Leslie Lamport

A biography essay | Lee Campbell | campbel2@tcd.ie

Back in 1974, we might have found Leslie Lamport at his local bakery, not mastering the art of baking but instead theorising over the concurrency of computer systems. It's an unlikely place to do so but Lamport noticed a problem there. While the bakery had multiple cashiers, if more than one customer approached an individual cashier at once, that cashier would try to communicate with them simultaneously and become confused [Association for Computing Machinery, 2013]. Approximately 10 years earlier, Edsger Dijkstra - a name synonymous with the field of Computer Science today - had posed the Dining Philosophers problem. [Lamport, 2018].The crux of the problem centred around each philosopher sharing a single set of cutlery with another, in order to eat and survive starvation. This problem of mutual exclusion, is much like the one Lamport faced in the bakery.

Indeed Dijkstra and Knuth among others had designed algorithms which solved the issue set out above but in both, contention is resolved by means of a complicated busy-waiting loop in which many shared variables are read and written [Anderson, 2001]. The bakery had given Lamport an idea for a more elegant solution to this problem. He envisioned a bakery in which each customer receives a number - incremented by one for each. In the bakery, a global counting system displays the number of the customer currently being served. Every other customer must wait until the cashier finishes serving the current customer and the next numbered is displayed. If that customer forgets something and wants to buy an additional item, they must take a new number as if they have just entered the store and wait again in line. In this analogy the customers are said to be threads, identified by a global variable.

The bakery algorithm is significant because it implements mutual exclusion without relying on any lower-level mutual exclusion [Lamport, 2018]. While this parable might seem frivolous, it was this simple idea which planted the seed for what became the eventual influential algorithm.

Of course much of Lamport's work was about planting seeds which is what has made is work so impactful. We have to look no further than Prof. James H. Anderson's 2011 paper entitled "Lamport on Mutual Exclusion: 27 Years of Planting Seeds" to see just how much 'planting seeds' became synonymous with Lamport. These seeds have grown into research topics of independent interest from researchers around the world. They form the foundation for broad areas of concurrency, and have influenced the specification, development and verification of concurrent systems. His work in distributed systems has also seen him go onto win numerous prestigious awards most notably perhaps the A.M. Turing award in 2013 the de facto nobel prize for Computer Scientists. [Association for Computing Machinery, 2013]. He may not be one of the many posterboys of the industry but his work and impact undoubtedly spreads far and wide. Everytime you access a modern computer over a network, you can be guaranteed you're using one of Lamport's ideas.

Born in February 1941 in New York, Lamport graduated from M.I.T. with a B.S. in Mathematics in 1960 and subsequently with M.A. and Ph.D. degrees in Mathematics from Brandeis University, respectively in 1963 and 1972 [Lamport, 2018]. Much of Lamport's success has come from his ability to approach problems differently than most, in particular through the lense of abstraction. Being able to see the essence in a problem and not to be distracted by the details is what set him apart. He often attributes part of his unique approach to his grounding in mathematics and moreover his need in mathematics to always understand things from first principles, this he said, helped him think 'above the code'.

A colleague of Lamport's, Michael Burrows - a distinguished engineer at Google - describes how "it's so simple when Leslie describes it you cannot imagine how you didn't think of it yourself, which is simultaneously wonderful and extremely frustrating". This true of many of Lamport's discoveries but in particular his paper on Time, Clocks and the Ordering of Events in a Distributed System, one of the most cited papers in Computer Science today [Microsoft Research, 2014]. In those days, real time was the only way to capture the communication delay in distributed systems, which many realised was not natural for such a system. Lamport envisioned the first real alternative to logical clocks, which involved imposing partial order on events based on the causal relation induced by sending messages from one part of the system to another. This notion was a game changer, and in fact using ordering as a means of proving system correctness in concurrent systems is still widely used today.

However, perhaps the most important concept setout in the paper was that of replicated state machines, which demonstrated how to replicate a state machine using logical clocks. In order for messages to be naturally replicated, Lamport stated that you must start a set of deterministic state machines in the same state and then make sure they process the same set of messages in the same order. Lamport had devised what was an elegant but incomplete solution in terms of agreement problem; a key proponent to making sure the processors in the system agree the order of the messages. Of course in typical Lamport fashion it was merely another seed planted. He then went on to delve deep into the agreement problem within distributed systems, a journey that would later lead him to Paxos.

Paxos is a fault tolerant algorithm which addressed many of the failures in Lamport's initial approach to state machine replication. Alongside being the almost ubiquitous industry standard framework for designing and reasoning about agreement and replication, Paxos has a colourful history. In the late 1980s, Lamport having fleshed the idea out into a research paper, true to form decided to present the work as a parable, this time solving the problem of achieving consensus on measures voted for by a rather lazy parliament on the ancient island of Paxo. The Transactions on Computer System among other journals rejected the paper, insisting he drop the "paxos stuff". He refused to relent, and while this delayed the algorithm being picked up for a number of years, it didn't stop it's eventual widespread success. Many companies building critical information systems, including Google, Yahoo, Microsoft, and Amazon, have adopted the Paxos foundations [Association for Computing Machinery, 2013]. Lamport continued to make key contributions over the following years to the theory of specification and verification of concurrent programs. Not least in developing the TLA (Temporal Logic of Actions) language and the TLA+ toolset, for modeling and verifying distributed algorithms and systems which would later be used by companies such as Intel and Microsoft to detect major flaws in their hardware before release.

Lamport, as you have seen had a history of theorising and then doing by implementing various tools based on his papers and ideas. When it came to typesetting that was no difference. Lamport as he was planning on writing the 'Great American Concurrency Book' figured he need a set of macros for TeX, a typesetting tool developed by Don Knuth. He pushed the boundaries, taking TeX and adding a flexibility that would see it become the default for technical publishing. Lamport remarks how he "was planning to write a user manual, but it never occurred to me that anyone would actually pay money for it." That was until Peter Gordon, an Addison-Wesley editor, and his colleagues visited him at SRI. The LaTex manual he would later publish with Addison-Wesley has since sold a few hundred thousand copies worldwide. [Lamport, 2018].

Lamport in his early days had a grand goal of developing a theory of concurrency and while that never happened, he instead devoted his career to solving many of the smaller problems we explored throughout this essay. Of course now as you look back, you realise that in solving so many small problems he has indeed created a grand theory.

A two time winner of the Dijkstra Prize, an IEEE John von Neumann Medal holder and a winner of the A.M. Turing award amongst numerous other accolades; Lamport's impact across the field of distributed systems and concurrency with LaTex as a footnote has undoubtedly changed the face of software engineering today. His tools help programmers around the world think above the code to determine what applications and services should do and ensure that they do it. Lamport, now 77, continues to plant seeds for the future through his work with Microsoft research labs today that will undoubtedly inspire the next generation [Lamport, 2018].

Bibliography

Lamport, Leslie (2018) *The Writings of Leslie Lamport*. Available at:

http://lamport.azurewebsites.net/pubs/pubs.html.

Britannica, T. E. of E. (2017) “Leslie B. Lamport,” *Encyclopædia Britannica*, 15 September.

Available at: https://www.britannica.com/biography/Leslie-Lamport.

Microsoft Research (2014) “A Conversation with Turing Award Winner Leslie Lamport,”

YouTube, 14 April. Available at: https://www.youtube.com/watch?v=pgWTmOyUjtM.

Association for Computing Machinery (2013) *Leslie Lamport - A.M. Turing Award Laureate*.

Available at: https://amturing.acm.org/award_winners/lamport_1205376.cfm.

Anderson, JH (2001)*Lamport on mutual exclusion: 27 years of planting seeds.*

Available at:

https://www.cs.unc.edu/~anderson/papers/lamport.pdf