P_i\}$ for that purpose. Figure 1 shows the status at one such instant $t = 0$. The load existing at P_i is represented by the quantity x_i measured in some convenient unit, which we shall refer to henceforth as *load unit.* Let X denote the *total load* existing at time $t = 0$ on all processors:

$$X = x_1 + x_2 + \ldots x_n$$

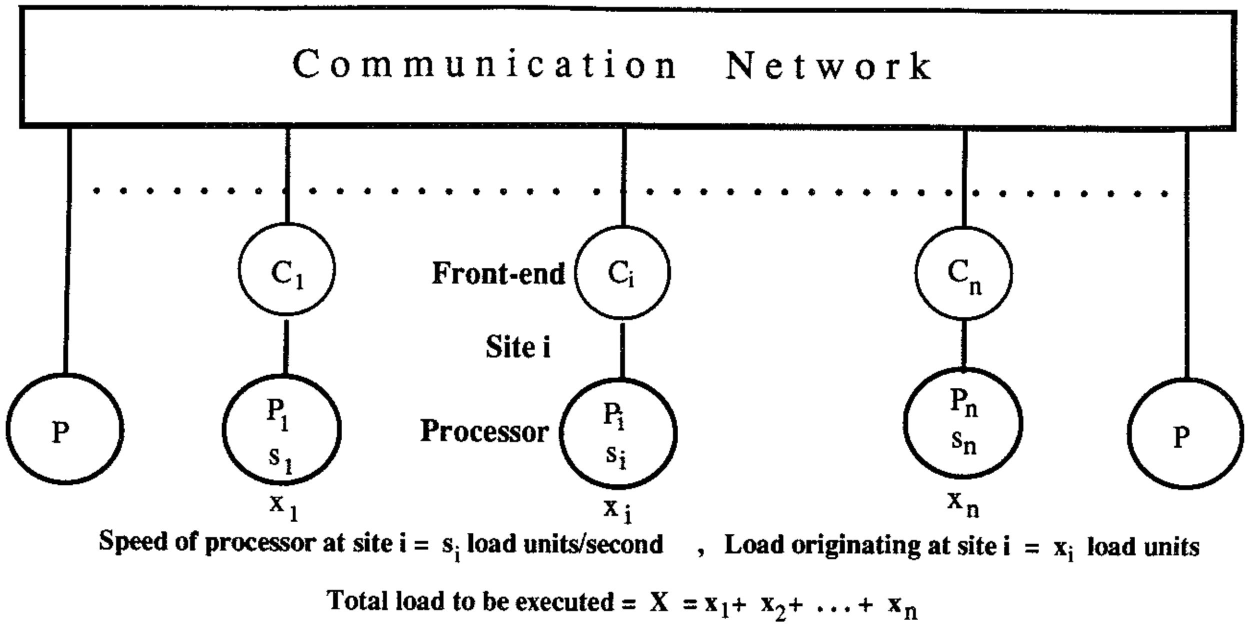
We assume that the load is *homogeneous* and *arbitrarily divisible*, such that any fraction of the total load X can be transferred and executed on any member of the processors $\{P_i\}$. We assume that the execution time of x_i on any processor is directly proportional to its magnitude, and the proportionality constant is indicative of the processor speed. Accordingly, we shall characterize x_i as a *continuous variable.* This modeling assumption would, for example, be adequate for cases where the load is generated by a distributed real-time process and consists of large amounts of the same type of data, such as sensor data, collected at the various sites for processing by the system. Each processor P_i is characterized by its *speed* s_i, measured in load units per second, representing the rate with which it can process any amount of load assigned to it for execution. If the load assigned to P_i is L load units, the *execution time* would be L/s_i seconds. If each processor P_i executes the entire load it holds at $t = 0$ locally, such that no load is exchanged across the network, then the *execution time* of P_i would be x_i/s_i:

$$x_i/s_i \equiv T_i = \text{execution time of processor } P_i \text{ (no load transfer)} \tag{1}$$

For convenience, and for reasons that will become apparent later, we shall assume that the processors $\{P_i\}$ are indexed (or re-indexed) such that corresponding values of $\{T_i\}$ form a *nondecreasing* sequence:

$$T_1 \le T_2 \le \cdots \le T_n \tag{2}$$

On the other hand, if we allow load exchange (balancing) across the network, then each front-end C_i may *send* or *transmit* some of the load, existing at its site at $t = 0$, to be executed by



Communication Network
C_1
Front-end
C_i
C_n
Site i
P
P_1 s_1
Processor
P_i s_i
P_n s_n
P
x_1
x_i
x_n
Speed of processor at site i = s_i load units/second , Load originating at site i = x_i load units
Total load to be executed = $X = x_1 + x_2 + \ldots + x_n$

Figure 1. The distributed load at time t = 0

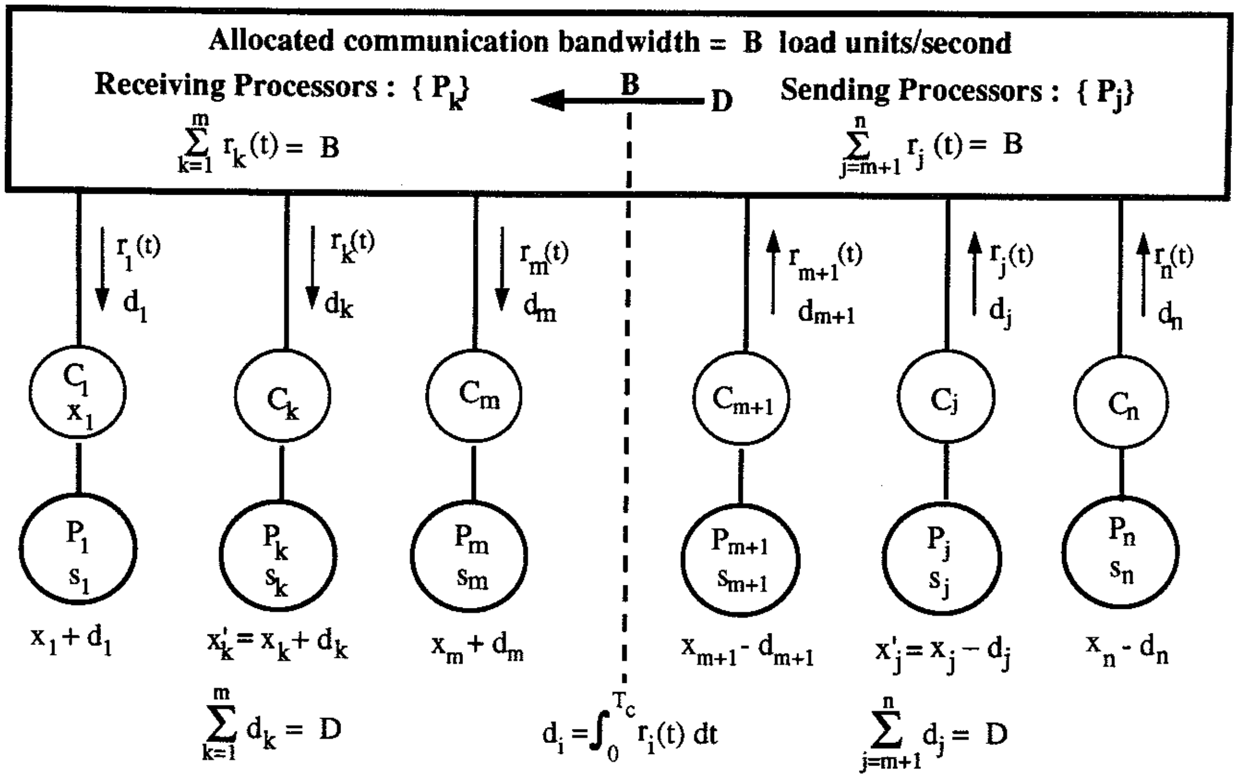


Figure 2. The run-time load transfer for $0 \le t \le T_c$

processors of other sites, or may *receive* additional load from other sites to be executed by its local processor. The *communication* of the load transfer over the network takes place *concurrently with load execution* at the various sites, and incurs an amount of communication delay proportional to the amount of transferred load (transmission time). This exchange is represented in Figure 2, where the receiving sites are indexed by $k = 1, 2, \ldots m$; and the sending sites are indexed by $j = m+, m+2, \ldots n$. Each P_k receives an additional load of d_k for a total load to be executed of $x'_k = x_k + d_k$. Each P_j executes a total load of $x'_j = x_j - d_j$, with the excess load of d_j sent by its front-end across the network to $\{P_k\}$. The total load transferred across the network is denoted by D:

$$D = d_1 + d_2 + \ldots + d_m = d_{m+1} + d_{m+2} + \ldots + d_n \qquad (3)$$

In order to service this load transfer, a certain amount of communication capacity (bandwidth) of the network must be *allocated* to the processors $\{P_i\}$ starting at $t = 0$ until the transfer is completed. Let W denote the total communication capacity (or transfer rate) of the network measured in *load units per second.* Let $B \leq W$ represent the portion of the *communication bandwidth* allocated at $t = 0$ to the processors $\{P_i\}$ for the purpose of servicing the balancing load transfer:

Communication bandwidth allocated for load balancing $\equiv$ B load units per second

Thus, the time needed to complete the transfer of the D load units from processors $\{P_j\}$ to processors $\{P_k\}$ across the network, i.e. the *communication completion time*, is given by

$$\textit{Communication completion time} \equiv T_c = D/B \qquad (5)$$

where we assume that the allocated communication bandwidth B is *fully utilized* for load transfer during the communication interval $[0, T_c]$, i.e. the network is carrying load transfer from processors $\{P_j\}$ to processors $\{P_k\}$ at the rate of B load units per second for the entire period. Figure 2 shows the variables $r_j(t)$ and $r_k(t)$ which represent the *load transfer rates* (measured in load units per second) corresponding to *real-time load flow* from the sites of processors $\{P_j\}$ and into the sites of processors $\{P_k\}$ respectively. It is important to note that these parameters are represented as *functions of time* . We shall demonstrate in the next section that the optimal load balancing is achieved by specifying the precise real-time scheduling of the load flow from/into the various system sites, i.e. by specifying the exact expressions of $r_j(t)$ and $r_k(t)$ as functions of time over the communication interval $[0, T_c]$. We shall require that

$$r_j(t) \geq 0 \quad , \quad r_k(t) \geq 0 \qquad\qquad \text{for all } t \in [0, T_c]$$

This requirement implies that each P_i will be designated as *either* a sending *or* a receiving processor for the entire duration of load transfer. Allowing the same processor to both send and receive loads would obviously be counterproductive since our problem formulation models the load as being arbitrarily divisible and homogeneous, with no special affinity to any particular processor. The scheduling of load communication is controlled and executed by the front-end processors, and proceeds concurrently with load execution by the host processors at the various system sites. As a preview of the nature of the functions $r_k(t)$, the reader is referred to Figure 3, which depicts the time variation of these functions for a specific example to be discussed later. In effect, these functions represent the dynamic algorithm for optimizing load migration in real-time, concurrently with execution. For the time being, note that Figure 2 indicates that the summation of load flow rates *from* the sites of $\{P_j\}$ must be equal to the summation of load flow rates *into* the sites of $\{P_k\}$, and both these summations are equal to B, the load transfer rate (bandwidth) on the network allocated for the process of load balancing:

$$r_1(t) + r_2(t) + \cdots + r_m(t) = r_{m+1}(t) + r_{m+2}(t) + \cdots + r_n(t) = B \qquad (6)$$

We shall show later that this general formulation of the multiplexing of transmitted load rates and de-multiplexing of received load rates is necessary if absolute optimality of load execution is to be achieved. The partitioning of the total communication bandwidth B allocated for load transfer among the individual transfer rates $\{r_i(t)\}$ can be implemented using any appropriate scheme available for the system, such as time-division or frequency-division multiplexing. Note also that the definite integral of $r_i(t)$ over the time interval $[0, T_c]$ of load transfer represents the total amount of load d_i transferred form/to the corresponding processor site:

$$d_i = \int_0^{T_c} r_i(t)\, dt \qquad (7)$$

We shall use the notation $d_i(t)$ to represent the total amount of load transferred from/to processor P up to time t, viz. over the interval [0 , t] :

$$d_i(t) = \int_0^t r_i(t)\, dt \qquad , \qquad t \in [0, T_c] \qquad (8)$$

Note that in the time modeling of the load balancing process as presented above, we have ignored any consideration of propagation delay. We assume that propagation time across the network is negligible compared to load transmission time and to the duration of communication completion time T_c.

The remainder of the paper is organized as follows: Section 3 examines the problem of synchronization delay of a receiving processor P_k and derives the necessary and sufficient condition for its existence in terms of x_k, s_k , and $r_k(t)$. Section 4 derives the optimal redistribution of the given load to minimize job completion time under the assumption that the receiving processors experience no synchronization idleness. The sending and the receiving processors are identified and the corresponding amounts of exchanged load are determined. In Section 5, a time schedule for implementing the optimal redistribution across the network, using minimum communication bandwidth, is specified in terms of piece-wise constant load transfer rate functions. Section 6 presents an example to illustrate the application of the various results. The paper concludes with a discussion of the basic issues related to implementing the optimal load balancing scheme under specific network communication architectures.

3. Synchronization Delay

An important aspect of the real-time modeling of load transfer, as described above, is the possibility that any member of the *receiving* processors $\{P_k\}$ might experience *synchronization delay* or *forced idleness* due to a temporary depletion of load. Note that the sending processors $\{P_j\}$ do not experience synchronization delay. Such a processor starts executing its new load allocation x'_j at time $t = 0$ and continues executing without interruption until completion time

$T'_j = x'_j / s_j$, constantly executing load at its nominal (maximum) speed s_j. Let $x_j(t)$ denote the total remaining load existing on processor P_j at time t :

$$\textit{Un-executed load on } P_j \textit{ at time } t \equiv x_j(t) = x'_j - s_j t \quad , \qquad t \in [0, T'_j]$$

Note that $x_j(t)$ does not include any amount of the excess load $d_j = x_j - x'_j$ which is being concurrently transmitted across the network by the front-end processor at the rate specified by $r_j(t)$. Note that $x_j(t) > 0$ for all t during the execution interval except at the completion instant T'_j where $x_j(T'_j) = 0$. Things are not so simple when we examine the corresponding considerations for a *receiving* processor P_k. Again, Let $x_k(t)$ denote the total load existing on processor P_k at time t :

$$\textit{Un-executed load on } P_k \textit{ at time } t \equiv x_k(t) \geq 0 \quad , \qquad t \in [0, T'_k]$$

where T'_k is the *execution completion time.* The expression for $x_k(t)$ in terms of the given parameters is

$$x_k(t) = x_k + d_k(t) - e_k(t) = x_k + \int_0^t r_k(t)\,dt - e_k(t), \qquad t \in [0, T'_k] \tag{9}$$

where

$$e_k(t) \equiv \textit{total load executed by } P_k \textit{ during the interval } [0, t] \tag{10}$$

This simply states that the net load $x_k(t)$ existing on P_k at time t is equal to the initial load x_k plus the extra load $d_k(t)$ received during the time interval [0, t] minus the total load $e_k(t)$ executed by the processor during the same interval. The difficulty arises when we attempt to determine an expression for $e_k(t)$. We are tempted to write, as before,

$$e_k(t) = s_k t \quad ? \tag{11}$$

which would be true only if the processor has been continuously executing for the entire period [0, t] at its nominal speed s_k. This would be the case if the processor was not idled or slowed down for any length of time due to total depletion of existing load $x_k(t)$ coupled with a temporary shortage of replenishment from incoming flow of extra load intended for the processor, in which case $e_k(t)$ would be *less than* $s_k t$. Thus

$$e_k(t) = s_k t, \qquad \text{if} \quad x_k(\alpha) = x_k + d_k(\alpha) - e_k(\alpha) > 0 \quad \text{for all } \alpha \in [0, t]$$

In general, when the above condition is not satisfied, the equality of $e_k(t)$ to $s_k t$ is not guaranteed, and we should write

$$e_k(t) \le s_k t \qquad \text{for all } t \in [0, T'_k] \tag{12}$$

To account for the fact that $e_k(t)$ may be less than $s_k t$, we introduce the parameter $\tau_k(t) \ge 0$ defined as

$$\tau_k(t) \equiv t - e_k(t)/s_k \equiv \textit{Effective total idle time of } P_k \textit{ during } [0, t]$$

$$e_k(t) = s_k t - s_k \tau_k(t) = s_k [t - \tau_k(t)] , \qquad \tau_k(t) \ge 0 \tag{13}$$

The last equality states that during the time period [0, t] the processor has, in effect, been executing for an aggregate time equal to $t - \tau_k(t)$ and *idle* for an aggregate time equal to $\tau_k(t)$. If the physical operation of the processor is such that its execution speed s_k remains constant, then the value of $\tau_k(t)$ would represent the *actual* total of the time periods it was experiencing synchronization delay during [0, t] . In practical implementations, this behavior might occur in load transfer schemes where the load is packetized and the receiving processor is allowed to process a packet only after its reception is completed. If the processor runs out of existing load (i.e. $x_k(t) = 0$) and the packet inflow rate is less than the execution rate (i.e. $r_k(t) < s_k$), then the processor will alternate between periods of packet execution and periods of idleness awaiting the completion of the next packet transfer. Alternatively, we may equivalently model the same effects by introducing the parameter $s_k(t) \le s_k$ which measures the *effective average speed* of the processor over the interval [0, t] :

$$s_k(t) \equiv e_k(t)/t \equiv \textit{Effective average execution speed of } P_k \textit{ during } [0, t]$$

$$e_k(t) = s_k(t) t , \qquad s_k(t) \le s_k \tag{14}$$

Equating the expressions for $e_k(t)$ from the two representations in (13) and (14), we have the following relationship between the parameters $\tau_k(t)$ and $s_k(t)$:

$$s_k(t) = s_k [1 - \tau_k(t)/t]$$

Returning to the expression of $x_k(t)$ in (9) and substituting for $e_k(t)$ from (13), we obtain:

$$x_k(t) = x_k + \int_0^t r_k(t)\,dt - e_k(t) = x_k + \int_0^t r_k(t)\,dt - s_k [t - \tau_k(t)]$$

$$x_k(t) = x_k + \int_0^t r_k(t)\,dt - s_k t + s_k \tau_k(t) \tag{15}$$

For a given processor P_k, the magnitude of cumulative effective idle time $\tau_k(t) \geq 0$ at any given t depends on the relative magnitudes of three parameters: the amount of initial load x_k, the speed of load execution s_k, and the time schedule of load delivery over $[0, t]$, as specified by the load transfer rate function $r_k(t)$. The following Assertion states a *necessary and sufficient condition* for a receiving processor P_k not to experience any synchronization delay during the communication interval $[0, T_c]$. The absence of synchronization delay is essential for achieving optimal load transfer schedules which minimize the processing completion time of the given load, as we demonstrate in a subsequent section. The characterization of the absence of synchronization delay, expressed in the Assertion in terms of the parameters x_k, s_k, and $r_k(t)$, will be used to verify that a specified load transfer schedule entails no such delays for all the receiving processors, viz. $\tau_k(t) = 0$ for all t and all k.

Assertion 1: *The necessary and sufficient condition for a receiving processor* P_k *to experience no synchronization delay during the communication interval* $[0, T_c]$ *is that*

$$g_k(t) \equiv x_k + \int_0^t r_k(t)\,dt - s_k t \geq 0 \qquad \textit{for all} \quad t \in [0, T_c] \tag{16}$$

Proof: *Sufficiency :* Given (16) is satisfied, we prove that $\tau_k(t) = 0$ for all $t \in [0, T_c]$. Note that since $\tau_k(t)$ represents the *cumulative* idleness from time 0 to time t, we have $\tau_k(0) = 0$ and $\tau_k(t)$ is a *nondecreasing* function of t for all $t > 0$. This implies that either $\tau_k(t) = 0$ for all $t \geq 0$ which proves the sufficiency of the condition, or that there exists a $t_1 \geq 0$ such that

$$\tau_k(t) = 0 \qquad \text{for all } t \in [0, t_1]$$

$$\tau_k(t) > 0 \qquad \text{for all } t > t_1$$

This implies that P_k experienced cumulative synchronization delay over the interval $(t_1, t]$ equal to $\tau_k(t) > 0$. This in turn implies that there must have been at least one instant $t_2 \in (t_1, t]$ at which the processor load $x_k(t)$, as expressed in (15) was totally depleted:

$$x_k(t_2) = g_k(t_2) + s_k\tau_k(t_2) = 0$$

But this is impossible since by hypothesis $g_k(t_2) \geq 0$ and $\tau_k(t_2) > 0$. Hence, the only valid possibility is $\tau_k(t) = 0$ for all $t \geq 0$.

Necessity : given $\tau_k(t) = 0$ for all $t \geq 0$, we prove that $g_k(t) \geq 0$ for all $t \in [0, T_c)$. Assume , to the contrary, that $g_k(t') < 0$ for some $t' \in [0, T_c)$. Since $\tau_k(t') = 0$, we obtain

$$x_k(t') = g_k(t') + s_k\tau_k(t') = g_k(t') < 0$$

which is absurd since $x_k(t')$ is the un-executed load on P_k at time t', which can not be negative.

The condition stated in (16) will be used to verify that the optimal load transfer schedule, discussed in a subsequent section, does not entail synchronization delay for any of the receiving processors $\{P_k\}$.

4. Optimal Load Redistribution

Consider that the total initial load $X = \Sigma_i x_i$ existing on all processors at $t = 0$ is to be *redistributed* in order to minimize execution time, such that processor P_i would execute a new value x'_i of total load instead of the load x_i initially available at its site. We must have

$$\Sigma_i x'_i = x'_1 + x'_2 + \cdots + x'_i + \cdots + x'_n = X \qquad (17)$$

The load migration implied by this redistribution is to be carried across the communication network starting at time $t = 0$. If $x'_i > x_i$, then processor P_i would *receive* an additional load of $x'_i - x_i = d_i$. If $x'_i < x_i$, then processor P_i would *send* the excess load of $x_i - x'_i = d_i$. Let T'_i denote new value of *completion time* for processor P_i executing the new value of load x'_i , which can be expressed as

$$\textit{Completion time of } P_i \equiv T'_i(x'_i, \tau_i) = x'_i/s_i + \tau_i \qquad (18)$$

$$\textit{Execution time of } P_i \equiv x'_i/s_i \qquad (19)$$

$$\textit{Effective total idle time of } P_i \textit{ during } [0, T'_i] \equiv \tau_i = \tau_i(T'_i) \geq 0 \qquad (20)$$

For a sending processor P_j , $\tau_j = 0$. For a receiving processor P_k , τ_k might have a nonzero positive value which depends on the function $r_k(t)$ specifying the rate of load transfer to P_k, as elaborated in the preceding section. We define the *job completion time*, denoted by T', as the completion time of the processor which finishes execution last, i.e.

$$\textit{Job completion time} \equiv T' = \max_i \{ T'_i \} = \max_i \{ (x'_i / s_i) + \tau_i \} \qquad (21)$$

The *optimization objective* is to minimize T' by choosing an optimal *load redistribution*, represented by the set of load values $x' = \{x'_i\}$; and specifying a real-time *load transfer schedule*, represented by the set of transfer rate functions $r = \{r_i(t)\}$. Expressing the dependence of T' on x' and r by writing $T'(x', r)$, the minimization is stated as:

$$T'_{min} \equiv \min_{x', r} T'(x', r)$$

The following assertion establishes an absolute *lower bound* T on the value of T'.

Assertion 2 : *For any load redistribution* $x' = \{x'_i\}$ *with* $\Sigma_i x'_i = X$, *the job completion time* T' *cannot be less than* $T \equiv X/S$:

$$T' \geq T \equiv X/S$$

$$S \equiv \Sigma_i s_i = s_1 + s_2 + \cdots + s_n$$

Proof: Assume, to the contrary, that $T' < X/S$ and reach a contradiction. From (21) we have

$$T' = \max_i \{ T'_i \} = \max_i \{ (x'_i / s_i) + \tau_i \} < X/S \qquad (22)$$

This implies

$$(x'_i / s_i) + \tau_i < X/S, \qquad \text{for all } i$$

From (20) the synchronization delay is non-negative, $\tau_i \geq 0$, hence

$$(x'_i / s_i) < X/S, \qquad \text{for all } i$$

$$x'_i < s_i X/S, \qquad \text{for all } i$$

Taking the summation of both sides over all i, we obtain

$$\Sigma_i x'_i < \Sigma_i s_i X/S = (X/S)\Sigma_i s_i = SX/S = X$$

$$X < X, \qquad \text{a contradiction.}$$

The minimum value of T', namely T'_{min}, must therefore be no less than the lower bound T:

$$T'_{min} \geq T = X/S \qquad (23)$$

The following assertion indicates how the load redistribution x' can be specified to achieve a job completion time *equal to* T.

Assertion 3: *If the load is redistributed such that*

$$x'_i = s_i T = s_i X/S, \quad \textit{for all } i \tag{24}$$

and the load transfer schedule $r = \{r_i(t)\}$ *can be specified such that no processor experiences synchronization delay, then the redistribution is optimal, with all processors exhibiting identical completion times*

$$T' = T \quad \text{and} \quad T'_i = T \quad \textit{for all } i \tag{25}$$

Proof: First verify that the summation of values of load redistribution specified in (24) is X:

$$x'_i = s_i X/S, \qquad \text{for all i}$$

$$\Sigma_i x'_i = \Sigma_i s_i X/S = SX/S = X$$

Next verify that if $\tau_i = 0$ for all i, then all processors have the same completion time:

$$T'_i = (x'_i / s_i) + \tau_i = x'_i/s_i = s_i X/S s_i = X/S = T, \qquad \text{for all i}$$

Hence,

$$T' = \max_i\{T'_i\} = T$$

It should be emphasized that the attainment of the optimal completion time T through the load redistribution $x'_i = s_i X/S$, as specified by the assertion, is contingent upon the precise determination of the load transfer rate functions $\{r_i(t)\}$ that would deliver the *exact amounts* of migrating loads from the sending processors $\{P_j\}$ to the receiving processors $\{P_k\}$ with *no synchronization delays* experienced by the latter. Before explaining how this can be achieved, we must identify the sets of sending processors $\{P_j\}$ and receiving processors $\{P_k\}$ and the corresponding exact amounts of load that must be transferred from the first set to the second. This is presented in the following assertion. Recall from (1) and (2) that the given processors $\{P_i\}$ are indexed, for convenience, according to the nondecreasing values of the parameters T_i

$$T_i \equiv x_i/s_i \tag{26}$$

$$T_1 \le T_2 \le \cdots \le T_n \tag{27}$$

The identification of $\{P_j\}$ and $\{P_k\}$ is obtained by simply comparing each T_i to T.

Assertion 4: *If* $T_i > T$ *then* $x_i > x'_i$ *and excess load is sent by* P_i *in the amount of*

$$d_i = x_i - x'_i = x_i - (s_i X/S) \tag{28}$$

If $T_i < T$ *then* $x_i < x'_i$ *and additional load is received by* P_i *in the amount of*

$$d_i = x'_i - x_i = (s_i X/S) - x_i \tag{29}$$

The sets $\{P_k\}$ *and* $\{P_j\}$ *correspond to* $k = 1, 2, \cdots m$ *and* $j = m+, m+2, \cdots n$ *such that*

$$T_1 \leq T_2 \leq \cdots \leq T_k \leq \cdots \leq T_m \leq T < T_{m+1} \leq T_{m+2} \cdots \leq T_j \leq \cdots \leq T_n \tag{30}$$

where $m = \max\{i : T_i \leq T\}$

The total amount of load D *transferred across the network is*

$$D = d_1 + d_2 + \ldots + d_m = d_{m+1} + d_{m+2} + \ldots + d_n \tag{31}$$

The communication completion time, viz. the load transfer completion time, is

$$T_c = D/B \tag{32}$$

Note that if $T_i = T$ then $x'_i = x_i$ and $d_i = 0$, in which case we qualify P_i as a "receiving" processor with "zero load transfer".

5. Optimal Scheduling of Load Transfer

Having specified how the load should be optimally redistributed, we turn our attention to the task of specifying how the load transfer should be scheduled in real-time in order to ensure the requirements mandated by Assertions 2 - 4 above. We do this by specifying the set of time functions $r = \{r_i(t)\}$ representing the rates of load transfer for each processor over the communication interval $[0, T_c]$. These requirements for optimal redistribution and scheduling are expressed as conditions on the time functions $\{r_i(t)\}$ as follows:

Condition 1 : A given load transfer schedule $r = \{r_i(t)\}$ must exhibit a *finite* communication completion time T_c beyond which all load transfer rates $r_i(t)$ are zero:

$$r_i(t) = 0 \qquad \text{for all } t > T_c, \text{ all } i \tag{33}$$

$$T_c = \min\{t_c : r_i(t) = 0 \text{ for all } t > t_c, \text{ all } i\}$$

Condition 2 : At all times during the communication interval $[0, T_c]$, the aggregate rate of load transfer (sending or receiving) must be equal to the communication bandwidth B allocated on the network for servicing the load transfer :

$$\Sigma_j r_j(t) = B = \Sigma_k r_k(t) \quad , \text{ for all } t \in [0, T_c] \tag{34}$$

Condition 3 **:** The total amount of load d_j transmitted from a sending processor and the total amount of load d_k delivered to a receiving processor during the communication interval $[0, T_c]$, as expressed in (7), must be equal to the optimal values stated in (28) and (29):

$$\int_0^{T_c} r_j(t)\, dt = x_j - (s_j X/S) \quad , \quad \int_0^{T_c} r_k(t)\, dt = (s_k X/S) - x_k \tag{35}$$

Condition 4 **:** Non of the receiving processors should experience synchronization idleness at any time during the communication interval $[0, T_c]$, which is equivalent to the necessary and sufficient condition stated in (16) of Assertion 1:

$$\int_0^{t} r_k(t)\, dt \geq s_k t - x_k \qquad \text{for all } t \in [0, T_c] \tag{36}$$

Any given $r^* = \{r_i(t)\}$ that satisfies the above conditions is said to be an *optimal dynamic load transfer schedule* , and results in the minimum job execution time $T = X/S$, as stated in Assertions 2 - 4. The characterization of such schedules as "dynamic" is to reflect the fact that load transfer across the network takes place concurrently with load execution by the processors. We shall shortly show how such schedules can be *specified* with concrete simple expressions for the functions $r_i(t)$. We first demonstrate that the above requirements for an optimal load transfer schedule imply that the bandwidth B, allocated on the network for load transfer, should not be less than a certain minimum value R to be specified. In other words, an optimal schedule r^* is not feasible or realizable unless $B \geq R$.

Assertion 5: *For an optimal schedule* $r^* = \{r_i(t)\}$ *to be feasible, the communication bandwidth* B *allocated for load transfer must be greater or equal to* R:

$$B \geq R \equiv (1/2)\Sigma_i \,|\, s_i - (S/X)\, x_i \,| \tag{37}$$

$$B = R \qquad , \qquad \text{when } T_c = T = X/S \tag{38}$$

Proof: Given an optimal load transfer schedule $r^* = \{r_i(t)\}$ which satisfies conditions 1-4 above, we prove that the inequality in (37) must be true. From (34) in condition 2, we have

$$2B = \Sigma_j r_j(t) + \Sigma_k r_k(t) \quad , \qquad \text{for all } t \in [0, T_c]$$

Integrating both sides with respect to t over the interval $[0, T_c]$, we obtain

$$\int_0^{T_c} 2B\,dt = \Sigma_j \int_0^{T_c} r_j(t)\,dt + \Sigma_k \int_0^{T_c} r_k(t)\,dt$$

Substituting, from (35) in condition 3, the expressions for the integrals, we obtain

$$2BT_c = \Sigma_j\ x_j - (s_jX/S) + \Sigma_k\,(s_kX/S) - x_k = \Sigma_i\,|\,(s_iX/S) - x_i\,| \qquad (39)$$

From (36) in condition 4 with $t = T_c$, we have

$$\int_0^{T_c} r_k(t)\,dt \ \geq\ s_kT_c - x_k$$

Substituting, from (35) in condition 3, the expression for the integral, we obtain

$$(s_kX/S) - x_k \geq s_kT_c - x_k$$

$$T_c \leq X/S \qquad (40)$$

This inequality states that, in an optimal schedule r^*, the load transfer (communication) completion time T_c should not exceed the minimum job completion time $T = X/S$, which is intuitively expected if no synchronization delay is to occur. Substituting the last inequality into (39), we obtain the required inequality in (37)

$$2BX/S \ \geq\ \Sigma_i\,|\,(s_iX/S) - x_i\,|$$

$$B \ \geq\ (1/2)\Sigma_i\,|\,s_i - (S/X)\,x_i\,| \ \equiv R \qquad (41)$$

Note that the equality in (41) holds when the equality in (40) holds:

$$B = (1/2)\Sigma_i\,|\,s_i - (S/X)\,x_i\,| \equiv R, \qquad \text{when } T_c = X/S \qquad (42)$$

In applications where the network communication capacity is a scarce or costly resource, it would be advantageous to achieve bandwidth conservation by seeking optimal load transfer schedules r^* that use the *minimum bandwidth allocation* , namely $B = R$. Let $r^{**} = \{r_i(t)\}$ denote such a schedule, which satisfies conditions 1-4 with the additional constraint of $B = R$ or equivalently $T_c = X/S = T$. We refer to such a schedule as a *minimal-bandwidth optimal load transfer schedule*. The following Theorem formulates a simple specification for such a schedule in terms of the various given problem parameters. For each sending processor P_j, $r_j(t)$ is specified as a *constant* function over the interval [0, T]. For each receiving processor P_k, $r_k(t)$ is specified as a *piece-wise constant* function over the interval [0, T], as illustrated in Figure 3 , with its values changing in a step-wise fashion at certain time instants $\{t_p\}$ whose values are computed by a simple recursive relationship. The parameters used in the statement of the Theorem are recapitulated below for the convenience of the reader.

x_i = initial load of i-th processor, $X = \Sigma_i x_i$ = total load to be processed

s_i = speed of i-th processor, $S = \Sigma_i s_i$ = aggregate speed of processors

$T_i = x_i/s_i$ = completion time of i-th processor if no load transfer is invoked

$T = X/S$ = lower bound on job completion time = optimal job completion time

$R = (1/2)\Sigma_i | s_i - (S/X)x_i |$ = minimal bandwidth allocated for optimal load transfer

k = index of receiving processors = 1, 2, . . . , m. $m = \max \{i : T_i \le T\}$

j = index of sending processors = m+1, m+2, . . . , n. n = number of processors

Theorem

The following specification of the load transfer rate functions $\{r_i(t)\}$ *over the communication interval* $[0, T_c]$, with $B = R$ *and* $T_c = T = X/S$, *represents a minimal-bandwidth optimal load transfer schedule consisting of piece-wise constant functions* .
For all i, *we require* $r_i(t) = 0$ *for all* $t > T$, *and*

1. *For a sending processor* P_j, *with* $j = m+1, m+2, \ldots, n$, :

$$r_j(t) = (S/X)\, x_j - s_j \qquad \text{for all } t \in [0, T] \tag{43}$$

2. *For a receiving processor* P_k, *with* $k = 1, 2, \ldots, m$:

Let
$$\{t_p\} = \{ t_1 = 0, t_2, \ldots t_{m+1} \} \tag{44}$$

$$t_{p+1} = t_p + (T_{p+1} - T_p)S_p/R, \qquad p = 1, 2, \ldots, m. \tag{45}$$

where
$$S_p \equiv s_1 + s_2 + \cdots + s_p ,$$

and $T_p = x_p/s_p$ *except for* T_{m+1} *which is locally re-defined in* (45) *as*

$$T_{m+1} \equiv T \tag{46}$$

then
$$r_k(t) = 0 \qquad \text{for all } t \in (t_p, t_{p+1}), \quad p = 1, 2, \ldots, k-1. \tag{47}$$

$$= (s_k/S_p)R \qquad \text{for all } t \in (t_p, t_{p+1}), \quad p = k, k+1, \ldots, m. \tag{48}$$

Proof: We now demonstrate the optimality of the schedule r specified in the Theorem by verifying that its functions $\{r_i(t)\}$ satisfy conditions 1 - 4 stated in (33) - (36), with $B = R$ and $T_c = T = X/S$, representing the requirement of *minimal bandwidth* allocation for load transfer.

Condition 1 : Since for all i, $r_i(t) = 0$ for all $t > T$, the condition is satisfied.

Condition 2 : For the sending processors we have, for all $t \in [0, T]$,

$$\Sigma_j r_j(t) = \Sigma_j (S/X) x_j - s_j = (X/S)\Sigma_j x_j - s_j X/S$$

Recalling from (28), (31), and (32) that $x_j - s_j X/S = d_j$, $\Sigma_j d_j = D$, and $D/T = B$, we obtain

$$\Sigma_j r_j(t) = (1/T)\Sigma_j d_j = D/T = B = R$$

For the sending processors we have, for any $t \in (t_p, t_{p+1})$, by substituting the expressions for $r_k(t)$ from (47) and (48) and noting that $r_k(t) = 0$ for all $k > p$:

$$\Sigma_k r_k(t) = r_1(t) + r_2(t) + \cdots + r_p(t) = (s_1/S_p)R + (s_2/S_p)R + \cdots + (s_p/S_p)R$$
$$= R(s_1 + s_2 + \cdots + s_p)/S_p = R\, S_p/S_p = R$$

Condition 3 : For a sending processor, we have

$$\int_0^T r_j(t)\,dt = \int_0^T [(S/X)x_j - s_j]dt = T[(S/X)x_j - s_j] = (X/S)[(S/X)x_j - s_j] = x_j - (s_j X/S)$$

For a receiving processor, since $r_k(t)$ is *piece-wise constant* over the intervals $(t_{p+1} - t_p)$, the integral may be expressed as a discrete summation:

$$\int_0^T r_k(t)\,dt = \sum_{p=1}^{m} r_k(t)[t_{p+1} - t_p]$$

Substituting the expressions for $r_k(t)$ and $[t_{p+1} - t_p]$ from (47), (48) and (45), and noting that $r_k(t) = 0$ for $p < k$, we obtain

$$\int_0^T r_k(t)\,dt = \sum_{p=k}^{m} s_k(T_{p+1} - T_p) = s_k \sum_{p=k}^{m} (T_{p+1} - T_p) = s_k(T_{m+1} - T_k)$$

Substituting from (46) $T_{m+1} = T$, we obtain the required result

$$\int_0^T r_k(t)\,dt = s_k(T - T_k) = (s_k X/S) - x_k$$

Condition 4 : This condition pertains only to the receiving processors. We need to verify that

$$\int_0^t r_k(t)\,dt \geq s_k t - x_k \qquad \text{for all } t \in [0, T]$$

The validity of this condition for the functions $r_k(t)$ of the Theorem is given in the Appendix.

6. Example

Consider the example with the numerical values shown in Table 1 below. The calculated value of the minimum job completion time, T = 50, lies between T_4 and T_5, which implies m=4. Thus the receiving processors are the first four and the sending processors are the remaining two, as indicated in Table 2. The minimal communication bandwidth that should be allocated on the network to service the optimal redistribution is

$$R = (1/2)\Sigma_i \mid s_i - (S/X)x_i \mid = (1/2T)\Sigma_i \mid Ts_i - x_i \mid = (1/2T)\Sigma_i \mid x'_i - x_i \mid = (1/2T)\Sigma_i d_i = 200/100 = 2$$

The minimal-bandwidth optimal load transfer schedule for the receiving processors is shown in Table 3 and also in Figure 3. For the sending processors, we have $r_5(t) = 1.1$ and $r_6(t) = 0.9$.

TABLE 1. Parameters of the example

P_i	P_1	P_2	P_3	P_4	P_5	P_6	Sum
x_i	14	36	30	20	80	70	X = 250
s_i	1.0	1.5	1.0	0.5	0.5	0.5	S = 5.0
$T_i = x_i/s_i$	14	24	30	40	160	140	T=X/S=50

TABLE 2. Optimal load redistribution

x_i	14	36	30	20	80	70
$x'_i = s_i T$	50	75	50	25	25	25
$d_i=\lvert x'_i - x_i\rvert$	36	39	20	5	55	45
$\{P_k\}$	P_1	P_2	P_3	P_4		
$\{P_j\}$					P_5	P_6

TABLE 3. Optimal load transfer schedule for the receiving processors

t_p	$t_1 = 0$	$t_2 = 5$	$t_3 = 12.5$	$t_4 = 30$	$t_5 = T = 50$	$\int_0^T r_2(t)dt$	d_i
$[t_p, t_{p+1}]$	[0 , 5]	[5 , 12.5]	[12.5 , 30]	[30, 50]			
$r_1(t)$	2	4/5	4/7	1/2		36	36
$r_2(t)$	0	6/5	6/7	3/4		39	39
$r_3(t)$	0	0	4/7	1/2		20	20
$r_4(t)$	0	0	0	1/4		5	5
Sum	R = 2	R = 2	R = 2	R = 2		D = 100	D = 100

6. Implementation Issues

We now examine the basic issues involved in implementing the general result we have derived for the optimal scheduling of divisible load redistribution, which can be recapitulated as follows: given the initial values of distributed loads $\{x_i\}$ and the values of processor speeds $\{s_i\}$, one can specify an optimal runtime schedule $\{r_i(t)\}$ of concurrent piece-wise constant load transfer rates that minimizes job completion time while using the least amount of communication capacity. Thus, there are two obvious aspects to the optimality of the prescribed schedule: its minimization of completion time *and* communication bandwidth. There is yet a third aspect that needs further elaboration: its *simplicity* of specification and implementation in terms of piece-wise constant transfer rates $\{r_i(t)\}$. We should emphasize that such an optimal schedule is *not unique,* in the sense that there are other ways of specifying $\{r_i(t)\}$ for achieving minimum time and bandwidth. This can be easily verified with specific examples. Refer again to the example of the previous section, and instead of the constant functions $r_5(t) = 1.1$ and $r_6(t) = 0.9$ over the communication interval [0, 50], we can verify that the following *linear* transfer rate functions for the sending processors satisfy conditions 1-3, and are therefore also optimal:

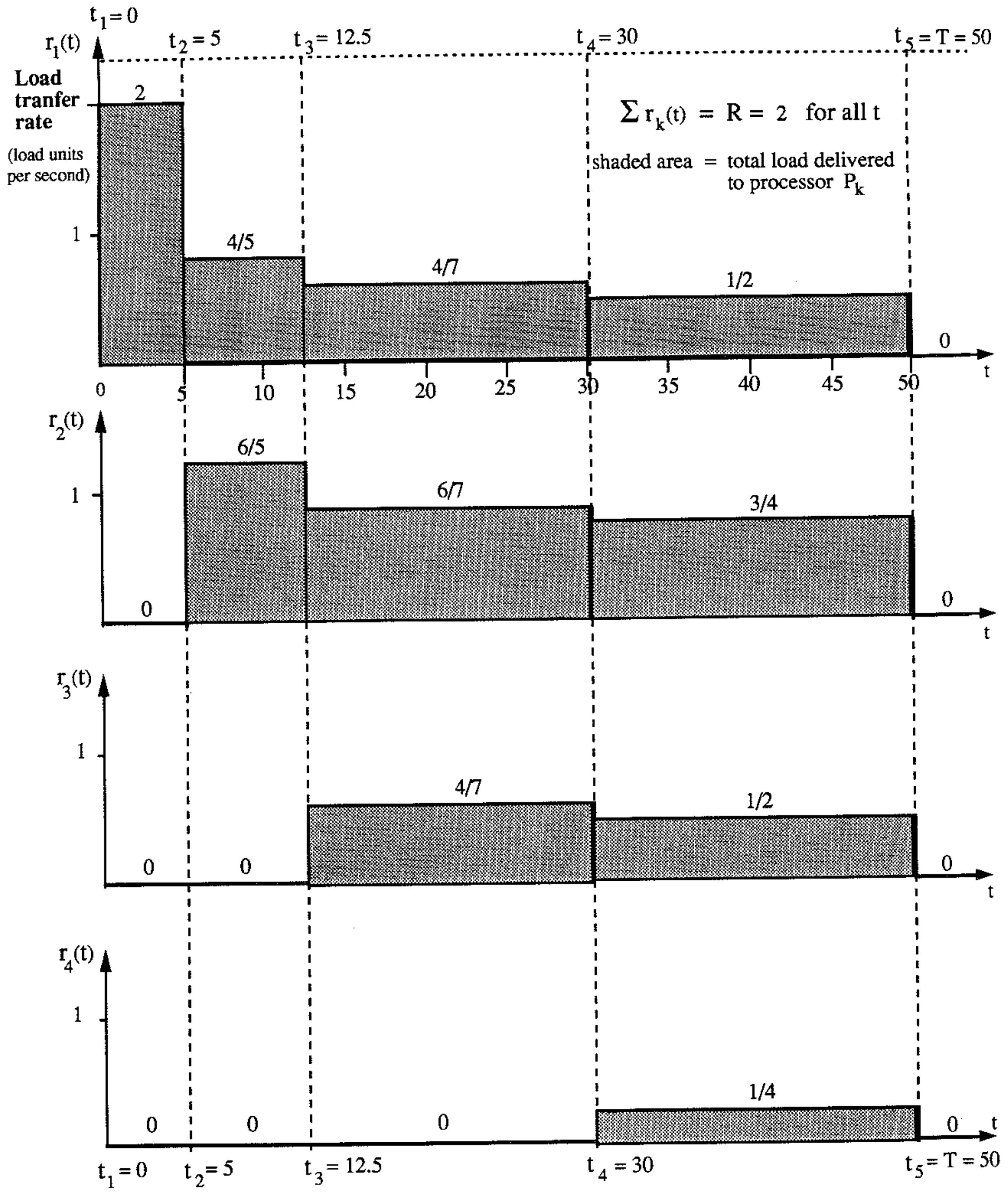


Figure 3. Minimal-bandwidth optimal load transfer schedule of the example

$$r_5(t) = 0.6 + t/50 \quad , \quad r_6(t) = 1.4 - t/50 \quad , \quad t \in [0, 50]$$
$$r_5(t) + r_6(t) = 2 \quad , \quad \int_0^{50} r_5(t)dt = 55 \quad , \quad \int_0^{50} r_6(t)dt = 45$$

One can plausibly assume that, in general, using constant rates over specified time intervals results in simpler implementations of the communication protocol.

Recall that in our formulation of the problem at hand we have characterized the communication network only by its transmission capacity or bandwidth. The optimal load balancing scheme we have derived is therefore applicable to any communication architecture and is independent of the topology or protocols of the underlying network. We only require that the network provide the minimal transfer rate R, with the capability of multiplexing the specified transmission rates from the sending processors into R, and de-multiplexing R into the specified rates for the receiving processors, as implied by the requirement of our scheme:

$$\Sigma_j r_j(t) = R = \Sigma_k r_k(t)$$

Note also that, since the divisible load is homogeneous, its elements have no distinct affinity to any particular processor, and there is no need to control the *routing* of the transfer from any specific source-processor to any specific destination-processor. In networks with communication architectures that provide frequency-division-multiplexing (FDM) protocols, the load transfer rates $r_i(t)$ can be directly implemented as frequency bandwidths of data channels. In networks with communication architectures that provide packet switching with time-division-multiplexing (TDM) protocols, the load transfer rates $r_i(t)$ can simulated by allocating to each processor a time-division channel corresponding to a nonuniform (but *cyclic*) pattern of packet time-slots on the medium. This is illustrated in Figure 4 for the receiving load transfer schedule of the example, in which we have assumed, for convenience, that each load unit is transmitted as one packet. Such a scheme can be *efficiently* implemented as a *distributed control* protocol (algorithm) with little overhead. At the time of invoking the load balancing algorithm, each processor informs all other processors of the value of its initial load x_i, which enables each processor to compute the optimal load transfer schedule and determine its status as a sending or receiving processor and the identity of the exact packets (by packet sequence number) it should

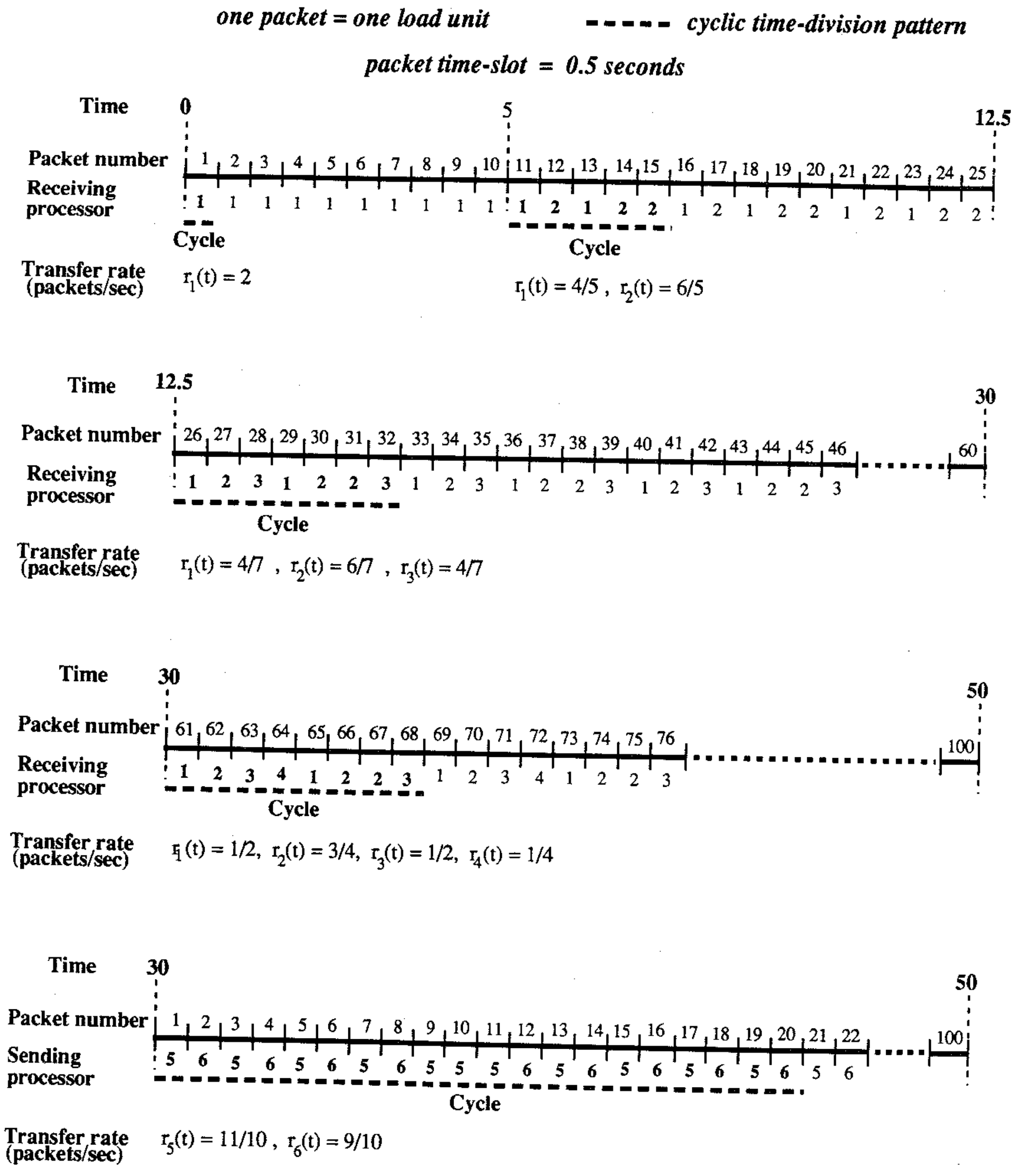

one packet = one load unit
cyclic time-division pattern
packet time-slot = 0.5 seconds
Time 0 5 12.5
Packet number 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Receiving processor 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 2 1 2 2 1 2 1 2 2
Cycle
Transfer rate (packets/sec) r1(t) = 2
r1(t) = 4/5 , r2(t) = 6/5
Time 12.5 30
Packet number 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 60
Receiving processor 1 2 3 1 2 2 3 1 2 3 1 2 2 3 1 2 3 1 2 2 3
Cycle
Transfer rate (packets/sec) r1(t) = 4/7 , r2(t) = 6/7 , r3(t) = 4/7
Time 30 50
Packet number 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 100
Receiving processor 1 2 3 4 1 2 2 3 1 2 3 4 1 2 2 3
Cycle
Transfer rate (packets/sec) r1(t) = 1/2, r2(t) = 3/4, r3(t) = 1/2, r4(t) = 1/4
Time 30 50
Packet number 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 100
Sending processor 5 6 5 6 5 6 5 6 5 5 5 6 5 6 5 6 5 6 5 6 5 6
Cycle
Transfer rate (packets/sec) r5(t) = 11/10 , r6(t) = 9/10

Figure 4. Implementing the example load transfer schedule in packet switching networks with time-division-multiplexing protocols

be sending or receiving. The total number of messages needed initially for information exchange is $(n-1)^2$, and the time complexity of computing the optimal schedule, as specified in the Theorem, is $O(n^2)$, since computing each of the n functions $r_i(t)$ requires at most n steps.

Appendix

To establish the validity of Condition 4 for the functions $r_k(t)$, we need to verify that

$$g_k(t) \equiv x_k + \int_0^t r_k(t)\,dt - s_k t \geq 0 \quad \textit{for all} \quad t \in [0, T] \qquad \text{(A1)}$$

Consider a value of t in the interval (t_q, t_{q+1}), and since $r_k(t)$ is piece-wise constant, we have

$$\int_0^t r_k(\alpha)\,d\alpha = \sum_{p=1}^{q-1} r_k(t)[t_{p+1} - t_p] + r_k(t)[t - t_q], \qquad t \in (t_q, t_{q+1})$$

Substituting the expressions for $r_k(t)$ and $[t_{p+1} - t_p]$ from (47), (48) and(45) and noting that $r_k(t) = 0$ for $p < k$, we obtain

$$\int_0^t r_k(t)\,dt = \sum_{p=1}^{q-1} s_k(T_{p+1} - T_p) + R(s_k/S_q)[t - t_q]$$

$$= s_k(T_q - T_k) + R(s_k/S_q)[t - t_q]$$

Substituting into (A1) and noting that $s_k T_k = x_k$, we obtain

$$g_k(t) = s_k T_q - s_k t + R(s_k/S_q)[t - t_q]$$

Applying the recurrence relationship in (45) repeatedly results in the following expression for t_q

$$t_q = (1/R)\left[T_q S_{q-1} - \sum_{k=1}^{q-1} x_k\right]$$

Substituting in the expression for $g_k(t)$, we obtain

$$g_k(t) = (s_k/S_q)\left[T_q S_q - S_q t + R t - T_q S_{q-1} + \sum_{k=1}^{q-1} x_k\right]$$

Noting that $T_q S_q - T_q S_{q-1} = T_q s_q = x_q$, we obtain

$$g_k(t) = (s_k/S_q)\left[Rt - S_q t + \sum_{k=1}^{q} x_k\right]$$

Recall that $T_c = T$, and from (31), (32) and (35) we have

$$T_c = D/R = (1/R)\sum_{k=1}^{m} d_k = (1/R)\sum_{k=1}^{m} [(s_k X/S) - x_k] = (1/R)\left(T S_m - \sum_{k=1}^{m} x_k\right) = T$$

$$T S_m - \sum_{k=1}^{m} x_k = TR$$

Returning to the expression of $g_k(t)$ and using the above equation to substitute for R, we obtain

$$g_k(t) = (s_k/S_qT)\left[T R t - T S_q t + T \sum_{k=1}^{q} x_k\right] = (s_k/S_qT)\left[tTS_m - t\sum_{k=1}^{m} x_k - tTS_q + T\sum_{k=1}^{q} x_k\right]$$

$$= (s_k/S_qT)\left[\ t\sum_{k=q+1}^{m} x_k - t\sum_{k=1}^{m} x_k + T\sum_{k=1}^{q} x_k\right]$$

$$= (s_k/S_qT)\left[T\sum_{k=1}^{q} x_k - t\sum_{k=1}^{q} x_k\right] = (s_k/S_qT)\left(\sum_{k=1}^{q} x_k\right)[T - t]$$

Since $t \in [0, T]$, the last expression is non-negative, hence $g_k(t) \geq 0$.

References

[1] T. G. Robertazzi, "Processor equivalence for daisy chain load sharing processors", *IEEE Trans. Aerosp. Electron. Syst.* vol. 29, no. 4, pp. 1216-1221. Oct. 1993.

[2] Y. C. Cheng and T. G. Robertazzi, "Distributed computation for a tree network with communication delays", *IEEE Trans. Aerosp. Electron. Syst.* vol. 26, May 1990.

[3] S. Bataineh and T. G. Robertazzi, "Network-oriented load sharing for a network of sensor driven processors ", *IEEE Trans. Syst. Man Cybernet.* vol. 21, no. 5, Oct. 1991.

[4] Z. Chair and P. K. Varshney, "Optimum data fusion in multiple sensor detection systems", *IEEE Trans. Aerosp. Electron. Syst.* vol. AES-22, pp. 98-101. Jan. 1986.

[5] E. Haddad, "Partitioned load allocation for minimum parallel processing execution time" *Proc. 19th International Conference on Parallel Processing,* Aug. 1989, St. Charles, Ill.

[6] R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Analysis of the effects of delays on load sharing", *IEEE Trans. Comp.* vol. C- 38, pp.1513-1525. Non 1989.

[7] H. S. Stone, "Multiprocessor shceduling with aid of network flow algorithms," *IEEE Trans. Software Eng.*, vol. SE-3, Jan. 1977, pp. 85-93.

[8] B. Indurkhya, H. S. Stone, and l. Xi-Cheng, " Optimal partitioning of randomly generated distributed programs," *IEEE Trans. Software Eng.*, vol. SE-12, March 1986, pp. 483-495.

[9] S. H. Bokhari, Assignment Problems in Parallel and Distributed Computing, Kluwer Acadmic Publishers, Boston, 1987.

[10] V. M. LO, " Heuristic algorithms for task assignment in distributed systems", *IEEE Trans. on Computers*, vol C-37, no. 11, pp. 1384 - 1397, Nov. 1988.

[11] E. Haddad, "Optimizing the parallel execution time of homogeneous random workloads" *Proc. 21st International Conference on Parallel Processing,* Aug. 1991, St. Charles, Ill.

[12] E. Haddad "Load distribution optimization in heterogeneous multiple processor systems", *Seventh Int. Parallel Processing Symposium* (WHP 93) April 1993, pp 42-47, April 1993.

[13] E. Haddad "Optimal distribution of random workloads over heterogeneous processors with loading constraints". *Proc. 1992 Int. Conf. Parallel Processing,* Aug. 1992, St. Charles, Ill.