
```

1.  procedure ONE_TO_ALL_BC( $d, my\_id, X$ )
2.  begin
3.       $mask := 2^d - 1;$            /* Set all  $d$  bits of  $mask$  to 1 */
4.      for  $i := d - 1$  downto 0 do   /* Outer loop */
5.           $mask := mask \text{ XOR } 2^i;$  /* Set bit  $i$  of  $mask$  to 0 */
6.          if  $(my\_id \text{ AND } mask) = 0$  then /* If lower  $i$  bits of  $my\_id$  are 0 */
7.              if  $(my\_id \text{ AND } 2^i) = 0$  then
8.                   $msg\_destination := my\_id \text{ XOR } 2^i;$ 
9.                  send  $X$  to  $msg\_destination;$ 
10.             else
11.                  $msg\_source := my\_id \text{ XOR } 2^i;$ 
12.                 receive  $X$  from  $msg\_source;$ 
13.             endelse;
14.          endif;
15.      endfor;
16.  end ONE_TO_ALL_BC

```

Algorithm 4.1 One-to-all broadcast of a message X from node 0 of a d -dimensional p -node hypercube ($d = \log p$). AND and XOR are bitwise logical-and and exclusive-or operations, respectively.

```

1. procedure GENERAL_ONE_TO_ALL_BC( $d, my\_id, source, X$ )
2. begin
3.    $my\_virtual\_id := my\_id \text{ XOR } source$ ;
4.    $mask := 2^d - 1$ ;
5.   for  $i := d - 1$  downto 0 do   /* Outer loop */
6.      $mask := mask \text{ XOR } 2^i$ ; /* Set bit  $i$  of  $mask$  to 0 */
7.     if ( $my\_virtual\_id \text{ AND } mask$ ) = 0 then
8.       if ( $my\_virtual\_id \text{ AND } 2^i$ ) = 0 then
9.          $virtual\_dest := my\_virtual\_id \text{ XOR } 2^i$ ;
10.        send  $X$  to ( $virtual\_dest \text{ XOR } source$ );
        /* Convert  $virtual\_dest$  to the label of the physical destination */
11.       else
12.          $virtual\_source := my\_virtual\_id \text{ XOR } 2^i$ ;
13.         receive  $X$  from ( $virtual\_source \text{ XOR } source$ );
        /* Convert  $virtual\_source$  to the label of the physical source */
14.       endelse;
15.     endfor;
16. end GENERAL_ONE_TO_ALL_BC

```

Algorithm 4.2 One-to-all broadcast of a message X initiated by $source$ on a d -dimensional hypothetical hypercube. The AND and XOR operations are bitwise logical operations.

```

1. procedure ALL_TO_ONE_REDUCE( $d, my\_id, m, X, sum$ )
2. begin
3.   for  $j := 0$  to  $m - 1$  do  $sum[j] := X[j]$ ;
4.    $mask := 0$ ;
5.   for  $i := 0$  to  $d - 1$  do
6.     /* Select nodes whose lower  $i$  bits are 0 */
7.     if  $(my\_id \text{ AND } mask) = 0$  then
8.       if  $(my\_id \text{ AND } 2^i) \neq 0$  then
9.          $msg\_destination := my\_id \text{ XOR } 2^i$ ;
10.        send  $sum$  to  $msg\_destination$ ;
11.       else
12.          $msg\_source := my\_id \text{ XOR } 2^i$ ;
13.         receive  $X$  from  $msg\_source$ ;
14.         for  $j := 0$  to  $m - 1$  do
15.            $sum[j] := sum[j] + X[j]$ ;
16.         endelse;
17.          $mask := mask \text{ XOR } 2^i$ ; /* Set bit  $i$  of  $mask$  to 1 */
18.       endifor;
19.   endfor ALL_TO_ONE_REDUCE

```

Algorithm 4.3 Single-node accumulation on a d -dimensional hypercube. Each node contributes a message X containing m words, and node 0 is the destination of the sum. The AND and XOR operations are bitwise logical operations.

```
1. procedure ALL_TO_ALL_BC_RING(my_id, my_msg, p, result)
2. begin
3.   left := (my_id - 1) mod p;
4.   right := (my_id + 1) mod p;
5.   result := my_msg;
6.   msg := result;
7.   for i := 1 to p - 1 do
8.     send msg to right;
9.     receive msg from left;
10.    result := result ∪ msg;
11.  endfor;
12. end ALL_TO_ALL_BC_RING
```

Algorithm 4.4 All-to-all broadcast on a p -node ring.

```
1. procedure ALL_TO_ALL_RED_RING(my_id, my_msg, p, result)
2. begin
3.   left := (my_id - 1) mod p;
4.   right := (my_id + 1) mod p;
5.   recv := 0;
6.   for i := 1 to p - 1 do
7.     j := (my_id + i) mod p;
8.     temp := msg[j] + recv;
9.     send temp to left;
10.    receive recv from right;
11.  endfor;
12.  result := msg[my_id] + recv;
13. end ALL_TO_ALL_RED_RING
```

Algorithm 4.5 All-to-all reduction on a p -node ring.

```

1. procedure ALL_TO_ALL_BC_MESH(my_id, my_msg, p, result)
2. begin

   /* Communication along rows */
3.   left := my_id - (my_id mod  $\sqrt{p}$ ) + (my_id - 1) mod  $\sqrt{p}$ ;
4.   right := my_id - (my_id mod  $\sqrt{p}$ ) + (my_id + 1) mod  $\sqrt{p}$ ;
5.   result := my_msg;
6.   msg := result;
7.   for i := 1 to  $\sqrt{p} - 1$  do
8.     send msg to right;
9.     receive msg from left;
10.    result := result  $\cup$  msg;
11.  endfor;

   /* Communication along columns */
12.  up := (my_id -  $\sqrt{p}$ ) mod p;
13.  down := (my_id +  $\sqrt{p}$ ) mod p;
14.  msg := result;
15.  for i := 1 to  $\sqrt{p} - 1$  do
16.    send msg to down;
17.    receive msg from up;
18.    result := result  $\cup$  msg;
19.  endfor;
20. end ALL_TO_ALL_BC_MESH

```

Algorithm 4.6 All-to-all broadcast on a square mesh of p nodes.

```
1. procedure ALL_TO_ALL_BC_HCUBE(my_id, my_msg, d, result)
2. begin
3.   result := my_msg;
4.   for i := 0 to d - 1 do
5.     partner := my_id XOR  $2^i$ ;
6.     send result to partner;
7.     receive msg from partner;
8.     result := result  $\cup$  msg;
9.   endfor;
10. end ALL_TO_ALL_BC_HCUBE
```

Algorithm 4.7 All-to-all broadcast on a d -dimensional hypercube.

```

1. procedure ALL_TO_ALL_RED_HCUBE(my_id, msg, d, result)
2. begin
3.   recloc := 0;
4.   for i := d - 1 to 0 do
5.     partner := my_id XOR  $2^i$ ;
6.     j := my_id AND  $2^i$ ;
7.     k := (my_id XOR  $2^i$ ) AND  $2^i$ ;
8.     senloc := recloc + k;
9.     recloc := recloc + j;
10.    send msg[senloc .. senloc +  $2^i$  - 1] to partner;
11.    receive temp[0 ..  $2^i$  - 1] from partner;
12.    for j := 0 to  $2^i$  - 1 do
13.      msg[recloc + j] := msg[recloc + j] + temp[j];
14.    endfor;
15.  endfor;
16.  result := msg[my_id];
17. end ALL_TO_ALL_RED_HCUBE

```

Algorithm 4.8 All-to-all broadcast on a d -dimensional hypercube. AND and XOR are bitwise logical-and and exclusive-or operations, respectively.

```
1. procedure PREFIX_SUMS_HCUBE(my_id, my_number, d, result)
2. begin
3.   result := my_number;
4.   msg := result;
5.   for i := 0 to d - 1 do
6.     partner := my_id XOR  $2^i$ ;
7.     send msg to partner;
8.     receive number from partner;
9.     msg := msg + number;
10.    if (partner < my_id) then result := result + number;
11.  endfor;
12. end PREFIX_SUMS_HCUBE
```

Algorithm 4.9 Prefix sums on a d -dimensional hypercube.

```
1. procedure ALL_TO_ALL_PERSONAL( $d, my\_id$ )
2. begin
3.   for  $i := 1$  to  $2^d - 1$  do
4.     begin
5.        $partner := my\_id \text{ XOR } i$ ;
6.       send  $M_{my\_id, partner}$  to  $partner$ ;
7.       receive  $M_{partner, my\_id}$  from  $partner$ ;
8.     endfor;
9. end ALL_TO_ALL_PERSONAL
```

Algorithm 4.10 A procedure to perform all-to-all personalized communication on a d -dimensional hypercube. The message $M_{i,j}$ initially resides on node i and is destined for node j .