The Distributed Computing Column

Sergio Rajsbaum*

November 8, 2000

Abstract

The Distributed Computing Column covers the theory of systems that are composed of a number of interacting computing elements. These include problems of communication and networking, databases, distributed shared memory, multiprocessor architectures, operating systems, verification, internet, and the web.

This is the first Distributed Computing Column that I have edited. It consists of four sections: A few thoughts on the role of distributed computing theory, a reprint of the PODC 2000 Influential-Paper Award to statement, a review of the PODC 2000 Conference, and a summary of a paper on robotics presented at the conference. The column concludes with some news and acknowledgments.

Request for Collaborations: Please send me any suggestions for material I should be including in this column, including news and communications, and suggestions for authors willing to write a guest column or to review an event related to theory of distributed computing. In particular, I would welcome open problems with a short description of context and motivation.

1 Principles of Distributed Computing: An Exciting Challenge

In my first column, it is only natural to start with a few thoughts on the role of theory of distributed computing (TDC). Those of us who are working on the TDC have felt overwhelmed by the enormous speed at which things are changing in the distributed computing world, in particular, the advances in networking, distributed systems, and the web. This makes the field very exciting, but also very challenging. When trying to work on TDC, one often gets the feeling of aiming at a moving target. Consider the following statements about theory of computing, where I have replaced "computing" with "distributed computing."

> *Build the foundations of the discipline of [distributed computing] by developing models of computation and methods of analysis applicable to the models [1].*
>
> *Theory of [distributed] computing provides [distributed computing] with concepts, models, and formalisms to help reason about these concepts and models. Theory of [distributed] computing addresses the question of what is and is not feasibly computable and creates the algorithms for the intellectual processes that are being automated. Software and efficient algorithms are the base of today's technology and of technology to come. [5].*

*Compaq Cambridge Research Laboratory, One Cambridge Center, Cambridge, MA 02142-1612. rajsbaum@crl.dec.com. On leave from Instituto de Matemáticas, UNAM.

Indeed, the models and methods change, trying to reflect the changing technologies. The meaning of an efficient algorithm and what are the parameters we should be trying to optimize change. New problems and paradigms are often introduced. Therefore a dual challenge is presented: (i) To find and address fundamental problems that will be relevant over time and that concern fundamental questions of TDC, and (ii) at the same time, to keep up-to-date developing relevant theory for the technology of the moment. We must carry on with both tasks, if we want to build a solid scientific foundation for our field, while understanding what the fundamental questions of our field *are* and contributing to the exciting technological development we are witnessing. Moreover, in a world of rapidly evolving technology, theoretical work must come early, before aspects of the field have frozen in unprincipled, ad hoc states [5]. Putting it in poetic terms [2]:

> *Pure science is like a beautiful cloud of gold and scarlet that diffuses wondrous hues and beams of light in the west. It is not an illusion, but the splendor and beauty of truth. However, now the cloud rises, the winds blow it over the fields, and it takes on darker, more somber colors. It is performing a task and changing its party clothes–think of it as putting on its work shirt. It generates rain that irrigates the fields, soaking the land and preparing it for future harvests. In the end it provides humanity with its daily bread. What began as beauty for the soul and intellect ends by providing nourishment for the humble life of the body.*

For the vast majority of basic scientists, the ultimate purpose of their calling is to *understand* the world around them. Power is a byproduct of understanding. No external pressure is stronger than the internal force of curiosity [4]. To reinforce this point, consider Goldreich and Wigderson's paragraph from [3] about Theory of Computation (TOC), which applies equally well to TDC.

> *The success of TOC is directly correlated to the extremely high quality and creativity of researchers in TOC, to their independence, and to the fundamental (and exciting) nature of the questions TOC addresses. In order for the Theory of Computing to prosper in the future it is essential that TOC attracts the same calibre of researchers, that Theoretical Computer Scientists concentrate their research efforts in Theory of Computing and that they enjoy the freedom to do so. Free pursuit of their research interests may well lead individual scientists to work closely with/in application areas. Indeed, our field has already an admirable tradition where many TOC leaders (undirected and un-forced) chose to redirect part of their research so as to strongly influence application areas as well as other sciences. Yet, decisions taken by individual scientists following their own understanding of the discipline differ drastically from attempts to direct the whole discipline towards directions which are not intrinsic to it.*

The importance and extent of TDC was made evident in the last ACM STOC conference. The single invited speaker was Tom Leighton of the Department of Mathematics, MIT, and Akamai Technologies. His talk was entitled "Some Problems Related to Content Distribution." On May 20, 2000, also associated with STOC, there was the *Workshop on Challenges for Theoretical Computer Science.* Many of the challenging directions open to TOC at the beginning of the 21st Century discussed in the workshop have to do with distributed problems. Among the more interesting, are the connections with social sciences on problems involving group consensus, voting, and elections, some of which have been studied for a long time. See also the DIMACS Workshop on Bioconsensus, October 25 - 26, 2000, http://dimacs.rutgers.edu/Workshops/Bioconsensus/. Another direction considered in the Challenges Workshop is distributed decision making. The following quote is from Fred Roberts:

Today, decision making by the corporate, government, or military decision maker requires flawless, efficient, coordination among many different players. Communication and coordination among agents has been studied in the framework of distributed computing. This work addresses reliability, synchronization, and communication complexity, but often involves such unrealistic assumptions as essentially infinite computing power by agents. Procedures for merging or aggregating information have been widely studied by the decision science community, but without emphasis on the reliability of communication or the computational complexity of the protocol. The computational characteristics of the agents, especially when they include sensors, are extremely relevant, as is the structure of the communication network. We will need to combine algorithmic and decision science approaches, simultaneously taking into account requirements for good group decisions, constraints on the computational power of the agents, and the structure of the communication network.

Auctions and bidding are other directions related to TDC discussed at the workshop. With respect to game theory and economics, researchers (e.g. Nisan, Ronen in STOC99) are considering distributed *mechanism design* problems, where the participants act according to their own self-interest. Finally, there is no need to emphasize the importance of the connections to the web. The workshop devoted an entire area of discussion to this subject.

The last ACM PODC Conference was a great example of how the TDC community is working towards its complex goals. There were excellent papers on problems that have been addressed since the first PODC conference, as well as on new problems. The Conference was also a good example of an additional difficulty that TDC faces: the wide spectrum of technologies encountered, and within each one, low level problems through application issues. Papers that could appear in PODC, often appear in conferences on architecture, databases, networking, operating systems, web, and others. One of the goals of the following review of PODC 2000 is to illustrate how the Conference has played a central role in TDC, with relations to other conferences specializing in particular theoretical and applied subjects. There are rich interactions in all directions: principles that are integrated into systems, and PODC papers inspired by practical problems. In the rest of the column, I will review the PODC 2000 Conference. First I will concentrate on the Influential-Paper Award, dealing with a fundamental subject that has had significant impact on TDC for over a decade. Then I will concentrate on the paper by Walter, Welch, and Amato, which is inspired by problems of current technologies.

2 First PODC Influential-Paper Award

This is the first year at which the PODC Influential-Paper Award was presented. This award is given each PODC to a person with an outstanding paper on the principles of distributed computing, whose significance and impact on the theory and/or practice of distributed computing has been evident for at least a decade. This year's winner is Leslie Lamport, for his paper "Time, Clocks, and the Ordering of Events in a Distributed System." The following (up to the end of this section) is reprinted from the conference proceedings.

Leslie Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, July 1978, 21(7):558-565.

Members of the selection committee were James Anderson, Cynthia Dwork, Vassos Hadzilacos, and Fred Schneider. The following is a description of the winning paper's contributions, written by Keith Marzullo.

This paper contained two main ideas, each of which led to a line of research that has dominated distributed computing for a decade or more:

1) A precise characterization of causality in distributed systems (called the clock condition) and a framework (similar to Minkowski diagrams of special relativity) for explaining and reasoning about event ordering in distributed protocols. The simplest way to implement the clock condition is with what Lamport called in this paper "logical clocks." The logical clock abstraction had an immediate impact on the field and is the reason that this paper is cited so often. Research enabled by this contribution includes the vector and matrix clock abstractions, the notion of consistent cuts (answering the question of "what is a state in a distributed system"), stable and nonstable predicate detection, and the semantic basis for epistemological logics (such as the logics of Knowledge, Common Knowledge, and Belief that have been advocated for specifying distributed protocols). Finally, it is worth noting that this was among the very first papers to show how distributed systems were fundamentally different from other concurrent systems and was the first paper to show how a rigorous mathematical basis (the "happens before" relation) could be used to talk about such differences.

2) The state machine approach as a generalization of n-module redundancy. This is the contribution that has proven to be the most influential, both to the theory and practice of distributed computing. The paper gives a distributed mutual exclusion protocol that ensures access to the critical section is obtained in causal order. More importantly, the paper explains how this protocol is an example of a general approach (the so-called "state machine" approach) for managing replication. Topics that led from this approach include:

- Byzantine agreement. Such protocols ensure that all state machines get the same set of inputs in spite of failures. Much work has led from this problem, including fast protocols, impossibility results, failure model hierarchies, and so on.

- Byzantine clock synchronization and ordered multicast protocols. Such protocols are used to order concurrent requests in the same way. Combined with agreement protocols, nonfaulty deterministic state machines are ensured to have the same states.

It is not surprising that the first papers defining both of the above problem areas were authored or co-authored by Lamport, as he was simply working through the protocols and sub-problems that spun off of the results in the original paper.

A good deal of more practical systems work also traces its roots to the state machine approach. A large number of distributed system toolkits (e.g., ISIS and successors, Transis, Relacs, Spread, Newtop) embody the state machine approach or some variant. And the designs of most critical systems (e.g., Boeing 777 control system, US, English, and French air traffic control systems, stock trading systems, etc.) are based on active replication schemes that are derived from the work in this paper.

* * *

The following is a brief historical perspective of the paper written by the author himself.

Jim Gray once told me that he had heard two different opinions of this paper: that it's trivial and that it's brilliant. I can't argue with the former, and I am disinclined to argue with the latter.

The origin of this paper began as a note titled *The Maintenance of Duplicate Databases* by Paul Johnson and Bob Thomas. I believe their note introduced the idea of using message timestamps in a distributed algorithm. I happen to have a solid, visceral understanding of Special Relativity. This enabled me to grasp immediately the essence of what they were trying to do. Special Relativity teaches us that there is no invariant total ordering of events in space-time. Different observers can disagree about which of two events happened first. There is only a partial order in which an event e_1 precedes an event e_2 iff e_1 can causally affect e_2. I realized that the essence of Johnson and Thomas's algorithm was the use of timestamps to provide a total ordering of events that was consistent with the causal order. This realization may have been brilliant. Having realized it, everything else was trivial. Because Thomas and Johnson didn't understand exactly what they were doing, they didn't get the algorithm quite right. Their algorithm permitted anomalous behavior that essentially violated causality. Quickly I wrote a short note to them pointing this problem out and corrected the algorithm.

It did not take me long to realize that an algorithm for total ordering of events could be used to implement any distributed system. A distributed system can be described as a particular sequential state machine that is implemented with a network of processors. The ability to totally order the input requests leads immediately to an algorithm to implement an arbitrary state machine by a network of processors, hence implementing any distributed system. I then wrote this paper, which was about how to implement an arbitrary distributed state machine. As an illustration, I used the simplest example of a distributed system I could think of—a distributed mutual exclusion algorithm.

This is my most often cited paper. Many computer scientists claim to have read it. But I don't think I ever encountered anyone who was aware that the paper said anything about state machines. They seem to think that it is about either the causality relation on events in a distributed system, or the distributed mutual exclusion problem. People have insisted that there is nothing about state machines in the paper. I have even had to go back and reread it to convince myself that I really did remember what I had written.

The paper describes the synchronization of logical clocks. As something of an afterthought, I decided to see what kind of synchronization it provided for real-time clocks. I included a theorem about real-time synchronization. I was rather surprised by how difficult the proof turned out to be. This was an indication of what lay ahead in the work on Byzantine clock synchronization. But, that is another story.

3 The PODC 2000 Conference

The 19th ACM Conference on Principles of Distributed Computing, sponsored by SIGACT and SIGOPS, was held July 16-19 2000 in Portland, Oregon. Overall, this year's PODC was an excellent experience. There were 32 papers and 11 brief announcements. The papers were selected in electronic discussions and at a meeting of the program committee. There were 117 papers submitted to the regular track, with 32 accepted at a 27% acceptance rate. There were 31 papers submitted as brief announcements, with 11 accepted at a 35% acceptance rate. The submissions came from all over: USA (51), Israel (12), France, Japan, Switzerland (6 each), UK, China (5 each), Australia, Canada (4 each), Taiwan, Korea (3 each), Germany, Italy, Spain, Singapore (2 each), Brazil, Sweden, Lebanon, Belgium, Trinidad&Tobago (1 each). But the accepted papers where from a small subset of these countries: USA (15), Israel (8), France (3), Switzerland (2), Germany, Japan, Spain, and Canada (1 each). A wide variety of subjects were represented in the submissions: distributed databases, distributed operating systems, mobile computing, multiprocessor/cluster, concurrency control, cryptographic protocols, wait-free algorithms, specification/semantics, consistency conditions, synchronization, distributed OO, middleware, networking, internet/web, fault-tolerance, distributed algorithms and complexity.

Over the past 5-6 years, the attendance has been in the 90-125 person range. This can be best characterized as steady after accounting for factors such as location (beach or not, campus housing or not, hotel pricing, etc.), and co-location with other conferences (SPAA in 1998, Crypto and Self-stabilization workshops in 1997, and FCRC in 1996 and 1999, and this year with a Middleware Workshop). This year there were 105 attendees, which included 21 students.

The conference began on Sunday afternoon with a TLA tutorial presented by Leslie Lamport. There we three invited talks, by Craig Partridge, Mark D. Hill, and Eric A. Brewer. This year, PODC 2000 was run in concert with a Middleware Symposium, which had its own program committee and submission procedures. The PODC Influential-Paper Award was presented to Leslie Lamport. The winner of the Best Student Paper Award was Yehuda Hassin. Gil Neiger did an excellent job as Conference Chair and Local Arrangements. Everything went smoothly, the Marriott hotel was very comfortable and had efficient service, we had nice weather, and Portland's downtown is pleasant with good restaurants and cafes. In addition, Gil was Webmaster! The PODC web page www.podc.org has a link to the PODC 2000 conference page, which includes links to some of the papers, the rump session talks, and the slides for the tutorials and invited talks.

Best Student Paper Award, Tutorial, Invited Talks, Middleware Workshop

This year's Best Student Paper Award was given to Yehuda Hassin, for the paper "Sparse Communication Networks and Efficient Routing in the Plane," co-authored with David Peleg. This paper proposes an approach to network design termed routing-oriented network design, which is based on designing the network topology and its routing scheme together. It attempts to optimize some of the relevant parameters of both simultaneously, in the special case of points located in the Euclidean plane. The desirable network parameters that are considered include low degree and small numbers of communication links. The desirable routing parameters considered include small routing tables, small numbers of hops and low routing stretches. Two different schemes are presented, one based on direct navigation in the plane and the other based on efficient hierarchical tree covers.

Leslie Lamport's tutorial on TLA (Temporal Logic of Actions) was a very good introduction to the system he developed for describing and reasoning about concurrent systems. He managed to present a step by step particular example, while at the same time giving a global picture of how the system is used. The system is based on a logic, TLA, and its language, TLA+. These are used to specify an algorithm, its correctness properties, as well as rules for proving that the algorithm satisfies its specification. There is also TLC, a model checker for TLA+ specifications. Lamport insisted on the advantages of using ordinary math as much as possible, which is what TLA tries to enable when one writes an algorithm and its proof " math is simpler than any programming language." He argued that just being able to write an algorithm and its specificationin in TLA+ already forces you to understand the algorithm and what it is supposed to do. This makes it harder to cover gaps in a proof with handwaving. He described TLA+ with a detailed description of the mutual exclusion algorithm from his Award paper, including its specification, and techniques for proving its correctness, and including using the model checker. He then described in more general terms the state machine approach, also from his award paper. Lamport used the mutual exclusion algorithm as an example of a state machine. He argued for the importance of working in the domain of math, and pointed out that the mutual exclusion spec was 11 lines, 80% math, the algorithm 70 lines, 95% math, the system spec 2000 lines, 99% math. As another example of working very close to math, he brought our attention to the fact that composition is specified simply as conjunction in TLA.

The three invited talks were enlightening and motivating. They all stressed the need for more PODC work to help cope with emerging problems in their respective areas: networking, shared memory multiprocessor architectures, and systems. There are pointers to the slides in the PODC web page.

The first invited talk was " Data Communications vs. Distributed Computing," given by Craig Partridge of BBN Technologies. He said

> *one of the more distressing aspects of the past fifteen years is the evolution of data communications and distributed computing into distinct and somewhat disconnected disciplines.*

Focusing on key technical breakthroughs in data communications and distributed computing, he examined why the disconnection occurred, and what we can do to increase communication between the two disciplines. He gave an overview of the advancements in networking. The discovery of self-similarity to model traffic, something that queuing theory failed to do. He discussed the problems of achieving quality of service in a self-similar world, and solutions such as Weighted Fair Queuing (Demers, Keshav, Shenker SIGCOMM89), and Packetized General Processor Sharing (Parekh, Gallager Tans. Netw. 93). He challenged us to try to come up with simpler and cheaper to implement versions of WFQ and PGPS. Other various subjects were discussed with interesting open questions for the PODC community: Ad-Hoc Networks (thousands of wireless nodes are moving in a small space), robustness of the Internet, simulation (How do you simulate something 100 times bigger than anything ever built?), Measurement (How much can you learn just from the edge of the network?), Errors (Packets damaged frequently, what to do?), and Anycast (sending a message to anyone in a group).

The second invited talk was "How Computer Architecture Trends May Affect Future Distributed Systems: From InfiniBand Clusters to Inter-Processor Speculation," by Mark D. Hill of the University of Wisconsin-Madison. Mark started by recalling the impressive advancements in processor speeds, with the same increase from prehistory to 2000 as from 2000 to 2001, and radical cost reductions. These impressive advancements have enabled various computer architecture innovations. These remarks motivated his discussion on changes in distributed system design, system area networks, multiprocessor servers and clusters. He discussed the importance of building systems that scale well w.r.t. storage, bandwidth, computation and availability. In his discussion about storage, he concentrated on the emerging standard InfiniBand. InfiniBand replaces shared bus architectures with interconnections between servers, remote storage and networking through a center of its switches and links. Data scalability is achieved by adding nodes as needed. He pointed to the need for formal work on both specifying and designing the software. Hill's second subject was on designing scalable multiprocessor servers. He described how processor caches have been introduced to cope with the key problem of memory bandwidth. He discussed the coherence problems this introduces, with the two classic classes of coherence algorithms: snooping and directory. Hill then talked about multicast snooping (ISCA99), an algorithm he co-authored to get the best of both worlds using prediction, and the efficiency of snooping with directory scalability. The newer algorithm, timestamp snooping (ASPLOS00), uses logical clocks to go around traditional snooping that physically totally orders transactions. Again, Hill talked about some of the problems that the PODC community could work on: understanding locality, multicast and network topologies, fault-tolerance, etc.

The third talk was "Towards Robust Distributed Systems," by Eric A. Brewer, University of California, Berkeley, and Inktomi. He discussed issues related to fault models, high availability, graceful degradation, and data consistency. He talked about building reliable, scalable and easier to manage distributed systems. Brewer explained the impossibility of achieving simultaneously fault-tolerance, availability and consistency. This results in the need to build systems that degrade gracefully in the presence of these three requirements; trading availability vs. consistency vs. network partitions. He discussed two related applications built at Berkeley. Brewer raised some controversy when arguing against abstractions such as RPC, that hide semantics of the underlying system from the programmer, thereby creating the illusion of a simpler world. This is apparent in particular when hiding issues such as the complexity of the fault model, and the communication mechanisms (message passing).

The Middleware Workshop included an introductory tutorial by Michael Ogg, Synaptus, and 11 papers. It was interesting to see in the workshop new variants or solutions to traditional PODC problems inspired by the middleware systems being addressed. For example, the paper "X-Ability: A Theory of Replication" by Frolund and Guerraoui proposes a specification of a replication system, independent of particular replications strategies such as active or primary backup. It presents also a particular replication algorithm derived from these techniques. Other examples are multicast in the paper by Herlihy, Tirthapura and Wattenhofer, or atomic broadcast in the paper by Aguilera and Storm.

4 A Distributed Problem in Robotics

The technical program consisted mostly of high quality papers over a wide range of topics. Some classic topics remain active research areas: there were several papers on routing, consensus, clock synchronization, vector clocks, and self-stabilization. There was a interest in *adaptive algorithms.* The complexity of these algorithms depends only on the actual number of participating processes in an execution, as opposed to the total number of processes in the system. New techniques and insights have been found while tackling problems such as renaming, mutual exclusion, and snapshots in their adaptive context. The paper I will describe next deals with one of the more original problems: robotics.

We are well aware that PODC has an overlap with various conferences in other fields, such as networking, verification, databases, etc. There are also less obvious conferences dealing with problems where we have an opportunity to bring in our "distributed skills," like the Fourth International Workshop on the Algorithmic Foundations of Robotics, held in March in Dartmouth College. This has been done by Jennifer E. Walter, Jennifer L. Welch, and Nancy M. Amato in their paper "Distributed Reconfiguration of Metamorphic Robot Chains." A *self-reconfigurable* robotic system consists of a collection of independently controlled, mobile modules. The system changes its shape due to the motion of individual modules, which can connect, disconnect and move around adjacent modules. The paper deals with *metamorphic systems,* in which all modules are identical. The paper presents a model that abstracts from specific hardware details about the modules and how they achieve locomotion. In particular, it describes issues having to do with how a module can move using a substrate composed of other modules. The plane is partitioned into equal-sized hexagonal cells, labeled using the coordinate system shown in Figure 1 (the figures are reprinted from the proceedings). Each module is hexagonal, and is the same size as a cell of the plain. It always occupies one cell. Each module knows at all times its location, and which of its neighboring cells are occupied by modules. The modules move in synchronous steps with the

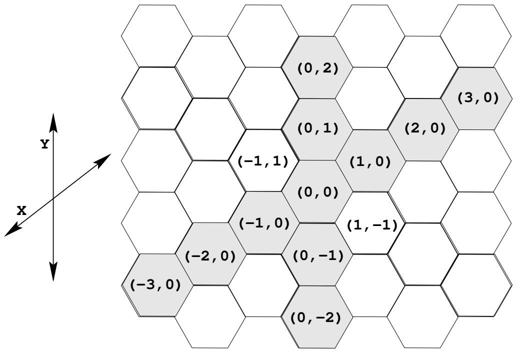

Figure 1: Coordinate System

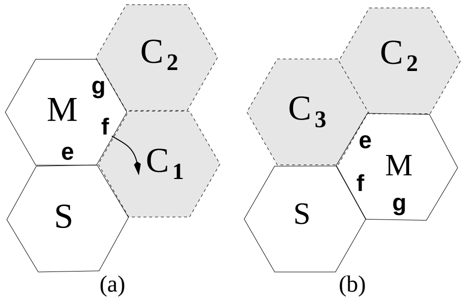

Figure 2: Legal Movements

following restrictions: In a round, only one module tries to move into a particular cell. A module M can move only to an adjacent cell C_1, and only if the cell is currently empty, and if: (1) module M has a neighbor S that does not move in the round and S is also adjacent to cell C_1, and (2) the neighboring cell to M on the other side of C_1 from S, C_2, is empty. See Figure 2. The problem is to design a distributed algorithm (a collection of identical programs, one for each module) that will cause the modules to move from an initial configuration I to a goal configuration G. It is assumed that the modules know the number of modules n, and G. Several algorithms and lower bounds are presented. The complexity that is measured is the number of rounds, and the number of module movements. The paper starts by presenting an algorithm for the case where I and G are straight line chains, and I and G intersect in one cell. The algorithm is described for the case where I and G are collinear, and then extended to the general case. The behavior of the algorithm is illustrated in Figure 3, for the case of 4 modules. The system reconfigures itself in 9 rounds using 15 module movements. The module (number 4 in the figure) in the intersection of I and G does not move. The other modules move downward, half of them rotate counter clockwise along the left of the chain, the other half rotate clockwise along the right of the chain. It is not hard to see that the algorithm is asymptotically optimal in both measures. It takes a linear number of rounds, and a quadratic number of module movements.

Something to consider is that the algorithms are distributed in a somewhat weak form. Each module executes a set of local instructions that use local parameters (its position, and the state of neighboring cells). However, the modules do not communicate with each other, and at least in the chain scenarios considered, each module can deduce what I is, and hence at all times can deduce the configuration of the system. So in principle, any (global) algorithm that gets from I to G could be executed locally by the modules.

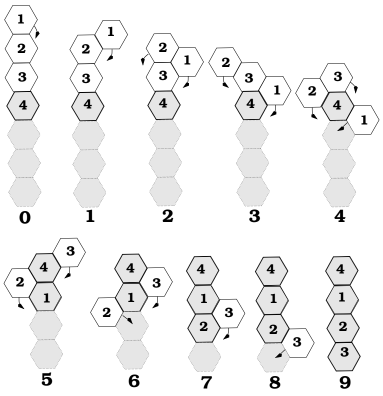

Figure 3: Example of execution for 4 modules

News

Gil Neiger was elected Steering-Committee Chair, taking over Michael Merritt that completed his term.

Next year PODC will be August 26-29, in Newport, Rhode Island with Nir Shavit as Program Committee Chair. Please check the website http://www.podc.org/podc2001/ and do not miss the 20th anniversary of PODC!

Acknowledgments

Many thanks to Hagit Attiya, Joanne Kaliontzis, and Alex Shvartsman for comments on an earlier version of this paper. Thanks also to Jim Anderson and Ajay Kshemkalyani for helping with information on PODC'00.

References

[1] "Contributions of Theoretical Computer Science" The SIGACT Long Range Planning Committee, Amihood Amir, Michael Loui, John Savage, Carl Smith (chair), Dec. 1997. http://sigact.csci.unt.edu/sigact/longrange/contributions.html

[2] José Echegaray, *Academia de Ciencias,* extract from formal session of March 12, 1916, Spain.

[3] Oded Goldreich and Avi Wigderson, "Theory of Computing: A Scientific Perspective," *ACM Computing Surveys,* **28A**(4), December 1996, http://www.acm.org/surveys/1996/GWtoc-sp/.

[4] Roger G. Newton, *The Truth of Science,* Harvard University Press, 2000.

[5] "Challenges for Theory of Computing" Report of an NSF-Sponsored Workshop on Research in Theoretical Computer Science, Chicago, April 1999. http://www.cs.buffalo.edu/pub/WWW/faculty/selman/report/