Journal of Parallel and Distributed Computing **62**, 1295–1313 (2002)
doi:10.1006/jpdc.2002.1861

Efficient Selection and Sorting Schemes Using Coteries for Processing Large Distributed Files

David S. L. Wei[1]

Department of Computer and Information Science, Fordham University, Bronx, New York 10458
E-mail: wei@dsm.fordham.edu

Sanguthevar Rajasekaran

Department of CISE, University of Florida, Gainesville, Florida 32611
E-mail: raj@cis.ufl.edu

Z. Cheng

Department of Computer Software, University of Aizu, Fukushima 965-80, Japan
E-mail: z-cheng@u-aizu.ac.jp

K. Naik

Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
E-mail: knaik@swen.uwaterloo.ca

and

Sy-Yen Kuo

Department of Electrical Engineering, National Taiwan University,Taipei, Taiwan, ROC
E-mail: sykuo@cc.ee.ntu.edu.tw

Received December 21, 1998; revised May 20, 2002; accepted June 1, 2002

In this paper, we develop efficient selection and sorting schemes for processing large files distributed over a network. The efficiencies of the schemes are expressed in terms of message count and communication delay. The schemes are developed using the concept of *coteries* which is a class of communication structures widely used in the development of some classical distributed algorithms, namely mutual exclusion, multiway rendezvous, etc. The development of the schemes is carried out as follows. First, we develop a ranking scheme. Second, using the ranking scheme, we develop a restricted version of sorting, where each node of the network contains exactly one key, and the sorting leads to the *i*th node holding the *i*th key of the sorted list. Third, using this restricted sorting, a selection scheme is developed. Given *n*

[1] To whom correspondence should be addressed.

0743-7315/02 $35.00

keys evenly distributed among p nodes, selection of the kth key means identifying the value of the key. Finally, using the idea of selection, we sort the n keys distributed among p nodes. Both the ranking and the restricted sorting steps need $O(p\sqrt{p})$ messages and suffer a two-round communication delay. The selection step needs $O(p^{3/2}\log n)$ messages with communication delay of $O(\tau \log p)$, where τ is the maximum of the times taken by a message to be sent to all the members of a quorum. The sorting scheme needs $O(n)$ messages and its communication delay is $O(\tau\frac{n}{p})$. Both of these complexities are optimal provided n is polynomial in p and $n = \Omega(p^{5/2}\log n)$.

Key Words: large distributed files; selection; sorting; coteries; sampling technique; consensus.

1. INTRODUCTION

In this paper, we develop selection and sorting schemes to process large files distributed over a network of computers. By a large file we mean a file size several times the number of nodes in the network, such as of the order of p^{10} keys in the file, where p is the number of network nodes. Distributed large files must be processed in applications such as national census, personnel information of large multinational companies, etc. Hereafter, by a file we mean a large file. In this context, selection and sorting are explained as follows. Given a file $\mathscr{F}$ with n keys evenly distributed over a network of p nodes and an integer k, $1 \leqslant k \leqslant n$, the selection problem is to find the kth smallest key of $\mathscr{F}$. Sorting $\mathscr{F}$ means relocating the n keys among the p nodes such that all the keys at the ith node are smaller (greater) than all the keys at the jth node for $i < j (i > j)$. Before and after sorting, the keys should be evenly distributed among the nodes.

In the development of our schemes, we logically organize the nodes into *coterie* structures. Informally, a *quorum* is a subset of the nodes, and a coterie is a collection of quorums, such that any two distinct quorums have at least one common node and no quorum is a subset of another [4, 7]. Using the coterie structures, several distributed algorithms have been developed for such classical problems as mutual exclusion [9], k-entry mutual exclusion [6], multiway rendezvous [2], and consensus [8, 10]. A central idea in a coterie-based algorithm is that a node communicates only with the members belonging to the same quorum. Information flows between two nodes belonging to different quorums through an intersecting node of the two quorums. Thus, any two nodes can indirectly communicate in at most two rounds of communication. The low message complexities of coterie-based algorithms is due to a node communicating only with a small subset of the nodes, say of size $\sqrt{p}$ or $\log p$, where p is the number of nodes in the coterie.

We develop the selection and sorting schemes using the ideas of ranking and a restricted version of sorting. Let X be any set of keys from linear order. Then the rank of any key k in X is defined to be the one plus the number of elements of X that are less than k. Let each node in a network have a single key. The ranking problem is to compute the rank of each key. The keys are not moved. In the restricted version of sorting (i.e., each node has exactly one key), we place the kth ranked key at the kth

node using our ranking scheme. We develop the selection scheme (for selecting the kth key from n keys evenly distributed over a coterie of p nodes) by using the restricted version of sorting and a consensus protocol as primitives. For efficiency, we use sampling techniques [11] in the selection scheme. Finally, given n keys evenly distributed among p nodes, we sort the n keys. The sorting scheme is a variation of the quicksort, where selections of the pivots are done using the developed selection scheme.

We express the performance of the selection and sorting schemes in terms of message count and rounds of communication involved. The importance of message count is due to the fact that local processing time is much smaller than communication time including transmission, reception, and queueing delays. The rounds of communication involved also contributes to processing time. Our ranking and restricted sorting need $O(p\sqrt{p})$ messages and suffer 2τ communication delay, where τ is the maximum of the times taken by a message to be sent to each member of a quorum. The selection scheme needs $O(p^{3/2}\log n)$ messages with communication delay of $O(\tau \log p)$. The sorting scheme needs $O(n)$ messages and its communication delay is $O(\tau\frac{n}{p})$. Both of these complexities are optimal provided n is polynomial in p and $n = \Omega(p^{5/2}\log n)$.

In Section 2, we introduce two coterie structures and state their properties. Ranking and restricted sorting schemes are developed on the coterie structures in Section 3—the restricted sorting uses the idea of ranking. Using the idea of sampling technique and the restricted sorting and consensus as building blocks, in Section 4, we develop a selection scheme. Finally, in Section 5, a distributed file is sorted using the selection scheme.

2. PRELIMINARY FACTS

In this section, we introduce two coterie structures. Let $S = \{s_1, s_2, \ldots, s_i, \ldots, s_p\}$ be the set of nodes of a network. A coterie is a family $C = \{S_j \mid S_j \subseteq S\}$ of subsets of nodes such that any pair of subsets in C has at least one common node, and no member of C is a subset of another member. Members of a coterie are called quorums. The development of a coterie-based algorithm depends on the way the quorums of the coterie are constructed. Our ranking scheme depends on the coterie structures, whereas the selection and sorting schemes are independent of the structure of a coterie. In the following, we explain two kinds of coteries to be used in this paper.

2.1. Coterie Structure 1 (CS1)

We first consider a kind of coterie called a finite projective plane. A finite projective plane π of order $m \geqslant 2$ is a structure consisting of a set, S, of points and a set, L, of lines, each of which contains a subset of points of S, and satisfies the following axioms [5, 10].

1. Any two lines cross at one and only one point.
2. There exist four lines, no three of which cross at the same point.
3. Each line comprises of exactly $(m + 1)$ points.

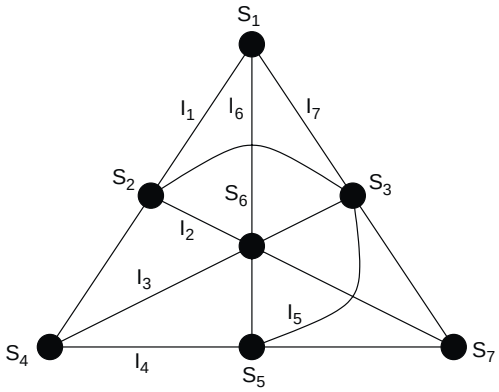

FIG. 1. A finite projective plane of order two.

Also, the following properties can be obtained from the above three axioms.

1. Each point is on $(m + 1)$ lines.
2. $|S| = (m^2 + m + 1)$ (i.e., the size of a finite projective plane is determined by its order).

An example of a finite projective plane of order two is shown in Fig. 1. From the figure, one can see that the collection of the points on line l_i form quorum i. To use CS1 as a communication structure for developing our algorithms, we refer a point in CS1 as a node and a line in CS1 as a broadcast communication link.

During the course of computation, each node behaves the same without being controlled by any other node, which consists of sending messages over adjoining lines, waiting for incoming messages, and processing received messages. We assume that messages can be transmitted independently in both directions on a line, and reach their destinations in sequence after a finite delay without error. In order to reduce the number of messages, a node s_i can directly send messages only to a subset of S, which is called the *sending set* of node s_i. Likewise, a node s_i can directly receive messages only from a subset of S, which is called the *receiving set* of node s_i. Both sending set and receiving set are defined in Definition 2.1. For the purpose of developing our ranking algorithm, the sending and receiving sets are chosen in such a way that after the first round of sending and receiving, we have

$$\bigcup_{s_j \in S_i(1)} R_j(1) = S \qquad \forall i.$$

DEFINITION 2.1. Letting $S_i(j)$ denote the jth round[2] sending set of node s_i, and $R_i(j)$ denote the jth round receiving set of node s_i, $1 \leqslant j \leqslant 2$, we have:

$$\begin{aligned} S_i(1) &= \{s_a | s_i \in l_a\} \ (1 \leqslant a, i \leqslant m^2 + m + 1), \\ S_i(2) &= \{s_a | s_a \in l_i\} \ (1 \leqslant a, i \leqslant m^2 + m + 1), \\ R_i(1) &= S_i(2), \\ R_i(2) &= S_i(1). \end{aligned}$$

[2] A message originated from any node can reach its destinations in two rounds (hops) in a finite projective plane.

Referring to Fig. 1, we have the following sending and receiving sets for node s_1: $S_1(1) = \{s_1, s_6, s_7\}$, $S_1(2) = \{s_1, s_2, s_4\}$, $R_1(1) = \{s_1, s_2, s_4\}$, and $R_1(2) = \{s_1, s_6, s_7\}$.

2.2. *Coterie Structure 2 (CS2)*

The second kind of coterie structure for a p-node network is constructed as follows.

Step 1. Create a completely connected auxiliary graph with m nodes, where $m \geqslant \frac{1+\sqrt{8p+1}}{2}$.

Step 2. Associate each node of the network with an edge of the auxiliary graph. For discussion convenience, we only consider the case where $m = \frac{1+\sqrt{8p+1}}{2}$, i.e., the number of edges, say p, is equal to $\frac{m(m-1)}{2}$, such that the one-to-one mapping is possible.

Step 3. A coterie structure is constructed as follows. A node of the auxiliary graph is represented by a line, and an edge is represented by a node. All the nodes on a line form a quorum, and the collection of the lines is the desired coterie structure.

It is then not hard to see that the line clique has the following properties:

1. Each line consists of exactly $m - 1$ nodes.
2. Each node is on exactly two lines.
3. Every two lines cross at one and only one node.
4. There are m lines in the structure.
5. There are p nodes in the structure.

EXAMPLE 2.1. To construct a coterie structure 2 of 6 nodes, i.e., $S = \{s_1, s_2, \ldots, s_6\}$, $m = 4$, $p = \frac{m(m-1)}{2} = 6$, an auxiliary graph is shown in Fig. 2a; Fig. 2b shows the association of nodes in S with edges of the auxiliary graph; The desired line clique is shown in Fig. 2c.

We define the *communication set* of a node s_k as a set of nodes residing on the two lines which cross at s_k. The communication set of s_k does not include s_k itself. A node can directly send messages only to nodes in its communication set. Also, a node can directly receive messages only from the nodes in its communication set. In Fig. 2, the communication set of s_1 is given by $\{s_4, s_6, s_2, s_5\}$.

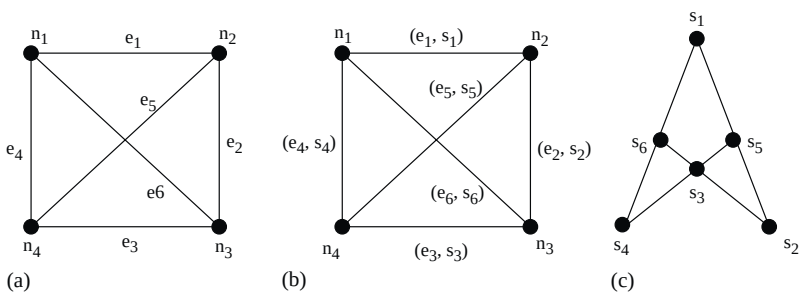

FIG. 2. Construction of a coterie structure 2.

3. RANKING AND RESTRICTED SORTING

We first develop a ranking scheme for each of the two coterie structures. Next, we develop a restricted sorting for both of the structures.

3.1. Ranking on the Coterie Structure 1

We define a set of variables managed by each node, intuitively explain the ranking scheme followed by its formal presentation, and finally show its correctness. Node s_i manages the following local variables.

- s_i: The identifier of the ith node.
- k_i: The key of the data item s_i possesses.
- $S_i(1)$: s_i's sending set for the first round.
- $S_i(2)$: s_i's sending set for the second round.
- $R_i(1)$: s_i's receiving set for the first round ($R_i(1) = S_i(2)$).
- $R_i(2)$: s_i's receiving set for the second round ($R_i(2) = S_i(1)$).
- des: The identifier of a destination node, $des \in S_i(2)$.
- num_i^{des}: The number of keys, smaller than the key of des, received from $R_i(1)$. Initially, $num_i^{des} := 0$ for each des.
- $rank_i$: The rank of the key of s_i.

The nodes communicate by exchanging the following two types of messages.

- KEY(k_i, s_i): sent by s_i to $\forall s_j \in S_i(1)$ to notify that the key in s_i is k_i.
- NUM(num_i^{des}, s_i): sent by s_i to $\forall des \in S_i(2)$.

The ranking scheme works in two rounds of communications and computations. In the first round, each node sends its own key to all nodes in its sending set, and receives keys from all nodes in its receiving set. Then each node computes partial results for the second round. In the second round, each node uses the partial results from the first round to compute the rank of its own key. The scheme is formally presented as follows.

ALGORITHM: Each Node s_i Independently Executes the Following Procedure

Round 1:
Step 1: (Sending and receiving the keys)
• Send KEY(k_i, s_i) to s_j $\forall s_j \in S_i(1)$.
• Receive the KEY(k_j, s_j) from s_j $\forall s_j \in R_i(1)$.
Step 2: (Local computation and sending the result of the computation)
$num_i^{des} := 0$
for each $des \in S_i(2)$ **do**
begin
(1) **for** each KEY(k_j, s_j) received **do**
  begin
    if $k_j < k_{des}$ **then**
      $num_i^{des} := num_i^{des} + 1$
  end
(2) Send NUM(num_i^{des}, s_i) to s_{des}.
end

Round 2:

Step 3: (Receiving the result of the computation)

• Receive NUM(X_j, s_j) from s_j $\forall s_j \in R_i(2)$.

Step 4: (Rank Computation)

• Compute the rank of s_i using all NUM(X_j, s_j) received:
$rank_i := (\sum_j \mathsf{X}_j) + 1$,
where X_j represents the number received by s_i from s_j.

Now we give an example of rank computation on a seven node network for which the coterie structure is shown in Fig. 1. Let nodes 1–7 hold key values $6, 9, 7, 5, 2, 1$, and 4, respectively.

After each node sends its key with its identifier to its sending set $S_i(1)$, and receives the keys and identifiers from all nodes in its receiving set $R_i(1)$, each node has the following information (keys with identifiers).

s_1: $R_1(1) = \{(6, s_1), (9, s_2), (5, s_4)\}$
s_2: $R_2(1) = \{(9, s_2), (1, s_6), (4, s_7)\}$
s_3: $R_3(1) = \{(7, s_3), (5, s_4), (1, s_6)\}$
s_4: $R_4(1) = \{(5, s_4), (2, s_5), (4, s_7)\}$
s_5: $R_5(1) = \{(2, s_5), (9, s_2), (7, s_3)\}$
s_6: $R_6(1) = \{(1, s_6), (6, s_1), (2, s_5)\}$
s_7: $R_7(1) = \{(4, s_7), (6, s_1), (7, s_3)\}$

Let us focus on the nodes in the sending set of s_1, namely

s_1: $R_1(1) = \{(6, s_1), (9, s_2), (5, s_4)\}$
s_6: $R_6(1) = \{(2, s_6), (6, s_1), (2, s_5)\}$
s_7: $R_7(1) = \{(4, s_7), (6, s_1), (7, s_3)\}$

Note that the key of each node in $S = \{s_1, s_2, \ldots, s_7\}$ appears in some nodes of the sending set, and no key appears more than once in a node.

At each node of the sending set of s_1, the following computation is done.

in s_1: $num_1 = 1$ (since 5 (*of* s_4) < 6 (*of* s_1)),
in s_6: $num_1 = 2$, and
in s_7: $num_1 = 1$.

Thus, after s_1 received these numbers, it can compute the rank of its key in round 2 as rank $= (1 + 2 + 1) + 1 = 5$. It can be easily verified that the rank of s_1's key of value 6 is 5 as it is evident from the sorted key list of {1, 2, 4, 5, 6, 7, 9}.

3.1.1. Correctness and analysis of the algorithm

LEMMA 3.1. *Let K be the set of keys in the network, and let $K_j(1)$ be the set of keys received by s_j from $R_j(1)$ during the first round. After sending and receiving of the first round, we have*

$$k_i \in K_j(1) \qquad \forall s_j \in S_i(1), \tag{1}$$

$$\forall s_{j1}, s_{j2} \in S_i(1), \qquad K_{j1}(1) \cap K_{j2}(1) = \{k_i\}, \tag{2}$$

$$\bigcup_{s_j \in S_i(1)} K_j(1) = K, \qquad \text{where } |K| = m^2 + m + 1. \tag{3}$$

Proof. (1) Obvious.

(2) Consider any two lines l_{j1} and l_{j2} going through the node s_i. Let s_{j1} and s_{j2} be two nodes in the set $S_i(1)$. The receiving sets $R_{j1}(1)$ and $R_{j2}(1)$ contain nodes on the lines l_{j1} and l_{j2}, respectively. According to axiom 1, l_{j1} and l_{j2} only meet at s_i. Thus, s_{j1} and s_{j2} receive different keys from different nodes except the key k_i from s_i.

(3) According to Definition 2.1, each node s_j in the sending set $S_i(1)$ is on line l_j, and all nodes in receiving set of s_j are on line l_j. Therefore, to count the number of keys received by all nodes in $S_i(1)$, we only need to count the number of points on those lines crossing at node s_i. There are $m+1$ lines going through s_i (property 1). Each line l_j contains m nodes excluding s_i (axiom 3). These $m+1$ lines cross at s_i only (axiom 1 and property 1). Therefore, $m+1$ lines contain $(m \times (m+1)+1)$ points, which is the total number of nodes in the finite projective plane of order m. Together with (2), the proof of (3) is completed. ∎

LEMMA 3.2. *After sending and receiving of the first round, a node never receives duplicate keys.*

Proof. Follows from the definition of sending and receiving sets. ∎

THEOREM 3.1. *By performing our ranking algorithm, every node s_i can compute the rank of its own key correctly.*

Proof. By Lemmas 3.1(1) and 3.1(3), the sending set $S_i(1)$ of each node s_i receives all keys in K, and each node in $S_i(1)$ receives the key k_i of s_i. Therefore, it is possible to compare k_i with every key in K at the nodes of $S_i(1)$. In addition, by Lemmas 3.2 and 3.1(2), no node can have duplicate keys and any two nodes in $S_i(1)$ do not have the same key except k_i. Thus, the summation of num_j's which is locally calculated at the nodes of $S_i(1)$ represents the number of keys smaller than the key of s_i. ∎

3.1.2. Analysis of our algorithm

THEOREM 3.2. *Our algorithm uses $O(p\sqrt{p})$ messages for the ranking problem.*

Proof. Obviously, message receiving only happens in Steps 1 and 3. In Step 1, since the number of nodes in receiving set $R_i(1)$ is m excluding s_i, where $m \leqslant \lfloor \sqrt{p} \rfloor$, every node receives $O(\sqrt{p})$ messages. Thus totally p nodes will receive $O(p\sqrt{p})$ messages. Likewise, in step 3, p nodes will receive $O(p\sqrt{p})$ messages. Thus, the total number of messages needed is $O(p\sqrt{p})$. ∎

3.2. Ranking on the Coterie Structure 2

The presentation of the ranking scheme for coterie structure 2 is very similar to that for coterie structure 1. The difference between the two structures leads to different ways of communications among nodes and different local computations. Once again, we define a set of variables managed by each node, intuitively explain the ranking scheme followed by its formal presentation, and finally show its correctness. Node s_i manages the following local variables.

3.2.1. Local variables

- s_i: The identifier of the ith node.
- k_i: The key of the data item s_i possesses.
- g_i: The communication set of s_i.
- *des*: The identifier of a destination node s_{des}, $s_{des} \in g_i$.
- g^i_{des}: The communication set of $s_{des} \in g_i$.
- $num1_{des}$: The number of keys, smaller than the key of *des*, received from each node $s_j \in g^i_{des}$. Initially $num1_{des} := 0$ for each *des*.
- $num2_{des}$: The number of keys, smaller than the key of *des*, received from each node $s_j \notin g^i_{des}$. Initially $num2_{des} := 0$ for each *des*.
- $rank_i$: The rank of the key of s_i.

The nodes communicate by exchanging the following two types of messages.

- KEY(k_i, s_i): sent by s_i to s_j $\forall s_j \in g_i$ to notify that the key in s_i is k_i.
- NUM$(num1_{des}, num2_{des}, s_i)$: sent by s_i to s_{des} $\forall s_{des} \in g_i$.

THE ALGORITHM

Node s_i autonomously executes the following procedure.

Round 1

Step 1: (Sending and receiving the keys)

(1) Send KEY(k_i, s_i) to s_j $\forall s_j \in g_i$.
(2) Receive KEY(k_j, s_j) from s_j $\forall s_j \in g_i$.

Step 2: (Local computation and sending the result of the computation)

for each $s_{des} \in g_i$ **do**
begin
(1) $num_1^{des} := 0$; $num_2^{des} := 0$
(2) **for** each KEY(k_j, s_j) received **do**
if $k_j < k_{des}$ **then**
if $j \in g^i_{des}$ **then** $num1_{des} := num1_{des} + 1$
else $num2_{des} := num2_{des} + 1$
(3) Send NUM$(num1_{des}, num2_{des}, s_i)$ to s_{des}.
end

Round 2

Step 3: (**Receiving the result of the computation**)

Receive Num$(num1_i, num2_i, s_j)$ from s_j $\forall s_j \in g_i$.

Step 4: (**Rank Computation**)

Compute the rank of s_i, using all NUM$(num1_i, num2_i, s_j)$ received :

$$rank_i := \frac{\sum num1_i}{m-2} + \frac{\sum num2_i}{4} + 1.$$

Now we give an example of rank computation on a six node network for which the coterie structure is shown in Fig. 2. Let nodes 1–6 hold key values 3, 5, 2, 9, 1, and 7, respectively. For $p = 6$, we have $m = 4$. After each node sends its key with the

identifier of the node to its communication set, and receives the keys and identifiers from all the nodes in its communication set, each node holds the following information (keys with identifiers):

s_1: $\{(9, s_4), (7, s_6), (5, s_2), (1, s_5)\}$
s_2: $\{(7, s_6), (2, s_3), (3, s_1), (1, s_5)\}$
s_3: $\{(7, s_6), (5, s_2), (1, s_5), (9, s_4)\}$
s_4: $\{(1, s_5), (2, s_3), (3, s_1), (7, s_6)\}$
s_5: $\{(3, s_1), (5, s_2), (2, s_3), (9, s_4)\}$
s_6: $\{(5, s_2), (2, s_3), (3, s_1), (9, s_4)\}$

Let us focus on the nodes in the communication set of s_1 given by the set $\{s_2, s_4, s_5, s_6\}$. In the first round, the messages received by these nodes are as follows.

s_4: $\{(1, s_5), (2, s_3), (3, s_1), (7, s_6)\}$,
s_6: $\{(5, s_2), (2, s_3), (3, s_1), (9, s_4)\}$,
s_2: $\{(7, s_6), (2, s_3), (3, s_1), (1, s_5)\}$,
s_5: $\{(3, s_1), (5, s_2), (2, s_3), (9, s_4)\}$.

Note that the key of each node in $S = \{s_1, s_2, \ldots, s_6\}$ appears in some nodes of the communication set, and no key appears more than once in a node. $(2, s_3)$ appears in four nodes because s_3 is not in the communication set of s_1. $(9, s_4)$, $(7, s_6)$, $(5, s_2)$, and $(1, s_5)$ appear in $2 = m - 2$ nodes. This is because s_4, s_6, s_2, and s_5 are in the communication set of s_1. $(3, s_1)$ appears in $4 = 2(m - 2)$ nodes. At each node of the communication set of s_1, the following computation is done.

s_4: $num1_1 = 1$ (since 1 $(of\ s_5) < 3$ $(of\ s_1)$) and $num2_1 = 1$
s_6: $num1_1 = 0$ and $num2_1 = 1$
s_2: $num1_1 = 1$ and $num2_1 = 1$
s_5: $num1_1 = 0$ and $num2_1 = 1$

Thus, after s_1 receives these numbers, it can compute the rank of its key using the formula in Step 4 as $rank_1 = (1 + 0 + 1 + 0)/(4 - 2) + (1 + 1 + 1 + 1)/4 + 1 = 3$. It can be seen that the rank of the key value 3 of s_1 in the sorted key list $\{1, 2, 3, 5, 7, 9\}$. is indeed 3.

3.2.2. *Correctness and analysis of our algorithm*

LEMMA 3.3. *After Step* 1, *the communication set of each node* s_k *receives all keys in the network*.

Proof. Consider the communication set of an arbitrary node s_k. The communication set is formed by two lines, say L and L', crossing at s_k. Now consider either of the two lines, say line L. L crosses with every other line in the coterie structure at exactly one node according to property 3 of the structure. The key of each node on the line which crosses with L will reach the node at the crossing node (notice that the crossing node would not be s_k), which implies that each key of each node on every other line will reach some nodes on L besides s_k. Also, for the nodes on line L', one might think that these keys will only reach s_k which is not in the communication set of s_k. However, according to property 2, these nodes also reside on some different

lines which cross with L at some other nodes but not s_k. In summary, after Step 1, each key in the network will reach some nodes in the communication set of s_k. ∎

LEMMA 3.4. *After Step* 1, *a node never receives duplicate keys.*

Proof. The proof follows from the definition of communication set and property 3. ∎

LEMMA 3.5. *After Step* 1, *each communication set holds some redundant keys. More specifically, the redundant information is as follows.*

(1) *Each node of the communication set of* s_k *receives the key of node* s_k. (*Note that there are* $2(m-2)$ *nodes in a communication set.*)

(2) *Exactly* 4 *nodes of each communication set, say the communication set of* s_k, *receive the key of a node* s_i *which is not in the communication set of* s_k.

(3) $m-2$ *nodes of the communication set, say the communication set of* s_k, *receive the key of a node* s_i *which is in the communication set of* s_k.

Proof. (1) We first show that the number of nodes in the communication set of a node is $2(m-2)$. The communication set of a node s_k is the set of nodes on the two lines which cross at node s_k (except s_k). The number of nodes on a line is $m-1$ including s_k according to property 1. Thus the number of nodes on the two lines excluding s_k is $2(m-2)$. A node s_k sends one message with its key to each node of its own communication set, and thus $2(m-2)$ nodes receive and hold the key of s_k.

(2) A node s_i, which is not in the communication set of s_k, is not on the lines of s_k. According to properties 2 and 3, two lines of s_i cross with two lines of s_k at exact 4 nodes as shown in Fig. 3. Thus, the key of node s_i reaches 4 nodes in the communication set of s_k.

(3) A node s_i, which is in the communication set of s_k, is on the same line as s_k is (see Fig. 4.) Consider an arbitrary node s_i in the communication set of s_k. According to property 1, there are $m-3$ nodes, excluding s_k and s_i, on the line on which s_k and s_i reside. These $m-3$ nodes will undoubtedly receive the key from s_i. In addition, the other line on which s_i resides crosses with the other line on which s_k resides at exactly

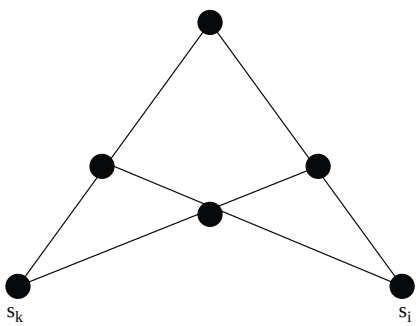

FIG. 3. The lines of s_i and s_k crossing at four nodes.

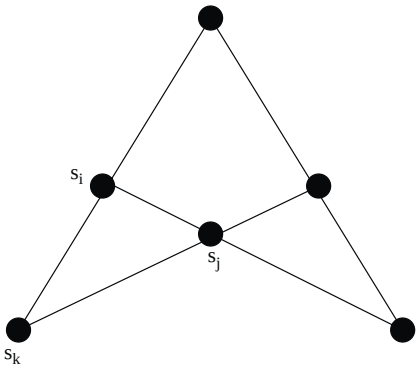

FIG. 4. Two lines of s_i and s_k crossing at s_j.

one node, say s_j (see Fig. 4). Therefore, totally there are $m-3+1=m-2$ nodes (in the communication set of s_k) which receive the key from s_i. ∎

THEOREM 3.3. *By performing our ranking algorithm, each node computes the rank of its key.*

Proof. Consider an arbitrarily chosen node, say s_k. The communication set of s_k receives all the keys on the network in Step 1 (Lemma 3.3). Each node, say s_j, in the communication set of s_k, divides the received keys into two groups depending on whether those keys are received from nodes on the same line as s_k or not. Also, s_j counts the number of keys which are smaller than the key of s_k in Step 2. Specifically, s_j counts two numbers *num*1 and *num*2. Since s_j receives the key of s_k (Lemma 3.5(1)), comparison between the key of s_k and other keys is possible. s_k computes the sum of all *num*1's received from all nodes in its communication set in Step 4. Similarly, the sum of all *num*2's is computed. In order to remove the redundancy, the sum of *num*1's and the sum of *num*2's are divided by $m-2$ and 4, respectively, according to Lemmas 3.5(2) and 3.5(3). Therefore, the quantity $\frac{\sum num1_i}{m-2}+\frac{\sum num2_i}{4}$ represents the exact number of keys smaller than the key of s_k. ∎

THEOREM 3.4. *The ranking scheme uses $O(p\sqrt{p})$ messages.*

Proof. We assume that message sent by a node eventually reaches its destination. A node receives message from a node in the same order as seen by the sender. Thus, only counting the number of received messages by all nodes is enough.

In Step 1, since the number of nodes of a communication set is $2(m-2)$ where $m=\frac{1+\sqrt{8p+1}}{2}$, every node receives $O(\sqrt{p})$ messages. Thus, totally p nodes will receive $O(p\sqrt{p})$ messages. Likewise, in Step 3, p nodes will totally receive $O(p\sqrt{p})$ messages. Thus, the total number of messages needed is $O(p\sqrt{p})$. ∎

3.3. Restricted Sorting

With our ranking algorithm, the sorting problem turns into a permutation routing problem. Precisely, after ranking all keys, our purpose in sorting is to send key k_i

with rank r_i to s_{r_i} for all keys. In the following, we first explain how to perform routing on coterie structure 2. Next, we show how the routing on coterie structure 2 is slightly different from that on coterie structure 1.

There are two cases in our routing scheme. In the first case, the destination s_{r_i} is in the communication set of s_i. Because a node knows each node in its own communication set, s_i simply sends key k_i directly to s_{r_i} in one step using only one message. In the other case, the destination s_{r_i} is not in the communication set of s_i. For this, s_i first sends key k_i to each node in its own communication set; next only those nodes whose communication sets contain s_{r_i} directly forward k_i to s_{r_i}. Based on Lemma 3.5(2), there are exactly four nodes which will forward k_i to s_{r_i}. Obviously, the first step uses $O(\sqrt{p})$ messages and the second step uses only four messages. Thus, the total number of messages sent by all the p nodes is $O(p(\sqrt{p}+4)) = O(p\sqrt{p})$. The number of messages for the restricted sorting problem is the sum of the messages for ranking and the messages for routing, which is still $O(p\sqrt{p})$.

Routing on coterie structure 1 can similarly be explained using the idea of sending sets instead of communication sets. Since, two quorums of a coterie structure 1 have exactly one common node, the message complexity is $O(p(\sqrt{p}+1)) = O(p\sqrt{p})$.

LEMMA 3.6. *The sorting of p keys, with one key per node, on a p-node network organized as a coterie structure, can be done in* $O(p\sqrt{p})$ *messages with* 4τ *communication delay.*

4. SELECTION SCHEME

Before we present our selection scheme, we explain two ideas, namely consensus and sampling technique. The consensus protocol is in fact a two phase decentralized prefix computation [8]. We summarize these two ideas in what follows.

Here we explain the prefix computation using coterie structure 2. In the first phase of the prefix computation, every node sends a message containing the value possessed by the node to all nodes in its communication set. After receiving messages from all nodes in its communication set, a node performs a simple associative operation (e.g., summation) and sends the result to all nodes in its communication set. This value represents a partial result of consensus (e.g., partial summation). In the second phase, after receiving the values of the function (e.g., partial summation) from all nodes in its communication set, a node again performs the same operation using the received partial results. Every node will produce the same value of the function (e.g., total summation) from this two-phase computation, which means the consensus is achieved. It is not hard to see that the message complexity of this simple but elegant scheme is $O(p\sqrt{p})$ which leads to the following lemma.

LEMMA 4.1. *A distributed prefix-computation can be realized on a p-node network organized as a coterie structure using* $O(p\sqrt{p})$ *messages and suffering a communication delay of* 2τ.

The sampling technique we employ is a variant of [1, 11]. It works as follows: (1) Group the keys into groups with l elements in each group (for an appropriate l); (2) Sort each group independently; (3) Collect each qth element from each group (for some q). This collection serves as a "sample" for the original input. Most likely, the sample set contains the useful information for further processing. For example, the median of this sample can be shown to be an approximate median for the input. However, we adopt a different approach. Our algorithm is a twist one of [1]. Initially, there are $\frac{n}{p}$ keys (or records) at each node. As the algorithm proceeds, keys are dropped from future consideration. We do not perform any load balancing. The remaining keys from each node will form the groups. Instead of picking the median of these medians as the element for partition, we choose a *weighted median* of these medians. Each group median is weighted with the number of remaining keys in that node.

DEFINITION 4.1. Let $X = k_1, k_2, \ldots, k_n$ be a sequence of keys where key k_i has an associated weight w_i, for $1 \leqslant i \leqslant n$. Also let $W = \sum_{i=1}^{n} w_i$. The weighted median of X is that $k_j \in X$ which satisfies $\sum_{k_l \in X \& k_l \leqslant k_j} w_{k_l} \geqslant \frac{W}{2}$ and $\sum_{k_l \in X \& k_l \geqslant k_j} w_{k_l} \geqslant \frac{W}{2}$. In other words, the total weight of all keys of X that are $\leqslant k_j$ should be $\geqslant \frac{W}{2}$ and the total weight of all the keys that are $\geqslant k_j$ also should be $\geqslant \frac{W}{2}$.

EXAMPLE 4.1. Let $X = 9,\ 15,\ 12,\ 6,\ 5,\ 2,\ 21,\ 17$ and let the respective weights be $1,\ 2,\ 1,\ 2,\ 3,\ 1,\ 7,\ 5$. Here $W = 22$. The weighted median of X is 17. One way of identifying the weighted median is to sort X; let the sorted sequence be $k'_1, k'_2, \ldots, k'_n$; let the corresponding weight sequence be $w'_1, w'_2, \ldots, w'_n$; compute the prefix sums, $y_1, \ldots, y_n$, on this weighted sequence, if y_j is the leftmost prefix sum that is $> \frac{W}{2}$, then k'_j is the weighted median.

For the above X, the sorted order is 2, 5, 6, 9, 12, 15, 17, 21 and the corresponding weights are 1, 3, 2, 1, 1, 2, 5, 7. The prefix sums of this weight sequence are 1, 4, 6, 7, 8, 10, 15, 22. The leftmost prefix sum that exceeds 11 is 15 and hence the weighted median is 17.

In Fig. 5, we present a high-level description of the selection algorithm. Each node s_i individually executes the algorithm, assuming that the algorithm is for finding the kth key from a file of size n evenly distributed over a p-node coterie. Node s_i needs to manage the following local variables in order that the algorithm can correctly performed.

N_i: The number of keys remaining in s_i. Initially, N_i is equal to $\lceil n/p \rceil$ or $\lfloor n/p \rfloor$. During the selection process, some keys will be labelled dead, which will be viewed as not remaining in the node.

M_i: A variable used to hold the median of the remaining keys in s_i. It also carries N_i with it when it is sent as a message to the neighboring nodes. The sending of M_i as a message is needed due to the invoked restricted sorting in Step 2 (M_i of each node will be repermuted in the sorted order, i.e., M_i will be moved from s_i to $s_{r_{M_i}}$, where r_{M_i} is the rank of M_i.).

M: A variable used to hold the weighted median.

c: A constant independent of n and p, and is preassigned to each node by the system.

0. Let q be the number of keys in the node. Contribute q to a prefix sum computation. Trigger the prefix sum computation to obtain the size n of the entire file.
(* To begin with, each key is "alive". During the computation, some keys will no more be considered and will be called "dead".*)
$N = n$.
repeat
1. Find the median of alive keys in local memory. Let M_i be the median and N_i be the number of remaining keys.
2. Contribute M_i to the (restricted) sorting and trigger the sorting. M_i carries N_i with it. Let M_i' be the key received after sorting and N_i' be the value carried with M_i'.
3. Contribute N_i' to the prefix computation and trigger a prefix sum computation. The value ω obtained by node s_i will be $\sum_{m=1}^{i} N_m'$.
If ω is $< \frac{N}{2}$, then notify node s_{i+1}; Otherwise, notify node s_{i-1}.
4. If the obtained value ω is $\geq \frac{N}{2}$ and node s_i itself receives a notification from node s_{i-1}, then M_i' is the weighted median M and a prefix computation is triggered to broadcast M.
5. Count the number of alive keys in the node, which are smaller than M and contribute this number to the prefix sum computation. Trigger this computation to compute the total number of alive keys which are smaller than M.
(* This is to count the rank of M, denoted r_M, out of all the remaining keys.*)
6. **if** $k \leq r_M$ **then** (* r_M is the rank of M.*)
mark those alive keys (in the local memory) that are $> M$ as dead
else
mark those alive keys that are $\leq M$ as dead.
7. Let D_i be the number of dead keys.
Contribute D_i and trigger a prefix computation to compute D, the total number of dead keys (keys eliminated).
8. **if** $k > r_M$ **then** $k = k - D$; $N = N - D$.
until $N \leq c$, c being a constant defined above
9. Trigger a prefix computation to elect a leader, e.g. the node with maximum identity, and route the surviving keys to the leader.
10. If s_i is elected, perform a local sorting on the surviving keys.
Report the kth key out of the surviving keys.

FIG. 5. The selection algorithm.

4.1. Analysis

Step 0 performs a prefix sum computation and thus requires $O(p\sqrt{p})$ messages and needs 2τ communication delay according to Lemma 4.1. Steps 1 needs only local computation and does not require message passing. In Step 2, we sort the medians and, thereby, compute the weighted median. This can be done using $O(p\sqrt{p})$ messages with 6τ communication delay according to Lemmas 4.1 and 3.6. Steps 3–4, and 5 also perform a prefix computation and thus also require $O(p\sqrt{p})$ messages and suffer 2τ communication delay (cf. Lemma 4.1). Step 6 is a local computation. Step 7 is similar to Steps 0, 3, 4, and 5. Step 8 is a local computation. Therefore, each run of the **repeat** loop uses $O(p\sqrt{p})$ messages and suffers $O(\tau)$ communication delay.

Our way of identifying the weighted median guarantees that at least $\frac{N}{4}$ keys are dropped out in each run of the **repeat** loop. Assume that $k > r_M$ in a given run. (The

other case can be proved similarly.) The number of keys dropped out is at least $\sum_{m=1}^{j} \lceil \frac{N'_m}{2} \rceil$ which is $\geqslant \frac{N}{4}$. Consequently, the repeat loop will be executed for $O(\log n)$ times. Besides, Step 9 also needs $O(p\sqrt{p})$ messages and 2τ communication delay. And Step 10 needs only local computations. Therefore, the algorithm uses $O(p \times \sqrt{p} \log n)$ messages to finish the selection with $O(\tau \log n)$ communication delay. This leads to the following theorem.

THEOREM 4.1. *Assuming that a large file of size n is distributed over a p-node network logically organized as coterie structure* 1 *or* 2, *selection on the file can be done in* $O(p^{3/2} \log n)$ *messages with communication delay* $O(\tau \log n)$.

5. SORTING LARGE DISTRIBUTED FILES

In this section, we present an enumeration sorting scheme which uses the selection of previous section for sorting a large distributed file. A common indexing scheme used for sorting a distributed file F of size n in a p-node network is that each key, key_i, will be residing at the $\lceil rank(key_i, F) \cdot \frac{p}{n} \rceil$th node after sorting. To show the optimality of our sorting scheme, we first present a lower bound.

LEMMA 5.1. *Sorting a distributed file F of size n on a p-node coterie structure, in the worst case, requires at least* $\Omega(n)$ *messages and* $\Omega(\tau \frac{n}{p})$ *delay.*

Proof. In the worst case, the destination of each key is different from its source, and thus each key has to move. This introduces at least n messages. Also, each node has to sequentially send out each of its $\frac{n}{p}$ keys, which introduces at least $\tau \frac{n}{p}$ communication delay. ∎

We then present our sorting algorithm which can sort a distributed file of size n in a p-node coterie structure network in $O(n)$ messages and suffering a communication delay of $O(\tau \frac{n}{p})$. This algorithm is optimal in the sense of both message complexity and communication delay. Now we give the basic idea behind our algorithm: perform the selection algorithm for p times; at the ith time, $i\lceil \frac{n}{p} \rceil$th key is selected and broadcast to each node of the network and each unmarked key—initially all keys are unmarked—is compared with the selected key. This computation is done so that after p iterations, each key will know its own right residence (node). Finally, we route each key to its right node. To do so, each node of the network individually performs the following algorithm.

0. $i = 1$
1. **repeat**
1.a Perform the selection algorithm of Fig. 5 to select the $i\lceil \frac{n}{p} \rceil$th key. (* The node which holds the selected key will broadcast the key to every node by triggering a broadcast prefix computation. *)
1.b Compare each unmarked key in the local memory with the selected key. Label an unmarked key as i if it is less than or equal to the selected key. (* If a key is labelled as i, it means that the key belongs to node i. *)

1.c $i = i + 1$
until $i = p + 1$
$i = 1$
2. **repeat**
2.a Route the key in memory cell i to node r if the key is labelled r.
2.b $i = i + 1$
until $i = \frac{n}{p} + 1$

THEOREM 5.1. *Sorting of a distributed file of size n can be distributedly performed on a p-node coterie structure using $O(n)$ messages and suffering a communication delay of $O(\tau \frac{n}{p})$ provided n is polynomial in p and $n = \Omega(p^{5/2} \log n)$, which is optimal.*

Proof. The selection requires $O(p^{3/2} \log n)$ messages and $O(\tau \log p)$ delay, and is executed for p times. Also, in the worst case, each of the n keys needs to move to its new home. In total, it thus requires $O(\max\{p^{5/2} \log n, n\}) = O(n)$ messages provided $n = \Omega(p^{5/2} \log n)$. Since each node has $\frac{n}{p}$ keys, it will take $\frac{n}{p}$ time for each node to send out and receive $\frac{n}{p}$ keys in the worst case. ∎

6. CONCLUSIONS

We presented two schemes for selecting and sorting the keys of a large distributed file. These schemes use the ideas of ranking and restricted sorting. The novelty of our approach lies in logically organizing the network into coterie structures. The efficiency of the schemes follows from using the properties of coterie structures in the algorithm developed. So far, the problems studied on coterie structures are mutual exclusion, consensus, and multiway rendezvous. Our selection and sorting schemes complement the application of coterie structures in distributed processing.

With a small modification, our selection and sorting schemes can be adapted to other types of coteries. Also, the message complexities of our algorithms are given based on the chosen coteries. There is a hidden cost due to the mapping of a coterie onto a real network. This hidden cost applies to all the distributed algorithms so far developed using a coterie. The study of efficient mapping or embedding of a coterie into the topology of a real network is thus important and would be interesting.

ACKNOWLEDGMENTS

Preliminary versions of portions of this paper were presented in [3, 12, 13, 14].

REFERENCES

1. M. Blum, R. Floyd, V. R. Pratt, R. Rivest, and R. Tarjan, Time bounds for selection, *J. Comput. System Sci.* **7**(4) (1972) 448–461.

2. Z. Cheng, T. Huang, and N. Shiratori, A new distributed algorithm for implementation of LOTOS, *in* "Proceedings of the 7th International Conference on Formal Description Techniques," pp. 483–494, 1994.
3. Z. Cheng, and D. Wei, Efficient distributed ranking and sorting schemes for a coterie, *in* "International Symposium on Parallel Architectures, Algorithms, and Networks," pp. 180–185, Beijing, China, June 12–14, 1996.
4. H. Garcia-Molina and D. Barbara, How to assign votes in a distributed system, *J. ACM* **32**(4) (October 1985), 841–860.
5. M. Hall Jr., "Combinatorial Theory," 2nd ed., John Wiley and Sons, New York, 1986.
6. H. Kakugawa, S. Fujita, M. Yamashita, and T. Ae, Availability of k-coterie, *IEEE Trans. Comput.* **42**(5) (May 1993), 553–558.
7. Y. Kuo and S. Huang, A geometric approach for constructing coteries and K-coteries, *IEEE Trans. Parallel Distrib. Systems* **8**(4) (April 1997), 402–411.
8. T. V. Lakshman and A. K. Agrawala, Efficient decentralized consensus protocols, *IEEE Trans. Software Eng.* **SE-12**(5) (May 1986), 600–607.
9. M. Maekawa, A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems, *ACM Trans. Comput. Systems* **3**(2) (May 1985), 145–159.
10. A. Nakajima, $2n\lfloor\sqrt{n}\rfloor$ symmetric communication structure for decentralized consensus protocols using a duality of indices, *IEICE Trans. Inform. Systems* **E77-D**(6) (June 1994), 669–675.
11. S. Rajasekaran, W. Chen, and S. Yooseph, Unifying themes for parallel selection, *in* "Proceedings of the 5th International Symposium on Algorithms and Computation," August 1994.
12. S. Rajasekaran and D. Wei, Designing efficient distributed algorithms using sampling techniques, *in* "11th International Parallel Processing Symposium," IEEE, Geneva, Switzerland, April 1–5, 1997.
13. D. S. L. Wei and Z. Cheng, An efficient distributed ranking scheme based on finite projective planes, *in* "International Conference on Parallel and Distributed Processing Techniques and Applications," pp. 473–480, Athens, GA, November 1995.
14. D. S. L. Wei, S. Rajasekaran, and S. Y. Kuo, efficient selection and sorting schemes for processing large distributed files in finite projective planes, *in* "International Conference on Parallel and Distributed Processing Techniques and Applications," pp. 69–78, Sunnyvale, CA, August 9–11, 1996.

DAVID S.L. WEI received his Ph.D. in computer and information science from the University of Pennsylvania in 1991. He is currently an associate professor of Computer and Information Science Department at Fordham University. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan (as an Associate Professor and a Professor). Dr. Wei has authored and co-authored more than 50 technical papers in the areas of distributed and parallel processing, mobile computing, and optical communications. Currently, he focuses his research effort on wireless communications and mobile computing.

SANGUTHEVAR RAJASEKARAN received his M.E. in automation from the Indian Institute of Science (Bangalore) in 1983 and his Ph.D. degree in computer science from Harvard University in 1988. He has served in the faculty of the University of Pennsylvania (as an assistant professor) and the University of Florida (as an associate professor and a professor). During 2000–2002, Dr. Rajasekaran served as the chief scientist (architectures and algorithms) of Arcot Systems, Santa Clara. From August 2002, he is the UTC Named Professor of CSE and Director of GE E-Clinic at the University of Connecticut. Dr. Rajasekaran has worked in the general area of Applied Algorithms spanning such domains as randomized computing, parallel and high performance computing, data mining, computational biology, learning theory, out-of-core computing, computational geometry, combinatorial optimization, parsing, particle simulations, etc. He has published over 100 articles in refereed journals and conference proceedings. He has co-authored two texts on algorithms and co-edited four other books.

Z. CHENG received his M.E. and Ph.D. from Tohoku university in 1990 and 1993, respectively. He was an assistant professor from 1993 to 1999 and an associate professor from 1999 to 2002, and has been a

professor, since February 2002, at the Department of Computer Software, University of Aizu. Currently, he is working on distributed algorithms, network agents, and distance education. Dr. Cheng is a member of IEEE, ACM, IEICE, and IPSJ.

K. NAIK received his B.S., M. Tech., M. Math., and Ph.D. from University College of Engineering, Burla (Sambalpur, India), Indian Institute of Technology (Kharagpur), University of Waterloo, and Concordia University, respectively. From June 1993 to July 1999, he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan. He was an associate professor in the School of Computer Science at Carleton University in Ottawa from July 1999 to June 2000. Since July 2000, he has been an associate professor in the Department of Electrical and Computer Engineering at the University of Waterloo. His research interests include communication protocols, wireless systems, and power-aware computing.

SY-YEN KUO received the B.S. (1979) in electrical engineering from National Taiwan University, the M.S. (1982) in electrical and computer engineering from the University of California at Santa Barbara, and the Ph.D. (1987) in computer science from the University of Illinois at Urbana-Champaign. Since 1991, he has been with National Taiwan University, where he is currently a professor and the Chairman of Department of Electrical Engineering. He spent his sabbatical year as a visiting researcher at AT&T Labs-Research, New Jersey from 1999 to 2000. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan from 1995 to 1998, a faculty member in the Department of Electrical and Computer Engineering at the University of Arizona from 1988 to 1991, and an engineer at Fairchild Semiconductor and Silvar-Lisco, both in California, from 1982 to 1984. In 1989, he also worked as a summer faculty fellow at Jet Propulsion Laboratory of California Institute of Technology. His current research interests include mobile computing and networks, dependable distributed systems, software reliability, and optical WDM networks. Professor Kuo is an IEEE Fellow. He has published more than 170 papers in journals and conferences. He received the distinguished research award (1997–2001) from the National Science Council, Taiwan. He was also a recipient of the Best Paper Award in the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference (DAC), the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.