Term Project Complexity Analysis
6.852

Jeremy Brown and Greg Hudson

In the absence of failures, the previously described flooding algorithm achieves a time complexity of $O(diam(l_h + d_h))$ to propagate a single article initiated by a *post* action at some process i to every server. The proof of this is by induction on the distance of a server from the originating server i; if all servers at a distance k from server i have received the article in time $O(k(l_h + d_h))$, then within a constant number of local processing and message deliveries (a uid, *accept*, the article, and an *ack*; a *reject* by a server at distance $k+1$ implies that it already has the article), every server at distance $k+1$ from server i will have received the article.

The remainder of this section will be concerned with message complexity. If there are n servers arranged in a graph with E edges, the previously described algorithm requires $\Theta(E(b_u + b_a))$ message bits to propagate a single article to all servers, in the absence of failures. To see the upper bound, simply note that each edge sees a uid and either a *reject*or an *accept*and article retransmission. To see the lower bound, consider the following execution occurring in a densely connected graph of n processes:

$post_1(a)$
[Let $u = (1, 0)$, the uid for a]
$migrate_{1,j}(u)$ for $2 \leq j \leq n$
$send_{1,j}(u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor$
$receive_{1,j}(u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor$
$send_{j,1}(accept, u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor$
$receivej, 1(accept, u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor$
$send_{1,j}(u, a)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor$
$receive_{1,j}(u, a)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor$
$migrate_{j,k}(u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor < k \leq n$
$send_{j,k}(u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor < k \leq n$
$receive_{j,k}(u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor < k \leq n$
$send_{k,j}(accept, u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor < k \leq n$
$receive_{k,j}(accept, u)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor < k \leq n$
$send_{j,k}(u, a)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor < k \leq n$
$receive_{j,k}(u, a)$ for $2 \leq j \leq \lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor < k \leq n$
[...]

We would like to achieve a message complexity of $O(Eb_u + nb_a)$; that is, we would like to send unique identifiers for the article across every edge, but send the article itself only once to each process. Following are several candidate approaches to the problem:

1 Restricting the Input

In the real Usenet, worst-case scenarious such as the one above don't usually happen because servers are usually busy processing different articles at different times. One might hope for some way of describing a pattern of *post*inputs such that the amortized message bits of propagating each article is $O(Eb_u + nb_a)$. But this approach is a dead end, since for any pattern of *post*inputs, there is always an execution where each article is fully propagated before the next one is posted, giving $O(E(B_u + b_a))$ message bits for each article.

2 Restricting the Graph

One might hope to define some restriction on the structure of the graph such that a posted article is sent across fewer than a constant fraction of the edges. Unfortunately, for every graph there is always an execution which sends the article across all of the edges. To see this, imagine we are given some connected undirected graph of servers. Now pick an arbitrary server i on the graph, and let the execution begin with $post_i(a)$ for some article a. Let u be the UID for a (that is, $u = (i, 0)$). Now determine the distance of each server from i (the *level* of each server), and define S_j to be the set of servers at level j (for $0 \leq j \leq diam$). Our execution will proceed level by level. After the article has propagated to all servers in S_j, we propagate it to the servers in S_{j+1} as follows:

$T := \emptyset$
while $T \neq S_{j+1}$
    pick $s \in S_{j+1} - T$
    Let N be $\{n \in nbrs_s : a \in dbase_n\}$
    for each $n \in N$
        perform $migrate_n(u)$
        perform $send_{n,s}(u)$
        perform $receive_{n,s}(u)$
        perform $send_{s,n}(accept, u)$
        perform $receive_{s,n}(accept, u)$
    for each $n \in N$
        perform $send_{n,s}(u, a)$
        perform $receive_{n,s}(u, a)$
        perform $send_{n,s}(ack, u)$
        perform $receive_{n,s}(ack, u)$
    $T := T \cup \{s\}$

Now every server at level $j+1$ has received the article, and we go on to the next level. In this manner, we send the article across every edge. Therefore, in any connected graph, the worst case message complexity for propagating a single posted article is $\Theta(E(b_u+b_a))$ bits.

3 Modifying the Algorithm

If neither the input nor the graph can be restricted to show better message performance, then perhaps the algorithm can be modified to show up an upper bound on message complexity. The simplest modification we came up with is to weaken the algorithm's fault tolerance so that it tolerates most h halting failures; that is, the modified algorithm need not propagate the article to every server if more than h servers stop without ever starting again. (This is assuming, of course, that the graph is never partitioned by halting failures; in that case, no algorithm can propagate the article to every server.)

A simple way to take advantage of this weakened fault-tolerance requirement is to accept a given uid at most $h+1$ times; that is, in addition to *dbase*