An Optimal Multicast Algorithm for Cube-Connected Cycles

SONG Jianping (宋建平), HOU Zifeng (侯紫峰) and SHI Yuntao (史云涛)
Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, P.R. China
E-mail: {sjp,zfhou,syt}@ict.ac.cn

Received June 24, 1999; revised November 2, 1999.

Abstract This paper presents an efficient algorithm that implements one-to-many, or multicast, communication in one-port wormhole-routed cube-connected cycles (CCCs) in the absence of hardware multicast support. By exploiting the properties of the switching technology and the use of virtual channels, a minimum-time multicast algorithm is presented for n-dimensional CCCs that use deterministic routing of unicast messages. The algorithm can deliver a multicast message to $m-1$ destinations in $\lceil \log_2 m \rceil$ message-passing steps, while avoiding contention among the constituent unicast messages. Performance results of a simulation study on CCCs with up to 10,240 nodes are also given.

Keywords multicast, cube-connected cycle, wormhole routing, dimension-ordered routing, one-port architecture

1 Introduction

While the hypercube topology has been a very popular architecture, it has one major disadvantage: the node degree grows linearly with the hypercube dimension, which makes the hypercube ill-suited for VLSI implementation. Preparata and Vuillemin[1] considered a substitutive architecture, known as the cube-connected cycle (CCC) architecture. The CCC not only preserves all the attractive features of the hypercube, e.g., small diameter, large bisection width, and symmetry, but also has fewer links and smaller constant node degree, making it ideal for VLSI implementation. Preparata and Vuillemin also showed that the CCC can implement efficiently many widely used parallel algorithms, including merging, sorting, permutations, FFT, and several matrix operations[1]. Recently, a great deal of research has been conducted on the CCC topology[2−4].

Routing algorithms for CCC have been extensively studied in the context of *one-to-one* or *unicast* communication[5,6], in which a source sends its message precisely to one destination. More powerful communication primitive is *multicast* (or *one-to-many*), which involves the delivery of the same message from a source node to an arbitrary number of destination nodes. A special case of multicast is *broadcast*, in which the destination nodes contain every node in the network. Multicast is an important communication pattern found in many parallel numerical algorithms, including matrix multiplication, matrix transposition, Gaussian elimination[7], LU factorization[8], and tridiagonalization[9].

Most existing parallel computers support only unicast communication in hardware. In these environments, multicast must be implemented in software by sending multiple unicast messages. Such implementations are called *unicast-based*[10]. Sending a separate copy of the message from the source to every destination may require excessive time due to a bottleneck at the source node. An alternative approach is to use a *multicast tree*[10] of unicast messages. In a multicast tree, the source node actually sends the message to only a subset of the

The work of this paper is supported by the National Natural Science Foundation of China under grant No.69896250.

destinations. Each recipient of the message forwards it to some subset of the destinations that have not yet received it. The process continues until all destinations have received the message. Using this approach, the time required for the operation can be greatly reduced[10].

On one-port systems, each node can send only one message at a time, regardless of the topology. For an implementation of unicast-based multicast, this restriction means that at least $\lceil \log_2 m \rceil$ message-passing steps are required to deliver the message to $m-1$ destinations, since the number of nodes holding the message can at most double with each step. Achieving this bound is not always simple, however, and requires that an algorithm must avoid *channel contention*, in which two or more messages involved in the operation simultaneously require the same channel. McKinley *et al.*[10] previously developed unicast-based multicast algorithms for one-port n-dimensional meshes and hypercubes. Robinson *et al.*[11] extended the research to wormhole-routed n-dimensional torus networks. Recently, Hong Xu *et al.*[12] continued the research in multistage networks.

In this paper, we extend the research to one-port wormhole-routed n-dimensional cube-connected cycles. The remainder of the paper is organized as follows. Section 2 presents the system model under consideration, and Section 3 describes the unicast routing algorithm for CCCs. In Section 4, we develop the theoretical results regarding channel contention in CCCs. The optimal unicast-based multicast algorithm for CCCs is presented in Section 5 and the performance results of the proposed multicast algorithm are given in Section 6. Finally, conclusions are made in Section 7.

2 System Model

Formally, an n-dimensional cube-connected cycle can be defined as follows[13]:

Definition 1. *The n-dimensional Cube-Connected Cycle, denoted as n-CCC, has* $n(2^n)$ *nodes and* $3n(2^{n-1})$ *edges. We can label each node with a pair* (i, w), *where* w *is an n-bit binary address that denotes the cycle of the node and* i *is the dimension of the node* $(0 \le i \le n-1)$. *Two nodes* (i, w) *and* (i', w') *are linked by an edge in the CCC if and only if either*

1) $w = w'$ *and* $i - i' \equiv \pm 1 \bmod n$, *or*

2) $i = i'$ *and* w *differs from* w' *in precisely the i-th bit.*

Edges of the first type are called cycle edges in cycle w, while edges of the second type are referred to as hypercube edges in dimension i.

As an example, a 3-dimensional cube-connected cycle (3-CCC) is shown in Fig.1.

In order to reduce network latency and minimize buffer requirements, the system under consideration uses the *wormhole routing* switching strategy[14]. In wormhole routing, a message is divided into a number of *flits* for transmission. The header flit of a message governs the route, and the remaining flits follow in a pipelined fashion. An important feature of wormhole routing is that the network latency is distance-insensitive when there is no channel contention[15].

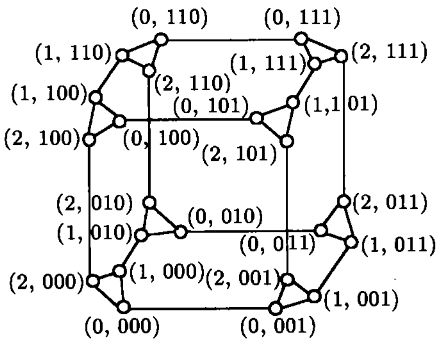


Fig.1. An example of 3D cube-connected cycle (3-CCC).

A great deal of research has been conducted in the last few years on the subject of wormhole routing algorithms[16-18]. Deadlock avoidance is the essential issue in the design of such algorithms. A widely used routing technique that can avoid deadlock is *dimension-ordered routing*[16]. Special cases of dimension-ordered routing include E-cube routing and XY routing for the hypercube and 2D mesh topologies, respectively[15]. In this paper, we adopt a wormhole routing algorithm in CCCs that is referred to as *HC routing* ('HC' for Hypercube edge and Cycle edge) for the purpose of discussion. HC routing is similar to dimension-ordered routing. In fact, HC routing is a combination of E-cube routing in hypercubes and XY routing in 1D torus. In order to describe HC routing, we should give the following definition:

Definition 2. *If x and y are two n-bit binary addresses, and if $x \neq y$, then the leftmost bit in which x and y differ is denoted by d_{xy}. That is, $d_{xy} = \max\{i | x_i \neq y_i,\ 0 \leq i \leq n-1\}$.*

Given the current node (i, x) and the destination node (j, y), HC routing can be described as follows:

1) If $x = y$, then the next node of the routing path is (k, x), where $k - i \equiv \pm 1 \bmod n$.

2) If $x \neq y$ and if $i = d_{xy}$, then the next node of the routing path is (i, w), where w differs from x in precisely the i-th bit. That is, $w = x_{n-1} \cdots x_{i+1} y_i x_{i-1} \cdots x_0 = y_{n-1} \cdots y_i x_{i-1} \cdots x_0$.

3) If $x \neq y$ and if $i \neq d_{xy}$, then the routing path consists of two sub-paths, one from node (i, x) to node (d_{xy}, x), the other from node (d_{xy}, x) to node (j, y).

Fig.2 shows the routing paths taken by two example unicast messages, one from source node (1, 000) to destination node (0, 111), and the other from source node (2, 001) to destination node (0, 011).

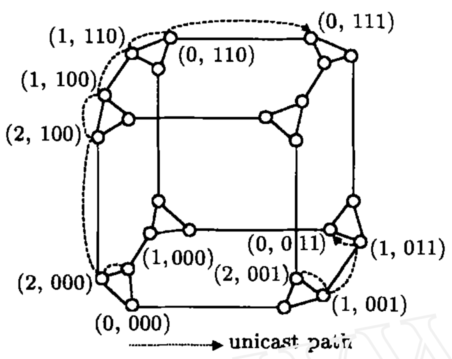


Fig.2. Examples of HC routing in CCCs. (For simplicity, most nodes are not labeled.)

3 Unicast Routing Algorithm

We now describe the unicast routing algorithm, which will be considered later in the context of its support of unicast-based multicast operations. In order to prevent channel contention in CCCs, single channels on some edges are replaced with multiple virtual channels, thus allowing the underlying unicast routing algorithm to choose among these virtual channels in such a way as to eliminate channel contention. Each virtual channel has its own flit buffer and control[19].

Virtual channels may be used in a variety of ways to eliminate channel contention, but how they are used has a significant effect on the design of efficient unicast-based collective communication operations, such as multicast. In CCCs, channel contention happens more frequently in a cycle edge than in a hypercube edge. In fact, we need no virtual channels in hypercube edges, while a bidirectional physical channel (or two unidirectional physical channels) in a cycle edge is replaced with four unidirectional virtual channels in order to design a minimum-time contention-free multicast algorithm.

The four unidirectional virtual channels can be divided into two groups. The channels in h-group ('h' for high-direction) are directed towards higher-address neighboring nodes, while lower-address neighboring nodes are reached through channels in l-group ('l' for low-direction). The virtual channels along a single cycle, w, of an n-dimensional cube-connected cycle are illustrated in Fig.3.

Formally, for each cycle w, $0 \leq w \leq 2^n - 1$, and for each node (i, w), let (j, w) be the node such that $j = i + 1$, and let (k, w) be the node such that $k = i - 1$. Then under CCCs,

1) there are two channels in h-group, $c_{w_i h_0}$ and $c_{w_i h_1}$, from (i, w) to (j, w) whenever $0 \leq i \leq n-2$, and

2) there are two channels in l-group, $c_{w_il_0}$ and $c_{w_il_1}$, from (i,w) to (k,w) whenever $1 \le i \le n-1$.

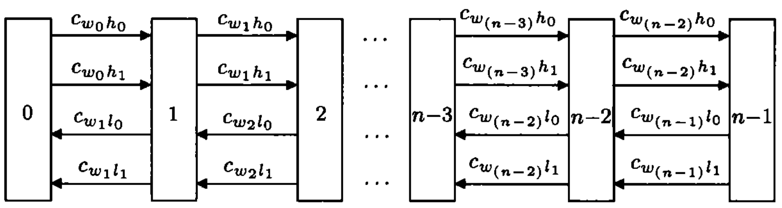


Fig.3. Virtual channels in row w of an n-CCC.

The unicast routing algorithm is described formally by the function R_{CCC}: $N \times N \to C$, which maps a pair (*current node, destination node*) into the next channel of the routing path. In order to define $R_{\mathrm{CCC}}((i,x), (j,y))$, let

$$k = \begin{cases} d_{xy}, & \text{if } x \neq y; \\ j, & \text{if } x = y \end{cases}$$

and let $\Delta = k - i$. If $x \neq y$ and $i = d_{xy}$, then $R_{\mathrm{CCC}}((i,x), (j,y))$ is the physical channel in the hypercube edge between nodes (i,x) and (i,w), where w differs from x in precisely the i-th bit. That is, $w = x_{n-1}\cdots x_{i+1}y_i x_{i-1}\cdots x_0 = y_{n-1}\cdots y_i x_{i-1}\cdots x_0$. Otherwise,

$$R_{\mathrm{CCC}}((i,x),(j,y)) = \begin{cases} c_{x_ih_0}, & \text{if } \Delta > 0 \text{ and } x \le y; \\ c_{x_ih_1}, & \text{if } \Delta > 0 \text{ and } x > y; \\ c_{x_il_0}, & \text{if } \Delta < 0 \text{ and } x < y; \\ c_{x_il_1}, & \text{if } \Delta < 0 \text{ and } x \ge y. \end{cases}$$

A message is routed from a source node to a destination node by applying the R_{CCC} function, first at source, and then at each node through which the message travels, until the destination is reached. Implementing the R_{CCC} function in a router is straightforward.

4 Contention in CCCs

The unicast routing algorithm directly affects the design of multicast algorithm in wormhole-routed systems, because it determines how the constituent messages must be scheduled in order to avoid channel contention. Before formally studying the contention between messages, we present some notations. The path from a source node α to a destination node β resulting from HC routing in an n-CCC is denoted by $P(\alpha,\beta)$. A unicast from node α to node β occurring at step t is denoted by $(\alpha,\beta,P(\alpha,\beta),t)$. If the path from node (i,x) to node (j,y) traverses a cycle edge $(r,w) \to (s,w)$, then the path can be denoted by $P((i,x),(j,y)) = (i,x) \Rightarrow (k,w) \Rightarrow (r,w) \to (s,w) \Rightarrow (t,w) \Rightarrow (j,y)$, where (k,w) and (t,w) are the first node and the last node of the path in cycle w, respectively, and the mark '$\Rightarrow$' denotes zero or more hops that a path traverses, and '$\to$' denotes one hop.

Definition 3[10] formally describes a *dimension order*, denoted by $<_d$, which is a lexicographical ordering (and thus, a total ordering) on the node labels of a CCC network. For example, given nodes (0, 00001), (3, 10000), (1, 01100) and (4, 00001) in a 5-CCC, we have (0, 00001) $<_d$ (4, 00001) $<_d$ (1, 01100) $<_d$ (3, 10000).

Definition 3. *The binary relation dimension order, $<_d$, is defined between two nodes (i,x) and (j,y) in CCCs as follows: $(i,x) <_d (j,y)$ if and only if either*

1) *$x < y$, or*

2) *$x = y$ and $i \le j$.*

The *reachable set*[10] of a node, say node α, in a multicast implementation is defined to be the set of nodes in the multicast that receive the message, either directly or indirectly, through node α. If the multicast is viewed as a tree of unicast messages, then the reachable set of node α is the set of nodes in the subtree rooted at α. We now formally define the concept of reachable set[10]:

Definition 4. *A node β is in the reachable set of node α, denoted as R_α, if and only if one of the following holds:*

1) $\beta = \alpha$; or

2) the multicast implementation contains a unicast $(\gamma, \beta, P(\gamma, \beta), t)$ such that $\gamma \in R_\alpha$.

We now present several theorems regarding the contention in wormhole-routed CCCs. These theorems are important in verifying that the proposed multicast algorithm, presented in Section 5, produces only multicast operations whose constituent unicast messages are *depth contention-free*[10], or pairwise contention-free. McKinley *et al.* previously presented the following theorem[10]:

Theorem 1. *Given a multicast implementation, if at least one of the following four conditions holds for every pair of resultant unicasts $(\alpha, \beta, P(\alpha, \beta), t)$ and $(\gamma, \delta, P(\gamma, \delta), \tau)$, where $t \le \tau$, then the multicast is depth contention-free.*

1) $\gamma \in R_\beta$;

2) $P(\alpha, \beta)$ and $P(\gamma, \delta)$ are arc-disjoint;

3) $\gamma = \alpha$;

4) $\gamma \in R_\eta$ and $(\alpha, \eta, P(\alpha, \eta), t + i)$ is a product of the multicast, for some node η and positive integer i.

Before developing Theorem 2 that identifies the situation in an n-CCC, in which pairs of unicast messages are contention-free, we first give some lemmas. Note that throughout the paper, we will assume that addresses are resolved from higher to lower bits.

Lemma 1. *For any two n-bit binary addresses x and y, if $x_k \ne y_k$ for some integer k, then $d_{xy} \ge k$.*

Lemma 2. *For any two n-bit binary addresses x and y, if $x \le y$ and $x_k > y_k$ for some integer k, then $d_{xy} > k$.*

Above two lemmas are obvious.

Lemma 3. *If there is a hypercube edge $(t, u) \to (t, v)$ in path $P((i, x), (j, y))$, then $x_t \ne y_t$.*

Proof. Since $(t, u) \to (t, v)$ is a hypercube edge in path $P((t, u), (j, y))$, then according to HC routing, $t = d_{uy}$. That is, $u_t \ne y_t$. Since the addresses are always resolved from higher to lower bits, then we have $x_t = u_t$. Therefore, we have $x_t \ne y_t$. □

Lemma 4. *For any two nodes (i, x) and (j, y) in an n-CCC, if there is a cycle edge $e = (r, w) \to (s, w)$ in path $P((i, x), (j, y))$, then let path $P((i, x), (j, y)) = (i, x) \Rightarrow (k, w) \Rightarrow (r, w) \to (s, w) \Rightarrow (t, w) \Rightarrow (j, y)$. If $w \ne x$, then $x_k \ne y_k$. If $w \ne y$, then $x_t \ne y_t$. If $w \ne x$ and $w \ne y$, then $k \ge r > s \ge t$.*

Proof. By the notation described above, we know that (k, w) and (t, w) are the first node and the last node of path $P((i, x), (j, y))$ in cycle w, respectively. If $w \ne x$, then since (k, w) is the first node of path $P((i, x), (j, y))$ in cycle w, so the last edge of path $P((i, x), (k, w))$ is a hypercube edge. By Lemma 3, we have $x_k \ne y_k$. Similarly, if $w \ne y$, then since (t, w) is the last node of path $P((i, x), (j, y))$ in cycle w, so the first edge of path $P((t, w), (j, y))$ is a hypercube edge. By Lemma 3, we have $x_t \ne y_t$. Since the HC routing resolves addresses from higher to lower bits, then $k > t$. Since all the edges in path $(k, w) \Rightarrow (r, w) \to (s, w) \Rightarrow (t, w)$ are cycle edges, then we have $k \ge r > s \ge t$. □

Lemma 5. *For any four nodes (k_1, u), (k_2, v), (k_3, x) and (k_4, y) in an n-CCC, if $(k_2, v) <_d (k_3, x) <_d (k_4, y)$, then paths $P((k_1, u), (k_2, v))$ and $P((k_3, x), (k_4, y))$ have no common hypercube edges.*

Proof. The proof is done by contradiction. Assume that there exists a hypercube edge e in dimension k shared by paths $P((k_1,u),\ (k_2,v))$ and $P((k_3,x),\ (k_4,y))$. Let $e = (k,p) \rightarrow (k,q)$. Since the edge e is in the path $P((k_1,u),(k_2,v))$, then according to HC routing, we have $q_i = v_i$ for $k \le i \le n-1$. Similarly, $q_i = y_i$ for $k \le i \le n-1$. Thus, $v_i = y_i$ for $k \le i \le n-1$. Since $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, then by the definition of dimension order, $v \le x \le y$. Thus $v_i = x_i = y_i$ for $k \le i \le n-1$. So $x_k = y_k$. But since e is a hypercube edge in the path $P((k_3,x),(k_4,y))$, then by Lemma 3, $x_k \ne y_k$, which contradicts the result that $x_k = y_k$. Therefore, the assumption that there exists a common hypercube edge e in paths $P((k_1,u),(k_2,v))$ and $P((k_3,x),(k_4,y))$ does not hold, and the lemma is proved. □

Lemma 6. *For any four nodes $(k_1,u),(k_2,v),\ (k_3,x)$ and (k_4,y) in an n-CCC, if $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, and paths $P((k_1,u),(k_2,v))$ and $P((k_3,x),(k_4,y))$ have a common cycle edge $e = (r,w) \rightarrow (s,w)$, and $r < s$, then $w \le y$ and $w > v$.*

Proof. Since $r < s$, then by Lemma 4, $w = x$ or $w = y$.

Assume that $w > y$. Since $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, then $v \le x \le y$. Thus $w \ne x$ and $w \ne y$, which contradicts the result that $w = x$ or $w = y$.

Assume that $w < v$. Since $v \le x \le y$, then $w \ne x$ and $w \ne y$, which contradicts the result that $w = x$ or $w = y$.

Assume that $w = v$. Since $w = x$ or $w = y$, and $v \le x \le y$, so we have $w = v = x$. Since $(k_2,v) <_d (k_3,x)$, then $k_2 \le k_3$. Since $w = v = x$, then we can let path $P((k_1,u),(k_2,v)) = (k_1,u) \Rightarrow (k,w) \Rightarrow (r,w) \rightarrow (s,w) \Rightarrow (k_2,w)$, and let path $P((k_3,x),(k_4,y)) = (k_3,w) \Rightarrow (r,w) \rightarrow (s,w) \Rightarrow (t,w) \Rightarrow (k_4,y)$. Since $r < s$, then $s \le k_2$ and $k_3 \le r < s$. Thus $k_3 < k_2$, which contradicts the result that $k_2 \le k_3$.

Since each of the above three assumptions leads to a contradiction, then we can conclude that $w \le y$ and $w > v$. □

Lemma 7. *For any four nodes $(k_1,u),(k_2,v),(k_3,x)$ and (k_4,y) in an n-CCC, if $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, and paths $P((k_1,u),(k_2,v))$ and $P((k_3,x),(k_4,y))$ have a common cycle edge $e = (r,w) \rightarrow (s,w)$, and $r > s$, then $w \ge v$.*

Proof. The proof is done by contradiction. Assume that $w < v$. Let path $P((k_1,u),(k_2,v)) = (k_1,u) \Rightarrow (k,w) \Rightarrow (r,w) \rightarrow (s,w) \Rightarrow (t,w) \rightarrow (k_2,v)$, and let path $P((k_3,x),(k_4,y)) = (k_3,x) \Rightarrow (k',w) \Rightarrow (r,w) \rightarrow (s,w) \Rightarrow (t',w) \Rightarrow (k_4,y)$. Since $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, then $v \le x \le y$. Thus, by the assumption of $w < v$, we have $w \ne v$ and $w \ne y$. Then, according to HC routing, we have

$$w = v_{n-1}\cdots v_{t+1}u_t\cdots u_0 = y_{n-1}\cdots y_{t'+1}x_{t'}\cdots x_0 \tag{1}$$

Since $w < v$, then from (1), we have

$$u_t < v_t \tag{2}$$

There are two cases to be considered with regard to t: $t > t'$ and $t \le t'$.

Case 1. $t > t'$. Then from (1), $u_t = y_t$ and $v_i = y_i$ for $t < i \le n-1$. Thus, from (2), $y_t < v_t$. Since $v_i = y_i$ for $t < i \le n-1$ and $y_t < v_t$, we have $y < v$, which contradicts the condition that $(k_2,v) <_d (k_4,y)$.

Case 2. $t \le t'$. Then from (1), $v_i = y_i$ for $t'+1 \le i \le n-1$. Since $v \le x \le y$, then $v_i = x_i = y_i$ for $t'+1 \le i \le n-1$. Then from (1), we have $w = x$. Thus, by the assumption of $w < v$, we have $x < v$, which contradicts the condition that $(k_2,v) <_d (k_3,x)$.

Since each of the above two cases leads to a contradiction, then the assumption that $w < v$ does not hold. □

Lemma 8. *For any four nodes $(k_1,u),(k_2,v),(k_3,x)$ and (k_4,y) in an n-CCC, if $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, and paths $P((k_1,u),(k_2,v))$ and $P((k_3,x),(k_4,y))$ have a common cycle edge $e = (r,w) \rightarrow (s,w)$, and $r > s$, then $w < y$.*

Proof. The proof is done by contradiction.

Assume that $w > y$. Let path $P((k_1,u),(k_2,v)) = (k_1,u) \Rightarrow (k,w) \Rightarrow (r,w) \rightarrow (s,w) \Rightarrow (t,w) \Rightarrow (k_2,v)$, and let path $P((k_3,x),(k_4,y)) = (k_3,x) \Rightarrow (k',w) \Rightarrow (r,w) \rightarrow (s,w) \Rightarrow$

$(t',w) \Rightarrow (k_4,y)$. Since $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, then $v \le x \le y$. Thus, by the assumption of $w > y$, we have $w \ne v$ and $w \ne y$. Then, according to HC routing, we have

$$w = v_{n-1} \cdots v_{t+1} u_t \cdots u_0 = y_{n-1} \cdots y_{t'+1} x_{t'} \cdots x_0 \tag{3}$$

Since $w > y$, then from (3), we have $x_{t'} > y_{t'}$. Thus, by Lemma 2, we have $d_{xy} > t'$. From (3), we have $v_i = y_i$ for $\max(t,t') < i \le n-1$. Since $v \le x \le y$, then $v_i = x_i = y_i$ for $\max(t,t') < i \le n-1$. Thus, $d_{xy} \le \max(t,t')$. If $t \le t'$, then $d_{xy} \le t'$, which contradicts the result that $d_{xy} > t'$. Thus $t > t'$ and $d_{xy} \le t$. Since $w > y \ge x$, then by Lemma 4, $x_{k'} \ne y_{k'}$. Thus, by Lemma 1, we have $d_{xy} \ge k'$. By the results that $d_{xy} \le t$ and $d_{xy} \ge k'$, we have $k' \le t$. Since e is a cycle edge in path $P((k_1,u),(k_2,v))$ and $r > s$, then according to HC routing, we have $k \ge r > s \ge t$. Similarly, we also have $k' \ge r > s \ge t'$. Thus, $k' > t$, which contradicts the result that $k' \le t$. Thus, the assumption that $w > y$ does not hold.

Assume that $w = y$. If $w = x$, then $P((k_3,x),(k_4,y)) = (k_3,w) \Rightarrow (r,w) \to (s,w) \Rightarrow (k_4,w)$. Since $r > s$, then according to HC routing, we have $k_3 \ge r > s \ge k_4$. That is, $k_3 > k_4$, which contradicts the condition that $(k_3,x) <_d (k_4,y)$. Thus, $w \ne x$. Since $v \le x \le y$, $w = y$ and $w \ne x$, then $w > x \ge v$. Let path $P((k_1,u),(k_2,v)) = (k_1,u) \Rightarrow (k,w) \Rightarrow (r,w) \to (s,w) \Rightarrow (t,w) \Rightarrow (k_2,v)$, and let path $P((k_3,x),(k_4,y)) = (k_3,x) \Rightarrow (k',w) \Rightarrow (r,w) \to (s,w) \Rightarrow (k_4,w)$. Since e is a cycle edge in path $P((k_1,u),(k_2,v))$ and $r > s$, then according to HC routing, we have $k \ge r > s \ge t$. Similarly, we also have $k' \ge r > s \ge t'$. Thus, $k' > t$. Since $w \ne x$, then by Lemma 4, $x_{k'} \ne y_{k'}$. Thus, by Lemma 1, we have $d_{xy} \ge k'$. So $d_{xy} > t$. Since $w > x \ge v, w \ne v$. Then according to HC routing, we have $w = v_{n-1} \cdots v_{t+1} u_t \cdots u_0$. By the assumption of $w = y$, we have $v_i = y_i$ for $t+1 \le i \le n-1$. Since $v \le x \le y$, then $v_i = x_i = y_i$ for $t+1 \le i \le n-1$. Thus, $d_{xy} \le t$, which contradicts the result that $d_{xy} > t$. Thus, the assumption that $w = y$ does not hold.

Since neither of the above two assumptions holds, then we can conclude that $w < y$. □

Paths with no common channels are said to be *arc-disjoint*. We now present the theorem that gives sufficient conditions under which unicast message paths will be arc-disjoint in a cube-connected cycle using the unicast routing algorithm described in Section 3.

Theorem 2. *For any four nodes $(k_1,u),(k_2,v),(k_3,x)$ and (k_4,y) in an n-CCC, if $(k_2,v) <_d (k_3,x) <_d (k_4,y)$, then paths $P((k_1,u),(k_2,v))$ and $P((k_3,x),(k_4,y))$ are arc-disjoint.*

Proof. The proof is done by contradiction. Assume that paths $P((k_1,u),(k_2,v))$ and $P((k_3,x),(k_4,y))$ have a common channel e. By Lemma 5, e can't be a channel in hypercube edges. Thus, e should be a channel in a cycle edge. Let $e = (r,w) \to (s,w)$. There are four cases to be considered with regard to e.

Case 1. e is an h_0-channel. Since e is in path $P((r,w),(k_2,v))$, then by the definition of function R_{CCC}, we have $r < s$ and $w \le v$, which contradicts Lemma 6.

Case 2. e is an h_1-channel. Since e is in path $P((r,w),(k_4,y))$, then by the definition of function R_{CCC}, we have $r < s$ and $w > y$, which contradicts Lemma 6.

Case 3. e is an l_0-channel. Since e is in path $P((r,w),(k_2,v))$, then by the definition of function R_{CCC}, we have $r > s$ and $w < v$, which contradicts Lemma 7.

Case 4. e is an l_1-channel. Since e is in path $P((r,w),(k_4,y))$, then by the definition of function R_{CCC}, we have $r > s$ and $w \ge y$, which contradicts Lemma 8.

Since each of the above four cases leads to a contradiction, then the assumption that paths $P((k_1,u),(k_2,v))$ and $P((k_3,x),(k_4,y))$ have a common channel e does not hold. □

5 Optimal Multicast Algorithm

In this section, we use the theorems in Section 4 to develop a unicast-based multicast algorithm for wormhole-routed cube-connected cycles. We show that this algorithm produces pairwise contention-free unicast messages and completes the multicast in the minimum possible number of steps.

As shown in Fig.4, the algorithm uses a recursive doubling procedure. We assume for simplicity that m, the number of nodes involved in the multicast, is a power of 2. Fig.4 shows all unicasts occurring in the first three steps (labeled "[1]," "[2]," and "[3]," respectively), as well as two unicasts occurring in the final step (labeled "[$\log_2 m$]"). As shown, the source node, d_0, sends the message to destination node $d_{m/2}$ during the first step. This operation partitions the multicast problem of size m into two problems, each of size $m/2$, with source nodes d_0 and $d_{m/2}$, respectively. This process continues recursively until all destination nodes have received the message.

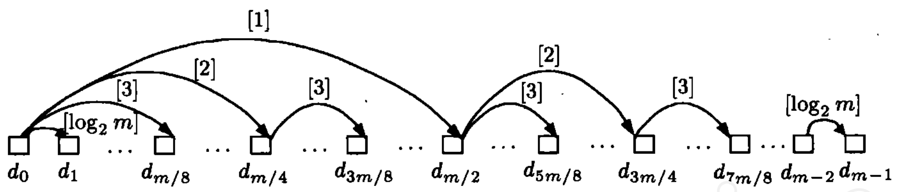


Fig.4. An optimal multicast for CCCs.

The key to avoiding contention among the constituent messages is the ordering of the destination nodes. Given a set of node labels, they can be arranged in a unique, ordered sequence according to the $<_d$ relation.

Definition 5[10]. *A sequence of nodes* $\{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}\}$ *is a dimension-ordered chain if and only if all the elements are distinct and* $\alpha_i <_d \alpha_{i+1}$ *for* $0 \le i < m-1$.

In order to develop a minimum-time, contention-free multicast algorithm for CCCs, we need to arrange the nodes involved in the multicast in the order described by Definition 6[11].

Definition 6. *If* $\Phi = \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}\}$ *is a dimension-ordered chain and* α_s *is an element of* Φ, *then* $\{\alpha_s, \alpha_{s+1}, \ldots, \alpha_{m-1}, \alpha_0, \alpha_1, \ldots, \alpha_{s-1}\}$ *is an R-chain with respect to* α_s.

An R-chain is an end-around rotation of a dimension-ordered chain. Note that, for any dimension-ordered chain $\Phi = \{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}\}$, Φ is an R-chain with respect to α_0. That is, any dimension-ordered chain is also an R-chain with respect to the first element. As an example of the construction of an R-chain, we consider the following nodes involved in a multicast from source node (3, 01010) in a 5-CCC.

$$\Phi = \{(3,01010),(1,00101),(0,10000),(4,01011),(2,10101),(3,00000),(1,01011),(0,11000)\}$$

First, we arrange Φ according to the $<_d$ relation, to obtain the dimension-ordered chain:

$$\Phi' = \{(3,00000),(1,00101),(3,01010),(1,01011),(4,01011),(0,10000),(2,10101),(0,11000)\}$$

Next, we rotate the chain Φ' so that the source node, (3, 01010), appears at the head of the list. This procedure results in the following R-chain:

$$\Phi'' = \{(3,01010),(1,01011),(4,01011),(0,10000),(2,10101),(0,11000),(3,00000),(1,00101)\}$$

```
The U-CCC algorithm
Input: R-chain $\{d_{left}, d_{left+1}, \ldots, d_{right}\}$, where $d_{left}$ is the local address.
Output: Send $\lceil \log_2(right - left + 1) \rceil$ messages
Procedure:
    while $left < right$ do
        $center = left + \lceil (right - left + 1)/2 \rceil$;
        $D = \{d_{center}, d_{center+1}, \ldots, d_{right}\}$;
        Send a message to node $d_{center}$ with the address field $D$;
        $right = center - 1$
    endwhile
```

Fig.5. The U-CCC algorithm.

Fig.5 gives the U-CCC algorithm, which implements the recursive doubling process described above in a cube-connected cycle. The algorithm takes an R-chain as input. The application of the U-CCC algorithm to the R-chain Φ'' from the previous example is illustrated in Fig.6.

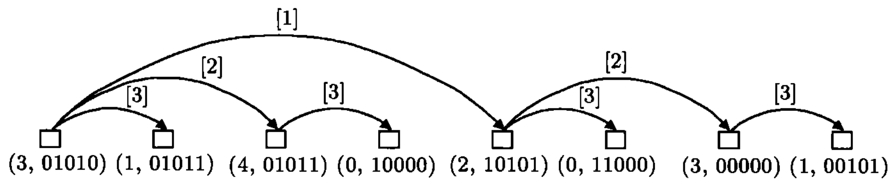


Fig.6. An example of multicast using the U-CCC algorithm.

Theorem 3 guarantees that the U-CCC algorithm is a minimum-time and contention-free multicast algorithm.

Theorem 3. *If $\Phi = \{d_0, d_1, \ldots, d_{m-1}\}$ is an R-chain, then the U-CCC algorithm applied to Φ, under CCCs, performs a minimum-time, contention-free multicast from source node d_0 to destinations $\{d_1, \ldots, d_{m-1}\}$.*

Proof. It's known that the lower bound on the number of message-passing steps required to multicast data to $m-1$ destinations is $\lceil \log_2 m \rceil$. From the above discussion and the U-CCC algorithm, it is easy to see that the algorithm produces a multicast from the source to the intended destinations, and that if contention is avoided, only $\lceil \log_2 m \rceil$ steps are required to complete a multicast to $m-1$ destinations. The more difficult task is to show that unicast messages produced by the algorithm are always pairwise contention-free.

Let $(\alpha, \beta, P(\alpha, \beta), t)$ and $(\gamma, \delta, P(\gamma, \delta), \tau)$ be any two unicasts produced by an invocation of the U-CCC algorithm, and assume, without loss of generality, that $t \le \tau$. As shown in Fig.7, there are three possible relationships between the two unicasts. In case 1, $\gamma = \alpha$, so by 3) of Theorem 1, the unicasts are contention-free. In case 2, 4) of Theorem 1 holds, so again, the unicasts are contention-free.

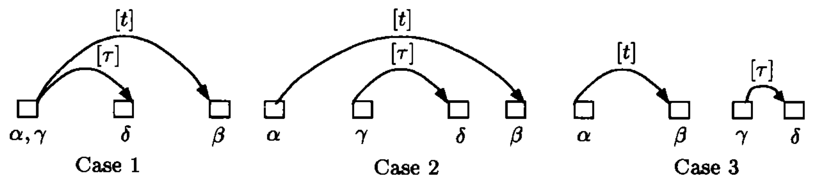


Fig.7. Possible relations between unicasts produced by U-CCC.

We now show that the unicasts represented by case 3 are arc-disjoint, and hence, by 2) of Theorem 1, contention-free. We note from Definition 6 that an R-chain, $\Phi = \{\alpha_s, \alpha_{s+1}, \ldots, \alpha_{m-1}, \alpha_0, \alpha_1, \ldots, \alpha_{s-1}\}$, consists of two concatenated sub-chains, $\Phi_a = \{\alpha_s, \alpha_{s+1}, \ldots, \alpha_{m-1}\}$ and $\Phi_b = \{\alpha_0, \alpha_1, \ldots, \alpha_{s-1}\}$. Since nodes α, β, γ, and δ appear in the given order $(\alpha, \beta, \gamma, \delta)$ in the R-chain, there are five possible subcases: 1) $\alpha, \beta, \gamma, \delta \in \Phi_a$, 2) $\alpha, \beta, \gamma \in \Phi_a$; $\delta \in \Phi_b$, 3) $\alpha, \beta \in \Phi_a$; $\gamma, \delta \in \Phi_b$, 4) $\alpha \in \Phi_a$; $\beta, \gamma, \delta \in \Phi_b$, 5) $\alpha, \beta, \gamma, \delta \in \Phi_b$.

Note that node α occurs before β in the R-chain. If α and β belong to the same sub-chain (that is, if either $\alpha, \beta \in \Phi_a$ or $\alpha, \beta \in \Phi_b$), then $\alpha <_d \beta$. Also, if the two nodes belong to different sub-chains (that is, if $\alpha \in \Phi_a$ and $\beta \in \Phi_b$), then $\beta <_d \alpha$. The situation is similar to the nodes γ and δ. Thus, we can conclude, for each of the above subcases, respectively, the following: 1) $\alpha <_d \beta <_d \gamma <_d \delta$, 2) $\delta <_d \alpha <_d \beta <_d \gamma$, 3) $\gamma <_d \delta <_d \alpha <_d \beta$, 4) $\beta <_d \gamma <_d \delta <_d \alpha$, 5) $\alpha <_d \beta <_d \gamma <_d \delta$.

Since Theorem 2 applies to each of the above five subcases, we can conclude that, in case 3 of Fig.7, paths $P(\alpha, \beta)$ and $P(\gamma, \delta)$ are arc-disjoint, and hence, by 2) of Theorem 1, contention-free. □

6 Performance Evaluation

In Section 5, we have shown that the U-CCC algorithm completes in a minimum number of steps, and the unicasts produced by the algorithm are pairwise contention-free. As a practical consideration, however, we note that it is possible for the U-CCC algorithm to generate pairs of unicasts that simultaneously use two different virtual channels joining the same pair of neighboring nodes. If each virtual channel of a CCC corresponds to a distinct physical communication link, then the above situation does not affect the performance of U-CCC algorithm. However, we must also consider CCCs in which pairs of parallel virtual channels are multiplexed onto a single physical communication link[15].

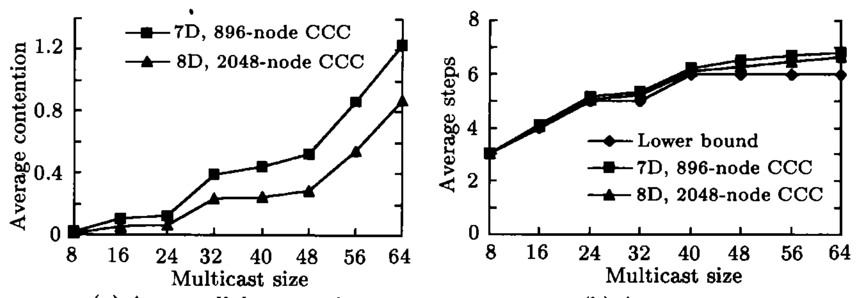
When two virtual channels are multiplexed onto a physical link, they share the bandwidth of that link. If one of the virtual channels is idle, then a message traversing the other virtual channel will use all the bandwidth of the physical link. If two messages simultaneously use the virtual channels that are multiplexed onto a single physical link, however, the speed at which these messages are delivered to their destinations is reduced by half.

In order to study the effect of virtual channel multiplexing on the performance of the U-CCC algorithm, we examined its behavior when executed on destination sets in which the nodes are randomly distributed throughout a network. A random distribution of destination nodes is consistent with many aspects of parallel computation, including support for barrier synchronization, implementation of distributed shared-memory, and in cases where processing nodes are allocated randomly to jobs.

For this study, we assume that whenever two unicast messages in the same step use virtual channels that are multiplexed onto the same physical link, one of them will be blocked. If a unicast message is blocked, then it requires time equivalent to at least two message-passing steps, rather than one. We also assume that whenever a unicast message delivered to destination node u is blocked, or delayed by one step, all the unicast messages delivered by the nodes in the reachable sets of node u will be delayed by one step. Note that a delayed message may result in new physical link contention with the messages produced in the next step.

For a CCC network, we assume that virtual channels in the same direction between a particular pair of neighboring nodes are multiplexed onto a single unidirectional physical link. For example, in Fig.3, nodes 0 and 1 are connected by two unidirectional physical links: one link supports virtual channels $c_{w_0h_0}$ and $c_{w_0h_1}$, the other link supports virtual channels $c_{w_0l_0}$ and $c_{w_0l_1}$.

Given the above assumption, Fig.8(a) plots the average physical link contention in a U-CCC multicast operation, while Fig.8(b) shows the average number of message-passing steps required to complete a U-CCC multicast operation. Each point in these plots was produced by averaging over a large number of uniformly distributed destination sets. Both 896-node 7D and 2048-node 8D CCCs are considered. Multicast set sizes from 8 through 64

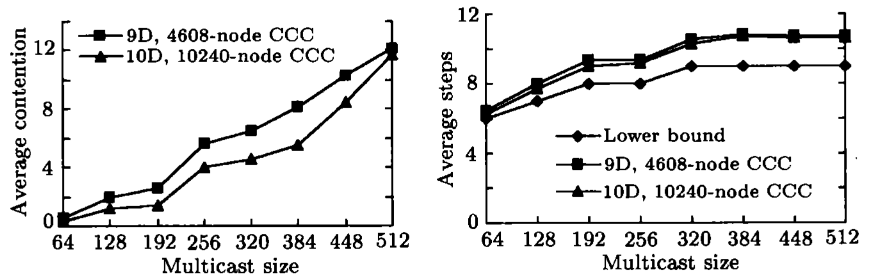


(a) Average link contention. (b) Average steps.

Fig.8. Link contention and message-passing steps for U-CCC multicast operation (896- and 2048-node CCC).

nodes were examined. The plotted lower bound in Fig.8(b) is calculated as the number of steps required when physical link sharing is not considered.

We also examined the effects of virtual channel multiplexing on larger CCC networks. Figs.9(a) and 9(b) plot the average physical link contention and the average number of message-passing steps for 4608-node 9D and 10240-node 10D CCC networks. For these larger configurations, multicast set sizes from 64 through 512 nodes were examined.



(a) Average link contention. (b) Average steps.

Fig.9. Link contention and message-passing steps for U-CCC multicast operation (4608 and 10240-node CCC).

As illustrated in Figs.8(b) and 9(b), the effect of virtual channel multiplexing on the average number of steps is small. In all cases, the number of steps is close to the theoretical lower bound.

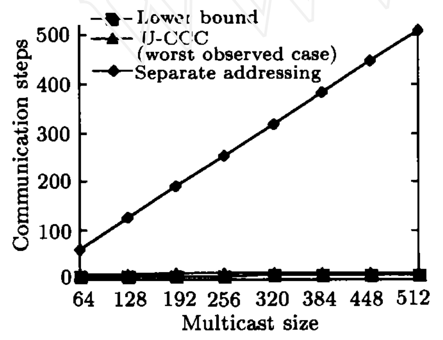


Fig.10. Communication steps for U-CCC and separate addressing.

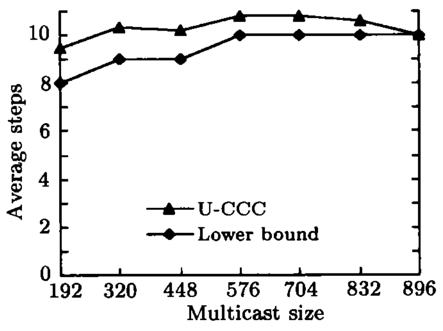


Fig.11. Communication steps for U-CCC (896-node CCC).

Fig.10 compares the worst-case observed performance of U-CCC with the number of steps required in *separate addressing*, in which the source directly sends a copy of message to every destination. The worst observed case occurred with a 4608-node 9D CCC network. As Fig.10 demonstrates, the difference between the performances of the U-CCC algorithm, with and without the practical consideration of link sharing, is extremely small compared to the performance of separate addressing.

As shown in Fig.11, if multicast set size is great enough, the performance of the U-CCC algorithm approaches the theoretical lower bound as the multicast set size increases. When the destination set contains every node in the network, the U-CCC algorithm completes a *broadcast* operation, and the number of steps is equivalent to the theoretical lower bound. Fig.11 is observed with an 896-node 7D CCC network.

7 Conclusions

This paper has presented an efficient algorithm for multicast communication on one-port, wormhole-routed cube-connected cycles. The algorithm produces multicast trees in which the constituent unicast messages do not contend for the same channels in spite of the use of deterministic routing. Moreover, the number of message-passing steps required to multicast data to $m-1$ destinations is $\lceil \log_2 m \rceil$, which is optimal for one-port architectures.

References

[1] Preparata F P, Vuillemin J. The cube-connected cycles: A versatile network for parallel computation. *Comm. ACM,* May 1981, 24(5): 300–309.

[2] Tzeng Nian-Feng. A cube-connected cycles architecture with high reliability and improved performance. *IEEE Trans. Computers,* Feb. 1993, 42(2): 246–253.

[3] Bruck J, Cypher R, Ho C-T. On the construction of fault-tolerant cube-connected cycles networks. *J. Parallel and Distributed Computing,* Feb. 1995, 25(1): 98–106.

[4] Klasing R. Improved compressions of cube-connected cycles networks. *IEEE Trans. Parallel and Distributed Systems,* Aug. 1998, 9(8): 803–812.

[5] Meliksetian D S, Chen Roger C Y. Optimal routing algorithm and the diameter of the cube-connected cycles. *IEEE Trans. Parallel and Distributed Systems,* Oct. 1993, 4(10): 1172–1178.

[6] Lo Hao-yung, Chen Jian-da. A routing algorithm and generalization for cube-connected cycle networks. *IEICE Trans. Information and Systems,* Sept. 1997, E80-D(9): 829–836.

[7] McKinley P K, Xu H, Kalns E, Ni L M. ComPaSS: Efficient communication services for scalable architectures. In *Proc. Supercomputing'92,* Nov. 1992, pp.478–487.

[8] Choi J, Dongarra J J, Pozo R, Walker D W. ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers. In *Proc. Fourth Symp. Frontiers of Massively Parallel Computation,* IEEE CS Press, 1992, pp.120–127.

[9] Dongarra J, van de Geijn R A. Reduction to condensed form for the eigenvalue problem on distributed memory architectures. *Parallel Computing,* 1992, 18: 973–982.

[10] McKinley P K, Xu H, Esfahanian A-H, Ni L M. Unicast-based multicast communication in wormhole-routed networks. *IEEE Trans. Parallel and Distributed Systems,* Dec. 1994, 5(12): 1252–1265.

[11] Robinson D F, McKinley P K, Cheng B H C. Optimal multicast communication in wormhole-routed torus networks. *IEEE Trans. Parallel and Distributed Systems,* Oct. 1995, 6(10): 1029–1042.

[12] Xu Hong, Gui Yadong, Lionel M Ni. Optimal software multicast in wormhole-routed multistage networks. *IEEE Trans. Parallel and Distributed Systems,* Jun. 1997, 8(6): 597–607.

[13] Lee I Y-Y, Wang S-D. Ring-connected networks and their relationship to cubical ring connected cycles and dynamic redundancy networks. *IEEE Trans. Parallel and Distributed Systems,* Sept. 1995, 6(9): 988–996.

[14] Dally W J, Seitz C L. The torus routing chip. *J. Distributed Computing,* 1986, 1(3): 187–196.

[15] Ni L M, McKinley P K. A survey of wormhole routing techniques in direct networks. *IEEE Computer,* Feb. 1993, 26: 62–76.

[16] Dally W J, Seitz C L. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.,* May 1987, C-36(5): 547–553.

[17] Jose Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel and Distributed Systems,* Oct. 1995, 6(10): 1055–1067.

[18] Libeskind-Hadas. A tight lower bound on the number of channels required for deadlock-free wormhole routing. *IEEE Trans. Computers,* Oct. 1998, 47(10): 1158–1161.

[19] Dally W J. Virtual channel flow control. *IEEE Trans. Parallel and Distributed Systems,* Mar. 1992, 3(2): 194–205.

SONG Jianping received the B.S. degree in computer science from the University of Science and Technology of China, Hefei in 1997. He is currently pursuing his Ph.D. degree in computer science at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His current research interests include interconnection networks, digital signal processing, and parallel and distributed computing.

HOU Zifeng is a professor and a Ph.D. supervisor at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His current research interests include high-performance system architecture and digital signal processing.

SHI Yuntao is currently pursuing her Ph.D. degree in computer science at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. Her current research interests include interconnection networks and digital signal processing.