On the Asymptotical Optimality of Multilayered Decentralized Consensus Protocol

Cheng-Ru Lin and Ming-Syan Chen, *Senior Member*, *IEEE*

Abstract—A decentralized consensus protocol refers to a process for all nodes in a distributed system to collect the information/status from every other node and reach a consensus among them. Two classes of decentralized consensus protocols have been studied before: the one *without an initiator* and the one *with an initiator*. While the one without an initiator has been well studied in the literature, it is noted that the prior protocols with an initiator mainly relied upon the one without an initiator and thus did not fully exploit the intrinsic properties of having an initiator. By exploiting the concept of multilayered execution, we develop in this paper an efficient multilayered decentralized consensus protocol for a distributed system with an initiator. By adapting itself to the number of nodes in the system, the proposed protocol can determine a proper layer for execution and reach the consensus in the minimal numbers of message steps while incurring a much smaller number of messages than required by prior works. Several illustrative examples are given and performance analysis of the proposed algorithm is conducted to provide many insights into the problem studied. It is shown that the decentralized consensus protocols developed in this paper for the case of having an initiator significantly outperform prior schemes. Specifically, it is proven that 1) the ratio of the average number of messages incurred by the proposed algorithm to that by the prior method approaches zero as the number of nodes increases and 2) the proposed algorithm is asymptotically optimal in the sense that the message number required by the proposed algorithm and that of the optimal one are asymptotically of the same complexity with respect to the number of nodes in the system, showing the very important advantage of the proposed algorithm.

Index Terms—Consensus protocol, distributed systems, multiport communication, performance analysis.

1 Introduction

It has been an important issue on how to link together many powerful and autonomous computers to build a distributed processing system for better availability and cost performance. In such a system, instead of using a shared memory and a global clock, all synchronization and communication among the processing nodes can be done via message passing [1]. One important scheme in distributed computations is the consensus protocol, which refers to a process for all nodes in a distributed system to collect the information/status from every other node and reach a consensus among them. Applications of consensus protocols include the extrema finding, coordination of distributed checkpoints [13], acquisition of a new global state [7] and the broadcasting of various system-dependent messages [12], [14].

As pointed out in [6], unlike a *centralized system* where there is a centralized coordinator to collect the information from all participants, a *decentralized system* is based on decentralized message passing, and does not rely upon the use of a coordinator. Though easier to design, centralized applications are apt to suffer from such drawbacks as the need of a coordinator, longer protocol execution time due to less parallelism and the vulnerability to a single point failure. Free of these drawbacks, decentralized applications have drawn a considerable amount of research attention [4], [5], [6], [9], [13], [18]. Performance of the consensus protocols is usually assessed by the number of message steps required to reach a consensus by all nodes in a distributed system, and also the total number of messages incurred during the execution of the protocol [9], [10], [13]. Several studies have been conducted to minimize the number of message steps (or time) and the number of messages for various communication networks/distributed environments [5], [6], [8], [13], [15], [16], [18]. A related survey can be found in [10]. Clearly, there is a trade off on the number of message steps required and the number of messages incurred for a consensus protocol. To reduce the overhead of the scheme without compromising its efficiency, we would naturally like to complete the consensus protocol in the minimal number of steps while incurring as few messages as possible.

Two classes of decentralized consensus protocols have been studied before: the one *without an initiator* and the one *with an initiator*. In the former (without an initiator), synchronization is usually assumed to be achieved a priori, and all processing nodes concurrently participate in the consensus protocol when the protocol is started.[1] In the latter (with an initiator), it is assumed that each node, except the initiator, will not participate in the consensus protocol until it is informed to do so by receiving a message from some other node. While the one without an initiator has been well studied in the literature [4], [10], [16], it is noted that the prior protocols with an initiator mainly relied upon the one without an initiator and, thus, did not fully exploit the intrinsic properties of having an initiator. As the

• *The authors are with the Electrical Engineering Department National Taiwan University, Taipei, Taiwan, Republic of China.*
E-mail: mschen@cc.ee.ntu.edu.tw.

Manuscript received 1 Feb. 2000; revised 28 June 2000; accepted 30 Nov. 2001.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 111364.

1. More precisely, the need of synchronization depends on the upper layer applications. Readers are referred to [3] for an instance of using a consensus protocol without an initiator while not requiring specific synchronization.

1045-9219/02/$17.00 © 2002 IEEE
Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

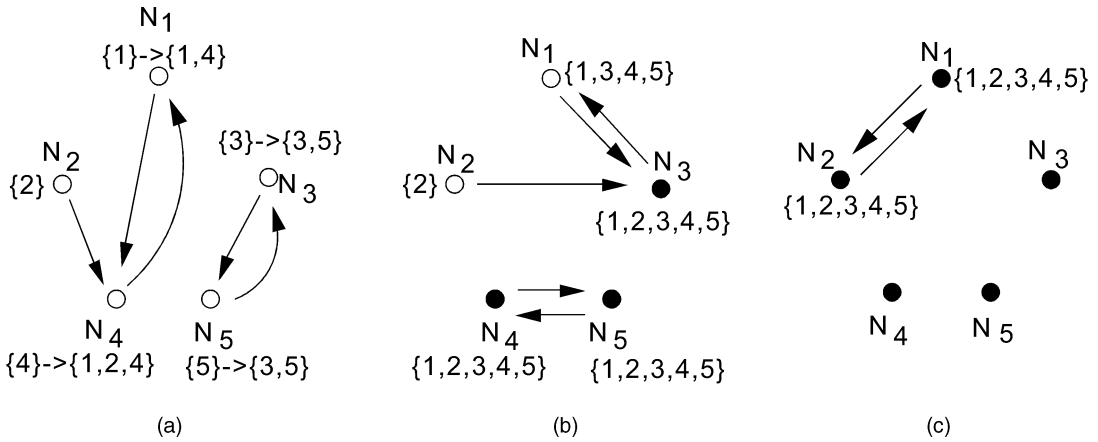


Fig. 1. A decentralize consensus protocol without an initiator (*1*-port).

practical importance of many related distributed applications increases nowadays, it is essential to devise an efficient decentralized consensus protocol with an initiator, which is consequently taken as the objective of this paper.

The system considered is completely connected with synchronous communication. In the proposed algorithm, both *1*-port communication, which means that every node in the system can send out one message at a time, and the general case, *k*-port communication, meaning that every node can send out k messages in one step, are considered. All nodes in the system are assumed to have the same number of communication ports, and every message sent takes one communication step. This model is the same as those in most related works [10]. To facilitate the presentation, we use the identification (id) of each node to denote the information that this node wants to send to every other node via the consensus protocol. In the end, consensus is reached at every node after each node receives all the id's from all other nodes.

The problems studied in this paper can be best understood by considering the case of reaching a consensus among five nodes.[2] A node having all the information from other nodes is called an *expert* [10]. We use black nodes to denote experts, and white nodes to denote those that still have incomplete information. Without an initiator, as shown in Fig. 1 for 1-port communication, the five nodes reach a consensus after three steps, with a total of 12 messages. The information collected so far after each step is shown in the bracket next to each node. An arrow pointing from node N_i to node N_j represents N_i is sending what it knows thus far to N_j. The case of having an initiator for 1-port communication is shown in Fig. 2 where, without loss of generality, node N_1 is the initiator. It can be seen that the five nodes reach a consensus after six steps, with a total of nine messages.

As pointed out earlier, although decentralized consensus protocols with an initiator were proposed in [6], those protocols mainly relied upon the one without an initiator and thus did not fully exploit the intrinsic properties of having an initiator. To remedy this, we shall address the development of efficient consensus protocols with an initiator in this paper. By exploiting the concept of multilayered execution, we devise in this paper an efficient decentralized consensus protocol, referred to as algorithm ML (standing for Multi-Layered), for a distributed system with an initiator. By adapting itself to the number of nodes in the system, algorithm ML is able to determine a proper layer for execution and to reach the consensus in the minimal numbers of message steps while incurring a much smaller number of messages than required by prior works. Several illustrative examples are given and performance analysis of algorithm ML is conducted to provide many insights into the problem studied. It is shown that algorithm ML significantly outperforms prior schemes for the case of having an initiator, and the performance improvement achieved by algorithm ML increases as the number of nodes in the system grows. Specifically, for a system of p nodes with *k*-port communication, the ratio of the average number of messages incurred by algorithm ML to that by the prior method is proved to be of the complexity $O(\log^{-1} p)$ which approaches zero as the number of nodes p becomes large. Furthermore, algorithm ML is proved to be asymptotically optimal in the sense that the message number required by algorithm ML and that required by the optimal one are asymptotically of the same complexity with respect to the number of nodes p in the system. It is worth mentioning that though results of limited applicability have been derived before [4], [10], [16], the issue of deriving the minimal number of messages required for a decentralized consensus protocol with *k*-port communication to complete in the minimal number of steps is still an open problem due to its inherent difficulty. Algorithm ML devised in this paper goes beyond prior schemes not only for its generality for *k*-port communication but also for its asymptotic optimality.

This paper is organized as follows: Preliminaries including the system model and descriptions of prior related protocols are given in Section 2. The multilayered decentralized consensus protocol with an initiator, i.e., algorithm ML, is described in Section 3. Performance analysis is conducted in Section 4. This paper concludes with Section 5.

2 Preliminaries

A system model considered in this paper is given in Section 2.1. Then, prior related protocols are presented in the remaining subsections to facilitate our later presentation of algorithm ML in Section 3. For better readability, we shall

2. Examples in Fig. 1 and Fig. 2 are extracted from [5].

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

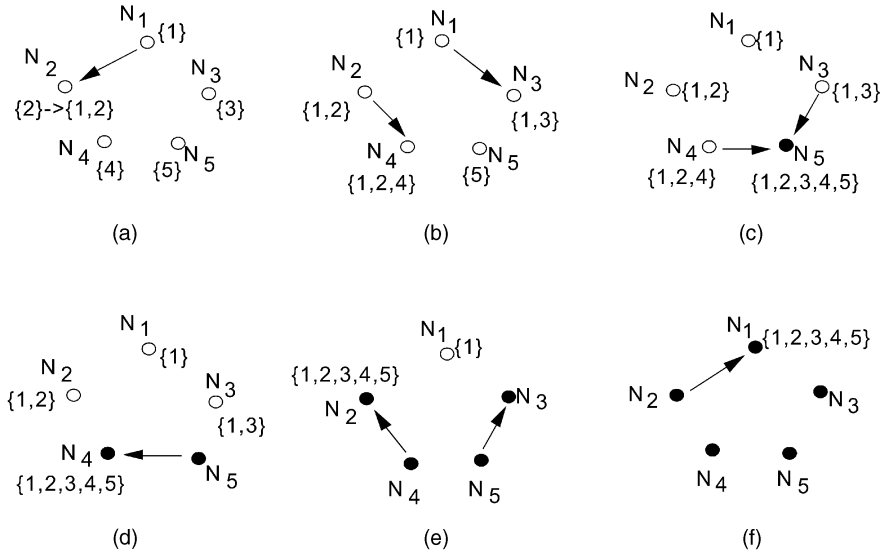


Fig. 2. A decentralized consensus protocol with an initiator (1-port).

present the case of 1-port communication first and then the case of multiport communication. Specifically, we shall describe briefly the decentralized consensus protocol without an initiator for 1-port communication (i.e., G_1, in Section 2.2), the one with an initiator for 1-port communication (i.e., G_2, in Section 2.3), the one without an initiator for k-port communication (i.e., G_3, in Section 2.4), the one with an initiator for k-port communication, (i.e., G_4, in Section 2.5) in that order.

2.1 System Model

We use the identification (id) of each node to denote the information that this node wants to send to every other node. Note that depending on the applications, the real content of id can be very general, such as a personalized information/database, the numbers to be sorted, a vector describing the local state of a node, and a "yes" or "no" vote of the commit protocol in a distributed transaction, to name just a few. Also, the *information* at each node means the set of id's that node collects thus far, and the content of the *message* of a transmission is referred to as the information of the sender at the time of transmission. One message might contain many id's. The system model we consider is similar to the one in [4], [6], [10], [12], [18], and is summarized as follows:

Model *M*

1. The system is completely connected with synchronous communication.
2. Every message sent in the system takes one communication step.
3. k-port communication means that each node is capable of sending k messages out in one step. (There is no restriction on the number of messages each node can receive in one step.)
4. All nodes in the system are assumed to have the same communication capability (i.e., same number of communication ports).

A node is said to become an expert if that node has received all id's of the nodes in the system [10]. The system is said to reach consensus if all nodes in the system are experts. Note that we do not exclude either the possibility of two-way transmission between two nodes, or the capability of each node to participate in both sending and receiving messages in one communication step, thus distinguishing our work in this paper from the one in [17]. The input port assumption of Model M can be justified by the sufficient input buffer space and also the simplicity of the CPU operation to handle the incoming data. Explicitly, it is noted that when users write message passing codes in the put/get (i.e., send /receive) model, we would usually have software bottleneck in puts rather than gets since most receives could be translated into local memory access. As a consequence, under such a put/get model which is supported by some existing message passing tools (e.g., Cray T3D and Meiko CS-1), fan-in could usually be less a restriction than fan-out in message passing. This is the very reason we consider this model. Note that this consensus protocol problem is the same as the gossiping problem addressed in [10] except that a concurrent two-way transmission, such as a phone conversation, is assumed for the latter. Also, we assume that nodes in the system will not be faulty or maliciously send wrong messages to others. Readers interested in issues related to fault-tolerance and Byzantine agreement are referred to [9], [11].

2.2 Decentralized Consensus Protocol without an Initiator (1-port)

A decentralized consensus protocol without an initiator for 1-port communication G_1 was presented in [5]. Highlights of G_1 are given below and readers interested in more details are referred to [5]. A balanced binary partitioning tree of a positive number p is a binary partitioning tree constructed by first labeling the root node with p, and then, for each node with a label $k \geq 2$, generating the left and right children of this node and labeling them with $\lceil \frac{k}{2} \rceil$ and $\lfloor \frac{k}{2} \rfloor$, respectively. Clearly, there are $n+1$ levels in the balanced binary partitioning tree of a number p where $n = \lceil \log_2 p \rceil$. For convenience, the level of the root is called level 0. Using the balanced binary partitioning tree, the nodes in the

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

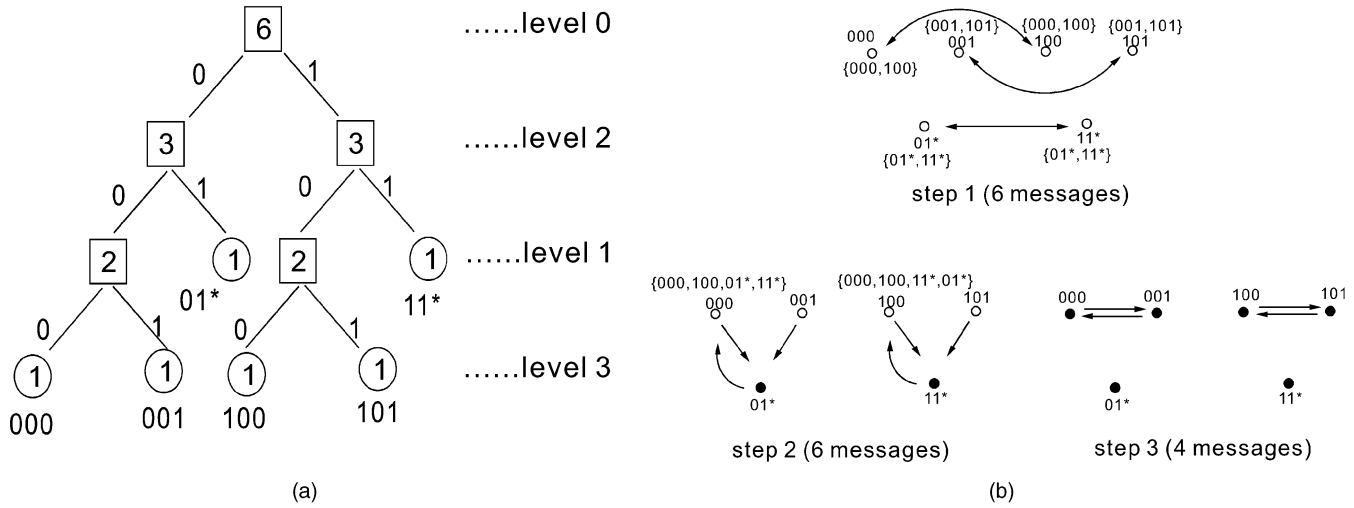


Fig. 3. Illustration for the partitioning tree and operations of algorithm G_1.

system can be addressed as follows. For a system of p nodes, obtain the balanced binary partitioning tree of p. Next, for every internal node, code the link to its left child with a bit "0" and that to its right child with a bit "1." Then, determine the address of each leaf node by the coded bits in the links on the path from the root to that node, and append a bit "*" to each leaf node in level $n-1$. An example partitioning tree for a system of six nodes is given in Fig. 3, where after assigning the p nodes in the system with the addresses of the p leaf nodes in the balanced binary partitioning tree in Fig. 3a, algorithm G_1 performs the consensus protocol by exchanging messages along each dimension one by one in Fig. 3b. It can be verified that the protocol is completed in three steps and the total number of messages sent is $6+6+4=16$. From [5], we have the following two propositions for G_1.

Proposition 1. *For a system of p nodes with one-port communication, algorithm G_1 completes a decentralized consensus protocol in n steps by incurring $N_{G1}(p) = np + p - 2^n$ messages, where $n = \lceil \log_2 p \rceil$.*

Proposition 2. *In a system of p nodes with one-port communication, algorithm G_1 requires the minimal number of steps, $\lceil \log_2 p \rceil$, to complete a decentralized consensus protocol.*

2.3 Decentralized Consensus Protocol with an Initiator (1-port)

By utilizing the concept of G_1, a decentralized consensus protocol with an initiator for 1-port communication, referred to as G_2 in this paper, was proposed in [5]. G_2 consists of three phases: 1) broadcasting phase, 2) shuffling phase, and 3) confirming phase. Recall that in the case of having an initiator, each node, except the initiator, will not start sending any message until it receives a message from some other node. The purpose of the broadcasting phase, started by the initiator, is mainly to inform each node to participate in the consensus protocol by letting every node receive some messages. As shown in [5], through sending messages, the information will be cumulatively collected during the broadcasting phase, and a *minimal complete set* is formed in the end of this phase.

Definition 1. *A* minimal complete set *of nodes is the minimal set of nodes that consists of all the information to reach a consensus.*

Definition 2. *The minimal complete set formed in the end of the broadcasting phase is termed the* shuffling set.

The purpose of the shuffling phase is in fact to reach consensus for the nodes in the shuffling set. The operations in the shuffling phase are essentially the same as those in G_1. After the consensus is reached by the nodes in the shuffling set, the entire information is sent to all other nodes in the system in the confirming phase. For an illustrative purpose, the operations of the proposed protocol when the number of processing nodes is 11 is given in Fig. 4, where 0*** is the set to execute the broadcasting and the confirming phases, and the shuffling set associated is {1,000, 1,001, 1,010} that reaches the consensus in two steps according to G_1. Note that in Fig. 4 the number labeled in an arrow is the step number of that message in the protocol. A detailed description of G_2 can be found in [5], where the following proposition for the performance of G_2 is given.

Proposition 3. *For a system of p nodes with 1-port communication and $n = \lceil \log_2 p \rceil$, G_2 can reach the consensus with an initiator in the minimal number of steps, i.e., $2n$ steps, while incurring*

$$N_{G2}(p) = 2^n - 1 + \max\{2^{n-2}, p - 2^{n-1}\} + r\lceil \log_2 r \rceil + r - 2^{\lceil \log_2 r \rceil}$$

messages, where $r = p - 2^{n-1}$.

2.4 Decentralized Consensus Protocol without an Initiator (k-port)

In [6], the results for G_1, based on the partitioning tree and the generation of minimal complete sets, were extended to the case of k-port communication, i.e., each node is capable of sending k messages at a time. The extension to the k-port communication, referred to as algorithm G_3 in this paper, can be depicted in light of the *generalized n-dimensional c-ary hypercube* [2], where c is chosen to be $k+1$. As shown in [6], decentralized consensus protocols for 2-port communica-

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

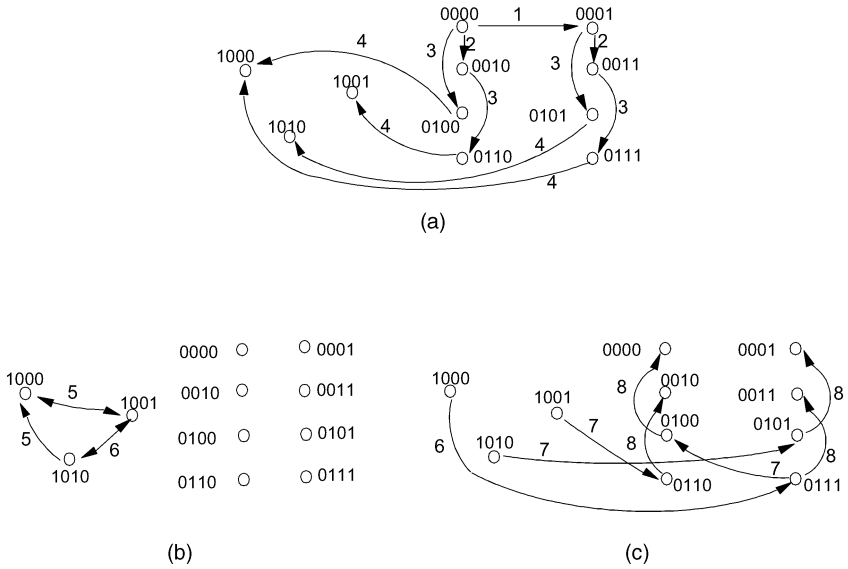


Fig. 4. The consensus protocol with an initiator when $\mathrm{p} = 11$. (a) Broadcasing, 11 messages. (b) Shuffling, five messages. (c) Confirming, eight messages.

tion can be developed from a ternary partitioning tree in such a way that each internal node in level i of the tree is taken as a minimal complete set generated after step i and the minimal complete set associated with an internal node is partitioned into those with its child nodes in one step. Essentially, based on generation of minimal complete sets in the corresponding partitioning tree, G_3 is applicable to the case of k-port communication. A procedure to build the optimal partitioning tree and an example for $p = 9$ are given in the Appendix A for interested readers. We then have the following two propositions for G_3 [6].

Proposition 4. *In a system of p nodes with k-port communication, the minimal number of steps required for all-to-all broadcasting is* $\lceil \log_{k+1} p \rceil$.

Proposition 5. *For a system of p nodes with k-port communication, the number of messages required by algorithm* G_3 *for all-to-all broadcasting in* $n = \lceil \log_{k+1} p \rceil$ *steps is,*

$$N_{G3}(p,k) = (d-2)n_1 p + (d-1)[n_2 p + p - (d-1)^{n_1} d^{n_2}],$$

where $n_1 + n_2 = n = \lceil log_{k+1} p \rceil$, *and* d *is the smallest positive integer such that* $p \leq (d-1)^{n_1} d^{n_2}$ *and* $p > (d-1)^{n_1+1} d^{n_2-1}$.

For example, consider the consensus protocol without an initiator in a system of 11 nodes with 3-port communication. Then, we have $p = 11$, $k = 3$ and $n_1 + n_2 = n = 2$, leading to $d = 4$, $n_1 = 2$ and $n_2 = 0$. We can obtain the optimal partitioning tree in Fig. 5a. It follows from Proposition 5 that $N_{G3}(11, 3) = 52$, agreeing with the scenario in Fig. 5.

2.5 Decentralized Consensus Protocol with an Initiator (k-port)

A decentralized consensus protocol with an initiator for k-port communication, referred to as G_4, was presented in [6]. Algorithm G_4 is similar to G_2 in that G_4 also has an initiator and the concept of three phases (i.e., the broadcasting phase, the shuffling phase, and the confirming phase), however, different from G_2 in that 1) the shuffling and confirming phases are overlapped in G_4 and 2) the operations in the shuffling phase of G_4 are based on those in G_3, rather than G_1.

As explained in [6], to achieve the minimal number of steps, it is not always necessary to use the maximal number of communication ports allowed. In fact, for some cases by not using the maximal number of communication ports, one can reduce the number of messages while still achieving the minimal number of steps. Let $n = \lceil \log_{k+1} p \rceil$, and h be the smallest number such that $\lceil \log_{h+1} p \rceil = n$. Clearly, h is the

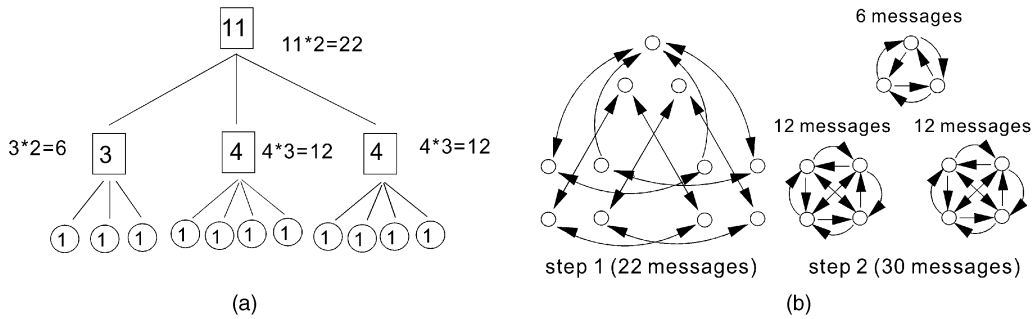


Fig. 5. The consensus protocol without an initiator for a 11 node system with 3-port communication. (a) Partitioning tree. (b) Consensus protocol in two steps.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

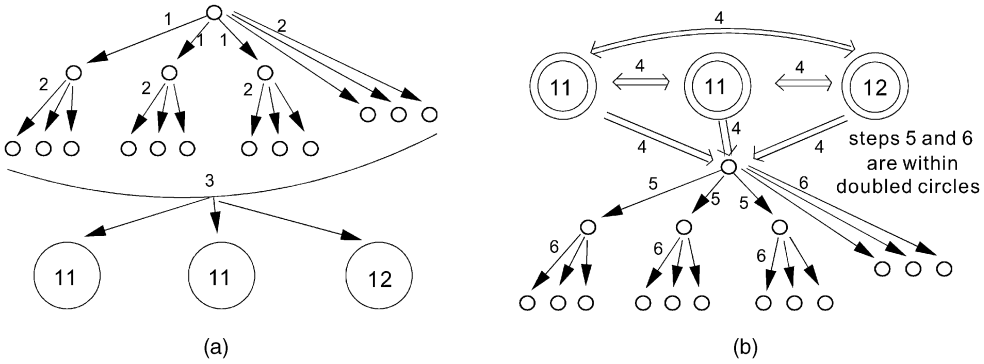


Fig. 6. The consensus protocol with an initiator for 3-port communication when $p = 50$. (a) Broadcasting (Step 1 to Step 3). (b) Shuffling and confirming (Step 4 to Step 6).

maximal number of ports needed to achieve the minimal number of steps. Conceptually, the p nodes in the system can be divided into two parts: 1) those forming the set for the broadcasting phase (containing $(h+1)^{n-1}$ nodes) and 2) those in the shuffling set (containing $r = p - (h+1)^{n-1}$ nodes). In essence, the broadcasting and the confirming phases are completed by the nodes in those $(h+1)^{n-1}$nodes whereas the shuffling phase is carried out by the nodes in the shuffling set. The operations of G_4 are outlined in the Appendix B and more details of G_4 can be found in [6]. From [6], we have the following proposition for the performance of G_4.

Proposition 6. *For a system of p nodes with k-port communication and $n = \lceil \log_{k+1} p \rceil$, G_4 can reach the consensus with an initiator in the minimal number of steps, i.e., $2n$ steps, while incurring*

$$N_{G4}(p,k) = 2(h+1)^{n-1} + r - 2 + \max\{(h+1)^{n-1} - (h+1)^{n-2}, r\} + N_{G3}(r,h)$$

messages, where h is the smallest number such that $\lceil \log_{h+1} p \rceil = n$, $r = p - (h+1)^{n-1}$ and $N_{G3}(r,h)$ is determined by Proposition 5.

The consensus protocol with an initiator for 3-port communication when $p = 50$ is given in Fig. 6. We then get $n = 3$, $h = 3$, and $r = 34$. The 3 steps of the broadcasting phase are shown in Fig. 6a. In Step 3, those 12 nodes receiving messages in Step 2 send messages to the shuffling set of 34 nodes, which are then divided into 3 sets of nodes, containing 11, 11 and 12 nodes, respectively, for the execution of the shuffling phase. In Fig. 6b, double arrows mean "group transmission" for clarity, and double circles denote the execution of consensus protocols without an initiator (based on G_3). The execution of the consensus protocol without an initiator for 11 nodes can be found in Fig. 5. It can be verified that the total number of messages incurred in Fig. 6 is 330, agreeing with Proposition 6.

3 Multilayered Decentralized Consensus Protocol

Note that though being able to perform the decentralized consensus protocol, G_4 in fact relies upon the execution of G_3 to handle its operations in the shuffling set. Since G_3 was originally designed for the protocol without an initiator, employing the operations in G_3 directly does not fully exploit the intrinsic property of the protocol with an initiator. Consequently, in light of the concept of multilayered execution, we shall develop an efficient decentralized protocol with an initiator, i.e., algorithm ML, in this section, and show that algorithm ML can significantly outperform algorithm G_4 by incurring a much smaller number of messages.

3.1 Description of Multilayered Decentralized Consensus Protocol

With its detailed operations presented later, the basic idea of algorithm ML is as follows: Similar to algorithm G_4, algorithm ML first has a broadcasting phase in which the initiator starts the protocol and informs all nodes to participate in the process. A shuffling set is formed in the end of the broadcasting phase. Note that as defined in Definition 2, the shuffling set is the minimal set that consists of all the information to reach the consensus. A node that has all the information is referred to as an *expert*. The very difference between algorithm ML and algorithm G_4 lies in their operations after the broadcasting phase. After the broadcasting phase, algorithm ML forms as many experts as needed from the shuffling set, and uses them to send the complete information back to other nodes. To facilitate our presentation of algorithm ML, we first introduce procedure STA (standing for Some-To-All broadcasting) below, which is essentially a procedure which algorithm ML will use to complete the consensus protocol after the above mentioned minimal complete set is formed in the end of the broadcasting phase.

Procedure STA: A some-to-all broadcasting protocol
/*Sending the id's of nodes in a set of *s* nodes, denoted by *S*, to all nodes of a system of $p = (k+1)^r$ nodes, where $r = \lceil \log_{k+1} s \rceil$ */.

Stmt 1. Build a balanced tree of height *r* as described by G_3 (which is given in the Appendix A). From level 0 to level $r-1$ of that tree, every node is an internal node of degree $k+1$.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

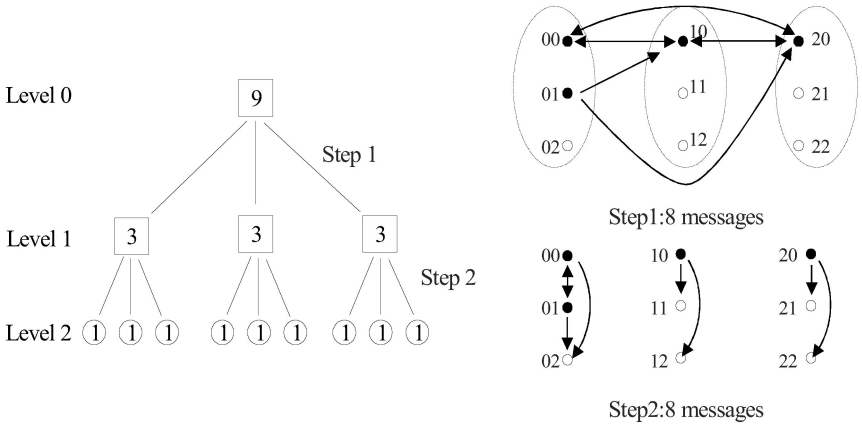


Fig. 7. An example of procedure STA when $p = 9,\ s = 4$ and $k = 2$.

Stmt 2. Perform the consensus protocol based on the generation of minimal complete sets in the partitioning tree in such a way that *from level 0 to level r* (which corresponds to *r* steps), *1) each minimal complete set contains at least one node in S and 2) only nodes in S will send messages.*

Fig. 7 is an example for a system of nine nodes where $s = 4$, $k = 2$ and $r = 2$. It can be seen that procedure STA incurs eight messages in the first step and eight messages in the second step. In the first step, three minimal complete sets are formed: one contains two nodes in *S* (i.e., nodes 00 and 01), and each of the other two sets contains only one node in *S* (i.e., node 10 and node 20). Procedure STA completes in the second step by forming nine minimal complete sets (i.e., every node is a complete set).

Theorem 1. *Suppose s is the number of nodes in a minimal complete set in a system of k-port communication. Procedure STA takes* $r = \lceil \log_{k+1} s \rceil$ *steps and* $r \cdot s \cdot k$ *messages to reach the consensus among* $p = (k+1)^r$ *nodes.*

Proof. It can be verified that Statement 2 of procedure STA is always feasible since $s \geq (k+1)^{r-1}$ and we can allocate at least one node in *S* for each minimal complete set in each step. The fact that procedure STA completes in *r* steps follows. On the other hand, in each step, each of these *s* nodes has to send a message to one node in each of its *k* sibling minimal complete sets. Thus, there will be at least $s \cdot k$ messages in each step, and the protocol will incur at least $r \cdot s \cdot k$ messages. In addition, note that in these *r* steps, only nodes in *S* will send messages, meaning that the number of messages in procedure STA is at most $r \cdot s \cdot k$. This theorem follows: □

With the introduction of procedure STA, algorithm ML can be described below.

Algorithm ML: The multilayered decentralized consensus protocol with an initiator for *k*-port communication. /* *p* is the total number of nodes in the system and $n = \lceil \log_{k+1} p \rceil$.*/

Stmt 1. In the first $n-1$ execution steps, use *k*-port communication to perform the broadcasting among $(k+1)^{n-1}$ nodes.

Smt 2. In the last step of the broadcasting phase, form a shuffling set of *s* nodes. Thus, we have

$$s = p - (k+1)^{n-1}.$$

Stmt 3. $p = (k+1)^n$ **then**
goto Statement 10
else

$$\begin{cases} y = \min\left\{ x \in N \,\middle|\, \sum_{i=1}^{x} k(k+1)^{n-i} \geq p, x \leq n \right\}, \text{where } N \text{ is the set of positive integers.} \\ r = \min\left\{ x \in N \,\middle|\, \sum_{i=1}^{y-1} k(k+1)^{n-i} + x(k+1)^{n-y} \geq p, x \leq k \right\}, \\ Z = \text{the shuffling set, and} \\ i = 1. \\ \textbf{if } s = 1 \textbf{ then} \text{ let the set } Z_A \text{ of a single node be the shuffling set, and } \textbf{goto} \text{ Stmt 9.} \\ \textbf{endif} \end{cases}$$

Stmt 4. if $(i = y)$ **then goto** Stmt 8 /* to determine the seed experts in the last layer */.

Stmt 5. Arbitrarily select a subset of *k* nodes from *Z*, and denote this subset as Z_A. /* Z_A contains *k* seed experts included in this run */
All nodes in *Z*, including Z_A, send their own id information to each node in Z_A, and nodes in Z_A also send their information to nodes in $Z - Z_A$.

Stmt 6. Each node in Z_A starts to send the information by a simple broadcast protocol with *k*-port communication subject to the condition that *nodes outside set Z are to receive the information first*.

Stmt 7. $Z = Z - Z_A$.
$i = i + 1$.
goto Stmt 4.

Stmt 8. Arbitrarily select a subset of *r* nodes from *Z*, and denote this subset as Z_A. /* Z_A contains *r* seed experts included in the last run of seed expert selection */ All nodes in *Z*, including Z_A, send their own information to Z_A.

Stmt 9. Each node in Z_A starts to send the information by a simple broadcast protocol with *k*-port communication subject to the condition that *nodes outside set Z are to receive the information first* **end** /* end of the protocol */

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

Stmt 10. if $(k = 1)$ **then**
Perform algorithm G_1 on the shuffling set in the next $n - 1$ execution steps. In the last step, send information from the shuffling set to the whole system.
else Perform procedure STA in the whole system, where those s nodes with the information to broadcast form the shuffling set.
endif
end /* end of the protocol */

For ease of description, we shall call those experts which are formed by nodes in Z "seed experts" so as to distinguish them from other experts. Note that these nodes could be arbitrarily chosen in the design stage as long as each node in the system knows the role of these nodes in advance. A non-seed expert mainly becomes an expert by receiving a single message of complete information from another expert. Algorithm ML can form as many as k seed experts in the first step after the broadcasting phase, and those seed experts can only send information to as many as $k(k+1)^{n-1} - k$ other nodes. If p is greater than $k(k+1)^{n-1}$, one needs to form more seed experts in the subsequent steps. Also, if algorithm ML forms another k seed experts in the second step after the broadcasting phase, then it can send information to as many as $k(k+1)^{n-1}+k(k+1)^{n-2}-2k$ other nodes. This procedure repeats until the remaining p is smaller than or equal to $k(k+1)^{n-1}+k(k+1)^{n-2}$. It can be seen that the instance for one node to receive the largest number of messages in one step occurs *either* in the last step of the broadcasting phase *or* the first step when forming the seed experts. Hence, the maximal number of messages one node will receive in one step is bounded by

$$\max\{\left\lceil \frac{(k+1)^{n-1}}{s} \right\rceil, s\},$$

where $s = p - (k+1)^{n-1}$ and $n = \lceil \log_{k+1} p \rceil$. It can also be verified that $k(y-1) + r$ seed experts will be formed in the entire protocol execution.

It is important to note that the parameter

$$y = \min\{x \in N | \sum_{i=1}^{x} k(k+1)^{n-i} \geq p, x \leq n\}$$

of algorithm ML corresponds to the number of message steps required to generate seed experts (from Statement 4 to Statement 8), and is in fact the *layer number* of the execution. Also, the parameter r is the number of seed experts selected in the last run of seed expert selection (Statement 8). Example consensus protocols that exploit different layers of execution are given in Section 3.2 for illustration. Also note that each node, after becoming an expert, is expected to be involved in sending messages to other nodes in the subsequent steps. That is the reason why we require the condition that "*nodes outside set Z are to receive the information first*" in Statement 6 and Statement 9 of algorithm ML. The following theorem states the correctness of algorithm ML.

Theorem 2. *For a system of p nodes with k-port communication and* $n = \lceil \log_{k+1} p \rceil$*, algorithm ML is able to complete the decentralized consensus protocol with an initiator in the minimal number of message steps.*

Proof. First, consider the case that $p = (k+1)^n$. In the broadcasting phase, algorithm ML forms a shuffling set of s nodes. Because $s = k(k+1)^{n-1}$, the whole system can achieve consensus by algorithm G_1 or procedure STA. Next, we discuss the case that $p < (k+1)^n$.

If $p = (k+1)^{n-1} + 1$, then after the broadcasting phase we have a simple broadcasting from the 1-node shuffling set. Now, consider the case that $p > (k+1)^{n-1}+1$. We first show the existence of parameters y and r. Clearly, the existence of

$$y = \min\left\{ x \in N \middle| \sum_{i=1}^{x} k(k+1)^{n-i} \geq p, x \leq n \right\}$$

follows from its definition and the fact that

$$\sum_{i=1}^{n} k(k+1)^{n-i} = (k+1)^n - 1 \geq p.$$

The fact that y is the minimal integer to satisfy

$$\sum_{i=1}^{y} k(k+1)^{n-i}$$

further leads to the existence of parameter r.

Second, we want to prove that in Statement 5 and Statement 8, there are always a sufficient number of nodes in Z to be selected. In other words, the size of the shuffling set is greater than or equal to the number of seed experts, i.e., $s \geq (y-1)k + r$. When $s = 2$ (i.e., $p = (k+1)^{n-1} + 1$, or $p = (k+1)^{n-1} + 2$), $(y, r) = (1, 2)$ for $k \neq 1$, and $(y, r) = (2, 1)$ for $k = 1$, both leading to $(y-1)k + r = s$. Clearly, as s increases, y and r grow much slower than s, meaning that $(y-1)k + r$ is always smaller than or equal to s.

Finally, it can be seen that the inequality: $p - num\_expert \geq s$, where num_expert is the number of experts after the $2n-1$ execution steps, ensures that nodes in the shuffling set do not have to become non-seed experts in the first $2n-1$ steps. That is, every seed expert needed can be simply formed from the shuffling set in accordance with a proper number of layer execution. To prove the inequality, note that since all experts are formed by the broadcasting of some seed experts and there are at most k experts generated in each execution step after the broadcasting phase. Thus, we have

$$num\_expert \leq \sum_{i=2}^{n} k(k+1)^{n-i},$$

leading to

$$p - num\_expert \geq p - \sum_{i=2}^{n} k(k+1)^{n-i} =$$
$$p - (k+1)^{n-1} + 1 > s.$$

Consequently, the correctness of algorithm ML follows.□

Theorem 3. *The number of the total messages required by algorithm ML is:*

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

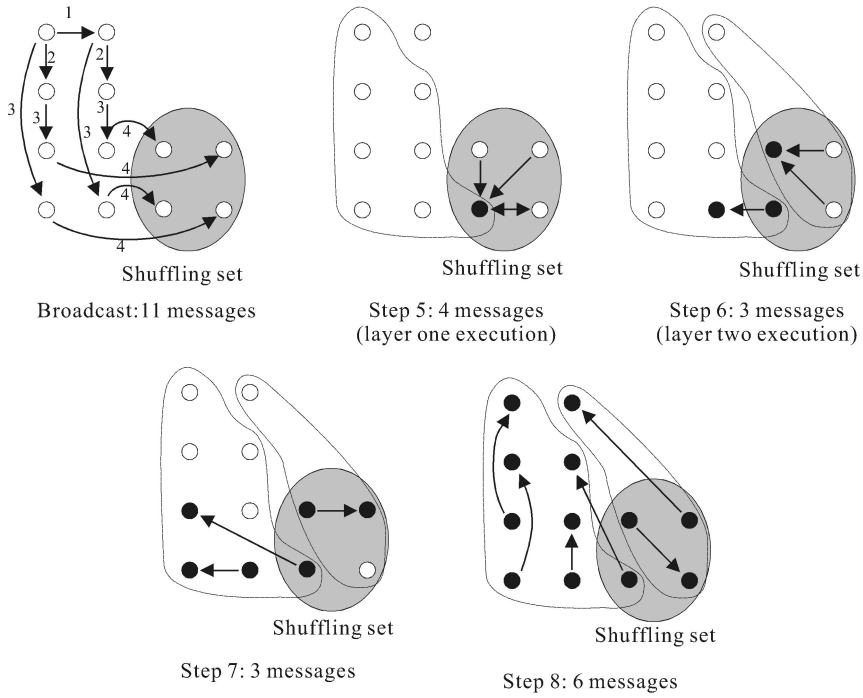


Fig. 8. An example of 2-layer execution of algorithm ML for a 12-node system with 1-port communication.

$$N_{ML}(p,k)=\begin{cases} p-1+nsk & \text{when } p=(k+1)^n, \\ 2p-3+k(k+1)^{n-2} & \text{when } p=(k+1)^{n-1}+1, \\ p-1+\max\left(s,k(k+1)^{n-2}\right)-s+((y-1)k+r)(s-1) & \\ \quad +p-\frac{1}{2}k(y-1)(ky-2k+2r)-r, & \text{otherwise,} \end{cases}$$

where p, n, s, y, and r are as defined in algorithm ML.

Proof. If $p=(k+1)^n$, algorithm ML takes $p-1$ messages in the broadcasting phase. If $k\neq 1$, algorithm ML incurs $n\cdot k\cdot s$ messages to perform procedure STA for the whole system, otherwise, algorithm ML performs G_1 for the shuffling set, and sends information to the other 2^{n-1} nodes outside the shuffling set. Hence, the total number of messages needed is $p-1+(n-1)s+(k+1)^{n-1}$, when $k=1$ and is $p-1+nsk$, when $k\neq 1$, both leading to the first formula of $N_{ML}(p,k)$.

If $p\neq(k+1)^n$, algorithm ML takes $p-1+\max(s,k(k+1)^{n-2})-s$ messages to inform every node to participate in the protocol and to form the shuffling set. If $s=1$, after the broadcasting phase, it needs $p-1$ messages sent from the shuffling set to others to complete the protocol. If $s\neq 1$, the number of messages required in Statement 5 of algorithm ML is

$$\left(\sum_{i=0}^{y-2}(s-i\times k)k\right),$$

that in Statement 8 is $((s-(y-1)k)r)-r$, and Statement 6 and Statement 9, as a whole, incur $p-(y-1)k-r$ messages. Hence, the total number of messages needed is $p-1+\max(s,k(k+1)^{n-2})-s+p-1$, when $s=1$, and is

$$p-1+\max(s,k(k+1)^{n-2})-s+\left(\sum_{i=0}^{y-2}(s-i\times k)k\right)+(s-(y-1)k)r+p-k(y-1)-2r,$$

otherwise, leading to the last two formulas of $N_{ML}(p,k)$ above. □

3.2 Illustrative Examples

Examples are given in this subsection to illustrate the operations of algorithm ML. Explicitly, illustration for multilayered execution is presented in Section 3.2.1 and that for multiport communication is given in Section 3.2.2.

3.2.1 Illustration for Multilayered Execution

Consider a system of 12 nodes with 1-port communication in Fig. 8. Since $p=12$ and $k=1$, we obtain $n=\lceil\log_2 12\rceil=4$ and $s=12-2^3=4$. From Step 1 to Step 4, the initiator informs every node of the system to participate in the protocol, and forms the shuffling set of s nodes. Because $p=2^3+2^2$, we obtain the layer number

$$y=\min\left\{x\in N\left|\sum_{i=1}^{x}2^{n-i}\geq 12, x\leq n\right.\right\}=2,$$

and $r=\min\{x\in N|8+4x\geq p, x\leq k\}=1$. Hence, we have $y=2$ layers and also $(y-1)\cdot k+r=(2-1)\cdot 1+1=2$ seed experts. As shown in Fig. 8, these two seed experts are formed in two layers, i.e., one in Step 5 and the other in Step 6. As it can be seen from Step 5 of Fig. 7, in addition to the messages that are needed to form the seed expert, the seed expert also sends its own information to its neighbors in the shuffling set. This is because of the fact that in Step 6, 7, and 8, that seed expert is dedicated to broadcasting its information. Hence, in Step 6, algorithm ML forms another seed expert, resulting in a 2-layer execution scenario in

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

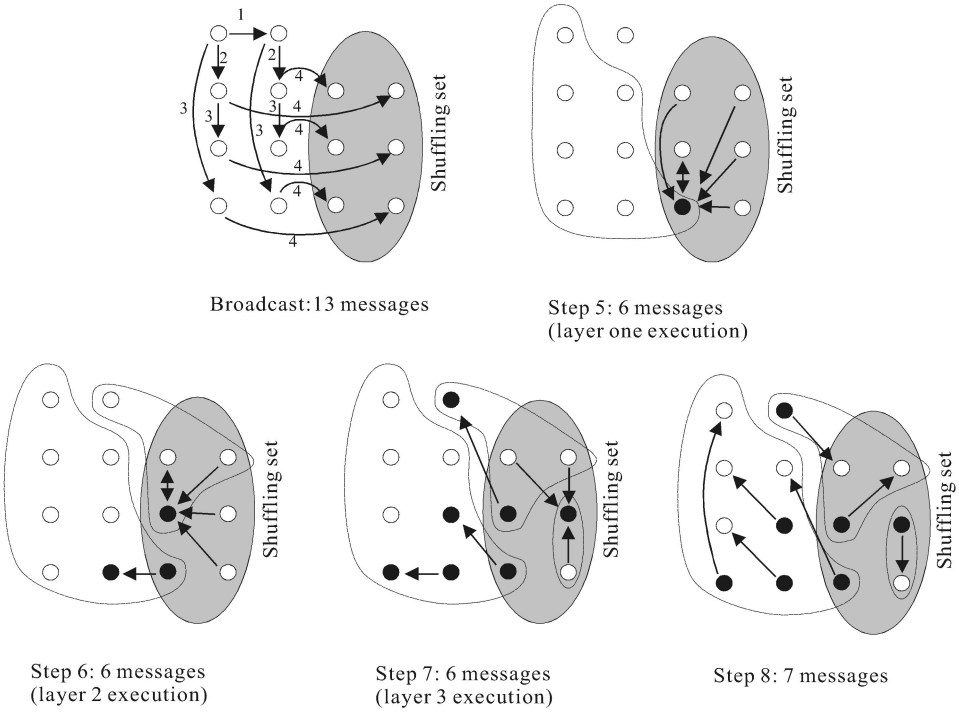


Fig. 9. An example of 3-layer execution of algorithm ML for a 14-node system with 1-port communication.

Fig. 8. Because the corresponding protocol takes eight steps, the seed expert formed in Step 5 can broadcast its information to other sevem nodes in Step 6, 7, and 8, and the one formed in Step 6 can broadcast its information to other three nodes, thus completing the protocol.

Fig. 9 is an example for a 3-layer execution of algorithm ML. Since $p = 14$ and $k = 1$ in this case, we obtain $n = \lceil \log_2 14 \rceil = 4$, and $s = 14 - 2^{4-1} = 6$. In the broadcasting phase, algorithm ML incurs 13 messages. Also, we get

$$y = \min\{x \in N | \Sigma_{i=1}^{x} 2^{n-i} \geq 14, x \leq 4\} = 3,$$
$$r = \min\{x \in N | 12 + 2x \geq p\} = 1,$$

i.e., algorithm ML needs one seed expert in each step from Step 5 to Step 7. In Step 5, algorithm ML incurs $5 = |Z| - 1$ messages to form an expert node and one message to maintain the completeness of set $Z - Z_A$. In Step 6, algorithm ML takes $|Z| - 1 = 4$ messages to form a new seed expert and another message to keep the completeness of set $Z - Z_A$. In Step 7, the formation of the last seed expert requires $|Z| - 1 = 3$ messages. Besides those three seed experts, every node in the system involves one message to become an expert. As a result, algorithm ML incurs a total number of $13 + 6 + 5 + 3 + (14 - 3) = 38$ messages for the 3-layer execution in Fig. 9. An example for algorithm ML to incur 4-layer execution is shown in Fig. 10 for illustrative purposes.

3.2.2 Illustration of Multiport Communication

To illustrate the operations of algorithm ML for multiport communication, we consider the system of 12 nodes with 2-port communication in Fig. 11. In this case, $p = 12$, $k = 2$, $n = 3$, $y = 1$, $r = 2$, and $s = 3$. Thus, the scenario in Fig. 11 is a 1-layer execution of algorithm ML for 2-port communication. In the broadcasting phase, algorithm ML incurs 14 messages to inform every node to participate in the protocol and to form the shuffling set. After the broadcasting phase, algorithm ML sends back the information from each node of the shuffling set to all the others. In Step 4, two seed experts are formed. In Step 5 and Step 6, those two seed experts send their information to other nodes by a broadcast protocol. In all, algorithm ML incurs a total number of

$$12 - 1 + \max(3, 6) - 3 + ((1 - 1) \times 2 + 2) \times (3 - 1) +$$
$$12 - \frac{1}{2} 2(1 - 1)(2 - 4 + 4) - 2 = 28$$

messages to complete the protocol, agreeing with $N_{ML}(12, 2) = 28$ as determined by Theorem 3.

Consider a 50-node system with 2-port communication in Fig. 12. The system is the same as the one in Fig. 6. In this case, $n = 3$ and $s = 34$. In the broadcasting phase (from Step 1 to Step 3), algorithm ML takes 49 messages to notify every node to participate in the protocol, while forming a shuffling set of 34 nodes. Also, we obtain $y = 2$, and $r = 1$. In Step 4, we select three nodes to be seed experts while incurring $31 \times 3 + 3 \times 2 = 99$ messages. Those seed experts need to send their own information to any other node in the shuffling set to ensure the completeness in Z, which process involves three messages. Thus, algorithm ML takes 102 messages in Step 4. In Step 5, to make another seed expert from set Z, algorithm ML incurs 30 messages. At the same time, the expert nodes formed in the previous step send their information to other nine nodes, while incurring nine messages. In Step 5, algorithm ML thus incurs 39 messages. Finally, in Step 6, we employ the 13 expert nodes formed earlier to send their information to other 37 nodes. In this case, algorithm ML incurs a total number of 227 messages to complete the

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

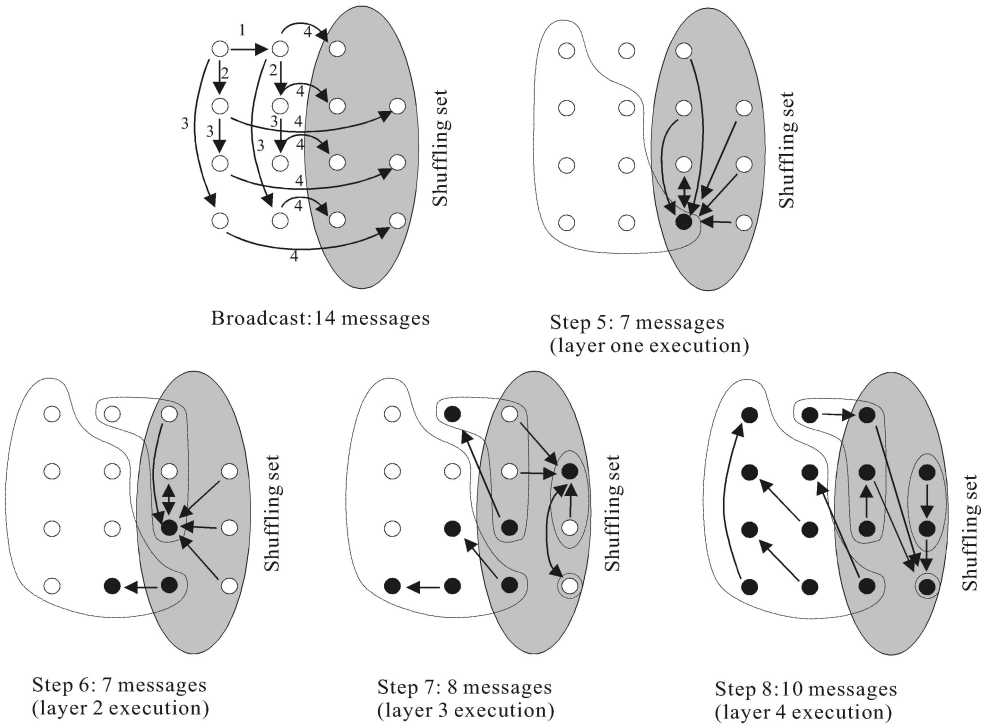


Fig. 10. An example of 4-layer execution of algorithm ML for a 15-node system with 1-port communication.

protocol, agreeing with Theorem 3. It can be verified that $N_{G4}(50,2) = 330$, which is significantly larger than the message number required by algorithm ML (i.e., $N_{ML}(50,2) = 227$), showing the advantage of algorithm ML.

4 Performance Analysis

In this section, some theoretical results of algorithm ML are derived in Section 4.1. Performance of algorithm ML and algorithm G_4 will be comparatively analyzed in Section 4.2. It is shown that algorithm ML significantly outperforms algorithm G_4 and the performance improvement achieved by algorithm ML increases as the number of nodes increases. Specifically, the ratio of $N_{ML}(p,k)$ to $N_{G4}(p,k)$ is proved to be of complexity $O(\log^{-1} p)$ which approaches zero as the number of nodes p becomes large. Algorithm ML will

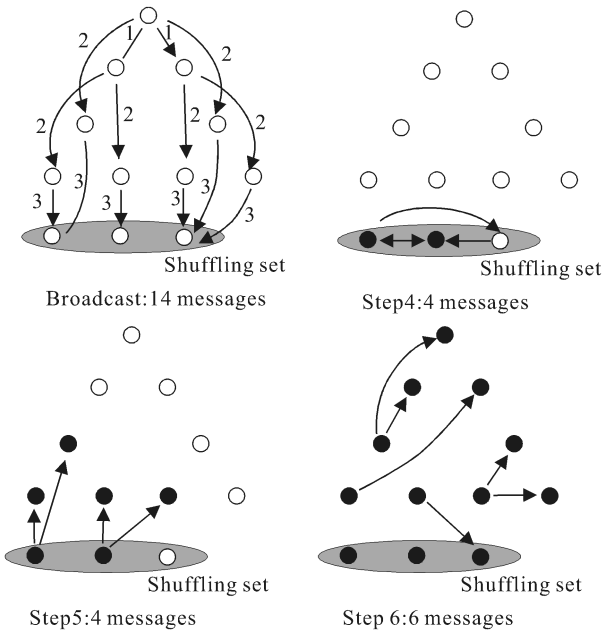


Fig. 11. An example of 1-layer execution of algorithm ML for a 12-node system with 2-port communication.

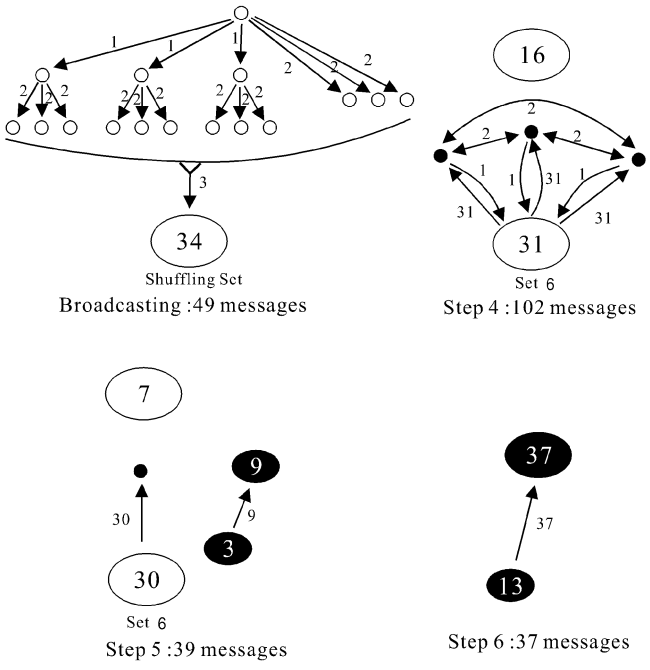


Fig. 12. An example of 2-layer execution of algorithm ML for a 50-node system with 2-port communication.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

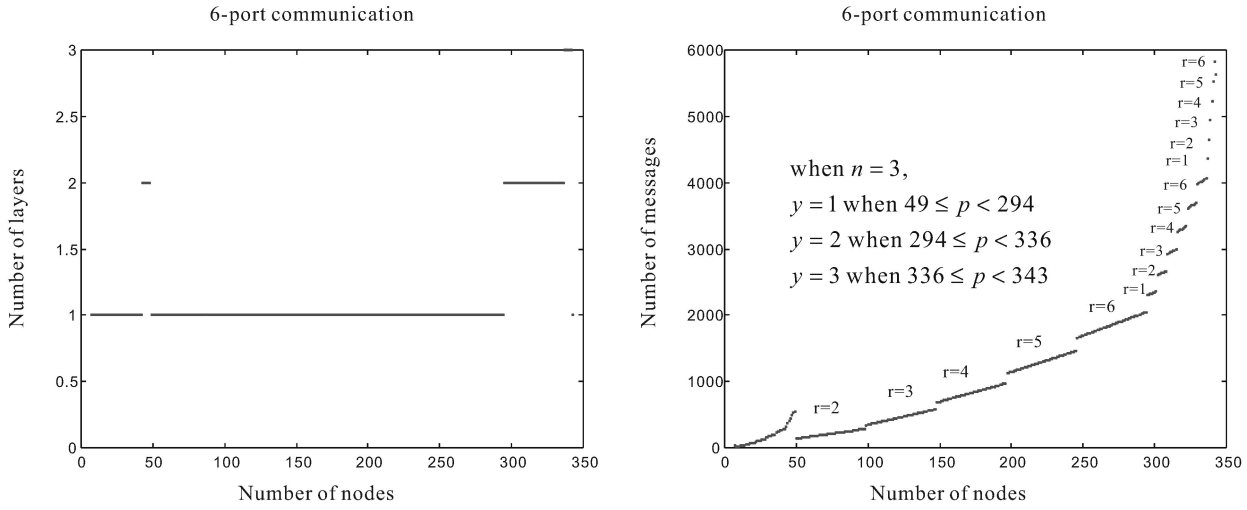


Fig. 13. Two plots for the analysis of algorithm ML with 6-port communication: (a) number of layers versus number of nodes. (b) Number of messages versus number of nodes.

furthermore be proved to be asymptotically optimal in the sense that the message number required by algorithm ML and its corresponding theoretical minimum are asymptotically of the same complexity with respect to the number of nodes in the system, indicating the very advantage of utilizing the concept of multilayer for protocol execution.

4.1 Theoretical Insights into the Decentralized Consensus Protocols

To provide more insights into algorithm ML, consider the two plots in Fig. 13, where the relationship between the number of layers y and the number of nodes p is shown in Fig. 13a, and the number of messages that are needed in algorithm ML for 6-port communication is shown in Fig. 13b.

Recall that in algorithm ML, $k(y-1)+r$ is the number of seed experts needed for protocol execution. However, forming more experts will cause a gap in the number of messages. Specifically, such gaps occur whenever we need another seed expert in the protocol. Explicitly, as illustrated in Fig. 13b, such gaps occur when

$$p = \sum_{i=1}^{y-1} k(k+1)^{n-i} + r(k+1)^{n-y},$$

where $y \leq n$, $r \leq k$, and $(y,r) \neq (1,1)$. Note that each seed expert formed at the x-th step after the broadcasting phase can send the complete information to as many as $(k+1)^{n-x}-1$ nodes. Thus, as shown in Fig. 13b, the segment of the curve that begins near 50 nodes corresponds to the case that the system needs two seed experts, and this segment ends when $p = 2 \times (6+1)^{3-1} = 98$ nodes. Similarly, the next segment of the curve corresponds to the case that three seed experts are needed, etc. It can be verified that the first five segments of the curve in Fig. 13b correspond to the case of 1-layer execution (i.e., with two to six experts), the next six segments correspond to the case of 2-layer execution (i.e., with seven to 12 experts), and the following six segments correspond to the case of 3-layer execution.

With the illustrative example in Fig. 13b, by omitting their straightforward proofs, the following properties of algorithm ML can be derived from Theorem 3.

Corollary 3.1. *The systems in the same segment of the curve have the same parameters y and r.*

Corollary 3.2. *The slope of each segment of the curve is 1) $((y-1)k+r)+2$, for $s \geq k(k+1)^{n-2}$, and 2)*

$$((y-1)k+r)+1,$$

for $s < k(k+1)^{n-2}$. Also, the effect of the singular function max *occurs in the first segment of the curve. Thus, each segment of the curve except the first one is a straight line with a slope equal to $((y-1)k+r)+2$.*

Corollary 3.3. *A y-layer execution of the consensus protocol involves k segments of the curves for $y>1$, and $k-1$ segments for $y=1$. Every segment of y-layer execution covers $(k+1)^{n-y}$ nodes.*

Next, we derive some theoretical insights into the decentralized consensus protocols studied. Define $M(s,p,k)$ as the minimal number of messages that are needed for a subset of s nodes in a system of p nodes for sending their id information in this s-node subset to all other nodes with k -port communication in the minimal number of steps. For example it is easy to verify that $M(2,4,1)=4$ as shown in the following figure.

We then have the following lemma for $M(s,p,k)$.

Lemma 1.

$$M(s+1,p,k) \geq M(s,p,k) + \left\lceil \frac{p}{(k+1)^{n-1}} \right\rceil - 1.$$

Proof. When sending information from $s+1$ nodes to p nodes, each of the $s+1$ nodes needs to send

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

$$\left\lceil \frac{p}{(k+1)^{n-1}} \right\rceil - 1$$

messages in the first step so as to multiply its own id by

$$\left\lceil \frac{p}{(k+1)^{n-1}} \right\rceil$$

times. Otherwise, it can be shown that the id's of those nodes cannot be delivered to all nodes in the minimal number of steps.

Since it sends information from only s nodes now, there exists one node that does not need to send information in the first step. From the definition of $M(s,p,k)$, we have

$$M(s+1,p,k) \geq M(s,p,k) + \left\lceil \frac{p}{(k+1)^{n-1}} \right\rceil - 1.$$

□

From Lemma 1, a lower bound of $M(s,p,k)$ follows:

Lemma 2. $M(s,p,k) \geq p - s + (s-1)\left\lceil \frac{p}{(k+1)^{n-1}} \right\rceil$.

Theorem 4. *For a decentralized consensus protocol with an initiator in a system of p nodes with k-port communication, when $s > k(k+1)^{n-2}$, the minimal number of messages required to complete the protocol in the minimal number of steps, i.e., n steps, is $M(s,p,k) + p - 1$, where $n = \lceil \log_{k+1} p \rceil$ and $s = p - (k+1)^{n-1}$.*

Proof. When $s > k(k+1)^{n-2}$, we need at least $p-1$ messages in the broadcasting phase to notify every node to participate in the protocol, and to form a shuffling set of s nodes. With k-port communication, we can notify at most $(k+1)^{n-1}$ nodes after step $n-1$. That is, at least $s = p - (k+1)^{n-1}$ nodes are notified in the last step of broadcasting phase. Since each node of the shuffling set is notified in the last step of the broadcasting phase, it cannot send its own id information to other nodes in the broadcasting phase, meaning that it is necessary to send back the information from the s nodes to all the p nodes.

Because $M(s+1,p,k) > M(s,p,k)$, we know that the smaller the shuffling set is, the smaller number of messages we need in sending the information back. From this phenomenon and the fact that $p-1$ messages are needed in the broadcasting phase, this theorem follows.□

4.2 Performance Comparison between Algorithm ML and Prior Results

In this section, we shall prove that algorithm ML will significantly outperform prior schemes for the case of having an initiator, and the performance improvement achieved by algorithm ML increases as the number of nodes increases. Comparison results between algorithm ML and G_2 are given first and general performance comparison between algorithm ML and G_4 is then conducted. As mentioned before, the ratio of the average number of messages incurred by algorithm ML to that by algorithm G_4 is proved to be of complexity $O(\log^{-1} p)$ which approaches zero as the number of nodes p becomes large, and algorithm ML is in fact asymptotically optimal in the sense that $N_{ML}(p,k)$ and its corresponding theoretical minimum are asymptotically of the same complexity with respect to the number of nodes in the system.

The following theorem compares the performance of algortihm ML and that of G_2.

Theorem 5. *When $p > 16$, algorithm ML in general outperforms algorithm G_2 by incurring fewer messages for protocol execution, i.e., $N_{ML}(p,1) \leq N_{G2}(p)$, except the three cases when $p = 2^n - 1$, $p = 2^n - 2$, and $p = 2^n - 3$, where $n = \lceil \log_2 p \rceil$.*

Proof. First, it can be shown that when $p = 2^n$, or $p = 2^{n-1} + 1$, $N_{ML}(p,1) = N_{G2}(p)$. Also, the numbers of messages in the broadcasting phase of the two algorithm are the same. Note that the formula of the number of messages in the shuffling phase and confirming phase of algorithm G_2 can be reduced to $rs + (p - 2^r)$, and that of algorithm ML can be reduced to

$$ys + p - \frac{y^2 + y + 2}{2},$$

where $n, s,$ and y are as defined in algorithm ML, and $r = \lceil \log_2 s \rceil$.

Consider the following two situations:

1. When $p \leq 2^{n-1} + 2^{n-2}$, i.e. $y = 2$, the two formulas for algorithm G_2 and algorithm ML become $rs + (p - 2^r)$ and $2s + p - 4$, respectively. It can be verified that when $s = 2,\ 3,$ or 4, the values of the two formulas are the same. However, when $s > 4$, we have $rs + (p - 2^r) > 2s + p - 4$, meaning that algorithm ML incurs fewer messages than algorithm G_2.
2. When $p > 2^{n-1} + 2^{n-2}$, say, $p = 2^n - x$ and $x < 2^{n-2}$, the formula of algorithm G_2 becomes $n(2^{n-1} - x)$ and that of algorithm ML becomes

$$(y+2)(2^{n-1} - x) + x - \frac{y^2 + y + 2}{2}.$$

   a. When $y + 2 < n$, the number of messages needed in algorithm G_2 is larger than that in algorithm ML since

$$(n - (y+2))(2^{n-1} - x) - x + \frac{y^2 + y + 2}{2} > 2^{n-1} - 2x > .$$

   b. When $y + 2 = n$, i.e., $4 \leq x < 8$, because $y = n - 2 \geq 3,$ we get

$$\frac{y^2 + y + 2}{2 \geq} 7,$$

   leading to

$$x - \frac{y^2 + y + 2}{2} \leq 0.$$

   Thus, the number of messages in algorithm G_2 is larger than or equal to that in algorithm ML.

   c. When $y + 2 > n$, the number of messages in algorithm ML is larger than that in algorithm G_2 since

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

$$(y+2-n)(2^{n-1}-x)+x-\frac{y^2+y+2}{2}$$
$$\geq 2^{n-1}-\frac{n(n-3)+4}{2}>0.$$

Since the case (c) above is only valid for the three cases when $p=2^n-1$, $p=2^n-2$, and $p=2^n-3$, this theorem follows. □

Similarly to the proof of Theorem 5, it can also be shown that algorithm ML in general outperforms algorithm G_4 by incurring fewer messages in the protocol execution in the case of *k*-port communication. For illustrative purposes, the values of $N_{ML}(p,2)$ and $N_{G4}(p,2)$ are shown in Fig. 14 where the number of nodes varies from $28=(2+1)^3+1$ to $81=(2+1)^4$.

Moreover, numbers of messages required by algorithm ML and algorithm G_4 in various cases are given in Fig. 15 for interested readers. It can be seen from Fig. 15 that the number of messages in algorithm ML is in general much smaller than that required by algorithm G_4. Instead of detailing each individual case, we shall prove that the performance improvement by algorithm ML increases as the number of nodes increases. Specifically, the ratio of the average number of messages incurred by algorithm ML to that by algorithm G_4 approaches zero as the number of nodes becomes large.

First, define the ration function $R(n,K)$ as:

$$R(n,k)=\frac{\frac{1}{k(k+1)^{n-1}}\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n}N_{ML}(p,k)}{\frac{1}{k(k+1)^{n-1}}\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n}N_{G4}(p,k)},$$

where $n=\lceil\log_{k+1}p\rceil$. Clearly, the smaller the value of $R(n,k)$ is, the more improvement achieved by algorithm ML over algorithm G_4. We then have the following important theorem which states that the improvement of algorithm ML over algorithm G_4 increases as the number of nodes p increases.

Theorem 6. $\lim_{n\to\infty}R(n,k)=\frac{1}{a\times n+b}$, *where* a *and* b *are functions of k,* $n=\lceil\log_{k+1}p\rceil$ *and p is the number of nodes in the system. That is, the complexity of* $R(n,k)$ *is* $O(\log^{-1}p)$.

Proof. First, in light of

$$N_{G3}(p,k)=(d-2)n_1p+(d-1)[n_2p+p-(d-1)^{n_1}d^{n_2}],$$

when p approaches infinity, $n=n_1+n_2$ will also approach infinity. Thus we have

$$k^n<<(k+1)^{n-1}\leq p\approx(d-1)^{n_1}\times d^{n_2}\leq(k+1)^n,$$

which implies that $d=k+1$, $n_2\gg n_1$ and $n_2\approx n$. Otherwise, the inequality will not hold. Then, we have $N_{G3}(p,k)\approx nkp$.

Recall that

$$N_{G4}(p,k)=2(h+1)^{n-1}+s-2+$$
$$\max\{(h+1)^{n-1}-(h+1)^{n-2},s\}+N_{G3}(s,h),$$

where *s* is the size of the shuffling set. When *n* is enough large, *h* is close to *k*. Also, since we shall consider the average number of messages only, we can assume that *s* is large enough to be taken in the previous approximation of $N_{G3}(p,k)$. $N_{G3}(s,k)$ can thus be approximated as rks, where $r=\lceil\log_{k+1}s\rceil$. $N_{G4}(p,k)$ can in turn be approximated as

$$2(k+1)^{n-1}+s+\max\Big\{k(k+1)^{n-2},s\Big\}+krs.$$

Next,

$$N_{ML}(p,k)=p-1+\max\Big(s,k(k+1)^{n-2}\Big)$$
$$-s+((y-1)k+r)(s-1)+p$$
$$-\frac{1}{2}k(y-1)(ky-2k+2r)-r$$

can also be approximated as

$$2(k+1)^{n-1}+s+\max\Big\{k(k+1)^{n-2},s\Big\}+yks+\delta s,$$

the term δs corresponds to the average effect of the term rs in the original expression. δ is thus a constant in the range $1\leq\delta\leq k$. From the definition of

$$y=\min\left\{x\in N\left|\sum_{i=1}^{x}k(k+1)^{n-i}\geq p,x\leq n\right.\right\},$$

we can approximate y as

$$n-\log((k+1)^n-p+1)=n-\log\Big(k(k+1)^{n-1}-s+1\Big).$$

Based on the foregoing, we can derive the following approximation for function $R(n,k)$:

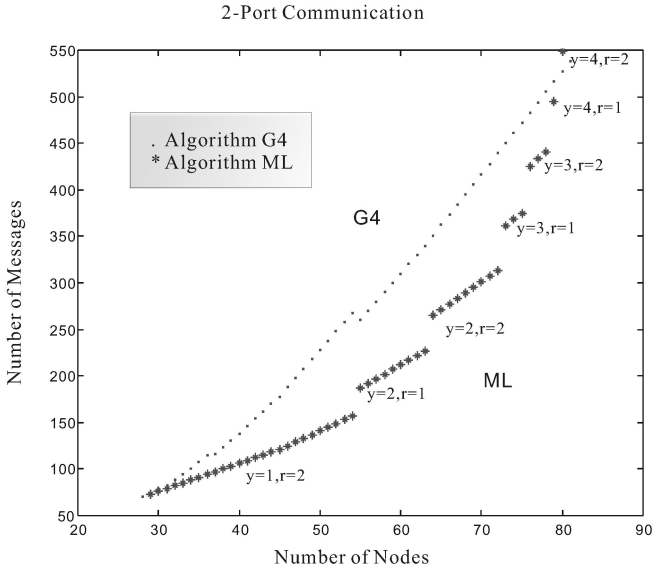


Fig. 14. The number of messages incurred by algorithm ML and algorithm G_4 for 2-port communication when the number of nodes varies.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

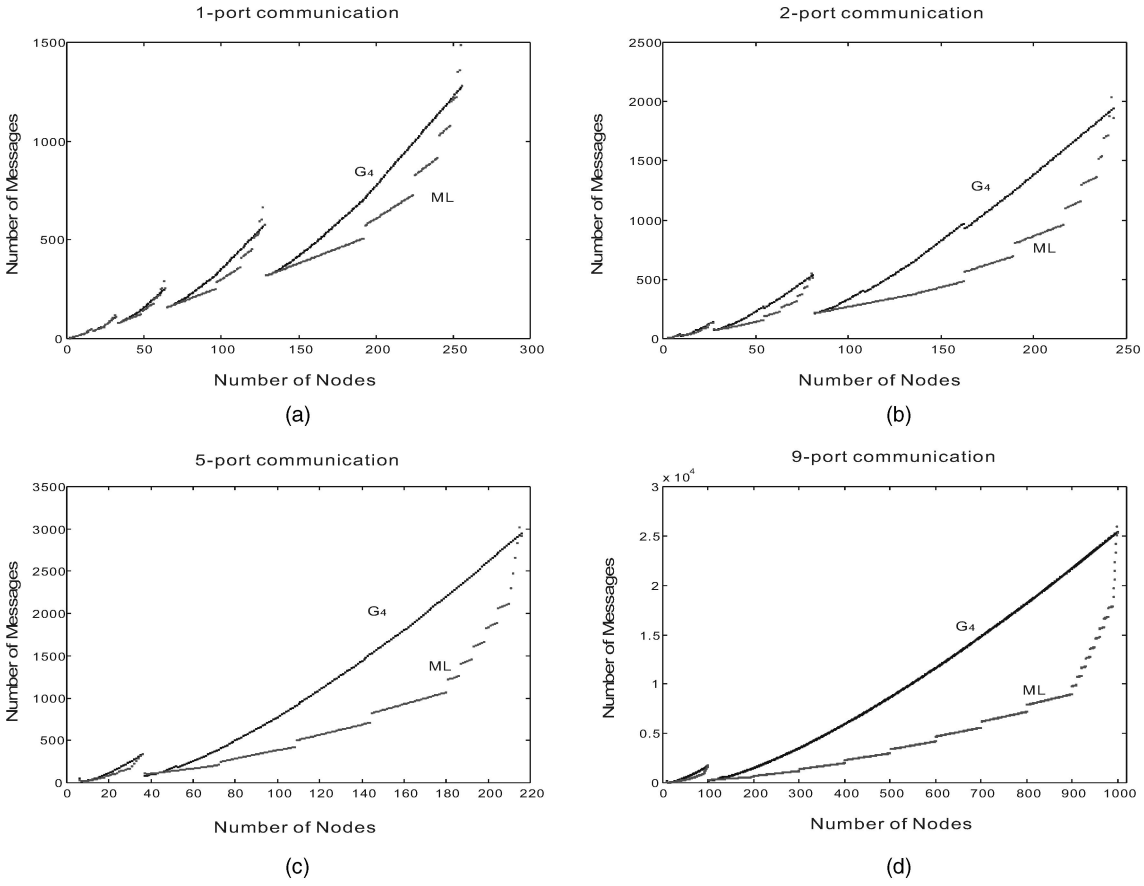


Fig. 15. Numbers of messages required by algorithm ML and algorithm G_4 in various cases: (a) 1-port, (b) 2-port, (c) 5-port, and (d) 9-port.

$$\lim_{n\to\infty} R(n,k) \approx \frac{\int_1^t \Big[2(k+1)^{n-1} + \max\Big\{k(k+1)^{n-2}, s\Big\} + s + kys + \delta s\Big] ds}{\int_1^t \Big[2(k+1)^{n-1} + \max\Big\{k(k+1)^{n-2}, s\Big\} + s + krs\Big] ds},$$

where $t = k(k+1)^{n-1}$.

We approximate the formula in the numerator first. Using the integration equality

$$\int_0^a [x \log x] dx = \frac{1}{4} a^2 (2 \log a - 1),$$

we have

$$\begin{aligned}
&\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{G4}(p,k) \approx \\
&\int_1^t \Big[2(k+1)^{n-1} + \max\Big\{k(k+1)^{n-2}, s\Big\} + s + krs\Big] ds \\
&= \int_1^t \Big[2(k+1)^{n-1} + s\Big] ds + \int_1^{k(k+1)^{n-2}} \Big[k(k+1)^{n-2}\Big] ds + \\
&\int_{k(k+1)^{n-2}+1}^t [s] ds + \int_1^t \big[k \times \big(\log_{k+1} s\big) \times s\big] ds \\
&= \alpha \cdot t^2 \log t + \beta \cdot t^2 + \varepsilon_1(n,k),
\end{aligned}$$

where α and β are functions of k, and $\lim_{t\to\infty} \frac{\varepsilon_1(n,k)}{t^2} = 0$. Similarly, using the integration equality

$$\int_0^{a-1} [(\log a - \log(a-x))x] dx = \frac{1}{4}\big(1 - 4a + 3a^2 + (2-4a)\log a\big),$$

the formula in the denominator can be approximated as below.

$$\begin{aligned}
&\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{ML}(p,k) \\
&\approx \int_1^t \Big[2(k+1)^{n-1} + \max\Big\{k(k+1)^{n-2}, s\Big\} + s + kys\Big] ds \\
&= \int_1^t \Big[2(k+1)^{n-1} + s\Big] ds + \int_1^{k(k+1)^{n-2}} \Big[k(k+1)^{n-2}\Big] ds + \\
&\int_{k(k+1)^{n-2}+1}^t [s] ds + \int_1^t \big[k \times \big(n - \log_{k+1}(t-s)\big) \times s\big] ds \\
&= \gamma t^2 + \varepsilon_2(n,k),
\end{aligned}$$

where γ is a function of k, and $\lim_{t\to\infty} \frac{\varepsilon_2(n,k)}{t^2} = 0$.

From these two approximations, we obtain

$$\lim_{n\to\infty} R(n,k) = \frac{\gamma}{\alpha \log t + \beta} = \frac{1}{an+b},$$

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

where a and b are functions of k and this theorem follows. □

From Proposition 6 and Theorem 3, values of $R(n,k)$, for $k=1,2,3,$ and $4,$ can be obtained and plotted in Fig. 16, whose results agree with Theorem 6. It can be seen from Fig. 16 that the performance improvement of algorithm ML over algorithm G_4 increases as the value of $n=\lceil \log_{k+1} p \rceil$ increases.

To provide insights into the optimality of algorithm ML, denote the minimal number of messages required by a decentralized consensus protocol with an initiator for k-port communication as $N_{OP}(p,k)$. Despite deriving the exact formula of $N_{OP}(p,k)$ is still an open problem, the following theorem states that with a given number of port k, $N_{ML}(p,k)$ and $N_{OP}(p,k)$ are asymptotically of the same complexity with respect to the number of nodes in the system p.

Theorem 7. *With the ratio function*

$$R^*(n,k)=\frac{\frac{1}{k(k+1)^{n-1}}\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{OP}(p,k)}{\frac{1}{k(k+1)^{n-1}}\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{ML}(p,k)},$$

we have $1 \geq \lim_{n\to\infty} R^*(n,k) \geq c$, *where c is a function of k.*

Proof. From the definition of $N_{OP}(p,k)$, we have $1 \geq \lim_{n\to\infty} R^*(n,k)$. We next prove the inequality in the right-hand side. Since we need at least $p-1$ messages to inform every node to participate in the protocol and to form a shuffling set of at least $p-(k+1)^{n-1}$ nodes, where $n=\lceil \log_{k+1} p \rceil$. With the lower bound of $M(s,p,k)$ in Lemma 2, we get a lower bound of $N_{OP}(p,k)$ as

$$2p-1-s+(s-1)\left\lceil \frac{p}{(k+1)^{n-1}} \right\rceil,$$

which is in turn reduced to $2p+s-3$ since $\left\lceil \frac{p}{(k+1)^{n-1}} \right\rceil \geq 2$. Hence, we have,

$$\begin{aligned}\lim_{n\to\infty} R^*(n,k) &= \frac{\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{OP}(p,k)}{\sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{ML}(p,k)} \\ &\geq \lim_{t\to\infty} \frac{\int_1^t \left[2(k+1)^{n-1}+3s\right] ds}{\gamma t^2+\varepsilon_2(n,k)} \\ &= \lim_{t\to\infty} \frac{\eta t^2+\varepsilon_3(n,k)}{\gamma t^2+\varepsilon_2(n,k)} = c,\end{aligned}$$

where $t=k(k+1)^{n-1}$, δ, η, and c are functions of k, and

$$\lim_{n\to\infty} \frac{\varepsilon_2(n,k)}{t^2} = \lim_{n\to\infty} \frac{\varepsilon_3(n,k)}{t^2} = 0.$$

□

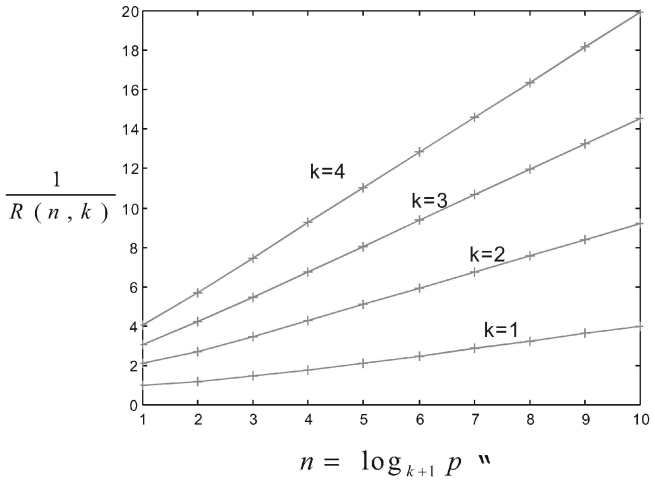


Fig. 16. Plots of $R(n,k)$, showing that the performance improvement of algorithm ML over algorithm G_4 increases as the value of $n=\lceil \log_{k+1} p \rceil$ increases.

5 Conclusion

By exploiting the concept of multilayered execution, we developed in this paper an efficient multilayered decentralized consensus protocol, algorithm ML, for a distributed system with an initiator. By adapting itself to the number of nodes in the system, algorithm ML is shown to be able to determine a proper layer for execution and reach the consensus in the minimal numbers of message steps while incurring a much smaller number of messages than required by prior works, indicating the advantage of utilizing the concept of multilayer for protocol execution. Several illustrative examples were given and performance analysis of the proposed algorithm was conducted to provide many insights into the problem studied. It has been shown that algorithm ML significantly outperforms algorithm G_4 and the performance improvement achieved by algorithm ML increases as the number of nodes increases. Specifically, the ratio of $N_{ML}(p,k)$ to that $N_{G4}(p,k)$ was proved to approach zero as the number of nodes becomes large. Moreover, algorithm ML was proved to be asymptotically optimal in the sense that $N_{ML}(p,k)$ and its corresponding theoretical minimum, i.e., $N_{OP}(p,k)$, are asymptotically of the same complexity with respect to the number of nodes in the system, showing the very important advantage of algorithm ML.

Appendix A

Algorithm G_3

To determine the minimal number of messages required for a decentralized consensus protocol in a system of p nodes with k-port communication, the corresponding optimal partitioning tree can be obtained as follows, where $n_1+n_2=n=\lceil log_{k+1} p \rceil$, and d is the smallest positive integer such that $p \leq (d-1)^{n_1} d^{n_2}$ and $p > (d-1)^{n_1+1} d^{n_2-1}$.

Procedure to build the optimal partitioning tree:

Step 1. Build a tree from level 0 to level n_1-1 in such a way that each node is an internal node and has a degree $d-1$.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

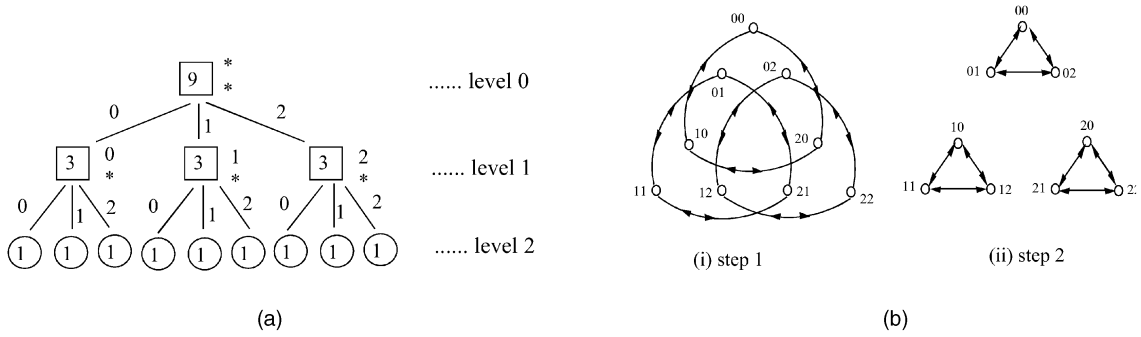


Fig. 17. The partitioning tree and the operations of G_3 when $k = 2$ and $p = 9$.

Step 2. In the next $n_2 - 1$ levels (i.e., from level n_1 to level $n - 2$), let each internal node have a degree d.

Step 3. In the last level of internal nodes (i.e., level $n - 1$), let $(d-1)^{n_1} d^{n_2} - p$ internal nodes have degree $d - 1$, and $p - (d-1)^{n_1+1} d^{n_2-1}$ internal nodes have degree d.

Step 4. In level n, attach leaf nodes to those internal nodes in level $n - 1$, according to the degree of each internal node in that level.

Step 5. Label each leaf node with one, and determine the number labeled in each internal node in the tree bottom up such that the number labeled in each node is the sum of those labeled in its child nodes.

Based on the above scheme, algorithm G_3 can be outlined as follows. consensus protocol without an initiator for *k*-port communication.

Algorithm G_3: The consensus protocol withour an initiator for *k*-port communication.

1. Build the optimal partitioning tree.
2. Perform the consensus protocol based on the generation of minimal complete sets in the partitioning tree.

For example, for the case of 2-port communication in a system of 9 nodes, we have the 3-ary partitioning tree and the corresponding execution of G_3 shown in Fig. 17.

APPENDIX B
ALGORITHM G_4

Algorithm G_4: The consensus protocol with an initiator for *k*-port communication.

/* p is the total number of nodes, $n = \lceil \log_{k+1} p \rceil$, and h is the smallest number such that $\lceil \log_{h+1} p \rceil = n$.
*/

1. From Step 1 to step $n - 1$, use h-port communication to perform the broadcasting among those $(h+1)^{n-1}$ nodes in part (A).
2. In the last step of the broadcasting phase (i.e., step *n*), form a shuffling set of $r = p - (h+1)^{n-1}$ nodes.
3. In step $n + 1$, each node in the shuffling set does two things: 1) sends one message to the initiator to start the confirming phase and 2) starts the shuffling phase within the *r* nodes in the shuffling set.
4. From step $n + 2$ to step $2n$, the initiator completes the confirming phase in part A, and nodes in the shuffling set, i.e., part B, complete the shuffling phase based on G_3.

ACKNOWLEDGMENTS

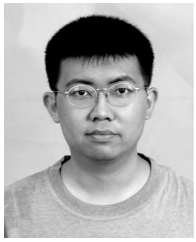
The authors would like to thank the anonymous referees for their helpful comments to improve this paper. Part of the early results of this study were presented in the Proceedings of the IEEE 20th International Conference on Distributed Computing Systems, April, 2000.

REFERENCES

[1] W.C. Athas and C.L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *Computer,* vol. 21, pp. 9-24, Aug. 1988.
[2] L. Bhuyan and D.P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network," *IEEE Trans. Computers,* vol. 33, no. 4, pp. 323-333, Apr. 1984.
[3] T.D. Chandra and D.P. Agrawal, "Unreliable Failure Detectors for Reliable Distributed System," *J. ACM,* vol. 43, no. 2, pp. 255-267, Apr. 1996.
[4] M.-S. Chen, J.-C. Chen, and P.S. Yu, "On General Results for All-to-All Broadcast," *IEEE Trans. Parallel and Distributed System,* vol. 7, no. 4, pp. 363-370, Apr. 1996.
[5] M.-S. Chen, K.-L. Wu, and P.S. Yu, "Efficient Decentralized Consensus Protocols in a Distributed Computing System," *Proc. 12th Int'l Conf. Distributed Computing Systems,* pp. 426-433, June 1992.
[6] M.-S. Chen, P.S. Yu, K.-L. Wu, "Decentralized Consensus Protocols with Multi-Port Communication," *Proc. 13th Int'l Conf. Distributed Computing Systems,* pp. 356-365, May 1993.
[7] S.B. Davidson, H. Garcia-Molina, and D. Skeen, "Consistency in Partitioned Networks," *ACM Computing Surveys,* vol. 17, no. 3, pp. 341-370, Sept. 1985.
[8] A.W. Fu, "Delay-Optimal Quorum Consensus for Distributed Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 8, no. 1, pp. 59-69, Jan. 1997.
[9] V. Hadzilacos and J. Halpern, "Message-Optimal Protocols for Byzantine Agreement," Technical Report RJ 7879, IBM Almaden Research Laboratory, San Jose, Calif., Dec. 1990.
[10] S.M. Hedetniemi, S.T. Hedetniemi, and A. Liestman, "A Survey of Broadcasting and Gossiping in Communication Networks," *NETWORKS,* vol. 18, pp. 319-351, 1988.
[11] M. Hurfin, A. Mostefaoui, and M. Raynal, "Consensus in Asynchronous Systems Where Processes can Crash and Recover," *Proc. 17th IEEE Symp. Reliable Distributed Systems,* pp. 280-286, 1998.
[12] S.L. Johnsson and C.T. Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Trans. Computers,* vol. 38, no. 9, pp. 1249-1268, Sept. 1989.
[13] T.V. Lakshman and A.K. Agrawala, "Efficient Decentralized Consensus Protocols," *IEEE Trans. Software Eng.,* vol. 12, no. 5, pp. 600-607, May 1986.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.

[14] S. Lee and K.G. Shin, "Interleaved All-to-All Reliable Broadcast on Meshes and Hypercube," *IEEE Trans. Parallel and Distributed Systems,* vol. 5, no. 5, pp. 449-458, 1994.
[15] Z.-R. Lin and M.-S. Chen, "An Asymptotically Optimal Multi-Layered Decentralized Consensus Protocol with an Initiator," *Proc. 20th IEEE Int'l Conf. Distributed Computing Systems,* 280-287, Apr. 2000.
[16] E.A. Monakhova, "Algorithms and Lower Bounds for P-Gossiping in Circulant Networks," *Proc. Third Int'l Symp. Parallel Architectures, Algorithms, and Networks (I-SPAN '97),* pp. 132-137, 1997.
[17] K.N. Venkataraman, G. Cybenko, and D.W. Krumme, "Simultaneous Broadcasting in Multiprocessor Networks," *Proc. Int'l Conf. Parallel Processing,* pp. 555-558, 1986.
[18] S.-M. Yuan and A.K. Agrawala, "A Class of Optimal Decentralized Commit Protocols," *Proc. Eighth Int'l Conf. Distributed Computing Systems,* pp. 234-241, 1988.



Cheng-Ru Lin received the BS degree in electrical engineering from National Taiwan University, Taipei, in 1999. He is currently a PhD candidate in the Electrical Engineering Department, at the National Taiwan University. His research interests include distributed systems, databases, data clustering, and data mining.

Ming-Syan Chen received the BS degree in electrical engineering from National Taiwan University, Taipei, the MS, and the PhD degrees in computer, information, and control engineering from the University of Michigan, Ann Arbor, in 1985 and 1988, respectively. Dr. Chen is currently a professor in the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, He has published more than 120 papers in his research areas. In addition to serving as program committee members in many conferences, Dr. Chen served as an associate editor of *IEEE Transactions on Knowledge and Data Engineering* on data mining and parallel database areas from 1997 to 2001, an editor of the *Journal of Information Science and Engineering*, a distinguished visitor of IEEE Computer Society for the Asia-Pacific from 1998 to 2000, and program chair of PAKDD-02 (Pacific Area Knowledge Discovery and Data Mining), program vice-chair of VLDB-2002 (Very Large Data Bases), general chair of Real-Time Multimedia System Workshop in 2001, program chair of IEEE ICDCS Workshop on Knowledge Discovery and Data Mining in the World Wide Web in 2000, and program cochairs of the International Conf. Mobile Data Management in 2003, International Computer Symposium (ICS) on Computer Networks, Internet and Multimedia in 1998 and 2000, and ICS on Databases and Software Engineering in 2002. He was a keynote speaker on Web data mining in International Computer Congress in Hong Kong, 1999, a tutorial speaker on Web data mining in DASFAA-1999 and on parallel databases in the 11th IEEE International Conference on Data Engineering in 1995 and also a guest coeditor for *IEEE Transactions Knowledge and Data Engineering* on a special issue for data mining in December 1996. He holds, or has applied for, eighteen US patents and seven ROC patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He received the Outstanding Innovation Award from IBM Corporate in 1994 for his contribution to parallel transaction design and implementation for a major database product, and numerous awards for his research, teaching, inventions, and patent applications. Dr. Chen is a senior member of IEEE and a member of ACM.

▷ **For more information on this or any computing topic, please visit our Digital Library at** http://computer.org/publications/dilb.

Authorized licensed use limited to: National Taiwan University. Downloaded on January 16, 2009 at 01:42 from IEEE Xplore. Restrictions apply.