the Availability Digest

www.availabilitydigest.com
@availabilitydig

Leslie Lamport Wins Turing Award for Distributed Computing

March 2014

Leslie Lamport, a Principal Researcher at Microsoft Research, has been named the recipient of the 2013 ACM A. M. Turing Award for his contributions to the reliability of distributed computing systems. He contributed to the theory and practice of building distributed computing systems that work as intended.

ACM (Association for Computing Machinery – www.acm.org) is widely recognized as the premier organization for computing professionals, delivering a broad array of resources that advance the disciplines of computing and information technology. The A.M. Turing Award, the ACM's most prestigious technical award, has been given annually since 1966 for major contributions of lasting importance to computing. The award is accompanied by a cash prize of $250,000, which in recent years has been underwritten by Intel Corporation and Google, Inc.

The award is named after Alan M. Turing. Turing (1912 - 1954) is widely considered to be the father of modern-day computer science. During World War II, he worked for the British code-breaking center and was responsible for breaking German ciphers, a substantial aid to the Allied war effort.[1]

Lamport's award citation reads as follows:

ACM – Association for Computing Machinery

For fundamental contributions to the theory and practice of distributed and concurrent systems, notably the invention of concepts such as causality and logical clocks, safety and liveness, replicated state machines, and sequential consistency. Along with others, he invented the notion of Byzantine failure and algorithms for reaching agreement despite such failures. He contributed to the development and understanding of proof methods for concurrent systems, notably by introducing the notions of safety and liveness as the proper generalizations of partial correctness and termination to the concurrent setting."

His research has imposed a well-defined coherence on the seemingly chaotic behavior of distributed computing systems, in which several autonomous computers communicate with each other by passing messages. His algorithms and formal modeling and verification techniques improve the quality of real distributed systems.

Safety and Liveness

Fundamental to his contributions to distributed computing are the concepts of *safety* and *liveness*. The property of *safety* requires that "something bad will never happen." The property of *liveness* means that

[1] In a devastating blow to information technology, Turing was convicted of homosexuality in 1952 and chemically castrated. He committed suicide in 1954. The British government has recently apologized, and the Queen gave him a posthumous pardon in 2013.

"something good eventually happens." Eventual consistency is an example, in which a distributed computing system will ultimately reach a consistent state without passing through inconsistent states that could be harmful.

Logical Clocks

A logical clock is a mechanism for capturing the chronology of causal relationships in a distributed system. It keeps accurate time independent of any precise reference time.

Many distributed transaction processing systems depend upon all nodes seeing the same time so that they may properly order events (for instance, which transaction occurred first?). However, if there are no critical ties with other independent systems, then the absolute time of these events may not be terribly important so long as the timestamps are properly ordered.

Lamport's logical clocks solve this problem with a simple algorithm.[2] As with any clock, a logical clock keeps track of time via clock ticks. The greater the value of the clock tick, the later the time. Lamport's algorithm depends only upon two obvious rules:

Rule 1:	Within a system, there is a minimum of one clock tick between any two events.
Rule 2:	If Node A sends a message to Node B, Node B receives that message after Node A sent it.

Replicated State Machines

Distributed systems are often structured in terms of clients and services. Each service comprises one or more servers that export operations that clients invoke by making requests. If fault tolerance is required, multiple servers that fail independently must be used. State machine replication is a method for implementing a fault-tolerant service by replicating servers and coordinating client interactions with the server replicas.

Replicas of a single server are executed on separate processors of a distributed system, and protocols are used to coordinate client interactions with these replicas. Logical clocks are used to keep the multiple servers time-synchronized.

Sequential Consistency

Sequential consistency is a property that requires that the result of the execution of any series of steps in a distributed processing system in which the servers are executing concurrently be in the same sequential order on all systems. The system provides sequential consistency if every node of the system sees the (write) operations on the same memory part (page, virtual object, cell, etc.) in the same order, although the order may be different from the order in which the operations were issued.

Sequential consistency in multicore processors is a particularly difficult problem, since a given execution sequence may be assigned to different cores in various processors.[3] Therefore, the sequence of instruction execution is indeterminate.

[2] Time Synchronization in Distributed Systems – Part 3, *Availability Digest*; February 2008.
http://www.availabilitydigest.com/public_articles/0302/time_sync_3.pdf
Time, Clocks, and the Ordering of Events in a Distributed System, *L. Lamport, Communications of the ACM*; July, 1978.
http://www.stanford.edu/class/cs240/readings/lamport.pdf

[3] This is a serious problem that had to be solved by Stratus. Its ftServer fault-tolerant systems use two processors running in lock-step, and the execution sequence in each must be deterministic.

Byzantine Fault Tolerance

The award citation went on to give credit to Lamport for inventing the notion of Byzantine failure and the algorithms for reaching agreement in a distributed system despite such failures. A Byzantine failure is one in which the components of a distributed system fail in arbitrary ways – not only by crashing but by incorrectly processing requests and generating what seems to be logical results. A system that appears to be properly working but which is generating erroneous outputs is a *traitor* (the Byzantine Empire's army was plagued by treachery at the highest levels).

If a distributed system has several traitors, what is the correct operation? Using Lamport's algorithms, the correct system services will be provided so long as there are not too many traitors. The number of servers in the distributed system must be at least three times the maximum number of traitors, plus one.[4]

TLA+

TLA+ (Terminal Logic of Actions) is a system specification language devised by Lamport. It describes the specifications for a system via simple mathematics. It is especially well-suited for writing high-level specifications of concurrent distributed systems. It couples with PlusCal, an algorithmic language that translates into a TLA+ specification, to which TLA+ tools can be applied.

The principal TLA+ tools include a model checker and a proof system.

Lamport also developed LaTeX (pronounced la-tek), a documentation system and markup language. LaTeX is now the de facto standard for the publication of scientific documents in several fields.

Leslie Lamport

Leslie Lamport holds a B.S. degree in mathematics from Massachusetts Institute of Technology as well as M.S. and Ph.D. degrees in mathematics from Brandeis University. Prior to his current position at Microsoft Research, he held tenures at SRI International and Digital Equipment Corporation (later Compaq Corporation and now HP).

The author or co-author of nearly 150 publications on concurrent and distributed computing and their applications, he is also the author of the book Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers, published by Addison-Wesley Publications.

Lamport has been recognized in many other ways. He received the IEEE Emanuel R. Piore Award for his contributions to the theory and practice of concurrent programming and fault-tolerant computing. He was also awarded the Edsger W. Dijkstra Prize in Distributed Computing for his paper "Reaching Agreement in the Presence of Faults." He won the IEEE John von Neumann Medal and was also elected to the U.S. National Academy of Engineering and the U.S. National Academy of Sciences.

The Award Presentation

ACM will present the 2013 A.M. Turing Award to Leslie Lamport at its annual Awards Banquet on June 21 in San Francisco, CA.

Summary

The theory and practice of concurrent distributed systems has been significantly advanced by Lamport's work. These systems, once impractical, are now common day and working well.

[4] Reaching Agreement in the Presence of Faults, *M. Pease, R. Shostak, L. Lamport*.
http://research.microsoft.com/en-us/um/people/lamport/pubs/reaching.pdf

Acknowledgement

Our thanks to our subscriber, Bruce Holenstein, for bringing this topic to our attention.