

An Asymptotically Optimal Multi-Layered Decentralized Consensus Protocol with an Initiator

Zheng-Ru Lin and Ming-Syan Chen
Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, ROC

Email: {owenlin@arbor.ee.ntu.edu.tw, mschen@cc.ee.ntu.edu.tw}

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 multi-layered execution, we develop in this paper an efficient multi-layered 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. 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 proved 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, multi-port 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 between the processing nodes can be done via message passing. 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 [6][8].

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 assumed to be achieved *a priori*, and all processing nodes concurrently participate in the consensus protocol when the protocol is started. 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 [1] [5] [7], 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 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 [5]. To facilitate the presentation, we use the identification (id) of each node to denote the information that this node wants to send

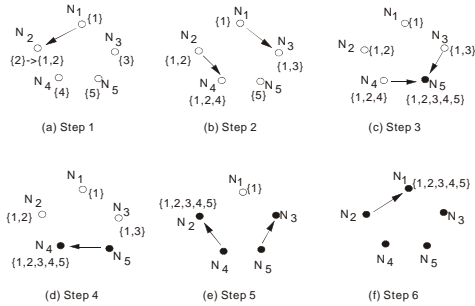


Figure 1. An example decentralized consensus protocol with an initiator (1-port) [3].

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 5 nodes. A node having all the information from other nodes is called an *expert* [5]. We use black nodes to denote experts, and white nodes to denote those that still have incomplete information. 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 Figure 1 [3] where, without loss of generality, node N_1 is the initiator. It can be seen that the &ve nodes reach a consensus after 6 steps, with a total of 9 messages.

Although decentralized consensus protocols with an initiator were proposed in [2], 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 multi-layered 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 al-

gorithm 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 [1] [5] [7], 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 summaries of prior related protocols are given in Section 2. The multi-layered 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

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. 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 [2] [5] [8], and is summarized as follows.

Model M

The system is completely connected with synchronous communication.

Every message sent in the system takes one communication step.

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.)

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 [5]. 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. Also, we do not assume that nodes in the system will be faulty or maliciously send wrong messages to others.

2.2 DCP without and with an initiator (1-port)

A decentralized consensus protocol without an initiator for 1-port communication G_1 was presented in [3]. From [3], 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 $N_{G_1}(p) = np + p - 2^n$ 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.

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 [3]. A detailed description of G_2 can be found in [3], 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 $N_{G_2}(p) = 2^n - 1 + \max\{2^{n-2}, p - 2^{n-1}\} + r \lceil \log_2 r \rceil + r - 2^{\lceil \log_2 r \rceil}$ messages, where $r = p - 2^{n-1}$.

2.3 DCP without and with an initiator (k-port)

In [2], 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 is referred to as algorithm G_3 in this paper. We have the following two propositions for G_3 [2].

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,

$$N_{G_3}(p, k) = (d-2)n_1p + (d-1)[n_2p + p - (d-1)^{n_1}d^{n_2}],$$

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 $p > (d-1)^{n_1+1}d^{n_2-1}$.

A decentralized consensus protocol with an initiator for k -port communication, referred to as G_4 , was presented in [2].

From [2], 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 $N_{G_4}(p, k) = 2(h+1)^{n-1} + r - 2 + \max\{(h+1)^{n-1} - (h+1)^{n-2}, r\} + N_{G_3}(r, h)$ messages, where h is the smallest number such that $\lceil \log_{h+1} p \rceil = n$, $r = p - (h+1)^{n-1}$ and $N_{G_3}(r, h)$ is determined by Proposition 5.

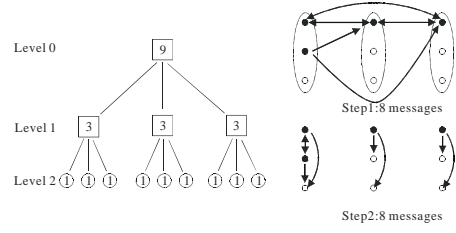


Figure 2. An example of procedure G^* when $p = 9$, $s = 4$ and $k = 2$.

3 Multi-Layered DCP

In light of the concept of multi-layered 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 Multi-Layered DCP

To facilitate our presentation of algorithm ML, we first introduce procedure G^* below, which is in essence extended from algorithm G_3 while exploiting the feature of having an initiator.

Procedure G^* : A some-to-all broadcasting protocol /* Sending the ids of nodes in a set of s nodes, denoted by S , to all nodes of a system of p nodes. */

- 1: Let $r = \lceil \log_{k+1} s \rceil$. Build a tree from level 0 to level r as in G_3 [2] in such a way that each node is an internal node and has degree $k + 1$.
- 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 $r - 1$, (1) each minimal complete set contains at least one node in S , and (2) only nodes in S can receive messages.*

Figure 2 is an example for a system of 9 nodes where $s = 4$ and $k = 2$. It can be seen that procedure G^* incurs 8 messages in the first step, and 8 messages in the second step. In the first step, 3 minimal complete sets are formed: one contains two nodes in S , and each of the other two sets contains only one node in S . Procedure G^* completes in the second step by forming 9 minimal complete sets (i.e., every node is an expert).

Theorem 1: Procedure G^* takes $r \cdot s \cdot k$ messages.

Proof: It can be verified that Step 2 of procedure G^* is always feasible since $s > (k+1)^{r-1}$ and we can allocate at least one node in S for each complete set at level $r - 1$. In addition, from level 0 to level $r - 1$, only nodes in S can receive messages, meaning that only those nodes need to send messages in the subsequent steps. This theorem follows. Q.E.D.

With the introduction of procedure G^* , algorithm ML can be described below.

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

- 1: In the first $n - 1$ execution steps, use k -port communication to perform the broadcasting among $(k + 1)^{n-1}$ nodes.
- 2: In the last step of the broadcasting phase, form a shuffling set of s nodes. Thus, we have $s = p - (k + 1)^{n-1}$.
- 3: if $p = (k + 1)^n$ then
 goto Step 10
 else

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

$$r = \min(x),$$
 that $\left\{ \begin{array}{l} x \in N, \\ \sum_{i=1}^{y-1} k(k+1)^{n-i} + x(k+1)^{n-y} \geq p, \\ x \leq k \end{array} \right.$
 $Z =$ the shuffling set, and
 $i = 1$.
 if $s = 1$ then
 let the set Z_A of a single node
 be the shuffling set,
 and goto Step 9.
 endif
 endif
- 4: if $(i = y)$ then goto Step 8 /* to determine the seed experts in the last layer */.
- 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 $Z - Z_A$.
- 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*.
- 7: $Z = Z - Z_A$. $i = i + 1$. goto Step 4.
- 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 .
- 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 */
- 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 G^* in the whole system, where those s nodes with the information to broadcast form the shuffling set.
 endif
 end /* end of the protocol */

Algorithm ML forms as many experts as needed from the shuffling set, and uses them to send the information of ids back to other nodes. For ease of description, we call those experts which are formed by nodes in Z , seed experts, so as to distinguish them from other experts. Note that 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 verified that $k(y - 1) + r$ seed experts will be formed in the entire protocol execution.

It is important to note that the parameter $y = \min\{x \in N \mid \sum_{i=1}^x k(k + 1)^{n-i} \geq p, x \leq n\}$ of algorithm ML corresponds to the number of message steps required to generate seed experts (from Step 4 to Step 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 (Step 8). Also note that each node, after becoming an expert, is expected to be involved in sending messages in 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 Step 6 and Step 9 of algorithm ML. With its proof omitted, 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.

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

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

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 G^* for the whole system, otherwise, algorithm

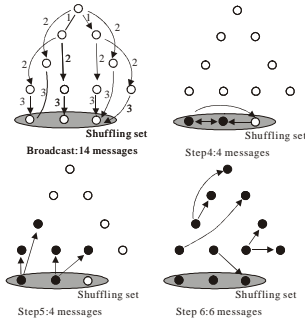


Figure 3. An example of 1-layer execution of algorithm ML for a 12-node system with 2-port communication.

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 Step 5 of algorithm ML is $\left(\sum_{i=0}^{y-2} (s - i \times k)k\right)$, that in Step 8 is $((s - (y - 1)k)r) - r$, and Step 6 and Step 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. Q.E.D.

3.2 Illustrative Examples

To illustrate the operations of algorithm ML, we consider the system of 12 nodes with 2-port communication in Figure 3. In this case, $p = 12$, $k = 2$, $n = 3$, $y = 1$, $r = 2$, and $s = 3$. Thus, the scenario in Figure 3 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.

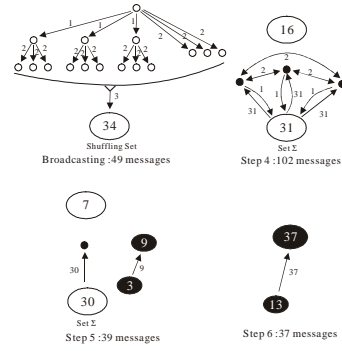


Figure 4. An example of 2-layer execution of algorithm ML for a 50-node system with 2-port communication.

Consider a 50-node system with 2-port communication in Figure 4. 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 3 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 3 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 9 nodes, while incurring 9 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 protocol, agreeing with Theorem 3. It can be verified that $N_{G_4}(50, 2) = 330$, 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 $N_{G_4}(p, k)$ is proved to be of complexity $O(\log^{-1} p)$ which approaches zero as the number of nodes p becomes large.

4.1 Theoretical Insights into the DCP

To provide more insights into algorithm ML, consider the two plots in Figure 5 where the relationship between the number of layers y and the number of nodes p is shown in Figure 5a, and the number of

messages that are needed in algorithm ML for 6-port communication is shown in Figure 5b .

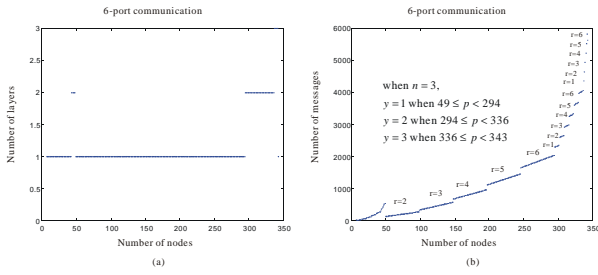


Figure 5. Two plots for the analysis of algorithm ML with 6-port communication : (a) number of layers v.s. number of nodes; (b) number of messages v.s. number of nodes.

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 Figure 5b, 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 Figure 5b, 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, and so on. It can be verified that the first five segments of the curve in Figure 5b correspond to the case of 1-layer execution (i.e., with 2 to 6 experts), the next 6 segments correspond to the case of 2-layer execution (i.e., with 7 to 12 experts), and the following 6 segments correspond to the case of 3-layer execution.

With the illustrative example in Figure 5b, by omitting their straightforward proofs, the following properties of algorithm ML can be derived from Theorem 3.

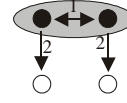
Corollary 3.1: The systems in the same segment of the curve have the same parameters y and r .

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. De&ne

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

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. Q.E.D.

4.2 Performance of Algorithm ML

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.

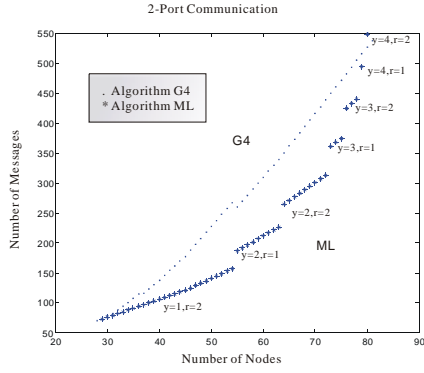


Figure 6. The number of messages incurred by algorithm ML and algorithm G_4 for 2-port communication when the number of nodes varies.

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 $N_{G_4}(p, 2)$ are shown in Figure 6 where the number of nodes varies from $28 = (2 + 1)^3 + 1$ to $81 = (2 + 1)^4$.

It can be seen from Figure 6 that the number of messages in algorithm ML is in much smaller than that required by algorithm G_4 . 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 ratio function $R(n, k)$ as:

$$R(n, k) = \frac{\frac{1}{k(k+1)^{n-1}} \sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{ML}(p, k)}{\frac{1}{k(k+1)^{n-1}} \sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{G_4}(p, k)},$$

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 5: $\lim_{n \rightarrow \infty} R(n, k) = \frac{1}{a \times n + b}$, 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 $N_{G_3}(p, k) = (d-2)n_1p + (d-1)[n_2p + p - (d-1)^{n_1}d^{n_2}]$, when p approaches infinity, we have $d = k + 1$ and $n_2 \gg n_1$. Thus, $n_2 \approx n$, and then, $N_{G_3}(p, k) \approx nkp$.

Recall that $N_{G_4}(p, k) = 2(h+1)^{n-1} + s - 2 + \max\{(h+1)^{n-1} - (h+1)^{n-2}, s\} + N_{G_3}(s, h)$, 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 $N_{G_3}(p, k)$. $N_{G_3}(s, k)$ can thus be ap-

proximated as rks , where $r = \lceil \log_{k+1} s \rceil$. $N_{G_4}(p, k)$ can in turn be approximated as $2(k+1)^{n-1} + s + \max\{k(k+1)^{n-2}, s\} + krs$.

Next, $N_{ML}(p, k) = p - 1 + \max\{s, k(k+1)^{n-2}\} - s + ((y-1)k + r)(s-1) + p - \frac{1}{2}k(y-1)(ky - 2k + 2r) - r$ can also be approximated as $2(k+1)^{n-1} + s + \max\{k(k+1)^{n-2}, s\} + yks + \delta s$, 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 $y = \min\{x \in N \mid \sum_{i=1}^x k(k+1)^{n-i} \geq p, x \leq n\}$, we can approximate y as $n - \log((k+1)^n - p + 1) = n - \log(k(k+1)^{n-1} - s + 1)$.

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

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

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

We approximate the formula in the numerator and use the integration equality

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

$$\begin{aligned} \sum_{p=(k+1)^{n-1}+1}^{(k+1)^n} N_{G_4}(p, k) &\approx \\ \int_1^t [2(k+1)^{n-1} + \max\{k(k+1)^{n-2}, s\} + s + krs] ds & \\ = \int_1^t [2(k+1)^{n-1} + s] ds + \int_1^{k(k+1)^{n-2}} [k(k+1)^{n-2}] ds + & \\ \int_{k(k+1)^{n-2}+1}^t [s] ds + \int_1^t [k \times (\log_{k+1} s) \times s] ds & \\ = \alpha \cdot t^2 \log t + \beta \cdot t^2 + \varepsilon_1(n, k), & \end{aligned}$$

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

$\int_0^{a-1} [(\log a - \log(a-x))x] dx = \frac{1}{4}(1 - 4a + 3a^2 + (2-4a) \log a)$, the formula in the denominator can be approximated as below.

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

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

From these two approximations, we obtain $\lim_{n \rightarrow \infty} R(n, k) = \frac{\gamma}{\alpha \log t + \beta} = \frac{1}{an+b}$, where a and b are functions of k and this theorem follows. Q.E.D.

From Proposition 6 and Theorem 3, values of $R(n, k)$, for $k = 1, 2, 3$ and 4, can be obtained and plotted in Figure 7, whose results agree with Theorem 5. It can be seen from Figure 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

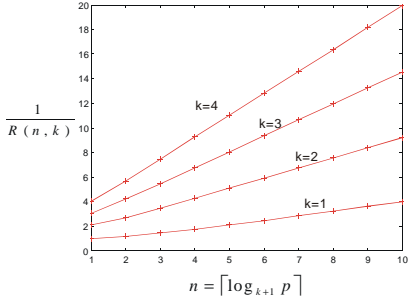


Figure 7. Plots of $R(n, k)$, showing that the performance improvement of algorithm ML over algorithm G_4 increases as the value of $n = \lceil \log_{k+1} p \rceil$ increases.

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 6: 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 $1 \geq \lim_{n \rightarrow \infty} R^*(n, k) \geq c$, where c is a function of k .

proof: From the definition of $N_{OP}(p, k)$, we have $1 \geq \lim_{n \rightarrow \infty} R^*(n, k)$. 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 $2p-1-s + (s-1) \lceil \frac{p}{(k+1)^{n-1}} \rceil$, which is in turn reduced to $2p+s-3$ since $\lceil \frac{p}{(k+1)^{n-1}} \rceil \geq 2$. Hence, we have,

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

where $t = k(k+1)^{n-1}$, δ , η , and c are functions of k , and $\lim_{n \rightarrow \infty} \frac{\varepsilon_2(n, k)}{t^2} = \lim_{n \rightarrow \infty} \frac{\varepsilon_3(n, k)}{t^2} = 0$. Q.E.D.

5 Conclusion

By exploiting the concept of multi-layered execution, we developed in this paper an efficient multi-layered decentralized consensus protocol, algorithm ML, for a distributed system with an initiator. 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 $N_{G_4}(p, k)$ 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.

Acknowledgments

The authors are supported in part by the National Science Council, Project No. NSC 89-2219-E-002-007 and NSC 89-2213-E-002-032, Taiwan, Republic of China.

References

- [1] M.-S. Chen, J.-C. Chen, and P. S. Yu. On General Results for All-to-All Broadcast. *IEEE Transactions on Parallel and Distributed System*, 7(4):363-370, April 1996.
- [2] M.-S. Chen, P. S. Yu, and K.-L. Wu. Decentralized Consensus Protocols with Multi-Port Communication. *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 356-365, May 1993.
- [3] M.-S. Chen, K.-L. Wu, and P. S. Yu. Efficient Decentralized Consensus Protocols in a Distributed Computing System. *Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 426-433, June 1992.
- [4] A. W. Fu. Delay-Optimal Quorum Consensus for Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 8(1):59-69, January 1997.
- [5] S. M. Hedetniemi, S. T. Hedetniemi, and A. Liestman. A Survey of Broadcasting and Gossiping in Communication Networks. *NETWORKS*, 18:319-351, 1988.
- [6] T. V. Lakshman and A. K. Agrawala. Efficient Decentralized Consensus Protocols. *IEEE Transactions on Software Engineering*, SE-12(5):600-607, May 1986.
- [7] E. A. Monakhova. Algorithms and Lower Bounds for P-Gossiping in Circulant Networks. *Proceedings of the 3rd Internl Symposium on Parallel Architectures, Algorithms, and Networks, 1997 (I-SPAN 97)*, pages 132-137, 1997.
- [8] S.-M. Yuan and A. K. Agrawala. A Class of Optimal Decentralized Commit Protocols. *Proc. of 8th Int. Conference on Distributed Computing Systems*, pages 234-241, 1988.