# Optimal Sharing of Partitionable Workloads in Heterogeneous Networks of Workstations*

*Arnold L. Rosenberg*†
Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA
rsnbrg@cs.umass.edu

April 17, 2000

**Abstract**

We optimally solve two problems related to work-sharing in a heterogeneous network of workstations (NOW). In both problems, we have access to a NOW comprising $n$ workstations of differing computational powers, to assist us with a large partitionable computational workload (e.g., from a data-parallel computation). In the NOW-Rental Problem, we must complete $W$ units of work, and we wish to "rent" the NOW for as short a time as is necessary to complete that work. In the NOW-Exploitation Problem, we have access to the NOW for a fixed duration of $L$ time units, and we wish to accomplish as much work as possible during that time. Using a single mathematical formulation that encompasses both of these problems, we develop a protocol which takes a suite of $2n + 3$ parameters that characterize the computational and communicational efficiency of the NOW and determines therefrom both an amount of work to allocate to each workstation in the NOW and a schedule for transmitting that work. The resulting work-allocation plus schedule yields either an optimal value of $L$, given $W$, or an optimal value of $W$, given $L$. Thus, the protocol solves both of the motivating problems.

*A portion of this work was presented at the *Intl. Wkshp. on Cluster Computing – Technologies, Environments, and Applications (CC-TEA'2000), Las Vegas, Nev. (2000).*

†On sabbatical at the Laboratoire de Recherche en Informatique, Université de Paris-Sud, 91405 Orsay cedex France.

**Keywords**: Cluster computing, Heterogeneous NOW, Network of workstations (NOW), Optimal scheduling, Parameterized models, Work-sharing

# 1 Introduction

## 1.1 The Scheduling Problem for a "Rented" Network of Workstations

Numerous sources eloquently argue the technological and economic inevitability of an increasingly common modality of parallel computation, the use of a network of workstations (NOW) as a parallel computer; cf. [1, 14]. Sources too numerous to list describe systems that facilitate the mechanics of NOW-based computing, often via the technique of *work-sharing*[1]—the use by one workstation of idle computing cycles of another—which is our interest here. To this point, however, rather few sources have sought rigorously analyzed guidelines for scheduling broad classes of individual computations on NOWs. In the current paper, we develop a formalism within which we optimally solve the following two scheduling problems related to work-sharing in a heterogenous NOW that is available for some prespecified period of time. In both problems, we have a large, partitionable supply of computational work (such as one might encounter in a data-parallel computation), and we have access to a NOW $\mathcal{N}$ comprising $n$ workstations of differing computational powers. We seek optimal work-allocation and scheduling regimens for the following two work-sharing problems.

**The NOW-Rental Problem.** We have $W$ units of work that we must complete. We wish to "rent" the NOW $\mathcal{N}$ for as short a period of time as is necessary to complete that work.

**The NOW-Exploitation Problem.** We have access to the NOW $\mathcal{N}$ for a duration of $L$ time units, and we want to accomplish as much work as possible during that time.

## 1.2 Our Results

In Section 2, we formulate a model for work-sharing, which encompasses both of our motivating problems. Section 2.1 isolates the parameters we use to characterize the computational efficiency of the workstations in the "rented" NOW $\mathcal{N}$ and the cost of

[1] As detailed imminently, we view work-sharing as a *cooperative* enterprise. When implemented as an *adversarial* enterprise, wherein the "borrowed" workstation can be interrupted by the return of its owner, work-sharing is often known as *cycle-stealing*.

inter-workstation communication; Sections 2.2 and 2.3 formulate the generic protocol we use to implement the process of work-sharing. In Section 3, we specialize our generic protocol to obtain the *LIFO Protocol*, which optimally solves both of our motivating problems. A high-level analysis of our generic protocol, in Section 3.1, exposes the general structure of any optimal work-sharing protocol. In Section 3.2, we demonstrate that this general structure uniquely specifies the LIFO Protocol, which, consequently, is uniquely optimal. In Section 3.3, we analyze the LIFO Protocol. We show that the Protocol is *self-scheduling*, in the sense that its allocations of work to $\mathcal{N}$'s workstations, and the timing of the transmissions of that work, are determined completely by the $2n + 3$ efficiency parameters identified in Section 2.1. We determine explicit expressions for both the optimal allocations of work to $\mathcal{N}$'s workstations and for the aggregate work-output $W$ of the Protocol, as a function of $L$. We thereby see how the resulting work-allocation plus schedule yields either an optimal value of $L$, given $W$—thereby solving the NOW-Rental Problem—or an optimal value of $W$, given $L$—thereby solving the NOW-Exploitation Problem. We further see that, during sufficiently long work-sharing opportunities, the LIFO Protocol provides good parallel speedup. Finally, in Section A, we suggest, by illustration, how our generic protocol can be used to solve yet other problems related to work-sharing. We present there a self-scheduling competitor of the LIFO Protocol, which we call the *FIFO Protocol*, which attempts to incorporate a notion of *fairness* into the allocation of work. While the FIFO Protocol produces less work-output than the LIFO Protocol, it also provides good parallel speedup.

## 1.3 Related Work

There have been relatively few rigorously analyzed studies of work-allocation/scheduling in NOWs, even fewer in heterogeneous ones. Among the most intimately related to our study is [2], which develops an "auction"-based model wherein one determines that subset of workstations which—according to the source's cost model—promises the best performance on one's workload. Indeed, one can view our study as a follow-up to [2], wherein one seeks to allocate segments of one's workload to the individual workstations in the selected subset, in a way that optimizes the amount of work that can be accomplished within the period of the subset's availability. The study in [9] is concerned with far-flung assemblages of NOWs, but its results are relevant to individual NOWs also. The study's focus, however, is on providing a "fair" allocation of resources to the members of its "Co-Op," (using a ticket-based resource-allocation scheme), rather than on optimizing either parallel speedup or work-throughput. The notion of fairness used in [9] is much stronger than the notion we use in our FIFO protocol in Section A. Finally, the model studied in [4, 11] bears strong similarities to the communication-oriented portion of the model we develop in Section 2.1, but those studies focus on the problem of scheduling

collective communications within a NOW, rather than entire computations.

There have also been several noteworthy studies of scheduling algorithms that share work with one workstation at a time within a NOW. Among such sources, [3, 5, 15, 16] deal with an *adversarial* model of cycle-stealing, wherein one is in danger of losing shared work if the "rented" workstation is interrupted by the return of its owner. One finds in [3] a work-sharing strategy that accomplishes within a polylogarithmic factor of an optimal amount of work on a randomly chosen workstation of the NOW, with high probability, as long as *some* workstation in the NOW will be available long enough to complete the work. In [5, 15, 16, 17], cycle-stealing is viewed as a game against a malicious adversary who seeks to interrupt the "rented" workstation in order to minimize the work-output of a cycle-stealing opportunity. One finds in [16] guidelines that maximize, to within low-order additive terms, the *guaranteed* work-output of a work-sharing opportunity, providing that the "master" workstation knows the duration of the opportunity, plus an upper bound on the number of potential interruptions by the adversary. One finds in [15, 17] guidelines that exactly maximize the *expected* work-output of a work-sharing opportunity, providing that the "master" workstation knows the instantaneous probability of the "rented" workstations being interrupted.

The CILK system studied in [6, 7, 8] implements a (*work-stealing*) multi-threading protocol wherein idle workstations borrow load asymptotically optimally, with respect to both speed of computation and space overhead.

We do not enumerate here the many studies of computation on NOWs, which focus either on systems that enable one workstation to steal cycles from another or on specific algorithmic applications. However, we point to [12] as an exemplar of the former type of study and to [18] as an exemplar of the latter.

# 2 A Formal Notion of Work-Sharing Protocol

## 2.1 The Basic Setting

We are the owners of workstation $P_0$, and we have a large, partitionable supply of computational work to do. We have the opportunity to "rent" $n$ workstations, of possibly differing computational powers, for a predetermined *lifespan*, during which the $n$ workstations are dedicated to our workload. Our goal is to develop a scheduling protocol which utilizes the "rented" NOW optimally, in the sense of the NOW-Rental and NOW-Exploitation problems described in Section 1.1. (1) When we have a fixed supply of $W$ units of work to complete, we wish to "rent" the NOW for as short as time as is necessary to complete that work. (2) When we have access to the "rented" NOW for

a fixed lifespan of $L$ time units, we wish to get as much work done on the NOW as possible.

**Computation rates.** We measure time in terms of *work units*, which are calibrated to workstation $P_0$'s computational power; that is, by convention, $P_0$ works at unit rate. For all of the workstations in question, we denote by $\rho(P)$ the time required by workstation $P$ to perform one unit of work[2]—so that $\rho(P_0) = 1$. We label the $n$ "rented" workstations $P_1, P_2, \ldots, P_n$, in *decreasing* order of computational power—which means *increasing* order of $\rho$-value; i.e., for each $i \in \{1, 2, \ldots, n-1\}$, $\rho(P_i) \leq \rho(P_{i+1})$. For simplicity, we henceforth use the notation $\rho_i \stackrel{\text{def}}{=} \rho(P_i)$.

**Work-related quantities.** We denote by $w_i$ the number of units of work that $P_0$ allocates to workstation $P_i$, for $i \in \{0, 1, 2, \ldots, n\}$. Hearkening back to the description (in Section 1.1) of our motivating problems, then, the *aggregate work-production* of a work-sharing opportunity is

$$\text{Aggregate Work-Production} \stackrel{\text{def}}{=} W = w_0 + w_1 + \cdots + w_n.$$

**Note.** The assumed partitionability of our workload is manifest in our ability to partition our aggregate $W$ units of work into $n+1$ allocations of respective sizes $w_0, w_1, \ldots, w_n$.

We assume throughout that a unit of work produces $\delta$ units of results; we expect that $\delta \leq 1$, although we make no use of this inequality. To enhance legibility, we henceforth denote the quantity $1 + \delta$, which pervades our analyses, by $\overline{\delta}$.

**Communication rates.** We assume a *single-ported* communication model, meaning that $P_0$ can communicate with only one "rented" workstation at a time. A communication consists either of $P_0$'s sending work to some $P_i$ or receiving work from some $P_i$. Every communication begins with a setup procedure whose duration is fixed, independent of the length of the transmission. As is argued in [4], this setup is likely to be faster in workstations having faster processors and memories; therefore, we assume that each workstation $P_i$, for $i \in \{0, 1, \ldots, n\}$, has communication-setup cost $c_i$, where $c_i \leq c_j$ whenever $\rho_i \leq \rho_j$;[3] in common with [10], we shall call the $c_i$ communication *overheads*. After the fixed overhead, communication proceeds at the *uniform* rate of $\varepsilon$ time units per unit of work. In common with the $\rho_i$, the rate $\varepsilon$ is calibrated to workstation $P_0$'s computation rate.

[2]Since the "rented" workstations are dedicated to our computation, we know exactly what their computation rates are on the tasks in our computation load.

[3]Our setup costs $c_i$ correspond to the message preparation times of [4]; they are the heterogeneous analogues of the (homogeneous) communication-cost parameter of [13] and the "overhead" parameter $o$ of [10].

**Note 1.** Our assumption of a uniform communication rate $\varepsilon$ reflects our focus on a single NOW whose workstations are interconnected by a single SAN or LAN.

**Note 2.** We expect $\varepsilon$ to be tiny compared both to a unit of work and to the maximum achievable uni-processor speedup $1/\rho_i$.

**Note 3.** We expect the work allocated to each "rented" workstation to be so coarse-grained that, for all workstation indices $i$ and $j$, every $c_i$ is smaller than every quantity $\delta\epsilon w_i$.

While the setup of a communication by a workstation $P$ ties up $P$'s processor, the actual transmission of data does not; therefore, $P$ can do "work of its own" while transmitting or receiving data.

**An Aside.** The model we study here differs in two respects from our earlier studies [5, 15, 16, 17]. Since those studies focus on sharing work with (or, stealing cycles from) a single "rented" workstation, we were able to simplify the mathematical setting there by the following two conventions.

1. In those sources, we let $c$ be the overhead for an inter-workstation *interchange*; the $c$ used in those studies is, therefore, replaced by an expression of the form $c_0 + c_i$ here.
2. In those sources, we absorbed $\varepsilon$ into $\rho_i$.

In the current, multi-workstation, setting, we cannot afford these simplifying luxuries, because the analyses of our work-sharing protocols must reflect:

1. the cost of a "rented" workstation's having to wait while $P_0$ transmits work to *other* workstations before transmitting work to it,
2. the degree to which one "rented" workstation can successfully "hide" the overhead time for its communication to $P_0$ within the work- or communication-time of another "rented" workstation.

## 2.2 Orchestrating the Use of a "Rented" Workstation

The orchestration underlying our work-sharing protocol takes the general form illustrated in Fig. 1, which depicts the timeline for $P_0$'s use of a single "rented" workstation $P_i$, from the vantage point of both $P_0$ and $P_i$. The protocol proceeds as follows.

**1. Transmission of work:** $P_0$ sends $w_i$ units of work to $P_i$ in aggregate time $c_0 + \varepsilon w_i$.

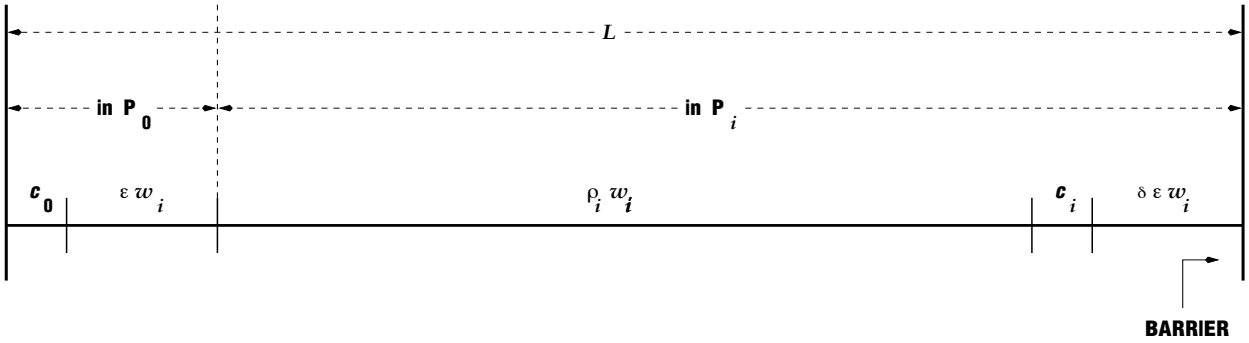


Figure 1: *The timeline for a single "rented" workstation.*

The first $c_0$ time units are used to set up the communication; the remaining $\varepsilon w_i$ time units are used to effect the transfer.

2. **The remote computation:** $P_i$ does the transmitted work in time $\rho_i w_i$.

3. **Transmission of results:** $P_i$ transmits the results of its work to $P_0$ in aggregate time $c_i + \delta\varepsilon w_i$.

   Here again, the first $c_i$ time units are used to set up the communication; the remaining $\delta\varepsilon w_i$ time units are used to effect the transfer.

**Note.** The amount of work $w_i$ must be chosen in a way that honors the lifespan $L$; that is, if $P_0$'s transmission begins at time $t$, then we must have $t+(\rho_i+\overline{\delta}\varepsilon)w_i+c_0+c_i \leq L$.

## 2.3 An Overview of a Multi-Workstation Protocol

We now extend the orchestration of Section 2.2 to a protocol for work-sharing within an $n$-workstation NOW. In common with many master-slave scheduling scenarios (cf. [2]), our protocol-formation strategy assumes the existence of some inter-workstation synchronization mechanism.

Our strategy for work-sharing is described most easily via a pair of ordinal-indexing schemes for the "rented" workstations, to complement the power-related indexing that yields the workstations' absolute names.[4] Each ordinal-indexing is a linear ordering of

[4]The "rented" workstations' multiple names, which are necessitated by $P_0$'s single-ported communication, will help us analyze our protocols. They should cause no confusion, as context will always indicate which name we are using at any given time.

the "rented" workstations, $P_1, P_2, \ldots, P_n$. The *startup indexing* specifies the order in which $P_0$ transmits work to the "rented" workstations; for this purpose, we label the $n$ workstations $P_{s_1}, P_{s_2}, \ldots, P_{s_n}$, with the understanding that $P_{s_i}$ receives work—hence, begins working—before $P_{s_{i+1}}$ does. The *finishing indexing* specifies the order in which the "rented" workstations return the results of their work to $P_0$; for this purpose, we label the workstations $P_{f_1}, P_{f_2}, \ldots, P_{f_n}$, with the understanding that $P_{f_i}$ ceases working—hence, transmits its results—before $P_{f_{i+1}}$ does. *The startup and finishing indexings are what distinguishes one protocol from another.*

1. **Transmission of work:** Starting at time 0, $P_0$ sends $w_{s_1}$ units of work to workstation $P_{s_1}$; when this transmission is completed ($c_0 + \varepsilon w_{s_1}$ time units later), $P_0$ sends $w_{s_2}$ units of work to workstation $P_{s_2}$; when this transmission is completed ($c_0 + \varepsilon w_{s_2}$ time units later), $P_0$ sends $w_{s_3}$ units of work to workstation $P_{s_3}$; and so on. The transmissions continue in this contiguous fashion until each "rented" workstation $P_{s_i}$ has been sent $w_{s_i}$ units of work, for $i \in \{1, 2, \ldots, n\}$,

   For each of these $n$ transmissions, $P_0$ devotes $c_0$ time units to set up the communication, then performs work "of its own" during the actual transmission.

2. **The computation:** Each "rented" workstation $P_i$ starts computing as soon as it receives work from $P_0$.

   After the $n$ work transmissions, $P_0$ performs work "of its own," uninterrupted until the barrier at time $L$.

3. **Transmission of results:** As soon as "rented" workstation $P_i$ completes its work (which, by definition, occurs $\rho_i w_i$ time units after it has received the work), it sends its results to $P_0$, in an aggregate time of $c_i + \delta\varepsilon w_i$ time units ($c_i$ time units for setup and $\delta\varepsilon w_i$ time units for transmission).

   For each $j \in \{2, 3, \ldots, n\}$, workstation $P_{f_j}$ "hides" the $c_i$-unit overhead for returning its results to $P_0$ within the result-transmission time of workstation $P_{f_{j-1}}$. (The assumed smallness of the $c_i$ allows such hiding.) $P_{f_j}$ begins its transmission immediately after $P_{f_{j-1}}$ completes its transmission.

**Note 1.** The work-allocations $w_i$ are determined in such a way that the "rented" workstations:

1. complete their work in the proper order (which is specified by the finishing indexing);
2. can transmit their results one after the other, with no intervening delays;
3. complete all work and all communications by the barrier at time $L$.

**Note 2.** The reader can verify that (not surprisingly) the mandated overhead-"hiding" during the transmission of results increases the aggregate work, $w_1 + w_2 + \cdots + w_n$, performed by the "rented" workstations.

**Aggregate work-production, revisited.** Since workstation $P_0$ does work "of its own" while it is transmitting data to a "rented" workstation—except during the setup phase for the transmission—the prescribed form of a multi-workstation protocol means that

**Fact 1** *Under any multi-workstation protocol, during a lifespan of $L$ time units, the "master" workstation $P_0$ completes $w_0 = L - nc_0$ units of work.*

In view of Fact 1, we can assess the work accomplished under a work-sharing protocol by focusing only on the aggregate work-production by the "rented" workstations, namely, the quantity $\widehat{W} \stackrel{\text{def}}{=} W - w_0 = w_1 + w_2 + \cdots + w_n$. *We shall, therefore, focus henceforth on $\widehat{W}$, rather than $W$.*

# 3 The LIFO Work-Sharing Protocol

## 3.1 The Motivation for the LIFO Protocol

In accord with the strategy of Section 2.3, let the "rented" workstations have the startup order $P_{s_1}, P_{s_2}, \ldots, P_{s_n}$ and the finishing order $P_{f_1}, P_{f_2}, \ldots, P_{f_n}$. We can analyze the resulting multi-workstation work-sharing protocol at a very high level, in a way that leads us toward our optimal LIFO Protocol.

Consider the $nL$ time units that our $n$ "rented" workstations will collectively have during the $L$ time units of the work-sharing opportunity. In particular, let us focus on those periods when some or all of the "rented" workstations *cannot* be performing work for $P_0$.

**Start-up delays:** Each $P_{s_i}$, where $i \in \{1, 2, \ldots, n\}$, cannot work until $P_0$ has distributed work to it and to all of its predecessors in the startup indexing. The resulting idle period for $P_{s_i}$ has length $ic_0 + \varepsilon(w_{s_1} + w_{s_2} + \cdots + w_{s_i})$.

**Mandate 1** *This reckoning mandates that workstations be sent work in* decreasing *order of their computation power, so that the faster a workstation is, the shorter startup delay it is subjected to.*

**Wrap-up delays:** Each $P_{f_i}$, where $i \in \{1, 2, \ldots, n\}$, cannot work while it and all of its successors in the finishing indexing transmit their results to $P_0$. The resulting idle period for $P_{f_i}$ has length $c_i + \delta\varepsilon(w_1 + w_2 + \cdots + w_i)$.[5]

**Mandate 2** *This reckoning mandates that workstations finish their work in* increasing *order of their computation power, so that the faster a workstation is, the shorter finishing delay it is subjected to.*

We can sum up the two mandates as follows.

**Fact 1** *One maximizes the work-output of a work-sharing protocol by setting the startup- and finishing-indices as follows: For each $i \in \{1, 2, \ldots, n\}$, $s_i = i$ and $f_i = n - i + 1$.*

The ordering of Fact 1 specifies the LIFO Protocol, which we turn to now.

## 3.2 The Detailed Specification of the LIFO Protocol

The overall structure of the LIFO Protocol is set by Fact 1 and the work-sharing strategy of Section 2.3. We turn now to the detailed specification of the Protocol.

Henceforth, we label the individual work-allocations, $w_i$, and the aggregate work-output, $\widehat{W}$, so as to identify the protocol in question; thus, we talk about $\widehat{W}^{(\mathrm{L})}$ and $w_i^{(\mathrm{L})}$ throughout this section, and about $\widehat{W}^{(\mathrm{F})}$ and $w_i^{(\mathrm{F})}$ in the Appendix.

In outline: under the LIFO Protocol one schedules, in turn: the largest possible amount of work, call it $w_1^{(\mathrm{L})}$, on the fastest workstation, namely, $P_1$; the next largest possible amount of work, call it $w_2^{(\mathrm{L})}$, on the next fastest workstation, namely, $P_2$; and so on. One determines the actual values of the $w_i^{(\mathrm{L})}$ via the following protocol-defining orchestration, which is illustrated in Fig. 2. From the figure, one sees that, under the LIFO Protocol:

- workstation $P_1$ is occupied during the entire $L$ time units of the opportunity: receiving work from $P_0$ for a duration of $c_0 + \varepsilon w_1^{(\mathrm{L})}$ time units, performing the work for a duration of $\rho_1 w_1^{(\mathrm{L})}$ time units, and returning its results to $P_0$ during the remaining $c_1 + \delta\varepsilon w_1^{(\mathrm{L})}$ time units.

[5] Recall that $P_{f_i}$ is delayed only by its own communication overhead, all subsequent ones being "hidden" in successors' transmissions.

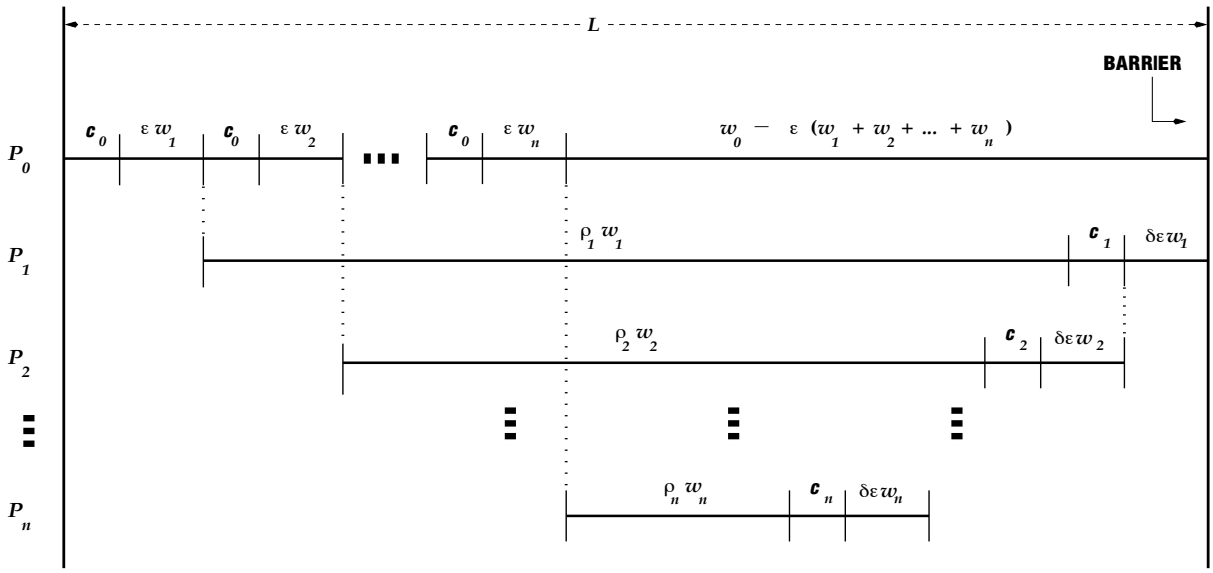


Figure 2: *The LIFO Protocol timeline for $n$ "rented" workstations. (The superscript "(L)" is elided to enhance legibility.)*

- Inductively, workstation $P_{i+1}$, where $i \in \{1, 2, \ldots, n-1\}$, is occupied for the $L - ic_0 - \overline{\delta}\varepsilon(w_1^{(\mathrm{L})} + w_2^{(\mathrm{L})} + \cdots + w_i^{(\mathrm{L})})$ time units during which workstations $P_1, P_2, \ldots, P_i$ are neither receiving work from nor returning results to $P_0$. During its allocated time, which begins at time $ic_0 + \varepsilon(w_1^{(\mathrm{L})} + w_2^{(\mathrm{L})} + \cdots + w_i^{(\mathrm{L})})$, $P_{i+1}$ receives work from $P_0$ for a duration of $c_0 + \varepsilon w_{i+1}^{(\mathrm{L})}$ time units, performs the work for a duration of $\rho_{i+1} w_{i+1}^{(\mathrm{L})}$ time units, and returns its results to $P_0$ during the remaining $c_{i+1} + \delta\varepsilon w_{i+1}^{(\mathrm{L})}$ time units.

Thus, under the LIFO Protocol, the $c_i$-unit overhead during which "rented" workstation $P_i$, where $i \in \{1, 2, \ldots, n-1\}$, prepares to return its results to $P_0$ are "hidden" within the $\delta\varepsilon w_{i+1}^{(\mathrm{L})}$ time units when (the slower) workstation $P_{i+1}$ is transmitting its results to $P_0$. $P_i$ begins to transmit its results immediately after $P_{i+1}$ completes its transmission.

## 3.3 The Analysis of the LIFO Protocol

We turn now to the detailed analysis of the LIFO Protocol in terms of work-production and parallel speedup. We shall see that the structure of the Protocol renders it self-scheduling, in the sense that the appropriate allocations of work, $\{w_i^{(\mathrm{L})} \mid 1 \leq i \leq n\}$, to

the $n$ "rented" workstations, as well as the times for all communications, are determined completely by the lifespan $L$ and the $2n+3$ performance-parameters

$$\delta,\ \varepsilon,\ \{c_i \mid 0 \le i \le n\},\ \{\rho_i \mid 1 \le i \le n\}.$$

We now determine explicit expressions for the work-allocations $w_i^{(\mathrm{L})}$, which will illustrate how the relative computation rates $\rho_i$ and overheads $c_i$ of the "rented" workstations influence the allocation of work under the Protocol.

The timeline in Fig. 2 indicates that, for each $i \in \{1, 2, \ldots, n\}$,[6]

$$(\rho_i + \overline{\delta}\varepsilon) w_i^{(\mathrm{L})} \ = \ L - ic_0 - c_i - \overline{\delta}\varepsilon \cdot \sum_{j=1}^{i-1} w_j^{(\mathrm{L})}.$$

These $n$ equations are represented most perspicuously via the matrix equation

$$\begin{pmatrix} \rho_1 + \overline{\delta}\varepsilon & 0 & \cdots & 0 & 0 \\ \overline{\delta}\varepsilon & \rho_2 + \overline{\delta}\varepsilon & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \overline{\delta}\varepsilon & \overline{\delta}\varepsilon & \cdots & \rho_{n-1} + \overline{\delta}\varepsilon & 0 \\ \overline{\delta}\varepsilon & \overline{\delta}\varepsilon & \cdots & \overline{\delta}\varepsilon & \rho_n + \overline{\delta}\varepsilon \end{pmatrix} \cdot \begin{pmatrix} w_1^{(\mathrm{L})} \\ w_2^{(\mathrm{L})} \\ \vdots \\ w_{n-1}^{(\mathrm{L})} \\ w_n^{(\mathrm{L})} \end{pmatrix} = \begin{pmatrix} L - c_0 - c_1 \\ L - 2c_0 - c_2 \\ \vdots \\ L - (n-1)c_0 - c_{n-1} \\ L - nc_0 - c_n \end{pmatrix}. \tag{3.1}$$

The system (3.1), being lower-triangular, hence nonsingular, determines the values of the $w_i^{(\mathrm{L})}$ uniquely. In the light of the timeline in Fig. 2, these values also determine uniquely the times of all communications under the Protocol.

The first equation of system (3.1) readily determines $w_1^{(\mathrm{L})}$ explicitly:[7]

$$(\rho_1 + \overline{\delta}\varepsilon) w_1^{(\mathrm{L})} \ = \ L \ - \ (c_0 + c_1). \tag{3.2}$$

Focusing next on any two consecutive equations in system (3.1), we find that

$$(\rho_i + \overline{\delta}\varepsilon) w_i^{(\mathrm{L})} \ = \ \rho_{i-1} w_{i-1}^{(\mathrm{L})} \ - \ (c_0 + c_i - c_{i-1}), \tag{3.3}$$

for each $i \in \{2, 3, \ldots, n\}$. We can unroll system (3.3) and invoke (3.2) to derive explicit,

[6] Recall that $\overline{\delta} = 1 + \delta$.

[7] To enhance legibility of our somewhat cumbersome expressions, we actually present explicit expressions for the products $(\rho_i + \overline{\delta}\varepsilon) w_1^{(\mathrm{L})}$.

albeit cumbersome, expressions for all of the $w_i^{(\mathrm{L})}$. For each $i \in \{1, 2, \ldots, n\}$,

$$\begin{aligned}(\rho_i + \overline{\delta}\varepsilon)w_i^{(\mathrm{L})} \;=\; & \left(\prod_{j=1}^{i-1} \frac{\rho_j}{\rho_j + \overline{\delta}\varepsilon}\right) L \\ & - \left(1 \;+\; \sum_{k=1}^{i-1} \prod_{j=i-k}^{i-1} \frac{\rho_j}{\rho_j + \overline{\delta}\varepsilon}\right) c_0 \;-\; c_i \\ & + \sum_{j=2}^{i-1} \left(1 \;-\; \frac{\rho_j}{\rho_j + \overline{\delta}\varepsilon}\right) \left(\prod_{k=j+1}^{i-1} \frac{\rho_k}{\rho_k + \overline{\delta}\varepsilon}\right) c_j \\ & + \left(1 \;-\; \frac{\rho_1}{\rho_1 + \overline{\delta}\varepsilon}\right) \left(\prod_{j=2}^{i-1} \frac{\rho_j}{\rho_j + \overline{\delta}\varepsilon}\right) c_1.\end{aligned} \tag{3.4}$$

The reader who will be interpreting our results numerically should have no problem with the complicated exact expressions for the $w_i^{(\mathrm{L})}$ in (3.4), especially since $n$ is likely to be fixed—and rather small. However, the reader who will be attempting to analyze the $w_i^{(\mathrm{L})}$ symbolically—as we shall—will likely appreciate somewhat simpler, albeit approximate, expressions for the $w_i^{(\mathrm{L})}$. It is not difficult to derive both a fine approximation, which may be suitable even for numerical estimation, and a coarse approximation, which may be useful for "back-of-the-envelope" calculations and for getting a general impression of how $\widehat{W}^{(\mathrm{L})}$ depends on the computation rates $\rho_i$ and the overheads $c_i$. Both approximations overestimate the true values of the $w_i^{(\mathrm{L})}$, hence of $\widehat{W}^{(\mathrm{L})}$.

One readily obtains a good fine approximation for the $w_i^{(\mathrm{L})}$ from the fact that, when $\overline{\delta}\varepsilon$ is tiny—as we expect it to be—then, for all $x$,[8]

$$\frac{x}{x + \overline{\delta}\varepsilon} \;=\; 1 - \frac{\overline{\delta}\varepsilon}{x + \overline{\delta}\varepsilon} \;\approx\; e^{-\overline{\delta}\varepsilon/(x+\overline{\delta}\varepsilon)}.$$

Applying this approximation to the exact expressions in (3.4), we obtain the following close approximate expressions for the $w_i^{(\mathrm{L})}$. Letting

$$Q[k, \ell] \;\stackrel{\text{def}}{=}\; \sum_{i=k}^{\ell} \frac{1}{\rho_i + \overline{\delta}\varepsilon},$$

we have

[8] Of course, this approximation overestimates the true value of $x/(x + \overline{\delta}\varepsilon)$.

A fine overestimate for $w_i^{(\mathrm{L})}$. For $i \in \{1, 2, \ldots, n\}$:

$$\begin{aligned}(\rho_i + \overline{\delta}\varepsilon) w_i^{(\mathrm{L})} \;\approx\; & e^{-\overline{\delta}\varepsilon Q[1,i-1]} L \\ & - \left(1 + \sum_{k=1}^{i-1} e^{-\overline{\delta}\varepsilon Q[i-k,i-1]}\right) c_0 - c_i \\ & + \sum_{j=2}^{i-1} \left(1 - \frac{\rho_j}{\rho_j + \overline{\delta}\varepsilon}\right) e^{-\overline{\delta}\varepsilon Q[j+1,i-1]} c_j \\ & + \left(1 - \frac{\rho_1}{\rho_1 + \overline{\delta}\varepsilon}\right) e^{-\overline{\delta}\varepsilon Q[2,i-1]} c_1,\end{aligned} \tag{3.5}$$

Our coarse overestimate for the $w_i^{(\mathrm{L})}$ merely ignores the tiny $\overline{\delta}\varepsilon$ terms in the denominators of all expressions of the form $(\rho_j + \overline{\delta}\varepsilon)$.

A coarse overestimate for $w_i^{(\mathrm{L})}$. For $i \in \{1, 2, \ldots, n\}$:

$$w_i^{(\mathrm{L})} \;\approx\; \frac{1}{\rho_i} L - \frac{1}{\rho_i}(i c_0 + c_i). \tag{3.6}$$

The coarse estimate (3.6) affords us an easy perspicuous overestimate for $\widehat{W}^{(\mathrm{L})}$:

$$\widehat{W}^{(\mathrm{L})} = w_1^{(\mathrm{L})} + w_2^{(\mathrm{L})} + \cdots + w_n^{(\mathrm{L})} \;\approx\; \left(\sum_{i=1}^{n} \frac{1}{\rho_i}\right) L - \left(\sum_{i=1}^{n} \frac{i}{\rho_i}\right) c_0 - \left(\sum_{i=1}^{n} \frac{1}{\rho_i} c_i\right). \tag{3.7}$$

One can obviously derive a finer, but less perspicuous, approximation via our finer estimates (3.5) of the $w_i^{(\mathrm{L})}$. Even from the coarse approximation (3.7), though, one can see that the LIFO Protocol yields good parallel speedup when $L$ is sufficiently large and $\varepsilon$ is sufficiently small.

In the next section, we specialize the analysis of the LIFO Protocol to the mathematically much simpler setting of a *homogeneous* NOW, wherein the expressions for all of the salient quantities are much more perspective.

## 3.4 The LIFO Protocol within a Homogeneous NOW

In order to get a better feeling for the parallel speedup one achieves via the LIFO Protocol, we turn now to an analysis of the Protocol within the mathematically much

simpler setting of a *homogeneous* NOW, wherein all $\rho_i \equiv 1$ and all communication overheads are equal: $c_i \equiv c$. In assessing the results of this section, one must keep in mind that this is the *least* favorable setting for the Protocol, since it minimizes the advantage of the strategy of sending the most work to the fastest workstations.

For a homogeneous NOW, the system of equations (3.1) simplifies to:

$$\begin{pmatrix} 1+\overline{\delta}\varepsilon & 0 & \cdots & 0 & 0 \\ \overline{\delta}\varepsilon & 1+\overline{\delta}\varepsilon & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \overline{\delta}\varepsilon & \overline{\delta}\varepsilon & \cdots & 1+\overline{\delta}\varepsilon & 0 \\ \overline{\delta}\varepsilon & \overline{\delta}\varepsilon & \cdots & \overline{\delta}\varepsilon & 1+\overline{\delta}\varepsilon \end{pmatrix} \cdot \begin{pmatrix} w_1^{(\mathrm{L})} \\ w_2^{(\mathrm{L})} \\ \vdots \\ w_{n-1}^{(\mathrm{L})} \\ w_n^{(\mathrm{L})} \end{pmatrix} = \begin{pmatrix} L-2c \\ L-3c \\ \vdots \\ L-nc \\ L-(n+1)c \end{pmatrix}.$$

The relations among the work-allocations $w_i^{(\mathrm{L})}$ then simplify as follows. For $i = 1$, equation (3.2) becomes

$$w_1^{(\mathrm{L})} \;=\; \frac{1}{1+\overline{\delta}\varepsilon}L \;-\; \frac{2}{1+\overline{\delta}\varepsilon}c. \tag{3.8}$$

For all other values of $i$, the relations between successive work-allocations, $w_i^{(\mathrm{L})}$ and $w_{i-1}^{(\mathrm{L})}$, simplifies from the system (3.3) to

$$w_i^{(\mathrm{L})} \;=\; \frac{1}{1+\overline{\delta}\varepsilon}w_{i-1}^{(\mathrm{L})} \;-\; \frac{1}{1+\overline{\delta}\varepsilon}c.$$

Unrolling this system of equations, we derive the following explicit analogues of (3.4). For each $i \in \{2, 3, \ldots, n\}$,

$$w_i^{(\mathrm{L})} \;=\; \left(\frac{1}{1+\overline{\delta}\varepsilon}\right)^i L \;-\; \frac{1}{\overline{\delta}\varepsilon}\left[1 \;-\; (1-\overline{\delta}\varepsilon)\left(\frac{1}{1+\overline{\delta}\varepsilon}\right)^i\right]c. \tag{3.9}$$

The structural simplicity of (3.8) and (3.9) allows us to compute an exact value for the aggregate work-output $\widehat{W}^{(\mathrm{L})}$ of the LIFO-Protocol within the homogeneous setting.

$$\widehat{W}^{(\mathrm{L})}_{(\mathrm{homo})} \;=\; \frac{1}{\overline{\delta}\varepsilon}\left[1-\left(\frac{1}{1+\overline{\delta}\varepsilon}\right)^n\right]\cdot L - \frac{1}{\overline{\delta}\varepsilon}\left[n+1-\frac{1}{\overline{\delta}\varepsilon}+\left(\frac{1}{\overline{\delta}\varepsilon}-1\right)\left(\frac{1}{1+\overline{\delta}\varepsilon}\right)^n\right]\cdot c. \tag{3.10}$$

The expression for $\widehat{W}^{(\mathrm{L})}_{(\mathrm{homo})}$ in (3.10) is certainly much simpler than its heterogeneous analogue would be, yet it is still too complex to render obvious the parallel speedup achieved under the LIFO Protocol within a homogeneous setting. Therefore, we combine (3.10) with the identity

$$\frac{1}{(1+\overline{\delta}\varepsilon)^n} \;=\; \left(1 \;-\; \frac{\overline{\delta}\varepsilon}{1+\overline{\delta}\varepsilon}\right)^n \tag{3.11}$$

to determine the following bounds on $\widehat{W}^{(\mathrm{L})}_{(\mathrm{homo})}$ (obtained by expanding the first few terms of the polynomial in (3.11)).

$$\widehat{W}^{(\mathrm{L})}_{(\mathrm{homo})} \;\geq\; \left[\frac{n}{1+\overline{\delta}\varepsilon} \;-\; (\text{l.o.t.})\right] \cdot L \;-\; \left[\frac{1-\overline{\delta}\varepsilon}{(1+\overline{\delta}\varepsilon)^2}\binom{n}{2} \;-\; (\text{l.o.t.})\right] \cdot c, \tag{3.12}$$

where the aggregate "low-order terms" in the coefficient of $L$ grows no faster than

$$\frac{\overline{\delta}\varepsilon}{(1+\overline{\delta}\varepsilon)^2}\binom{n}{2}.$$

We see from Fact 1 and (3.12) that, even in a homogeneous setting, the LIFO Protocol achieves close to $(n+1)$-fold—hence, maximal—parallel speedup via the use of $n$ "rented" workstations, as long as $L$ is sufficiently large—say, $L >> nc$—and $\varepsilon$ is sufficiently small—say, $\varepsilon < n^2/(1+\delta)$.

**Acknowledgments.** This research was supported in part by NSF Grant #CCR-97-10367. The author's sabbatical at the Lab. de Recherche en Informatique, Univ. Paris-Sud was supported in part by a Fulbright Senior Scholar Award and in part by a grant from CNRS.

# References

[1] T.E. Anderson, D.E. Culler, D.A.Patterson, and the NOW Team (1995): A case for NOW (networks of workstations). *IEEE Micro 15*, 54–64.

[2] M.J. Atallah, C.L. Black, D.C. Marinescu, H.J. Siegel, T.L. Casavant (1992): Models and algorithms for coscheduling compute-intensive tasks on a network of workstations. *J. Parallel Distr. Comput. 16*, 319–327.

[3] B. Awerbuch, Y. Azar, A. Fiat, F.T. Leighton (1996): Making commitments in the face of uncertainty: how to pick a winner almost every time. *28th ACM Symp. on Theory of Computing*, 519–530.

[4] M. Banikazemi and D.K. Panda (2000): Efficient collective communication on heterogeneous networks of workstations. Tech. Rpt., Ohio State Univ.

[5] S.N. Bhatt, F.R.K. Chung, F.T. Leighton, and A.L. Rosenberg (1997): On optimal strategies for cycle-stealing in networks of workstations. *IEEE Trans. Comp. 46*, 545–557.

[6] R. Blumofe and C.E. Leiserson (1993): Space-efficient scheduling of multithreaded computations. *25th ACM Symp. on Theory of Computing*, 362–371.

[7] R. Blumofe and C.E. Leiserson (1994): Scheduling multithreaded computations by work stealing. *35th IEEE Symp. on Foundations of Computer Science*, 356–368.

[8] R. Blumofe and D.S. Park (1994): Scheduling large-scale parallel computations on networks of workstations. *3rd Intl. Symp. on High-Performance Distributed Computing*, 96–105.

[9] W. Cirne and K. Marzullo (1999): The Computational Co-Op: gathering clusters into a metacomputer. *13th Intl. Parallel Processing Symp.*, 160–166.

[10] D. Culler, R.M. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, T. von Eicken (1996): LogP: towards a realistic model of parallel computation. *C. ACM 39*, 78–85.

[11] R. Kesavan, K. Bondalapati, D.K. Panda (1996): Multicast on irregular switch-based networks with wormhole routing. *3rd Intl. Symp. on High-Performance Computer Architecture.*

[12] M. Litzkow, M. Livny, M.W. Mutka (1988): Condor – A hunter of idle workstations. *8th Intl. Conf. on Distr. Computing Systs.*, 104–111.

[13] C.H. Papadimitriou and M. Yannakakis (1990): Towards an architecture-independent analysis of parallel algorithms. *SIAM J. Comput. 19*, 322–328.

[14] G.F. Pfister (1995): *In Search of Clusters*. Prentice-Hall.

[15] A.L. Rosenberg (1999): Guidelines for data-parallel cycle-stealing in networks of workstations, I: on maximizing expected output. *J. Parallel and Distr. Comput. 59*, 31–53.

[16] A.L. Rosenberg (2000): Guidelines for data-parallel cycle-stealing in networks of workstations, II: on maximizing guaranteed output. *Intl. J. Foundations of Computer Science 11*, to appear.

[17] A.L. Rosenberg (2000): Optimal schedules for data-parallel cycle-stealing in networks of workstations. *12th ACM Symp. on Parallel Algorithms and Architectures.*

[18] S.W. White and D.C. Torney (1993): Use of a workstation cluster for the physical mapping of chromosomes. *SIAM NEWS*, March, 1993, 14–17.

# A The FIFO Work-Sharing Protocol

While the LIFO Protocol is optimal in work-production within our model, it is completely "user-centric," in the sense that it occupies the most desirable "rented" workstation for the entire lifespan of the work-sharing opportunity, the next most desirable for just a bit less time, and so on. One can devise computationally good protocols which have at least a rudimentary notion of fairness built in, which also have good work-production, hence, achieve rather good parallel speedup. This section is devoted to presenting one such, the FIFO Protocol. The fairness in the FIFO Protocol is predicated on the relative computational powers of the "rented" workstations and is just one step in the direction of the stringent notions of fairness that are common in many environments where precious resources are contended for. The notion here, for instance, is not nearly as strict as the corresponding notion in [9], which implements fairness via the external mechanism of tickets which mandate how much work a workstation should get. However, the FIFO Protocol may be more appropriate than the LIFO Protocol in an environment wherein access to workstations is highly contentious, so that one cannot request too many cycles from any particular workstation.

## A.1 The Details of the FIFO Protocol

The FIFO Protocol has the "rented" workstations terminate their work in the same order as they begin their work; i.e., for all $i$, $s_i = f_i = i$. This results in the following orchestration, which defines the protocol and determines the values of the $w_i^{(\mathrm{F})}$; cf. Fig. 3. From the figure, one sees that, under the FIFO Protocol:

- workstation $P_1$ is occupied for $c_0 + c_1 + (\rho_1 + \overline{\delta}\varepsilon)w_1^{(\mathrm{F})}$ time units, receiving work from $P_0$ for a duration of $c_0 + \varepsilon w_1^{(\mathrm{F})}$ time units, performing the work for a duration of $\rho_1 w_1^{(\mathrm{F})}$ time units, and returning its results to $P_0$ during the remaining $c + \delta\varepsilon w_1^{(\mathrm{F})}$ time units.
- Inductively, workstation $P_{i+1}$, where $i \in \{1, 2, \ldots, n-1\}$, is occupied for the $c_0 + c_{i+1} + (\rho_{i+1} + \overline{\delta}\varepsilon)w_{i+1}^{(\mathrm{F})}$ time units beginning at time $ic_0 + \varepsilon(w_1^{(\mathrm{F})} + w_2^{(\mathrm{F})} + \cdots + w_i^{(\mathrm{F})})$. During its allotted time, $P_{i+1}$ receives work from $P_0$ for a duration of $c_0 + \varepsilon w_{i+1}^{(\mathrm{F})}$ time units, performs the work for a duration of $\rho_{i+1} w_{i+1}^{(\mathrm{F})}$ time units, and returns results to $P_0$ during the remaining $c_{i+1} + \delta\varepsilon w_{i+1}^{(\mathrm{F})}$ time units.

Thus, under the FIFO Protocol, each workstation $P_i$, where $i \in \{2, 3, \ldots, n\}$, "hides" the $c_i$-unit overhead for returning its results to $P_0$ within the $\delta\varepsilon w_{i-1}^{(\mathrm{F})}$ time units when work-

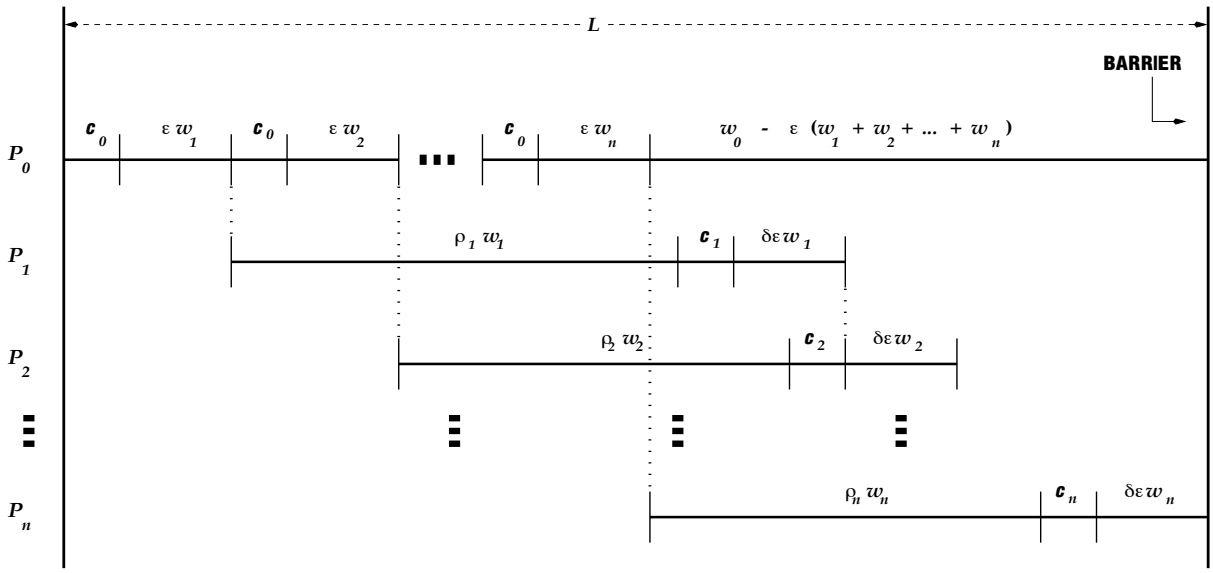


Figure 3: *The FIFO-protocol timeline for $n$ "rented" workstations. (The superscript "(F)" is elided to enhance legibility.)*

station $P_{i-1}$ is returning its results to $P_0$. $P_i$ then begins its transmission immediately after $P_{i-1}$ completes its transmission.

## A.2 The FIFO Protocol within a Homogeneous Setting

In order to suggest strongly that the FIFO Protocol achieves good parallel speedup—albeit not as good as the LIFO Protocol does (cf. (3.12)—we analyze its performance in a *homogeneous* NOW, wherein all $\rho_i \equiv 1$ and all $c_i \equiv c$. Note that this setting is the *most* favorable for the FIFO Protocol, for much the same reason that it is the *least* favorable for the LIFO Protocol.

One sees from the timeline in Fig. 3 that the interelationships among the FIFO Protocol's work-allocations in a homogeneous setting are given by the matrix equation:

$$\begin{pmatrix} 1+\overline{\delta}\varepsilon & \delta\varepsilon & \cdots & \delta\varepsilon & \delta\varepsilon \\ \varepsilon & 1+\overline{\delta}\varepsilon & \cdots & \delta\varepsilon & \delta\varepsilon \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \varepsilon & \varepsilon & \cdots & 1+\overline{\delta}\varepsilon & \delta\varepsilon \\ \varepsilon & \varepsilon & \cdots & \varepsilon & 1+\overline{\delta}\varepsilon \end{pmatrix} \cdot \begin{pmatrix} w_1^{(\mathrm{F})} \\ w_2^{(\mathrm{F})} \\ \vdots \\ w_{n-1}^{(\mathrm{F})} \\ w_n^{(\mathrm{F})} \end{pmatrix} = \begin{pmatrix} L-2c \\ L-3c \\ \vdots \\ L-nc \\ L-(n+1)c \end{pmatrix}. \quad \text{(A.1)}$$

The nonsingularity of the coefficient matrix in (A.1) shows that the FIFO Protocol is self-scheduling in the same sense that the LIFO Protocol is. By considering adjacent pairs of equations from system (A.1), one derives the following expressions for all but the first of the FIFO Protocol's work-allocations. For $i \in \{2, 3, \ldots, n\}$:

$$w_i^{(\mathrm{F})} \;=\; \left(\frac{1+\delta\varepsilon}{1+\varepsilon}\right)^{i-1} w_1^{(\mathrm{F})} \;-\; \frac{1}{(1-\delta)\varepsilon}\left[1 - \left(\frac{1+\delta\varepsilon}{1+\varepsilon}\right)^{i-1}\right] c.$$

Summing these quantities, we find that

$$\begin{aligned} w_2^{(\mathrm{F})} + \cdots + w_n^{(\mathrm{F})} \;=\; & \frac{1+\delta\varepsilon}{(1-\delta)\varepsilon}\left[1 \;-\; \left(\frac{1+\delta\varepsilon}{1+\varepsilon}\right)^{n-1}\right] w_1^{(\mathrm{F})} \\ & - \left[\frac{n-1}{(1-\delta)\varepsilon} \;-\; \frac{1+\delta\varepsilon}{(1-\delta)^2\varepsilon^2}\left\{1 \;-\; \left(\frac{1+\delta\varepsilon}{1+\varepsilon}\right)^{n-1}\right\}\right] c. \end{aligned} \tag{A.2}$$

Finally, the first equation of system (A.1) yields the following implicit expression for $w_1^{(\mathrm{F})}$.

$$w_1^{(\mathrm{F})} \;=\; \frac{1}{1+\overline{\delta}\varepsilon} \cdot L \;-\; \frac{2}{1+\overline{\delta}\varepsilon} \cdot c \;-\; \frac{\delta\varepsilon}{1+\overline{\delta}\varepsilon}\left(w_2^{(\mathrm{F})} + \cdots + w_n^{(\mathrm{F})}\right). \tag{A.3}$$

Completing the analysis of the FIFO Protocol to the point of obtaining explicit expressions for all quantities—notably, the work-allocations $w_i^{(\mathrm{F})}$ and the aggregate work-output $\widehat{W}^{(F)}$—requires extensive tedious and unenlightening computation, beginning with solving the simultaneous system of two equations in two unknowns in (A.2) and (A.3). We shall not burden the reader with these clerical details, proceeding instead directly to the "bottom line," namely the analogue for the FIFO Protocol of the bounds (3.12).

$$\widehat{W}_{(\mathrm{homo})}^{(\mathrm{F})} \;\geq\; \left[\frac{(1+\delta\varepsilon)n}{(1+\varepsilon)(1+\overline{\delta}\varepsilon)} \;-\; (\text{l.o.t.})\right] \cdot L \;-\; \left[\frac{1+\varepsilon}{(1+\delta\varepsilon)(1+\overline{\delta}\varepsilon)^2}\binom{n}{2} \;-\; (\text{l.o.t.})\right] \cdot c, \tag{A.4}$$

where the aggregate "low-order terms" in the coefficient of $L$ grows no faster than

$$\frac{(1-\delta)\varepsilon}{(1+\delta\varepsilon)^2}\binom{n}{2}.$$

Despite its rather good parallel speedup in the homogeneous setting, as exposed by (A.4), the FIFO Protocol does not compete with the LIFO Protocol even in this most favorable match-up.