Taxonomy of dynamic task scheduling schemes in distributed computing systems

H.G. Rotithor

Indexing terms: Distributed computing systems, Taxonomy, Dynamic Task scheduling, State estimation, Decision making

Abstract: System state estimation and decision making are the two major components of dynamic task scheduling in a distributed computing system. Combinations of solutions to each individual component constitute solutions to the dynamic task scheduling problem. It is important to consider a solution to the state estimation problem separate from a solution to the decision making problem to understand the similarities and differences between different solutions to dynamic task scheduling. Also, a solution to the state estimation problem has a significant impact on the scalability of a task scheduling solution in large scale distributed systems. In this paper we present a taxonomy of dynamic task scheduling schemes that is synthesised by treating state estimation and decision making as orthogonal problems. Solutions to estimation and decision making are analysed in detail and the resulting solution space of dynamic task scheduling is clearly shown. The proposed taxonomy is regular, easily understood, compact, and its wide applicability is demonstrated by means of examples that encompass solutions proposed in the literature. The taxonomy illustrates possible solutions that have not been evaluated and those solutions that may have potential in future research.

1 Introduction

Task scheduling in distributed computing systems (DCS) consists of 'local' scheduling and 'global' scheduling. A DCS refers to a loosely coupled collection of processing elements (PEs). Local scheduling involves assignment of tasks to time-slices of a single PE whereas global scheduling involves deciding where a task should be scheduled for execution [1]. In this paper task scheduling refers to global task scheduling. Advantages of task scheduling are well known and have been discussed in most of the literature pertaining to the subject [2, 3]. Task scheduling schemes are broadly classified as 'static' [4–7] or 'dynamic' [8–11]. Static schemes use enumerative [12], graph theoretic [13, 14], mathematical programming [15] and queuing theoretic [16–19] approaches to generate either optimal or suboptimal solutions. Static schemes use *a priori* knowledge about task behaviour and do not obtain information about the dynamically changing state of the system before making a task scheduling decision. Dynamic schemes make few assumptions about task characteristics and obtain information of the system state before making a task scheduling decision [20–22]. Dynamic schemes may be adaptive or non-adaptive [1].

Dynamic task scheduling may be implemented as 'dynamic task sharing' or 'dynamic task balancing' [23, 24]. Dynamic task scheduling schemes have been applied extensively in experimental distributed systems and have shown significant potential for performance improvement [8–11, 25, 26]. Following the present trend of price reduction in hardware components, it is anticipated that large scale DCS incorporating dynamic task scheduling will rapidly evolve in the future. The major thrust of research in dynamic task scheduling will be centred on providing solutions that are scalable to large scale DCS [8, 27]. To clearly understand current research trends in dynamic task scheduling and anticipate future trends, it is important to analyse carefully the characteristics of solutions proposed in the literature and classify them based on those characteristics. It is the objective of this paper to present a classification scheme for dynamic task scheduling. The taxonomy presented facilitates observing similarities and differences between dynamic task scheduling schemes based on the characteristics of policies used for system state estimation and decision making. A taxonomy based on a detailed analysis of estimation and decision making policies provides a lucid insight into the solution space of dynamic task scheduling because estimation and decision making are two major components of this process. A need for a detailed analysis of solutions to estimation and decision making becomes evident after studying the solutions proposed in the literature which are either monolithic in nature or combine components of estimation and decision making in an ambiguous manner. The taxonomy is applied to differentiate between solutions reported in the literature and it provides a common framework within which new solutions to dynamic task scheduling can be studied. Solutions to static task scheduling have been classified by Casavant *et al.* [1] and are not considered here.

2 Motivation for the proposed taxonomy

Dynamic task scheduling has two major components: 'system state estimation' and 'decision making'. System state estimation involves disseminating state information throughout the DCS and constructing an estimate of the

Paper 9630E (C1), first received 11th September 1992 and in revised form 15th March 1993

The author is with the Department of Electrical and Computer Engineering Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609

system state based on the states of individual PEs. Decision making involves choosing a PE to schedule a task based on the estimate of the system state. A solution to dynamic task scheduling should thus provide a solution to estimation and a solution to decision making. Although decision making involves use of the system state estimate, it is beneficial to treat solutions to the two components orthogonal to clearly understand the possible solutions to dynamic task scheduling. A careful analysis of solutions to dynamic task scheduling proposed in the literature reveals in many cases that the solutions do not clearly distinguish between estimation and decision making and a solution to the estimation problem is often implicit in the solution to dynamic task scheduling (overall solution). In some cases the distinction is maintained and state estimation is called 'information policy' [20, 28, 29] or 'information exchange' [30] and decision making is called 'placement policy' [20, 29], 'location policy' [28], or 'decision making' [30]. Even in those cases where an attempt is made to maintain a distinction, components of estimation and decision making are not clearly identified and a monolithic solution is provided to estimation and decision making.

A problem with a monolithic solution, where important components of estimation and decision making are not clearly identified, is that the solutions that differ with respect to these components appear identical. For example, demand driven information policies are classified as either sender, receiver, or symmetrically initiated [29]. This is an example where the distinction is unclear; a demand driven information policy indicates when *information* dissemination begins, whereas a sender, receiver, or symmetrically initiated policy relates to how *decision* making is done. Similarly, the polling location policy described in Reference 29 combines state estimation with decision making into a monolithic solution. Relating estimation and decision making as projected above blurs the distinction between them, makes it difficult to identify components of estimation and decision making, and limits their possible combinations that are usable as discussed in our analysis later. In this paper we maintain a clear distinction between estimation and decision making and provide a set of regular criteria that high-light the important components of solutions to estimation and decision making. A detailed analysis based on the selected criteria clearly separates the solution spaces of estimation and decision making and provides a better view of solutions to dynamic task scheduling. Our approach facilitates a study of solutions proposed in the literature under one framework and an understanding of their similarities and differences. This approach also illustrates some solutions that have not been evaluated and that may have potential for providing high performance in large scale DCS. Such an analysis of estimation and decision making policies has not been reported in the literature and is important for the following reasons.

First, it allows one to see different components of a dynamic task scheduling algorithm clearly and facilitates a detailed intercomparison of different algorithms as against a comparison based on a monolithic solution. Second, Zhou [31] and Ramamritham [32] have shown that the estimation performance has a significant impact on the overall performance due to the overhead incurred. Zhou [8] has also observed that in a large scale DCS, the scalability of a task scheduling algorithm depends significantly on the solution to the estimation problem. The importance of considering the estimation problem in detail is thus evident, and its significance is expected to increase further in future research as large scale DCS evolve. Third, it motivates choosing appropriate metrics to evaluate performance of estimation and decision making so that the effect of each component on the metric that measures overall performance can be clearly seen [33]. This can provide an important feedback that is useful in improving those components of a dynamic task scheduling algorithm that affect the overall performance most. Using a monolithic solution for estimation, decision making, or dynamic task scheduling may not provide such an insight.

Before describing the taxonomy it is imperative to put our work in perspective with respect to related work reported in the literature. Wang *et al.* [34] have proposed a taxonomy that classifies task scheduling schemes in one dimension as source initiative or server initiative and in the other dimension based on the level of information dependency in scheduling. However, this taxonomy does not consider estimation and decision making separately. Components of estimation are implicit in the scheduling policy, the number of solutions proposed is limited, and the taxonomy does not incorporate techniques that combine source and server initiated policies that have been implemented since the paper was published. An elaborate taxonomy of scheduling in general purpose DCS was proposed by Casavant [1]. This taxonomy includes static scheduling as well, thus has a much broader scope than the scope of this paper. However, the taxonomy presented for dynamic scheduling portion does not consider estimation and decision making separately in detail. We provide a better and a more detailed view of estimation and decision making policies using a well defined and a regular set of criteria. The criteria used for analysing estimation and decision making policies are represented by a set of questions that a designer would answer while designing the policies. It is not possible to include every issue that a designer would have to address as a possible criterion in the analysis because such an approach will result in the analysis getting too specialised or unwieldy and cloud up the important issues. Two important considerations are used in selecting the criteria. First, the selected criteria are relevant to all policies and any significant differences between policies can be isolated using these criteria. Second, the selected criteria address either directly or indirectly important goals that are common to the design of a large number of policies such that various policies incorporating different tradeoffs with respect to these goals can be differentiated. For example, system responsiveness and robustness goals are affected by choosing either centralised, decentralised, or hybrid control, communication overhead depends on the choice of components of an estimation policy whereas computational overhead depends on the choice of a decision making transformation as discussed later.

3 A taxonomy for dynamic task scheduling

First, solutions to estimation are examined in detail.

3.1 State estimation

When the PEs responsible for estimation and the PEs responsible for decision making are different, a state estimate that is constructed is sent to the PEs that make decisions. The state information disseminated depends on a specific implementation and different type of dissemination schemes may be used for different components of the state information based on the significance of those

components. Examples of state information that may be disseminated are: time-averaged task queue length [20, 35], memory surplus at a PE [36], computing capacity surplus at a PE [32], operational status of a PE, etc. CPU task queue length has been shown to be an effective candidate for load index [37]. It should be noted that the end user of system state information is the decision making policy which computes a task scheduling decision based on the available state information. Thus, a specification of the nature of state information and a specification of the type of transformations performed on the state information to compute a decision should be designed to suit the decision making policy. An estimation policy merely provides an infrastructure for exchanging state information and constructing an estimate of the system state. The computation involved in constructing an estimate of the system state from the state information of individual PEs may range from simply creating a vector of individual state of all PEs [8, 22] to computing a complex function of individual components of the state information [36, 38].

Solutions to state estimation are classified based on answers to the following questions:

(i) Which PEs are responsible for collecting state information and constructing an estimate of the system state?

(ii) During any state information exchange how many PEs are involved?

(iii) How does an individual PE choose to disseminate local state information?

(iv) At what instant does an individual PE choose to initiate information dissemination?

Note that it is possible to pose many more questions in order to capture finer details of an estimation policy. The answers to questions above contain enough detail to highlight the important characteristics of the possible solutions. Various possibilities are discussed in detail below and are shown in Fig. 1.

Answers to the first question point to various logical organisations of PEs that are possible for the purpose of estimation. Three types of organisations commonly used are: 'centralised', 'decentralised', and 'hybrid'.

Centralised [9, 39–42]: A central agent collects state information and constructs an estimate of the system state. The central agent may be a physical PE [39, 41] or a globally shared file that is accessed and updated by all PEs [9, 43]. This organisation has an advantage that it incurs low overhead during estimation [31]. The disadvantages are a poor responsiveness of a central resource in a large scale system resulting in poor scalability and the failure-prone nature of a central resource.

Decentralised [20, 25, 28, 44, 45]: In a decentralised organisation each PE is responsible for collecting state information and obtaining an estimate of the system state. This type of organisation has higher availability in the presence of failures, but it can potentially incur large

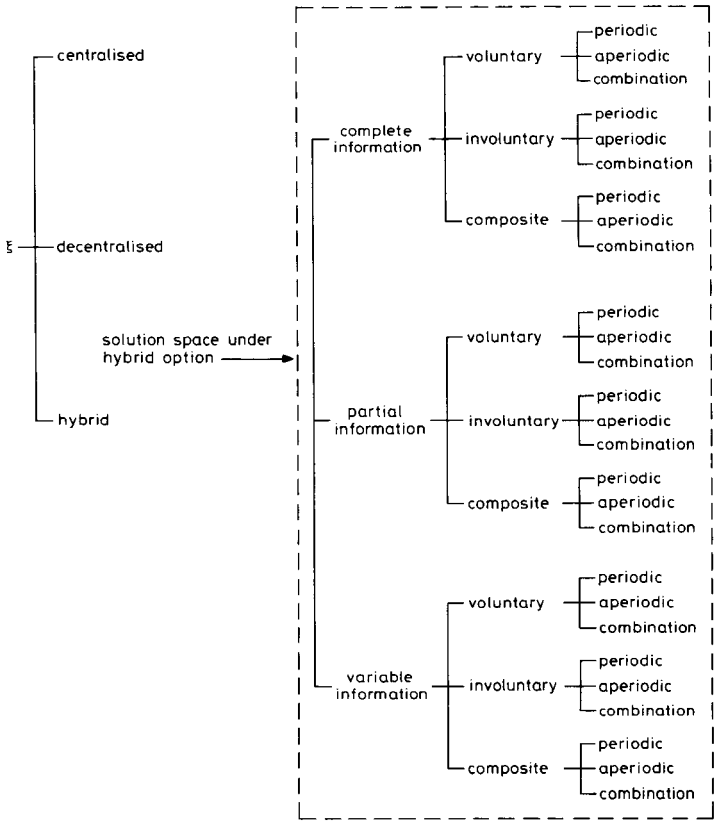

Fig. 1 *Solution space of estimation schemes*

Note: The solution space inside the box (dashed) is repeated under centralised and decentralised options

overhead to maintain accurate state infromation and therefore is not easily scalable to a large scale DCS [20]. Decentralised organisation has been considered for a number of small to medium size DCS.

Hybrid [8, 22]: A hybrid organisation combines centralised and decentralised organisations, inherits their properties, and attempts to extract advantages of both organisations. A hybrid organisation may be implemented in two ways. In the first case, PEs are divided into clusters and state information is exchanged within and between clusters. Membership within clusters may be decided by various factors such as network proximity, type of service performed by PEs etc. An example of this approach is Utopia [8], which contains physical clusters and virtual clusters. Virtual clusters contain powerful PEs with large resource capacities. State information within clusters is exchanged in a centralised manner whereas between clusters it is exchanged in a decentralised manner. In the second case, a part of the state information is exchanged in a decentralised manner and the remaining state information is exchanged in a centralised manner. Stankovic's [22] scheme is an example of this approach. In this scheme, all PEs exchange their estimates with neighbors in a decentralised manner and each PE sends its local state to a central monitor in a centralised manner. A cluster-based hybrid scheme has shown potential for providing the desired performance in large scale DCS [8].

Answers to the second question decide how many PEs are involved in exchanging state information. Three broad categories that can be used to answer this question are: *complete* information exchange, *partial* information exchange, and *variable* information exchange. When all PEs are involved in every exchange of state information it is called a 'complete information exchange'. When only a subset of all the PEs is involved in every exchange of state information it is called a 'partial information exchange'. Actual size of the subset and members of the subset depend on specific implementations. For example the members of the subset may be chosen based on network proximity and capacity considerations or at random. Shin *et al.* [45] have proposed an algorithm to specify members of a 'buddy set' such that accurate state information is maintained and instability is avoided. A 'variable information exchange' scheme combines complete information exchange and partial information exchange. Such a scheme may be implemented in two ways. In the first method, a part of the information is exchanged involving all PEs in the system whereas the remaining information is exchanged involving only a subset of PEs in the system. Stankovic's [22] scheme is an example of this method where state estimates are exchanged only between neighbours (partial) whereas the local state information is sent by all PEs to a central monitor (complete) for the purpose of updating the likelihood function and computing optimal strategies. Another method of implementing a variable information exchange scheme is to *dynamically* vary the number of PEs involved in information exchange during execution, based on some heuristic, with the intention of optimally utilising the communication network. The number of PEs chosen may vary depending on the system state [29]. Shivaratri *et al.* [29, 46] have described a variable information exchange policy that adapts to the system state by appropriately updating and accessing senders, receivers, and OK lists which contain PE names with whom state information is exchanged.

Three possibilities for answering the third question are: *voluntary* dissemination, *involuntary* dissemination, and *composite* dissemination. In a voluntary scheme, a PE voluntarily sends out state information to other PEs when certain conditions are met. Examples of these conditions are described later. In an involuntary scheme, a PE sends out its state information only when it is requested by another PE. A PE may request information from other PEs when certain conditions are met. The involuntary scheme has been called as 'information exchange on demand' [20] or 'query based' [47]. A composite scheme uses a combination of voluntary and involuntary schemes, where a PE sends information voluntarily when certain conditions are met and requests information when some other conditions are met. An advantage of the voluntary scheme is that those PEs which receive state information during the exchange always have a fairly accurate estimate of the system state which may be used for making a decision with a minimal delay; a factor that is important for real time systems [45]. A disadvantage of the voluntary scheme is that the information may be sent to PEs which may not have a use for all the information received. This is true when a PE receives successive state information updates from another PE and does not make any decision between those updates. The involuntary scheme has an advantage that the information is sent only to that PE which needs it, but this scheme involves a delay between the instant a request is made and the instant a reply is received, therefore any decision that needs to be made based on the information received may be delayed. A composite scheme attempts to extract advantages of the two schemes. Examples of the composite scheme are: 'bidding' and 'flexible' algorithms described by Ramamritham [32] and the 'microeconomic' algorithms discussed by Fergusson *et al.* [48]. In Ramamritham's algorithms, surplus information is broadcast periodically to a subset of PEs voluntarily whereas bids are obtained from a subset of PEs (involuntary) when a task is to be scheduled. It should be noted that bidding uses involuntary information dissemination. In the microeconomic algorithms, price change information is broadcast voluntarily whereas the Dutch auction algorithm obtains bids when a resource becomes idle (involuntary).

The final question involves deciding when information dissemination begins. Information dissemination involves either voluntarily sending out state information or involuntarily requesting state information. Three techniques commonly used are: *periodic*, *aperiodic*, and *combination*. In the 'periodic' case, information dissemination repeats after a prespecified interval. In the 'aperiodic' case, information dissemination begins when certain conditions have been met. Examples of conditions that might be met are: when a PE makes specific state transitions or when a PE enters a specific state. For example a PE may voluntarily disseminate state information when it makes a transition to a lightly loaded or a heavily loaded state. Similarly, a PE may request state information when it is in a lightly loaded state or a heavily loaded state before making a task scheduling decision. The exact specification is algorithm dependent. A 'combination' scheme disseminates a part of the state information periodically and a part of the state information aperiodically. Examples of a combination scheme are the 'bidding' and the 'flexible' schemes, proposed by Ramamritham [32], where each PE periodically broadcasts its surplus information and requests bids aperiodically, that is, only when a task is to be scheduled. For periodic dissemination

[22], it is usually difficult to decide a period *a priori*, because the system dynamics is not always predictable. Too small a period causes high overhead and unnecessary information being exchanged whereas too large a period may cause inaccurate estimates. Aperiodic schemes are relatively easier to specify but they too can incur high overhead under certain conditions. If the PEs change state often, a voluntary aperiodic scheme that disseminates all state changes when state transitions occur, can potentially incur high overhead because information about all state changes may not be useful to all the PEs. Involuntary aperiodic scheme that obtains information only at decision making time can circumvent this problem by avoiding redundant exchanges of state information. However, if PEs continue in a decision making state for a long time and consequently not change state very often, an involuntary scheme can incur high overhead because the system state between successive requests may not have changed. The overhead problems with aperiodic dissemination can be handled by use of intelligent heuristics [45, 46, 49]. It should be noted that the scheduler or the load information module may be invoked periodically, but information dissemination may still be aperiodic if it is initiated after specific conditions have been met.

A solution to the state estimation problem, denoted ξ, is given as:

$$\xi \in \{(\text{Centralised, Decentralised, Hybrid}) \times (\text{Complete, Partial, Variable}) \times (\text{Voluntary, Involuntary, Composite}) \times (\text{Periodic, Aperiodic, Combination})\} \quad (1)$$

Where ξ contains one member from each set above and '$\times$' denotes a Cartesian product. This formulation results in 81 possible schemes for estimation.

3.2 Decision making

Most task scheduling schemes use a threshold 'transfer policy' to initiate decision making [21, 28, 29]. When the resource requirements of tasks at a PE cross a threshold, the PE enters a decision making state and makes a decision to send or request tasks in that state. The selected threshold and/or the number of tasks transferred when a scheduling decision is made decide whether task sharing or task balancing is achieved. Solutions to decision making are classified based on answers to the following questions:

(i) Which PEs are responsible for computing and executing task scheduling decisions?

(ii) Whether a decision maker is a source or sink for tasks in the scheduling process?

(iii) What type of transformations are performed on the state information that affect computation of a task scheduling decision?

Various possibilities are discussed in detail below and are shown in Figs. 2 and 3.

The first question is answered along the line of the estimation problem. Three possibilities are: *centralised*, *decentralised*, and *hybrid*. In the 'centralised' case, a single PE makes task scheduling decisions for all PEs in the system [21, 39, 42]. In the 'decentralised' case, each PE is responsible for computing and executing its own decisions [20, 25, 28, 44, 45]. In the hybrid case, decisions within a cluster are made by a PE designated as a manager as described by Zhou [8] and Ahmad [50], that is, intracluster decisions are centralised at the cluster manager. Decisions made in one cluster do not directly depend on those made in the other clusters; thus intercluster decision making is decentralised. The only way

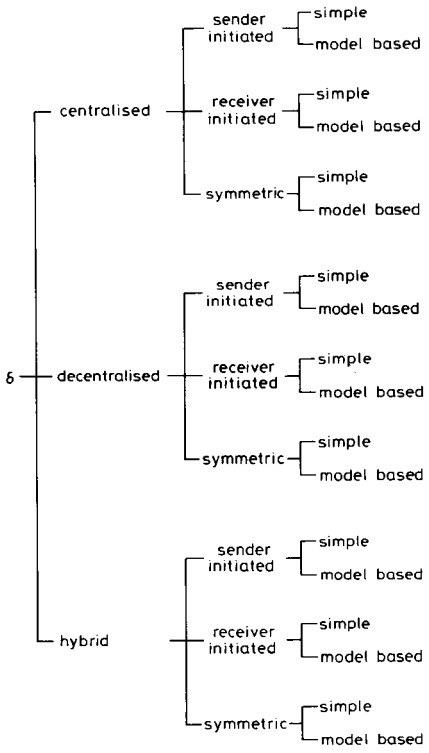

Fig. 2 *Solution space of decision making schemes*

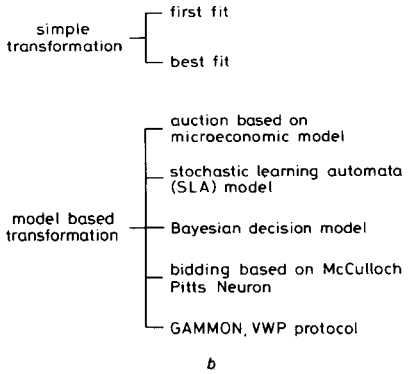

Fig. 3 *Transformations in decision making*

intercluster decisions interact is via the estimation scheme. A hybrid scheme thus combines centralised and decentralised decision making.

The second question is answered by choosing one of the three policies: *sender initiated* [8, 31], *receiver initiated* [49, 51, 52], and *symmetrically initiated* [46, 53]. In the 'sender initiated policy', a PE with tasks searches for

another PE with lesser number of tasks for remote execution whereas in a 'receiver initiated policy', a PE that has a relatively smaller number of tasks searches for another PE that has a higher number of tasks. A 'symmetrically initiated policy' combines sender initiated and receiver initiated policies in an attempt to extract advantages of both the policies. Sender initiated decision making provides good performance at low system loads but is susceptible to instability at high system loads. Receiver initiated decision making provides good performance at high system loads. Symmetrically initiated decision making has a potential to provide a quick redistribution of tasks at any load. A stable symmetric policy has been discussed by Shivaratri [29, 46] and Mirchandaney [53].

The third question can be answered by observing the transformations that are performed on the state information which affect computation of a task scheduling decision. A classification based on the transformations performed on the state information is a valuable factor in distinguishing between decision making policies because a specification of the state information and transformations performed on the state information are designed to suit the decision making policy. Such transformations may be performed when state information is exchanged [36] and/or at the time of computing a task scheduling decision [37, 54]. The transformations are classified into two types. The first type of transformations are simple and are not based on a specific model. The second type of transformations are based on a fomal model and the computations involved are dictated by the model [38, 55]. As an example of the first case, state information containing current load, memory size, and other resource utilisation may be directly exchanged between PEs with very simple transformation (like scaling) or no transformation. Simple transformations that a PE may employ while computing a decision include finding the first solution that meets the requirements (first fit) [28, 31, 46, 51] or the best solution (best fit) [8, 52, 56, 57] that meets the requirements using the available information. Examples of requirements that PEs might satisfy include having a minimum or a maximum value of load index, a value of load index above or below a threshold, being idle etc. If more than one candidate meets the requirements, then a solution may be chosen based a selected discipline like FCFS or random. For example, choose the PE with lightest load [8, 56] or heaviest load [52, 57], choose a PE with the highest surplus [32] or best bid [30, 54], or choose one of the idle PEs at random [58]. All such solutions are clubbed together into a transformation called 'simple'. Simple transformation includes weighting components of state information, linearly combining weighted components, and applying the first fit or the best fit strategies to the result in order to compute a decision. We do not further classify simple transformations in a finer detail. A disadvantage of such a decision is that it may not be possible to differentiate between schemes which differ only in those fine details, but this is acceptable, because in that case, it would be desirable to treat such schemes as identical. A large number of policies described in the literature use simple transformation for decision making. Examples of model based transformations are: Bayesian decision model [22], stochastic learning automata model [37, 54], bidding based on McCulloch Pitts Neuron [36], auction based on the microeconomic model [48, 59, 60], and GAMMON strategy based on a limited contention virtual window protocol [44]. Various transformations in decision making are shown in Fig. 3. Regardless of the type of transformation used, solutions to decision making are computed under uncertainty about the system state. The uncertainty is due to finite communication delays in the network and it may cause the state information to be inaccurate and decisions to be not optimal [61].

A solution to the decision making problem, denoted δ, is given as

$$\delta \in \{(\text{Centralised, Decentralised, Hybrid}) \times (\text{Sender initiated, Receiver initiated, Symmetric}) \times (\text{Decision making transformation})\} \quad (2)$$

$$\text{Decision making transformation} \in \{\text{Simple, Model based}\} \quad (3)$$

The solution space of decision making in Figs. 2 and 3 depicts 63 possible schemes. A solution to the dynamic task scheduling problem, denoted τ, is then given as

$$\tau = \xi \times \delta \quad (4)$$

The proposed taxonomy for generating τ is thus regular, easily understood, and provides a large number of solutions to dynamic task scheduling. Not all of the resulting solutions are interesting from practical considerations. For example, decentralised estimation paired with centralised decision making may not be of practical use. Some of the solutions have been implemented in previous works reported in the literature whereas some others with practical merit have not been explored. The utility of the taxonomy is discussed in the next section.

4 Application to previous works

Application of the taxonomy to previous works is discussed first. The examples are meant to be illustrative and not exhaustive.

1. S. Zhou [31], 'Trace-driven simulation study of dynamic load balancing', *IEEE Trans. Software Eng.*, Sept. 1988, **14**, (8), pp. 1327–1341, IEEE Computer Society
GLOBAL algorithm: **ξ = {Centralised, Complete, Voluntary, Periodic}, δ = {Decentralised, Sender initiated, Simple (least loaded PE)}**
DISTED algorithm: **ξ = {Decentralised, Complete, Voluntary, Periodic}, δ = {Decentralised, Sender initiated, Simple (least loaded PE)}**
CENTRAL algorithm: **ξ = {Centralised, Complete, Voluntary, Periodic}, δ = {Centralised, Sender initiated, Simple (least loaded PE)}**
THRHLD algorithm: **ξ = {Decentralised, Partial, Involuntary, Aperiodic}, δ = {Decentralised, Sender initiated, Simple (first fit)}**, *first fit:* select first PE whose load is below a threshold value
LOWEST algorithm: **ξ = {Decentralised, Partial, Involuntary, Aperiodic}, δ = {Decentralised, Sender initiated, Simple (least loaded PE)}**
RESERVE algorithm: **ξ = (Decentralised, Partial, Involuntary, Aperiodic}, δ = {Decentralised, Receiver Initiated, Simple (reserve PEs with load above threshold)}**, reservations are stacked when received, most recent reservation wins.

2. D. Eager, E. Lazowska, and J. Zahorjan [28], 'Adaptive load sharing in homogeneous distributed systems', *IEEE Trans. Software Eng.*, May 1986, **SE-12**, (5), pp. 662–675, IEEE Computer Society

Threshold algorithm: ξ = **{Decentralised, Partial, Involuntary, Aperiodic}**, δ = **{Decentralised, Sender initiated, Simple (first fit)}**
Shortest algorithm: ξ = **{Decentralised, Partial, Involuntary, Aperiodic}**, δ = **{Decentralised, Sender initiated, Simple ((best fit) PE with shortest queue length)}**.

3. K. Ramamritham, J. Stankovic, and W. Zhao [32], 'Distributed scheduling of tasks with deadlines and resource requirements', *IEEE Trans. Comput.*, Aug. 1989, **38**, (8), pp. 1110–1123, IEEE Computer Society
Focussed algorithm: ξ = **{Decentralised, Partial, Voluntary, Periodic}**, δ = **{Decentralised, Sender initiated, Simple (PE with maximum surplus)}**
Bidding algorithm: ξ = **{Decentralised, Partial, Composite, Combination}**, δ = **{Decentralised, Sender initiated, Simple (PE with best bid)}**
Flexible algorithm: ξ = **{Decentralised, Partial, Composite, Combination}**, δ = **{Decentralised, Sender initiated, Simple (max surplus + best bid)}** first check max surplus then best bid. Surplus information is broadcast periodically whereas bids are requested aperiodically when a task is to be scheduled.

4. K.G. Shin, and Y. Chang [45], 'Load sharing in distributed real time systems with state change broadcasts', *IEEE Trans. Comput.*, Aug. 1989, **38**, (8), pp. 1124–1142, IEEE Computer Society
ξ = **{Decentralised, Partial (buddy set), Voluntary, Aperiodic}**, δ = **{Decentralised, Sender initiated, Simple (underloaded PE at the top of the preferred list)}**.

5. I. Ahmad, and A. Ghafoor [50], 'Semi distributed load balancing for massively parallel multicomputer systems', *IEEE Trans. Software Eng.*, Oct. 1991, **SE-17**, (10), pp. 987–1003, IEEE Computer Society
ξ = **{Decentralised, Complete, Involuntary, Aperiodic}**, δ = **{Hybrid, Sender initiated, Simple (based on intersphere and intrasphere loads)}**
The PEs within a sphere do not maintain any load information, dissemination among schedulers is in a decentralised manner.

6. J.A. Stankovic [22], 'An application of bayesian decision theory to decentralised control of job scheduling', *IEEE Trans. Comput.*, Feb. 1985, **C-34**, (2), pp. 117–130, IEEE Computer Society
ξ = **{Hybrid, Variable, Voluntary, Periodic}**, δ = **{Decentralised, Sender initiated, Bayesian decision model)}**.

7. T. Kunz [37], 'The influence of different workload descriptions on a heuristic load balancing scheme', *IEEE Trans. Software Eng.*, July 1991, **17**, (7), pp. 725–730, IEEE Computer Society
ξ = **{Decentralised, Complete, Voluntary, Periodic}**, δ = **{Decentralised, Sender Initiated, SLA model}**.

8. K. Baumgartner and B.W. Wah [44], 'GAMMON: A load balancing strategy for local computer systems with multiaccess networks', *IEEE Trans. Comput.*, Aug. 1989, **38**, (8), pp. 1098–1109, IEEE Computer Society
ξ = **{Decentralised, Complete, Voluntary, Aperiodic}**, δ = **{Decentralised, Sender Initiated, GAMMON (VWP Protocol to search for min-max pair)}**.

9. S. Zhou, X. Zheng, J. Wang, and P. Delisle [8], 'Utopia: A load sharing facility for large, heterogeneous distributed computer systems', *Technical Report*, CSRI-257, (University of Toronto, Toronto, April 1992)
Global algorithm: ξ = **{Hybrid, Partial, Voluntary, Periodic}**, δ = **{Decentralised, Sender initiated, Simple (least loaded PE)}**
Central algorithm: ξ = **{Hybrid, Partial, Voluntary, Periodic}**, δ = **{Hybrid, Sender Initiated, Simple (least loaded PE)}**.

10. D. Ferguson, Y. Yemini, and A. Nikalaou [48], 'Microeconomic algorithms for load balancing in distributed computer systems'. Proceedings of the International Conference on *Distributed computing systems*, 1988, pp. 491–499
Sealed Bid Auction: ξ = **{Decentralised, Partial, Voluntary, Aperiodic}**, δ = **{Decentralised, Sender initiated, Microeconomic model)}**
Dutch auction: ξ = **{Decentralised, Partial, Composite, Aperiodic}**, δ = **{Decentralised, Receiver initiated, Microeconomic model)}**
Price changes are broadcast to all neighbours in both auctions. Bids are sent by processes to resources in the sealed bid auction whereas in the Dutch auction bids are requested when a resource is idle.

11. N.G. Shivaratri, and P. Krueger [46], 'Two adaptive location policies for global scheduling algorithms'. Proceedings of the International Conference on *Distributed Computing Systems*, 1990, pp. 502–509
Symmetric algorithm: ξ = **{Decentralised, Partial, Involuntary, Aperiodic}**, δ = **{Decentralised, Symmetric, Simple (first fit)}**.

12. R.M. Bryant, and R.A. Finkel [62], 'A stable distributed scheduling algorithm'. Proceedings of 2nd International Conference on *Distributed Computing Systems*, 1981, pp. 314–323
ξ = **{Decentralised, Partial, Voluntary, Aperiodic}**, δ = **{Decentralised, Sender Initiated, Simple (transfer tasks in PE pairs from a heavy loaded to a lightly loaded PE)}**.

The following observations are made after applying the taxonomy to different examples. The first observation relates to the fact that estimation and decision making are considered separate problems. Estimation and decision making do not always use the same type of logical control organisation. The taxonomy clearly distinguishes between the type of organisations used for estimation and decision making; a distinction that is not obvious with a monolithic solution to estimation or decision making. For example GLOBAL algorithm [31] uses *centralised* estimation and *decentralised* decision making, the algorithm presented in Reference 22 and the *Global* algorithm [8] use *hybrid* estimation and *decentralised* decision making, and the algorithm in Reference 50 uses *decentralised* estimation and *hybrid* decision making. The taxonomy helps identify significant similarities or differences that may exist between algorithms. For example GLOBAL and DISTED algorithms [31] use the same solution for decision making ({Decentralised, Sender initiated, Simple (best fit)}) but differ in estimation; GLOBAL uses centralised estimation whereas DISTED uses decentralised estimation. THRHLD and RESERVE algorithms [31] use the same estimation policy ({Decentralised, Partial, Involuntary, Aperiodic}) but use different decision making policies; THRHLD uses sender initiated decision making whereas RESERVE uses receiver initiated decision making. The taxonomy is applicable to dynamic task scheduling in real-time systems as well. The real-time systems differ mainly in their decision making transformations.

It is observed that the *simple* decision making transformation is applied in a large number of cases whereas estimation and decision making using a hybrid scheme is implemented only in a few recent cases. Estimation schemes that are commonly implemented are: {Decentralised, Partial, Involuntary, Aperiodic} and {Decentralised × (Complete, Partial) × Voluntary × (Periodic, Aperiodic)}, whereas a popular decision making scheme is {Decentralised, Sender initiated, Simple (best fit)}.

It should be noted that, individually, every solution for ξ and δ can be implemented in practice but all of their possible combinations may not be of interest. Application of the analysis to previous works indicates that less than 30% of the solution space for ξ and δ is used. Among those solutions for ξ and δ that have been implemented, a small subset of their possible combinations exists. A study of the solution space of τ reveals that at least one third of the space contains useful solutions. This portion consists of combinations of centralised estimation with centralised decision making, decentralised estimation with decentralised decision making, and hybrid estimation with hybrid decision making. In addition, combinations of centralised estimation with decentralised decision making, hybrid estimation with decentralised decision making further expand the usable space of τ.

The taxonomy illustrates new possibilities that may have potential to provide the necessary performance in large scale DCS as discussed below. The solutions developed in the earlier research were of a *puristic* nature. That is, the solutions used either centralised or decentralised policies, voluntary or involuntary dissemination, sender initiated or receiver initiated policy etc. Each puristic solution has merits and demerits of its own. In order to make solutions scalable and applicable to large scale DCS, an apparent trend in the recent literature is to develop *mixed* policies [8, 29, 50] that combine puristic approaches and have shown potential for providing high performance in large scale DCS. Such mixed policies inherit merits and demerits of the constituent puristic policies and it is possible to tune them beneficially with a good understanding of the engineering trade-offs involved. Examples of mixed policies for estimation are: hybrid policy that combines centralised and decentralised policies, variable information exchange policy that combines complete information exchange and partial information exchange, composite scheme that combines voluntary and involuntary initiation of information dissemination. Examples of mixed policies for decision making are: a hybrid policy that combines centralised and decentralised policies, a symmetric policy that combines sender initiated and receiver initiated policies. Examples of solutions that incorporate mixed policies which have not been evaluated in the literature and that may have potential for providing the required performance in large scale DCS are: {Hybrid × (Partial, Variable) × Composite × Combination} for estimation and {Hybrid × Symmetric × (Simple, Model based)} for decision making. It should be noted that different algorithms can be designed using each of these policies with different heuristics. A mixed policy can be more difficult to specify as compared to a puristic policy because a mixed policy may require the use of intelligent heuristics for efficient adaptation. Such heuristics are not always easy to incorporate because they are either difficult to design and/or they increase the overhead. Puristic policies will have an application in systems that have characteristics conducive for implementing them; for example, if the system size is small and robustness is important, decentralised control may be used, if the system state changes rapidly and if estimation delay should be minimised, voluntary dissemination may be used.

The taxonomy presented in this paper will help the designer of a dynamic scheduling algorithm in comparing various solutions using a common framework and enable the designer to precisely specify the components of estimation and decision making policies based on the system requirements. For example, if the system is moderately sized and the system state does not change very often, the designer may choose {Decentralised, Complete, Voluntary, Aperiodic} estimation policy whereas if the system is a large size DCS which rapidly changes state, the designer may choose {Hybrid, Variable, Involuntary, Aperiodic} estimation policy. Similar reasoning may be applied in choosing a decision making policy. The designer may also choose to switch between the policies making them adaptive by use of appropriate heuristics and evaluate the utility of solutions that have not been considered in the past.

5 Conclusions

A taxonomy for dynamic task scheduling has been presented. It consider solutions to state estimation and decision making problems in detail and treats them orthogonally. Treating the two problems orthogonally is conceptually beneficial because it allows one to study, understand, and classify the solutions using a regular and well defined set of criteria. A detailed understanding of the estimation solutions is important because the scalability of a dynamic task scheduling algorithm in a large scale DCS depends significantly on the estimation policies. Utility of the proposed taxonomy was demonstrated by applying it to classify solutions proposed in the literature. The proposed taxonomy illustrates solutions which may have potential for future research.

Although, treating solutions to estimation in a monolithic manner has shown that an estimation policy significantly affects the performance in a large scale DCS, the effects of varying each component of estimation and decision making have not been fully analysed. Such an analysis will be necessary to specify the components for estimation and decision making in large scale systems. This problem is being currently addressed as part of the ongoing research. It is the author's conjecture that combinations using mixed policies will provide solutions to dynamic task scheduling for large scale DCS in future research. Another area in which future research is expected to evolve is in design of new heuristics for estimation and decision making that better adapt to the changing system state.

6 References

1 CASAVANT, T.L., and KUHL, J.G.: 'A taxonomy of scheduling in general-purpose distributed computing systems', *IEEE Trans. Software Eng.*, Feb. 1988, **14**, (2), pp. 141–154

2 LIVNY, M., and MELMAN, M.: 'Load balancing in homogeneous broadcast distributed systems'. Proceedings of the ACM computer network performance symposium, (ACM, Maryland, April 1982), pp. 47–55

3 MUTKA, M.W., and LIVNY, M.: 'Profiling workstations' available capacity for remote execution'. Performance '87. Proceedings of the 12th IFIP WG 7.3 Symposium on *Computer performance* Brussels, 7–9 Dec. 1987

4 EFE, K.: 'Heuristic models of task assignment scheduling in distributed systems', *Computer*, IEEE Computer Society, June 1982, pp. 50–56

5 CHU, W.W., HOLLOWAY, L.J., LAN, M., and EFE, K.: 'Task allocation in distributed data processing', *IEEE, Computer*, Nov. 1980, **13**, pp. 57–69

6 HO, Y.C., and KURTARAN, B.: 'Distributed computing viewed as a team problem with constraints', *J. Opt. Theory & Appl. (JOTA)*, Nov. 1978, **26**, (3), pp. 352–365

7 NICOL, D.M., and SALTZ, J.H.: 'An analysis of scatter decomposition', *IEEE Trans. Comput.*, Nov. 1990, **39**, (11), pp. 1337–1345

8 ZHOU, S., ZHENG, X., WANG, J., and DELISLE, P.: 'Utopia: A load sharing facility for large heterogeneous distributed computer systems', *to appear in* 'Software — Practice and Experience'; *also available as* Technical report CSRI-257, University of Toronto, Toronto, Canada, April 1992

9 DOUGLIS, F., and OUSTERHOUT, J.: 'Process migration in the sprite operating system'. Proceedings of the 7th international conference of *Distributed computing systems*, Berlin, Germany, 21–25, Sep., 1987, pp. 18–25

10 EZZAT, A.: 'Load balancing in NEST: a network of workstations'. Proc. 1986 Fall Joint Computer Conference, Dallas, TX, No. 1986, pp. 1138–1149

11 BERSHAD, B.: 'Load balancing with maitre d', *Technical report UCB/CSD 85/276*, University of California, Berkeley, CA, Dec. 1985

12 SHEN, C., and TSAI, W.: 'A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion', *IEEE Trans. Comput.*, March 1985, **C-34**, 3, pp. 197–203

13 BOKHARI, S.H.: 'Dual processor scheduling with dynamic reassignment', *IEEE Trans. Software Eng.*, July 1979, **SE-5**, (4), pp. 341–349

14 STONE, H.S.: 'Critical load factors in two processor distributed systems', *IEEE Trans. Software Eng.*, May 1978, **SE-4**, (3), pp. 85–93

15 YI, P., MA, R., LEE, E., and TSUCHIYA, M.: 'A task allocation model for distributed computing systems', *IEEE Trans. Comput.*, Jan. 1982, **C-31**, (1), pp. 41–47

16 CHOU, T.C.K., and ABRAHAM, J.A.: 'Load balancing in distributed systems', *IEEE Trans.*, July 1982, **SE-8**, (4), pp. 401–412

17 KLEINROCK, L.: 'Queuing systems: volume 2, computer applications'. John Wiley & Sons (1976)

18 KLEINROCK, L., and NILSSON, A.: 'On optimal scheduling algorithms for time-shared systems', *J. ACM*, July 1981, **28**, (3), pp. 477–486

19 CHOW, Y., and KOHLER, W.H.: 'Models for dynamic load balancing in a heterogeneous multiple process or system', *IEEE Trans. Comput.*, May 1979, **C-28**, (5), pp. 354–361

20 ZHOU, S., and FERRARI, D.: 'An experimental study of load balancing performance', *Technical Report UCB/CSD 87/336*, University of California, Berkeley, CA, Jan. 1987

21 ZHOU, S., and FERRARI, D.: 'A measurement study of load balancing performance'. Proceedings of the international conference on *Distributed computing systems*, 1987, pp. 490–497

22 STANKOVIC, J.A.: 'An application of Bayesian decision theory to decentralized control of job scheduling', *IEEE Trans. Comput.*, Feb. 1985, **C-34**, (2), pp. 117–130

23 KRUEGER, P., and LIVNY, M.: 'When is the best load sharing algorithm a load balancing algorithm?', *Computer Sciences Technical Report 85-04-01*, University of Wisconsin, Madison, MD, April 1987

24 KRUEGER, P., and LIVNY, M.: 'The diverse objectives of distributed scheduling policies'. Proceedings of the International conference on *Distributed computing systems*, 1987, pp. 242–249

25 THEIMER, M.M., LANTZ, K.A., and CHERITON, D.R.: 'Preemptable remote execution facilities for the V-system'. Proceedings of the 10th symposium on *Operating systems principles*, Dec. 1985, pp. 2–12

26 FINKEL, R., and ARTSY, Y.: 'Designing a process migration facility: the charlotte experience', *IEEE, Computer*, Sept. 1989, **22**, (9), pp. 47–56

27 MULLENDER, S., VAN ROSSUM, G., TANNENBAUM, A.S., VAN RENESSE, R., and VAN STAVEREN, H.: 'Amoeba: a distributed operating system for the 1990s', *IEEE. Computer*, May 1990, **23**, (5), pp. 44–53

28 EAGER, D., LAZOWSKA, E., and ZAHORJAN, J.: 'Adaptive load sharing in homogeneous distributed systems', *IEEE Trans. Software Eng.*, May 1986, **SE-12**, (5), pp. 662–675

29 SHIVARATRI, N.G., KRUEGER, P., and SINGHAL, M.: 'Load distributing for locally distributed systems', *IEEE Computer*, Dec. 1992, **25**, (12), pp. 33–44

30 CASAVANT, T.L., and KUHL, J.G.: 'A communicating finite automata approach to modeling distributed computation and its application to distributed decision making', *IEEE Trans. Comput.*, May 1990, **39**, (5), pp. 628–639

31 ZHOU, S.: 'Trace-driven simulation study of dynamic load balancing', *IEEE Trans. Software Eng.*, Sept. 1988, **14**, (9), pp. 1327–1341

32 RAMAMRITHAM, K., STANKOVIC, J., and ZHAO, W.: 'Distributed scheduling of tasks with deadlines and resource requirements', *IEEE Trans. Comput.*, Aug. 1989, **38**, (8), pp. 1110–1123

33 ROTITHOR, H.G., and PYO, S.S.: 'Decentralized decision making in adaptive task sharing'. Proceedings of the IEEE symposium on *Parallel and distributed processing*, Dallas, TX, 9–13, Dec. 1990, pp. 34–41

34 WANG, Y.T., and MORRIS, R.J.T.: 'Load sharing in distributed systems', *IEEE Trans. Comput.*, March 1985, **C-34**, (3), pp. 204–217

35 FERRARI, D., and ZHOU, S.: 'A load index for dynamic load balancing'. Proceedings, 1986 Fall joint computer conference, Dallas, TX, 4–6 Nov., 1986, pp. 684–690

36 STANKOVIC, J.A., and SIDHU, I.S.: 'An adaptive bidding algorithm for proceses, clusters, and distributed groups'. Proceedings of International Conference on *Distributed computer systems*, 1984, pp. 49–59

37 KUNZ, T.: 'The influence of different workload descriptions on a heuristic load balancing scheme', *IEEE Trans. Software Eng.*, July 1991, **17**, (7), pp. 725–730

38 STANKOVIC, J.A., CHOWDHURY, N., MIRCHANDANEY, R., and SIDHU, I.: 'An evaluation of the applicability of different mathematical approaches to the analysis of decentralized control algorithms'. Proceedings of the International Conference on *Parallel processing*, IEEE Computer Society, 1982, pp. 62–69

39 HAGMANN, R.: 'Processor server: sharing power in a workstation environment'. Proceedings of the 6th IEEE *Distributed computing* conference, Cambridge, MA, May 1986, pp. 260–267

40 LITZKOW, M.J.: 'Remote Unix'. Proceedings of the summer USENIX conference, Phoenix, AZ, June 1987

41 LITZKOW, M.J., LIVNY, M., and MUTKA, M.W.: 'Condor — a hunter of idle workstations'. Proceedings of the 8th International Conference on *Distributed computing systems*, San Jose, CA, 13–17 June, 1988, pp. 104–111

42 LIN, H.C., and RAGHAVENDRA, C.S.: 'A dynamic load balancing policy with a central job dispatcher', *IEEE Trans. Software Eng.*, Feb., 1992, **18**, (2), pp. 148–158

43 MORRIS, J.H., SATYANARAYAN, M., CONNER, M.H., HOWARD, J.H., ROSENTHAL, D.S.H., and SMITH, F.D.: 'Andrew: a distributed personal computing environment', *Commun. ACM*, March 1986, **29**, (3), pp. 184–201

44 BAUMGARTNER, K., and WAH, B.W.: 'GAMMON: a load balancing strategy for local computer systems with multiaccess networks', *IEEE Trans Comput.*, Aug. 1989, **38**, (8), pp. 1098–1109

45 SHIN, K.G., and CHANG, Y.: 'Load sharing in distributed real time systems with state change broadcasts', *IEEE Trans. Comput.*, Aug. 1989, **38**, (8), pp. 1124–1142

46 SHIVARATRI, N.G., and KRUEGER, P.: 'Two adaptive location policies for global scheduling algorithms'. Proceedings of the international conference on *Distributed computing systems*, 1990, pp. 502–509

47 THEIMER, M.M., and LANTZ, K.A.: 'Finding idle machines in workstation-based distributed system'. Proceedings of the 8th international conference on *Distributed computing systems*, IEEE, San Jose, CA 13–17 June, 1988, pp. 112–122

48 FERGUSON, D., YEMINI, Y., and NIKOLAOU, C.: 'Microeconomic algorithms for load balancing in distributed computer systems'. Proceedings of the international conference on *Distributed computing systems*, IEEE 1988, pp. 491–499

49 NI, L.M., XU, C., and GENDREAU, T.B.: 'A distributed drafting algorithm for load balancing', *IEEE Trans. Software Eng.*, Oct. 1985, **SE-11**, (10), pp. 1153–1161

50 AHMAD, I., and GHAFOOR, A.: 'Semi distributed load balancing for massively parallel multicomputer systems', *IEEE Trans. Software Eng.*, Oct. 1991, **SE-17**, (10), pp. 987–1003

51 EAGER, D., LAZOWSKA, E., and ZAHORJAN, J.: 'A comparison of receiver-initiated and sender-initiated adaptive load sharing', *Sigmetrics*, 1985, **13**, (2), pp. 1–3

52 HAC, A., and JIN, X.: 'Dynamic load balancing in a distributed system using a decentralized algorithm'. Proceedings of the international conference on *Distributed computing systems*, 1987, pp. 170–177

53 MIRCHANDANEY, R., TOWSLEY, D., and STANKOVIC, J.A.: 'Analysis of the effects of delays on load sharing', *IEEE Trans. Comput.*, Nov. 1989, **38**, (11), pp. 1513–1525

54 STANKOVIC, J.A.: 'Stability and distributed scheduling algorithms', *IEEE Trans. Software Eng.*, Oct. 1985, **SE-11**, (10), pp. 1141–1152

55 STANKOVIC, J.A.: 'A comprehensive framework for evaluating decentralized control'. Proceedings of the international conference on *Parallel processing*, 1980, pp. 181–187

56 HWANG, K., CROFT, W., GOBLE, G., WAH, B., BRIGGS, F., SIMMONS, W., and COATES, C.: 'A Unix based local computer network with load balancing', *IEEE, Computer*, Apr. 1982, **15**, (4),

pp. 55–66
57 KALE, L.V.: 'Comparing the performance of two dynamic load distribution methods'. Proceedings of the international conference on *Parallel processings*, St. Charles, IL, 1988, pp. 8–12
58 DOUGLIS, F., and OUSTERHOUT, J.: 'Transparent process migration for personal workstations', *Technical Report, UCB/CSD 89/540*, University of California, Berkeley, CA, 1989
59 MILLER, M.S., and DREXLER, K.E.: 'Markets and computations: Agoric open systems', *in* HUBERMAN, B.S. (Ed.): 'The ecology of computation' (North-Holland, Amsterdam, 1988), pp. 133–176
60 MALONE, T.W., FIKES, R.E., GRANT, K.R., and HOWARD, M.T.: 'Enterprise: a market-like task scheduler for distributed computing environments', *in* HUBERMAN, B.A. (Ed.): 'The ecology of computations' (North-Holland, Amsterdam, 1988), pp. 177–205
61 STANKOVIC, J.A.: 'Achievable decentralized control for functions of a distributed processing operating system'. Proceedings on the international conference on *Parallel processing*. IEEE Computer Society, 1982, pp. 226–230
62 BRYANT, R., and FINKEL, R.A.: 'A stable distributed scheduling algorithm'. Proceedings of the 2nd international conference on *Distributed computing systems*, 1981, pp. 314–323