ASYNCHRONOUS SYSTEMS, CONSERVATION LAWS AND CONVERGENCE TO A STEADY STATE

E. W. Dijkstra

Rapporteurs: Dr. P. E. Lauer
Mr. P. Quarendon

In spite of the title, Professor Dijkstra discussed a class of machines comprising a large number of computational elements (mosquitoes) with a limited ability to inter-communicate. He described the implementation of two algorithms, a Fast Fourier Transform and a bitone sort, and commented on the new problems which arose in their development.

Professor Dijkstra pointed out that this would be a highly experimental lecture since he had been prevented from giving it before due to some patent difficulties. It was for this reason that he had given a title on asynchronous networks even though he would like to talk about synchronous networks at this seminar.

He went on to say that he would use this single one-hour period to describe a special purpose machine, or rather, a member of a whole class of machines about the design of which one could ponder. The most baffling aspect of the exercise of designing this machine was the high degree to which the usual notational techniques and conceptual abilities proved absolutely insufficient to cope with the problems that arose. The speaker motivated his interest in this particular machine and his choice of this machine as the basis of his talk by expressing the hope that this might make it possible to convey to the listeners a hint of a new class of unsuspected difficulties and problems which the design of similar machines might present in the future.

One way of describing what he had been about was to say that he was designing an elephant out of a multitude of mosquitoes, where one might visualize a mosquito as a little animal with some active possibilities. Thus one would have a whole network of these nasty little beasts and together they form an elephant. One would like them to turn in harmony and together they should form a powerful piece of equipment. The metaphor of mosquitoes was chosen in analogy to very little chips. In the case of the design and use of chips the number of pins, the number of ways in and out are limited by lack of space. Hence, one can imagine something like four-legged mosquitoes each with two input legs and two output legs. In a network each output leg of a mosquito would be connected with an input leg of another mosquito and these connections are used for one-way traffic only.

These mosquitoes are to be conceived as having certain data processing abilities and they have a very restricted ability of passing information, namely, only to those of their neighbours to which they are immediately connected. Furthermore, Professor Dijkstra suggested that one might think of them as being equipped with one or two antennae, one for reset and one for synchronization or something of that sort.

This sort of conceptualization brings one to the sort of problem referred to above. Supposing one has some sort of computational dependence on intermediate results as is known to us from the triangle of Pascal. There one has a set of intermediate results and there is a next set of intermediate results, which are staggered, and each one in the latter set depends on two previous ones. If one knows the upper line one can move down.

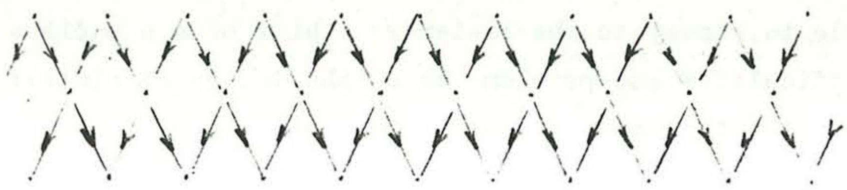

Figure 1

To see the point better, the speaker suggested one should however consider a Pascal's cylinder instead. So one may think of a cylinder with the intermediate results of one moment placed around the cylinder in a circle, and at the next time they are staggered, and then again they are in place. The downward arrows in the circular crescent sense arrangement convey the idea of dependence. Then one can think of the computation

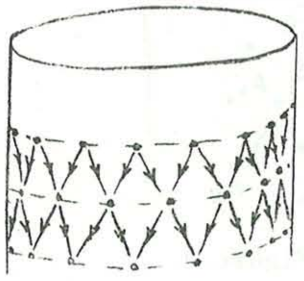

Figure 2

as performed by a circular elephant with mosquitoes only connected to their neighbours. Conceiving the time axis as inside the cylinder running from top to bottom one can visualize the progress of the computation as a downward movement of the ring of intermediate results plotted on the surface of the cylinder. At one moment in time the ring will be in a certain position and that will be reflected by the current intermediate results being stored in each mosquito.

Now one is in a position to see the problem. If one visualizes the progress of the computation as the movement of the ring of mosquitoes along the cylinder, one can think of it as moving screw-wise in either direction or even in zig-zag.

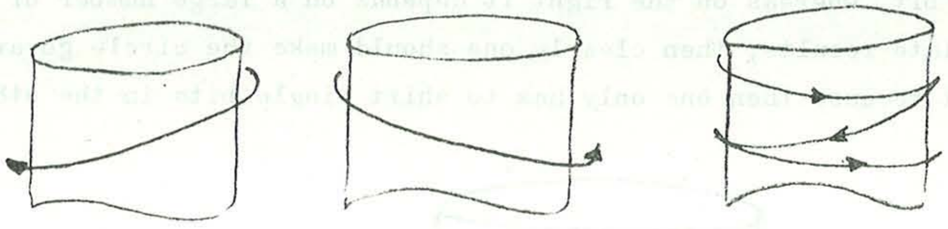

Figure 3

Supposing one has a mosquito A at time zero, when the ring is in the topmost position, and it stores the initial value. Then at time one, A will store the next intermediate result but it could be either at place A_1 or A_2. And even if A_1 were chosen then at the next step one can still choose A_3 or A_4.

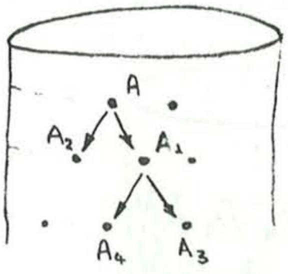

Figure 4

If one has a machine consisting of many mosquitoes and each mosquito has information in it, the information present in different mosquitoes has to be combined, has to be present in a single mosquito, so that it can be subjected to some sort of operation. If the mosquitoes concerned are not connected one has the problem of getting the information together. But even if they are connected, one still has the choice of shipping the information in either direction and therefore performing the next function at different places. This gives an additional degree of freedom which in the course of the design of an elephant is extremely confusing. It would be very nice if one could abstract from it, were it not for the fact that the total efficiency of the mosquito, for instance traffic density, can greatly depend on such choice.

If, for instance, this dependence is such that on the left it depends on a single bit, whereas on the right it depends on a large number of bits of intermediate results, then clearly one should make the circle go around to the right because then one only has to shift single bits in the other direction.

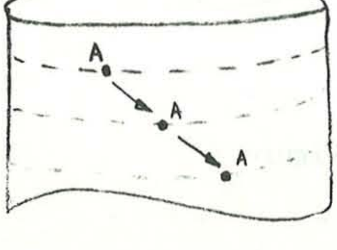

Figure 5

It is such difficulties one has to be able to cope with. The speaker added that in trying to design elephants that would be very powerful they had of course also focussed on computational processes where, at least conceptually, they could visualize a great amount of activity going on simultaneously.

As his first example, Professor Dijkstra chose the fast Fourier Transform.

Given the numbers a[i], i = 0, 1 ... 1023, say, the 1024 complex Fourier coefficients b[j] depend in the following way on the a[i]'s:

$$b[j] = \sum_{i=0}^{M-1} a[i]*d^{i*j} \qquad 0 \leq i,\ j < M$$

with $M = 2^N$, $d = e^{-2\pi i}$

Supposing M to be 1024, and N to be 10, then d is the 1024th complex root of 1.

This formula apparently requires 1024 series of 1024 terms to be summed. However, it is not as bad as that because if one introduces:

$$M1 = M/2 \qquad \text{and} \qquad d1 = d^2$$

it can be shown that

$$b[2j] = \sum_{i=0}^{M1-1} a'[i]*d1^{i*j}$$

$$b[2j+1] = \sum_{i=0}^{M1-1} a''[i]*d1^{i*j}$$

with $a'[i] = a[i] + a[i+M1]$

$a''[i] = (a[i] - a[i+M1])*d^i$

If the even coefficients are renamed b'[j], then these can be computed as the Fourier Coefficients of a series of values of half the original length. The same thing can be done for the odd coefficients.

The speaker illustrated the normal technique by a diagram showing the data dependencies of the new a's.

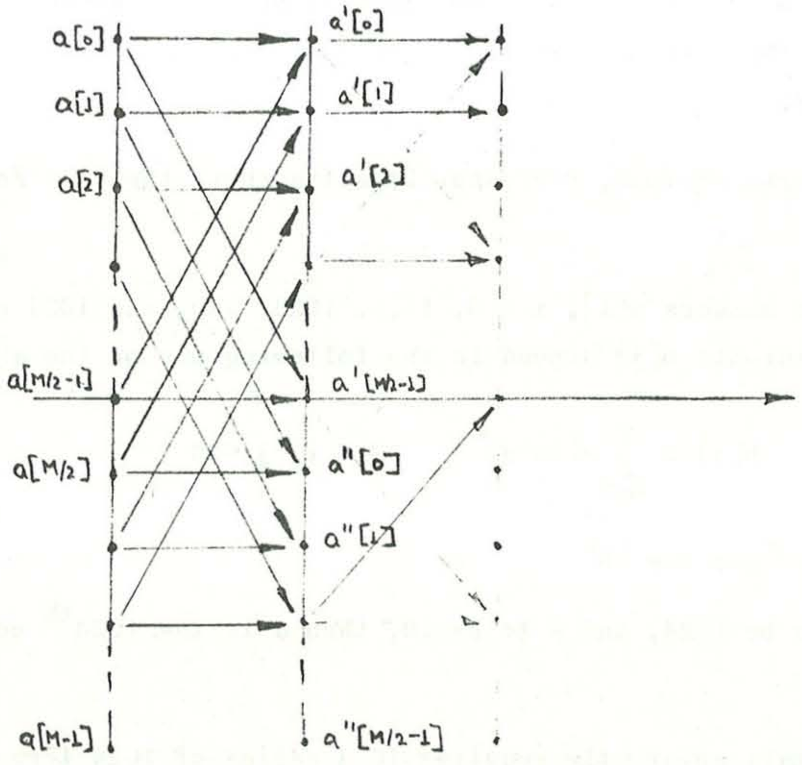

Figure 6

For example, a'[0] and a''[0] depend only on a[0] and a[M/2]. In each case, values are used M1 positions apart, and each application of the formula for $0 \le i < M1$ absorbs two values, and produces two values.

The result is that the original array is over-written by two new arrays each of half the length. Now the problem is reduced to computing the Fourier coefficients of these shorter arrays, and the game can be repeated. After this is repeated N times the array length is reduced to one, and the M Fourier coefficients are tabulated. This is a tremendous improvement, because if the original formula is applied, the amount of work to be done is M^2, whereas for a sequential machine like this it is M*N.

To arrange a set of mosquitoes to perform the task in N steps is quite different. A set of 1024 can be placed down the first column in Figure 6, each with two input legs and two output legs. Quite a complex wiring pattern arises. At the next stage, another 1024 mosquitoes are needed and the wiring pattern is different, and so on for all ten stages. The speaker suggested that this machine was not very attractive, and asked whether it was conceivable that a set of 1024 mosquitoes with two input legs and two output legs can be wired so as to perform the computation in 10 steps.

From the original definition of the Fourier coefficients, each of the b's depends on all the a's, or each a must broadcast in ten steps to all the mosquitoes which are going to contain the final answer. Put another way, given 1024 mosquitoes, is there a connection pattern so that there is a ten-step path from any node to any other? The speaker asserted that there was, and went on to demonstrate it in the following way:

Suppose one wants to go from node i to node j. If i and j are identified with a ten-bit numbers, these are conceptually concatenated, and a ten-bit window is placed over the first part of the resultant twenty-bit number. The number viewed through the window represents the current node. Starting at node i, the window is moved left one position at a time, so that in ten steps node j will be reached. This gives the connection pattern as follows: Two consecutive positions of the window

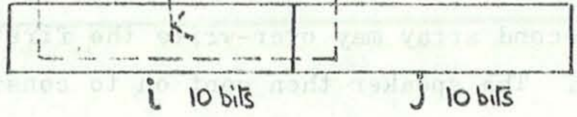

Figure 7

overlap in nine positions, so that there is a common number k (0≤1≤511), where the old position is k or k+512, and the new position is 2k or 2k+1.

$$\left.\begin{array}{l} k \\ k+512 \end{array}\right\} \rightarrow \left\{\begin{array}{l} 2k \\ 2k+1. \end{array}\right.$$

As there are four possibilities, four wires for node-node paths are needed, and applying this for the 512 values of k gives the required wiring pattern, with the possibility of passing from any node to any other in ten steps.

Professor Dijkstra then referred back to the Fourier analysis problem, and illustrated how this was effected. The result of the connection pattern is to stagger the resulting coefficients after the first stage in such a way that they are correctly placed for the same wiring pattern to be applied in the second stage.

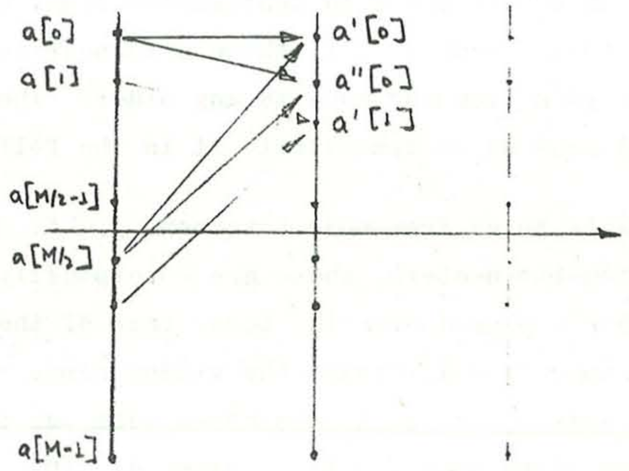

Figure 8

This means that the second array may over-write the first, and only 1024 mosquitoes are needed. The speaker then went on to consider relaxing the requirement for complete synchronism. He said that he had written a program comprising 1024 sequential processes, representing the mosquitoes, synchronizing themselves only with the mosquitoes to which they were directly connected. He said that what happens in this situation defies imagination, since one mosquito may have only taken one step, whereas at the other end the answers have already been generated. He compared the computation with a multi-dimensional bicycle chain wiggling through its progress. However he claimed that this was one technique for proving the correctness of such systems, that is, one can regard a normal sequential algorithm working in a single store, and then to visualize the process laid out in time. Each moment in time is called a stage. Realizing that all the variables are stored in different mosquitoes one can relax the requirement that a stage is realised all over the asynchronous network simultaneously. It turns out that studying the history of a single mosquito, this can be layed out in such a way that different snapshots can be regarded as successive stages in a sequential computation. This means that some of the known mathematical techniques for proving things about programs then become applicable.

The use of the specific wiring pattern described is not exhausted by the example of the hyper-fast Fourier Transform. It can also be used for sorting. This, while not so nice as the previous example, enables a sort to be carried out in N^2 steps. The sorting process appealed to is the 'bitone sort'.

A bitone sequence is a sequence with one maximum and one minimum at most. Strictly, one should describe it as a bitone cycle, since it is an undirected object. It is a cyclic arrangement of objects with one maximum and one minimum.

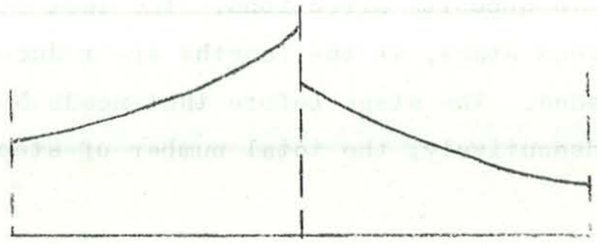

Figure 9

A bitone cycle of length 2k is easily split into two bitone cycles of length k, with the property that the maximum element of the left (right) hand cycle is at most equal to the minimum element of the right (left) hand cycle. The procedure is to inspect the first element and the one half-way, and place the smaller on the left. This is repeated for all k pairs of elements k positions apart. Having generated a pair of bitone sequences in this way, the same process is applied to these, where each application the largest value of one is at most the smallest value of the next. Repeating this until each sequence has been reduced to one element, produces a sorted array.

This procedure is very convenient for this kind of equipment, because element k and element 512+k can fire their values at two receivers. The receivers compare these values, and one retains the smallest, the other the largest. The operation of comparing all values 512 apart can take just one step. In the next step, the lengths of the sequences have been reduced by a factor of two, so values have to be combined which are now 256 apart. However, because the values are now not in their original positions, but at 2k and 2k+1, due to the wiring pattern, the values to be compared are still physically 512 apart.

Generating the sorted array from a bitone sequence then takes N steps. The remaining problem is to get the bitone sequence from the randomly ordered input. This was stated to be quite simple, because if the left hand half of the sequence is ordered, say, upward, and the other half downward, this is a bitone sequence. To create the sequence, then, the two halves are sorted in opposite directions. The last stage will need N steps. In the previous stage, as the lengths are reduced by a factor of two, N-1 steps are needed. The stage before that needs N-2 and so on. If one could do it consecutively, the total number of steps would be about $\frac{1}{2}N^2$, instead of N^2.

This, Professor Dijkstra said, shows some of the difficulties which arise, because one has no way of knowing that the dummy movements, which account for the other $\frac{1}{2}N^2$ steps are really necessary.

Professor Dijkstra concluded by asking whether making elephants out of mosquitoes was realistic in the sense that one could hope to find a methodology for the design of elephants. He ventured his personal opinion that it was, giving an argument to support it.

In a sequential program, consisting of statements, each completed statement transmits information to its dynamic successor. In the forms:

$$S_0 ;\ \underline{\text{If}}\ B\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2\ \underline{\text{fi}}\ ;\ S_3$$

$$S_0 ;\ \underline{\text{while}}\ B\ \underline{\text{do}}\ S_1\ \underline{\text{od}}\ ;\ S_2$$

each statement may follow, and be followed by only two others; so there is a way of looking at sequential programs in which each statement has at most two successors and two predecessors. Perhaps, he said, this indicates that there is some hope of being able to design elephants from many mosquitoes.

Discussion

Dr. McKay asked whether an equal time was assumed for each step.
Professor Dijkstra replied: 'Yes. In the synchronous elephants, I assume an equal time for each step, but I do not need to do this. If you say that some mosquitoes may be very lazy, and some very efficient, you want to synchronize them. Of course they are very small, and have very little buffer space, and the kind of synchronization regime that is required is that at count zero, when the elephant is created, each mosquito is allowed to send. It sends to both its receivers. The receivers do their computation and when they have absorbed those numbers, then signal back that the senders are allowed to send the next portion. You synchronize the computational activity on the fact that the operands have arrived, and the sending activity on the signals from the receivers that they are ready to receive.'
Professor Dijkstra went on to say that one can do this with a counting semaphore, because having heard 'you may send' twice, one knew that one had come from one receiver, and one from the other. However, proving it had been difficult originally.

Professor Michaelson asked what happened when one of the mosquitoes died, to which the reply was that one just replaced the elephant. Professor Michaelson then asked how one told that the elephant had died. The reply was that this is a problem in any large machine. Professor Dijkstra went on to say that whilst debugged programs may cost ten dollars a statement, LSI people are predicting chips at ten dollars each or less. Their use will depend on a methodlogy for designing elephants, and an environment in which they can be put to good purpose. He gave as an example of their use the validation of accesses to a data base, where the tests are extremely slow and nearly always give the same answer. The use of mosquitoes to perform such tests avoids using machine instructions which, at great expense, provide very little information.

Note

One week after his presentation the speaker's attention was drawn to an article by Harold S. Stone: "Parallel Processing with the Perfect Shuffle", IEEE Transactions on Computers, Vol. C20, No. 2, February 1971, in which his discoveries are described. The speaker would like to apologize for having presented something known in the public domain as if it were new.