# Assignment 11

Deadline: Friday **02.02.2024** 23:55

**Notes:**

- Solve the assignment **on your own** — no groups allowed.
- Hand-written solutions will not be accepted, except for graphs and diagrams.
- If you hand in non-pdf files or multiple files, name your submission as `stla23_11_SURNAME.zip`, replacing `SURNAME` with your surname. Otherwise `stla23_11_SURNAME.pdf`. Also include your full name in the submitted PDF.
- Submit your solution via **Ilias**.

This is a bonus assignment. It's not mandatory to solve it but you can get up to 20 bonus points if you do.

## Exercise 11.1 Vector Clocks — 20 Points

In a distributed system, it is often necessary to determine a ordering of events that happened on the different processes. Because it is not possible to determine a global time in an asynchronous distributed system, we can only determine a partial ordering of events. One way to do this is by using *vector clocks*. A vector clock can be used to determine a happened before relation between events denoted by $A \to B$ where $A$ and $B$ are events. For an event $A$ on process $i$ and an event $B$ on process $j$, $A \to B$ holds if and only if

- $i = j$ and $A$ happened before $B$ on process $i$, or
- $A$ is the sending of a message on process $i$ and $B$ is the receiving of the same message on process $j$ or
- there is an event $C$ such that $A \to C$ and $C \to B$.

Events $A$ and $B$ are said to be concurrent if neither $A \to B$ nor $B \to A$ holds.

A vector clock is a mapping from processes to natural numbers. Each process has it's own vector clock. Let $VC_i$ denote the vector clock of process $i$. The initial value of $VC_i$ is $VC_i[j] = 0$ for all processes $j$. When an event $A$ happens on process $i$, the entry of $VC_i$ for process $i$ is incremented by one, i.e., $VC'_i[i] = VC_i[i] + 1$. When process $i$ sends a message to process $j$, it increments the entry of $VC_i$ for process $i$ by one and sends the vector clock $VC_i$ with the message. When process $j$ receives the message, it increments the entry of $VC_j$ for process $j$ by one and updates the entry of $VC_j$ for all processes $k$ by taking the maximum of the current value and the value received in the message, i.e., $VC'_j[k] = max(VC_j[k], VC_i[k])$. The sending and receiving of a message is also considered an event. An example of this is shown in Figure 1.

For a event $A$ we denote the vector clock of the process on which $A$ happened by $VC(A)$. Let $VC(A)$ and $VC(B)$ be two vector clocks. We state that $VC(A) < VC(B)$ if and only if

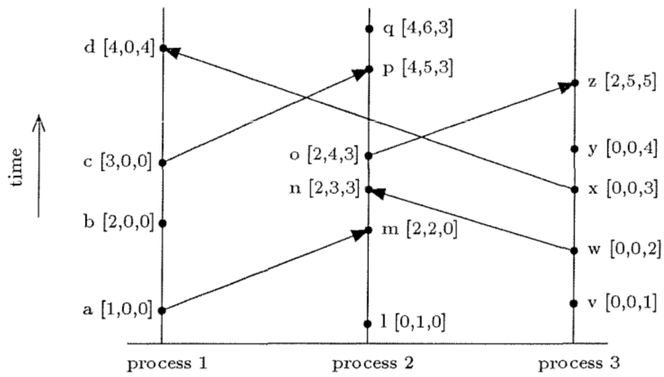


Figure 1: Vector clock example with three processes taken from https://sookocheff.com/post/time/vector-clocks/

$VC(A)[i] \leq VC(B)[i]$ for all processes $i$ and $VC(A)[j] < VC(B)[j]$ for at least one process $j$. The following holds:

$$A \rightarrow B \quad \Leftrightarrow \quad VC(A) < VC(B)$$

With this we can determine a partial ordering of events.

**a)** Write the module `VectorClock` in TLA$^+$ that specifies the vector clock operations for $N$ processes. Keep in mind that in asynchronous system there can happen events between the sending of a message and the receiving of the same message. Use constants to make the state space of the specification finite. **(10 points)**

**b)** Write the specification `VectorClockMC` in TLA$^+$ that keeps track of the happened before relation between events. Use the relation to check with the help of TLC if your specification `VectorClock` does imply the the happened before relation. **(10 points)**

---

**Total: 20 Points**