The Lamport Logical Clock problem

Given a distributed system with N processes, each process goes through a finite number of events. The events could be internal or send or receive. We assume that broadcast is a send event with more than two receive events. In a correct execution of the (entire) distributed system:

- each send event is followed by one or more receive events (no messages are lost)
- a process has only one receive event associated with a send event executed at another process (there is no duplication of messages received)
- there is no receive event without a send event that has happen at a different process earlier in the execution.

We know that there is no global (shared) clock in a distributed system. Since a common physical clock is difficult to maintain and we are not interested in exact global real time but only in the ordering of these occurrences in time, we use the notion of logical clocks to have a partial order between events. As we learnt in the class, a *logical clock* C maps occurrences of events in a computation to partially ordered set such that $a \lessdot b \Rightarrow C(a) < C(b)$. Two examples of logical clocks are Lamport's logical clock and the vector clock. In class we discussed and gave examples of how to calculate the Lamport logical clock values for events.

Each process encounters a number of events and each event will have a Lamport Logical Clock value, simply *LC-value* , associated with it. In the next paragraph we will describe how to calculate these values. After that we will discuss if,, given values, how to detect whether a process has a correct or incorrect execution. Both aspects are important for the two algorithms that you have to design and implement.

Let P be some process in the N-node distributed network. We need to calculate the LC-value for all the events that P encounters.

We will reword the definition of the Lamport logical clock to make it more recursive.

Let *a* be some event encountered by P.

1. If *a* is the first event and is an internal or send event, then LC(*a*) = 1.
2. If *a* is the first event and is a receive event, then LC(*a*) = *k* + 1 where *k* is the LC-value of the send event corresponding to *a* (that has occurred at a process other than P).
3. If *a* is not the first event and is an internal or send event, then LC(*a*) = *k* + 1 where *k* is the LC-value of the event just before *a* at process P.
4. If *a* is not the first event and is a receive event, let *b* be the send event corresponding to *a* (that has occurred at a process other than P) and *k* be the clock value of the event just before *a* at process P. Then LC(a) = max{ k, LC(b) } + 1

In class we worked the following example, with N=3:

Consider three processes and the following sequence of events at processes p0, p1, p2:
p0 : a   s1   r3   b p1 : c   r2   s3 p2 : r1   d   s2   e

Here s_i and r $_i$ are corresponding send and receive events, for i =
1,2,3   Lamport's logical clock values are: p0 : 1   2   8   9 p1 : 1
6   7 p2 : 3   4   5   6

Algorithm Calculate

Problem: You are given a number N of no more than 5 processes and a matrix of N rows and M columns where each element of the matrix is a string. Each row represents the sequence of events at a process: row 0 is the sequence of events at process p0, row 1 is the sequence of events at process p1, etc.. M represents the maximum number of events at a process and you can assume that M is less than 25. The number of processes N is not fixed but you can assume that is less than 6 and greater than 1, and will be given as input to your algorithm. You can assume that there are at most 9 send events. Each event is either:

- *internal, case in which the string associated with it is a single character, a letter from the English alphabet other than 's' and 'r'*
- *send, case in which the string associated with it is a two-character string, the first character is 's' followed by a digit in the range 1-9.*
- *receive, case in which the string associated with it is a two-character string, the first character is 'r' followed by a digit in the range 1-9.*
- *null if the process is done executing, case in which the string associated with it is the NULL string*

Your algorithm needs to calculate the LC-value for each of the events. So the output will be a NxM matrix with positive integer values. If a process has less than M events, the rest of the entries in the matrix until the column M will be filled with 0.

For the example considered in class and presented above, the input to the algorithm is N=3, M=4, and the matrix of events is:

```
a    s1    r3    b
c    r2    s3   NULL
r1   d    s2    e
```

The output is:

```
1    2    8    9
1    6    7    0
3    4    5    6
```

Algorithm Verify

Problem: You are given a number N of no more than 5 processes, a positive value M and a matrix of N rows and M columns where each element of the matrix is a positive integer in the range 0..24. Each row represents the sequence of LV-values of events at a process: row 0 is the sequence of LC-values, one value for each event at process p0, row 1 is the sequence of LC-values, one value for each event at process p1, etc.. M represents the maximum number of events at a process and you can assume that M is less than 25. If a process has less than M events, the rest of the entries in the matrix until the column M will be filled with 0. The number of processes N is not fixed but you can assume that is less than 6 and greater than 1, and will be given as input to your algorithm. You can assume that there are at most 9 send events. Given such an input, you need to identify a correct sequence of events at each process or output the message "INCORRECT" if there is an incorrect execution, i.e. at least one process has an incorrect execution.

<u>Example 1:</u> For the example considered in class and presented above, the input to the algorithm is N=3,

M=4, and the matrix of LC-values is

```
1   2   8   9
1   6   7   0
3   4   5   6
```

The output is the matrix of events:

```
a    s1    r3    b

c    r2    s3   NULL

r1   d    s2    e
```

<u>Example 2:</u> Consider the input to be N=3, M=4 and the matrix of LC-values is

```
1   2   8   9
1   6   7   0
2   3   4   5 The output is:
```

```
s1  b   r3   e a    r2
s3   NULL  r1     c
d    s2
```

<u>Example 3:</u> Consider the input to be N=3, M=4 and the matrix of LC-values is

```
1   2   8   9
1   6   7   0
2   4   5   6
```

The output is "INCORRECT".

Because process p2 does not have a correct execution. The second value on the third row cannot be 4.