A. V. Gerbessiotis
CIS 485/Spring 1999
Handout 9
Broadcasting

Broadcasting

Problem: p processors numbered $0 \ldots p-1$ are given. One of them, say processor 0 holds a message of length equal to one word. The problem of *broadcasting* involves the dissemination of this message to the remaining $p-1$ processors.
Algorithm 1. In one round of communication, processor 0 sends the message to be broadcast to processors $1, \ldots p-1$ in turn (a "sequential"-looking algorithm).
Algorithm 2. The message is broadcast in $\lg p$ rounds of communication (we assume p is a power of two) by binary replication (this is the PRAM algorithm presented in class). The number of processors holding the message in round i is twice as that of round $i-1$. In round $i = 1, \ldots, \lg p$, each processor j with index $j < 2^{i-1}$ sends the message it currently holds to processor $j + 2^{i-1}$. The number of processors holding the message at the end of round i is 2^i.

Communication Performance

We are going to express the performance of the two algorithms in terms of the BSP parameters L and g. A communication round would correspon to a superstep of the BSP model. The cost of a communication round would then be $\max\{L, mg\}$, if an m-relation is realized during the superstep, ie a processor sends/receives total information of size m words.

Algorithm 1.

The communication time of Algorithm 1 is $1 \cdot \max\{L, (p-1) \cdot g\}$ (in a single superstep the message is replicated $p-1$ times by processor 0).

Algorithm 2.

The communication time of Algorithm 2 is $\lg p \cdot \max\{L, 1 \cdot g\}$ (in each superstep each processor sends one message and one processor receives exactly one message). Superstep utilization may be low if $g \ll L$.

Algorithm 3

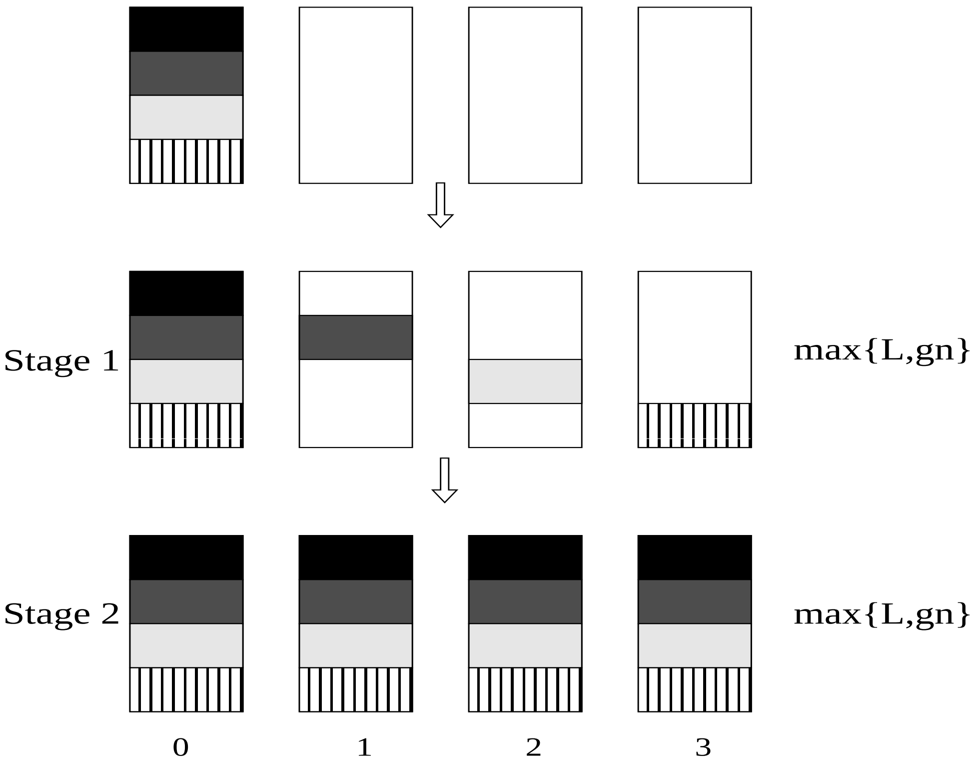
Both Algorithm 1 and Algorithm 2 can be viewed as extreme cases of an Algorithm 3. The main observation is that up to L/g words can be sent in a superstep at a cost of L. It makes sense then for each processor to send so many messages to other processors, say, $k-1$. At the end of each superstep the number of processors possessing the message is k times more than that of the previous superstep. During each superstep each processor sends the message to exactly $k-1$ other processors. Algorithm 3 consists of a number of supersteps between 1 and $\lg p$. For $k = p$, Algorithm 1 is derived from Algorithm 3; for $k = 2$, Algorithm 2.

Let k be the replication parameter. The broadcasting algorithm consists of $\lg p / \lg k$ supersteps (we assume p is a power of k so that the number of supersteps is an integer). In superstep $i = 0, 1, \ldots$, every processor j with index $j < k^i$ sends the message to $k-1$ distinct processors numbered $j + k^i \cdot l$, where $l = 1, \ldots k-1$. By induction, at the end of superstep i (the $(i+1)$-st superstep), the message is broadcast to $k^i \cdot (k-1) + k^i = k^{i+1}$ processors. The number of supersteps required is the minimum integer r such that $k^r \geq p$, i.e. $r = \lg p / \lg k$ (or $\lceil \lg p / \lg k \rceil$, if $\lg p / \lg k$ is not an integer).

```
    BROADCAST (0, p,k)
my_pid = bsp_pid();    mask_pid = 1;
while (mask_pid < p) {
    if (my_pid < mask_pid)
        for (i = 1, j = mask_pid; i < k; i++, j+= mask_pid) {
            target_pid = my_pid + j;
            if (target_pid < p)
                bsp_put(target_pid, &M, &M, 0, sizeof(M));
        }
    bsp_sync();
    mask_pid = mask_pid * k;
```

Two-phase broadcasting

Finally, suppose that processor 0 broadcasts a message of $n > p$ words (a new parameter, problem size is introduced; previously it was the case that $n = 1$). It is clear that applying Algorithms 1, 2 or 3 to solve this problem provides an inefficient solution as one of the processors sends or receives substantially more than n words of information. There is a broadcasting algorithm, call it Algorithm 4, that requires only two communication rounds/supersteps and is optimal (for the communication model abstracted by L and g) in terms of the amount of information (up to a constant) each processor sends or receives. The idea is to split the message into p pieces, have processor 0 send piece i to processor i in the first superstep and in the second superstep processor i replicates the i-th piece $p-1$ times by sending each copy to each of the remaining $p-1$ processors (see attached figure).



Exercise. What can you say about parallel prefix? Analyze the BSP performance of the PRAM algorithm for parallel prefix. Can you halve its number of supersteps yet maintain the same BSP cost?