An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes

Ernest Chang
University of Toronto

Rosemary Roberts
University of Waterloo

This note presents an improvement to LeLann's algorithm for finding the largest (or smallest) of a set of uniquely numbered processes arranged in a circle, in which no central controller exists and the number of processes is not known a priori. This decentralized algorithm uses a technique of selective message extinction in order to achieve an average number of message passes of order (n log n) rather than $O(n^2)$.

Key Words and Phrases: decentralized algorithms, distributed systems, operating systems
CR Categories: 4.32, 4.35, 5.25, 5.32

This work was made possible through the assistance of Health and Welfare, Canada.

Authors' present addresses: E. Chang, Computer Science Department, University of Waterloo, Waterloo, Ontario N2K 1N7, Canada; R. Roberts, Unity College, Unity, Maine 04988.

Introduction

Given a random circular arrangement of uniquely numbered processes where no a priori knowledge of the number of processes is known, and no central controller is assumed, we would like a method of designating by consensus a single unique process. The algorithm we propose works equally well for finding either the highest numbered or the lowest numbered process. Let us, without loss of generality, consider highest finding.

A situation in which this algorithm is important has been presented by LeLann [1]. In his example, a circle of controllers in which the control token is lost causes every controller to time out, and an election to find a new emitter for the control token is performed. LeLann's algorithm requires every controller to send a message bearing its number. Each controller thus collects, through the messages seen, the numbers of the other controllers in the circle. Every controller sorts its list, and the controller whose own number is the highest on its list is elected.

LeLann's algorithm, in a circle with n controllers, requires total messages passed proportional to n^2, written $O(n^2)$, where a message pass is a SEND of a message from a controller. This is clearly so, since each of the n controllers sends a message which is passed to all other nodes. Our algorithm requires, on the average, $O(n \log n)$ message passes.

The Algorithm

Each process is assumed to know its own number, and initially it generates a message with its own number, passing it to the left. A process receiving a message compares the number on the message with its own. If its own number is lower, the process passes the message (to its left). If its own number is higher, the process throws the message away, and if equal, it is the highest numbered process in the system.

Proposition: This algorithm detects one and only one highest number.

Argument: By the circular nature of the configuration and the consistent direction of messages, any message must meet all other processes before it comes back to its initiator. Only one message, that with the highest number, will not encounter a higher number on its way around. Thus, the only process getting its own message back is the one with the highest number.

Startup Conditions

It may not be the case that all processes are aware of the need to initiate a message before messages start arriving. Assume therefore that at least one process initiates a message. Then the rule is that each process initiating a message marks itself. A message arriving at

an unmarked process causes that process to mark itself and then generate a message according to the above algorithm. Otherwise, the algorithm performs as before. This minor modification ensures that all processes which would eventually be involved are indeed involved. The elected process must assume the responsibility of ensuring that every process unmarks itself so that subsequent elections can be held successfully.

Performance Analysis

The model we will use is a circle of processes numbered from 1 to n, with a clockwise movement of messages. We are interested in two measures—the time needed to find the highest and the number of message passes (as previously defined). Call the message initiated by process i, message i.

Time Behavior

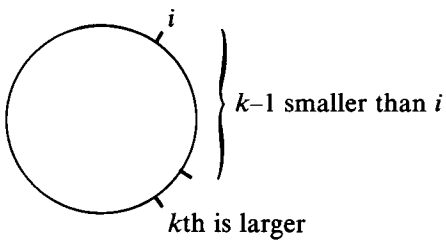
The algorithm succeeds when the highest number is found. If all processes start simultaneously, then since only one cycle through the ring is needed, the time required is $O(n)$, where n is the number of processes. If the highest numbered process starts first, then its message would take one cycle, and the time required is $O(n)$. However, if the process furthest away from the longest is the only one to initiate the election, then the time required would be $O(n-1)$ for a message to get to the largest process, and $O(n)$ for the largest to be elected. Thus, the time would be $O(2n-1)$. In all cases, nevertheless, the time behavior for election is linearly proportional to n.

Message Passes

(a) *Best Case.* Processes are ordered clockwise in increasing sequence so that each message (except message n) only goes once. There are $n-1$ of these, while message n requires n passes. Thus, the total number of message passes is $n + n - 1 = 2n - 1$.

(b) *Worst Case.* Processes are ordered clockwise in decreasing sequence so that message i must be passed i times. Thus, the total number of message passes is $\sum_{i=1}^{n} i = n(n+1)/2$.

(c) *Average Case.*



Let $P(i, k)$ be the probability that message i is passed k times, which is the probability that the $k-1$ clockwise neighbors of i are less than i and the kth clockwise neighbor of i is larger than i. There are $i-1$ processes less than i and $n-i$ processes larger than i.

Write $C(a, b)$ as the number of ways of choosing b things from a things. Then

$$P(i, k) = \frac{C(i-1, k-1)}{C(n-1, k-1)} \times \frac{n-i}{n-k}.$$

Knowing that the message n always takes n passes and there is only one such message, we therefore consider only $n-1$ messages, each making, at most, $n-1$ passes. Therefore, the expected number of passes for messages other than message n is

$$E_i(k) = \sum_{k=1}^{n-1} k\, P(i, k) \qquad i \neq n.$$

Therefore, the expected number of message passes, for all the messages, is

$$E(k) = n + \sum_{i=1}^{n-1} \sum_{k=1}^{n-1} kP(i, k).$$

This can be simplified to

$$E(k) = n + \sum_{k=1}^{n-1} \frac{n}{k+1}$$

$$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right).$$

The harmonic series has a partial sum of $C + \log_e n$, and therefore the average number of message passes is $O(n \log n)$.

Concluding Comments

Some simple variants of this algorithm are of interest. If the message originating at process i came to process j before j had emitted message j and $i > j$, then clearly j cannot be the largest node. Thus, it is unnecessary for j to emit its own message. In the best case, in which the highest process n is the only initiator, the number of required message passes improves from $2n-1$ to n.

We have assumed that all processes are to be involved in an election. A more "voluntary" situation is easily accommodated by modifying the start-up mechanism. If node i has not sent out message i by the time another message reaches it, then it simply does not participate in the current election. If it should subsequently wish to join in, it must wait till the next election.

Finally, consider briefly failure under the best of circumstances—a single node "vanishes" without taking any message or disrupting communications. If the failing node is not the highest node n, the election would not be affected. If it is the highest but message $(n-1)$ has not yet been extinguished, things would still be fine for node $(n-1)$ would be correctly elected. If node n fails after

message ($n - 1$) has been extinguished, however, then no node would be elected. Still, message n would keep circulating, and this condition is detectable with suitable modification to the algorithm.

In conclusion, the highest element of a set of n things can be found in $O(n)$ comparisons, but n must be known. Alternatively, if a single process can be designated a priori in a circle of processes, it can also find the highest numbered process in $O(n)$ comparisons. However, little study has been made of completely decentralized control, in which processes do not know how many other processes are involved, and a uniquely designated process does not exist. It is pleasing to know that even under these circumstances, decentralized algorithms are simple and efficient.

Received November 1977; revised October 1978

References

1. LeLann, G. Distributed systems–Towards a formal approach. Information Processing 77, North-Holland Pub. Co., Amsterdam, pp. 155–160.

Consumer Difficulties With Computerized Transactions: An Empirical Investigation

T.D. Sterling
Simon Fraser University,
British Columbia, Canada

The prevalence with which errors may be encountered by the *end targets* of a computerized process is assessed. How many and what type of errors occur? How easily are they corrected? What is the reaction of consumers to errors—to a failure to correct them? What can be learned by designers of large management packages from such data?

Results show that with the present state of the art, approximately 40 percent of individuals (or households) having *average* contacts with different types of accounts experience one or more errors per year. Eighty percent relate to billing. Attempts to correct errors often turned out to be difficult and not always successful.

There appears to be some conflict between computer-using organizations and their public. Also the role of poor management packages including poor software is indicated. While most management systems may be adequate, results of the survey raise concerns about the timeliness and the number of designs of very large linked program packages (as EFT for instance).

Key Words and Phrases: errors, systems errors, billing errors, management systems, consumers

CR Categories: 2.0, 2.1, 2.12, 2.2, 3.50, 3.52, 3.55, 4.19, 4.6

Author's address: T.D. Sterling, Dept. of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 156.