

Final Remarks

The book concerns main principles and features of distributed systems, composed of computers internally controlled, that is, the so-called von Neumann's architectures (von Neumann 1945). Presenting basic idea behind activity of such machine in Chap. 1, details of its physical construction have been omitted. All the nowadays realizations of the instruction execution cycle are electronic, but one may imagine, for instance, mechanical realization, involving cogwheels, levers, etc., with program and data supplied on punched cards as memory—as in analytical machine built by Babbage (1864). The principle is similar, but execution duration of arithmetic and other operations in case of present-day electronic processors is several billions times shorter. Along with physical realizations, the von Neumann's machine is the human concept. However this is not the only principle of activity of devices performing computation operations—arithmetical, logical and control. Inspiration for searching for different principles, called computational paradigms, are natural phenomena. Nowadays, the intensive research is being carried out on mechanisms of biological particles activity, in particular information flow among molecular structures and on making usage of their multitude for mass of simple operations, executed in parallel. Some computational architectures have already been created, that operate on such principle, though not yet carried into widespread, public usage. Such constructs are capable of solving tasks exceeding (owing to time complexity) capability of classical sequential processors, even interconnected into distributed systems. Computations in such experimental architectures, variously named: molecular, biomolecular, biochemical or DNA computing, have so far efficiently applied to solving some combinatorial problems, the so-called computationally hard, like finding Hamiltonian cycle in a graph (closed path crossing every vertex exactly ones), verification of satisfiability of propositional formulae of very many variables, modelling of the so-called self-organizing systems (e.g. evolution of organisms, weather phenomena, etc.) and many more problems of high complexity. Another natural phenomenon inspiring non classic work principle of computational devices comes from quantum mechanics. The information unit is there the so-called quantum bit, *qubit* in short, mathematically represented as a linear combination of two states of polarization of a quantum object, e.g. photon, where the coefficients of the combination are the so-called amplitudes of probability

of a certain state occurrence. Thus, the qubit is, in a sense, a superposition (or composition) of two binary states, whose sequence may be simultaneously in many states, due to the so-called quantum entanglement. This phenomenon has been used for simultaneous execution of mass computations. Each computation is a sequence of quantum states and a state is a sequence of n qubits, where n is their number which the so-called quantum computer can contain. The computer may be **simultaneously** in 2^n states, whereas its classic, sequential counterpart—in one only from among 2^n states **at a time**. The quantum architecture (Deutsch 1985), still in experimental phase, alike the molecular one, is capable of overcoming the complexity barriers of classic processors, also in distributed systems. Typical problems of high complexity come from number theory, especially their application in cryptography. Both aforesaid non classic paradigms of computation inspired by natural phenomena, along with usage of natural objects—biochemical and quantum—in their physical realization (hardware), are expected to be applied in the massively parallel computing. The principles belong to a research domain called natural computation. This domain encompasses also imitation of phenomena and processes encountered in the nature as problem solving techniques, but by creating algorithmic models executed in the classic computer system, or in grids of many simple and cheap sequential processors. Examples are neural computation, cellular automata, swarm intelligence (imitating behaviour of a “swarm” of objects striving to a joint objective), evolutionary computation (imitating Darwinian evolution), membrane computing (imitating of cell membrane functions), cf. Rozenberg et al. (2012). These and other models of computation—a result of observation of natural phenomena—are being devised for special tasks, but there are also research attempts to create the Universal Quantum Turing Machine (Turing 1937; Deutsch 1985; Penrose 1989), a system for all the effective computations, like a classic computer with arbitrarily large memory needed for a given class of problems. One may also mention other systems, where parallelization and distribution of great number of cooperating elements takes place, like the cyber-physical systems (Lamnabhi-Lagarrigue et al. 2014) (for control of technical devices) or multi agent systems (Russell and Norvig 2010). All them are dynamically developing scientific and engineering explorations. This book was concerned with distributed systems composed of computers of the classic von Neuman’s architecture, leaving to the interested reader a closer familiarity with endeavours leading to practical application of the above mentioned contemporary computational principles.

References

- Babbage, C. (1864). *The life of a philosopher*. London.
- Deutsch, D. (1985). Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London, A400*, 97–117.
- Lamnabhi-Lagarrigue, F., Di Benedetto, M. D., & Schoitsch, E. (2014). Introduction to the special theme cyber-physical systems. *Ercim News*, 94, 6–7.

- Penrose, R. (1989). *The emperor's new mind*. UK: Oxford University Press.
- Rozenberg, G., Baeck, T., & Kok, J. (2012). *Handbook of natural computing*. Heidelberg.
- Russell, S., & Norvig, P. (2010). *Artificial intelligence. A modern approach*. Englewood Cliffs, NJ: Prentice Hall.
- Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society Series 2*, 42, 230–265.
- von Neumann, J. (1945). *First draft of a report on the EDVAC*.

Bibliography

- Akl, S. G. (1997). *Parallel computation: Models and methods*. USA: Prentice Hall.
- Bedrouni, A., Mittu, R., Abdeslem Boukhtouta, A., & Berger, J. (2009). *Distributed intelligent systems. A coordination perspective*. Berlin: Springer.
- Belapurkar, A., Chakrabarti, A., Ponnappalli, H., Varadarajan, N., Padmanabhuni, S., & Sundarajan, S. (2009). *Distributed systems security. Issues, processes and solutions*. USA: Wiley.
- Ben-Ari, M. (1996). *Podstawy programowania współbieżnego i rozproszonego*. Warszawa: Wydawnictwa Naukowo-Techniczne 1996 (Polish translation of 1990).
- Boykin, J., Kirschen, D., Langerman, A., & Loverso, S. (1993). *Programming under Mach*. Reading, MA: Addison-Wesley.
- Broy, M., & Stølen, K. (2001). *Abracadabra protocol*. In: *Specification and development of interactive systems. Monographs in Computer Science*. NY: Springer.
- Comer, D. E. (1995). *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture*. USA: Prentice Hall.
- Coulouris, G., Dollimore, J., & Kindberg, T. (1998). *Systemy rozproszone, podstawy i projektowanie*. Warszawa: WNT (Polish translation of 1994).
- Czaja, L. (1973). Niektóre aspekty implementacji ALGOL-u 60 dla maszyn ZAM/21 ALFA. In *Materiały Sympozjum XV-rocznicy Instytutu Maszyn Matematycznych i Roku Nauki Polskiej*. Warszawa (Some aspects of implementation of ALGOL 60, in Polish).
- Czaja, L., & Szorc, P. (1967). Implementation of ALGOL for ZAM computers. *Algorytmy*, 4(7), 91–111.
- Dahl, O.-J., Dijkstra, E. W., & Hoare, C. A. R. (1972). *Structured programming*. London and New York: Academic Press.
- Dasgupta, P., LeBlanc, R. J., Jr., Ahamad, M., & Ramachandran, U. (1991). The clouds distributed operating systems. *IEEE Computer*, 24(11), 34–44.
- Davies, D. W., Holler, E., Jensen, E. D., Kimbleton, S. R., Lampson, B. W., LeLann, G., et al. (1983). In B. W. Lampson, M. Paul, & H. J. Siebert (Eds.), *Distributed systems—Architecture and implementation, an advanced course*. Berlin: Springer.
- Gabassi, M., & Dupouy, B. (1995). *Przetwarzanie rozproszone w systemie UNIX*. Warszawa: Lupus (Distributed processing in UNIX, in Polish).
- Gien, M. (1995). Evolution of the CHORUS open microkernel architecture: The STREAM project. In *FTDCS '95 Proceedings of the 5th IEEE Workshop on Future Trends of Distributed Computing Systems*. USA: IEEE Computer Society.
- Gościński, A. (1991). *Distributed operating systems, the logical design*. USA: Addison Wesley.
- Haddad, S., Kordon, F., Pautet, L., & Petrucci, L. (2011). *Distributed systems, design and algorithms*. USA: Wiley.
- Holenderski, L., & Szalas, A. (1988). Propositional description of finite cause-effect structures. *Information Processing Letters*, 27, 111–117.

- Kshemkalyani, A. D., & Singhal, M. (2011). *Distributed computing: Principles, algorithms, and systems*. Cambridge: Cambridge University Press.
- Milner, R. (1989). *Communication and concurrency*. In C. A. R. Hoare (Series Ed.). USA: Prentice-Hall.
- Shapiro, E. Y. (1987). *Concurrent PROLOG: Collected papers*. Cambridge MA, USA: MIT Press.
- Sinha, P. K. (1997). *Distributed operating systems—Concepts and design*. Piscataway: IEEE Press.
- Sorin, D. J., Hill, M. D., & Wood, D. A. (2011). A premier on memory consistency and cache coherence. In M. D. Hill (Ed.), *Synthesis lectures on computer architecture*. USA: Morgan and Calypool Publishers.
- Spector, A., & Gifford, D. (1984). The space shuttle primary computer system. *Communications of the ACM* 27(i), 874–900.
- Sportack, M. (1998). *Networking essentials unleashed* (1st ed.). USA: Sams Publishing.
- Starke, P. (1987). *Sieci petri*. Warsaw: PWN (Polish translation of the author's book in German: *Petri Netze*, 1980).
- Stevens, W. R. (1999). *UNIX Programowanie usług sieciowych*. Warszawa: WNT.
- SYSTEM AUTOMATYCZNEGO KODOWANIA SAKO. (1961). *Prace Zakładu Aparatów Matematycznych PAN*. Warszawa: PAN (SAKO—A system of automatic coding, in Polish).
- Tanenbaum, A. S., Van Renesse, R., Van Staveren, H., Sharp, G. J., Mullender, S. J., Jansen, J., et al. (1990). Experience with the amoeba distributed operating system. *Communication of the ACM*, 33(12), 46–63.
- Tanenbaum, A. S. (1997). *Rozproszone systemy operacyjne*. Warsaw: PWN (Polish translation of 1995).
- Tanenbaum, A. S. (2004). *Sieci komputerowe*. Gliwice: Wydawnictwo Helion (Polish translation of 2003).
- Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer networks*. USA: Prentice Hall.
- Tanenbaum, A. S., & van Steen, M. (2002). *Distributed systems: Principles and paradigms* (2nd ed.) UK: Pearson.
- Tanenbaum, A. S., & van Steen, M. (2003). *Computer networks*. USA: Prentice Hall.
- Tanenbaum, A. S., & van Steen, M. (2006). *Systemy rozproszone, zasady i paradygmaty*. Warszawa: WNT (Polish translation of 2002).
- Wierzbicki, A. (2010). *Trust and fairness in open, distributed systems*. Berlin: Springer.
- Wirth, N. (1988). *Programming in modula-2* (4th ed.), *Texts and Monographs in Computer Science*. Berlin: Springer.
- Wulf, W. A., & Bell, C. G. (1972). *C.mmp—A multi-mini processor*. In *Proceedings of AFIPS, FJCC* (Vol. 41, pp. 765–777). Montvale, N.J.: AFIPS-Press.

Index

A

Accumulator, 1, 3
ADA, 20, 125–127, 142, 143, 146
Algol, 142
Algorithm, 2, 8, 35–37, 43, 53, 79, 122, 162, 172–175, 177–179, 222, 223, 225
 bully, 172–174, 177
 ring, 79, 172, 177–179
Alternating Bit Protocol, 160, 227
ALU, 121, 122
Amoeba, 53, 125
Architecture, 1, 43, 50, 52, 59, 60, 62, 64, 120–123, 142, 188
 von Neuman, 1, 2
Arpanet, 49, 55
Asynchronous Transfer Mode (ATM), 120, 130, 133, 137, 138

B

Berkeley method, 90, 91
Binary relations, 87, 241
Binder, 145
BITNET, 49, 50
Blocking, 68, 69, 72, 74, 75, 127–129
Broadcast, 86, 91, 101, 120, 127, 130, 131, 166, 197, 205, 221–223
Busy waiting, 116
Byzantine generals, 165, 169

C

Causal broadcast, 223, 224
Causal consistency, 217, 218, 220, 223
Causal order, 217, 219, 220, 225
Centralized system, 51
c/e structures, 227, 228, 246–249

Channel, 4, 6, 12, 20, 35, 120, 122, 123, 127, 129, 163, 227–229
Checkpoint, 160
Chorus, 53, 125
Circuit switching, 49, 125
Clock, 3, 43, 52, 56, 85–91, 96–99, 102, 142, 195, 202, 229
Clouds, 53, 59
Common Object Request Broker Architecture (CORBA), 59, 144
Communicating Sequential Processes (CSP), 19, 125, 127
Communication, 2, 4, 12, 19, 20, 35–38, 42, 50–53, 56, 57, 59, 61, 63, 64, 74, 78, 90, 94, 96, 99, 116, 119–133, 137, 141, 142, 146, 157, 165, 166, 173, 189, 221
 asynchronous, 19, 37–39, 120, 121, 124, 126–131, 137
 connection oriented, 120
 connectionless, 120, 121, 127, 130, 131, 137, 138
 group, 19, 53, 94, 126, 127, 131, 132, 157, 165, 221
 synchronous, 19, 20, 35–38, 42, 120, 121, 124–128, 131, 137
Computer network, 2, 50, 188
Computer systems classification, 47
Concurrency, 56, 60, 61, 65, 66, 71, 74, 190, 195, 196, 200, 212, 220
Concurrent Pascal, 125, 143, 146
Concurrent Prolog, 125
Consensus, 164, 165, 172
Coordinator, 162, 166, 170, 172–175, 177–179
Cristian's method, 89–91, 94, 142
Critical section, 8, 11, 12, 67, 78, 111, 143
Crossbar, 61, 62

D

D’Alambert criterion, 159
 Deadlock, 65, 70, 73–75, 119, 162, 215
 Discrete random variable, 158, 245
 Distributed Computing Environment (DCE), 53
 Distributed Shared Memory (DSM), 60, 187–190, 194–196, 215–218, 220, 221
 Distributed system, 1, 2, 49, 50, 52, 55, 56, 60, 74, 85, 87, 96, 108, 119, 131, 141–143, 145, 157, 161, 187–189, 221

E

Election, 91, 162, 170, 173–175, 177–179, 184
 European Academic and Research Network (EARN), 50
 Event, 65, 73, 74, 85–87, 96, 97, 99, 100, 102, 116, 124, 132, 161, 162, 212, 215, 216, 218, 223, 244, 245
 Exceptions, 124, 141, 146
 Expected value, 158, 159, 244–246

F

Failure, 55, 66, 78, 90, 101, 112, 146, 157–162, 173, 174, 177, 178
 Fast-read algorithm, 222
 Fast-write algorithm, 221, 222
 Fault tolerance, 55, 56, 61, 157, 165, 172
 FIFO, 38, 131–133, 203, 222, 223
 Flynn’s taxonomy, 47

G

Global timestamp, 100–102, 108, 109
 Greatest common divisor, 20, 38

I

Independent events, 158, 244
 Inhibitor arcs, 116, 117
 Instruction execution cycle, 1–3, 43
 Instruction register, 1, 3
 Instruction set, 3, 19, 37
 Interleaving, 4, 190, 191, 194, 195, 199, 200, 202, 206, 210, 212, 220
 Internal control, 1
 Internet, 49, 50, 54, 61, 86, 94, 131, 188

J

Java, 20, 59, 123, 125, 142, 143, 145, 146

L

LINDA, 37, 125, 126, 129
 Linear order, 87, 96, 100
 Lock, 66, 67, 70
 Logarithmic switch, 62, 63

Logical clock, 87, 94, 97, 99, 100
 Logical time, 94
 LOGLAN, 125, 146

M

Mach, 53, 125
 Marshalling, 120, 121, 123, 133, 141, 142, 189
 Memory coherence, 199, 200, 205, 206, 208–210, 212–214
 Memory consistency, 125, 187–190, 195–197, 199, 205, 212, 215, 217, 220, 221
 causal, 217, 219, 220, 225
 Pipelined Random Access Memory (PRAM), 219, 220, 225
 sequential, 195, 197, 199, 200, 202, 205, 212, 217, 225
 strict, 187, 195, 200, 217, 220
 Mertens theorem, 159, 242–244
 Meta-program, 2
 Modula-2, 125, 127, 146
 Multicast, 120, 126, 127, 131, 157
 Multiprogramming, 4
 Mutual exclusion, 8, 19, 65–67, 75, 78–80, 83, 100–102, 109, 112, 116, 120, 143, 144, 162, 215

N

NASK, 50
 Network Time Protocol (NTP), 91, 94, 95
 Non Uniform Memory Access (NORMA), 64
 NO Remote Memory Access (NUMA), 62, 63

O

OCCAM, 20, 124–127
 Openness, 56, 59, 61
 OSI/RM, 120, 133, 134, 137, 138

P

Partial order, 96, 100, 196, 212, 247
 Petri nets, 116, 190, 227, 246, 248, 249
 Physical time, 85, 87, 89, 90
 PLEARN, 50
 Port, 121, 126, 127, 145
 Precedence, 85, 94, 96, 99
 Probability, 158, 159, 164, 196, 244–246
 Process, 3, 5, 8, 12, 19, 42, 52, 65–67, 70, 75, 76, 78, 79, 96–102, 108, 120, 121, 124–129, 131, 132, 142, 143, 145, 161, 162, 167, 173, 174, 177, 195, 197, 199, 205, 212, 217–219, 221
 Processor, 1–4, 43, 44, 54, 61, 63, 65, 85, 87, 96, 161, 197, 199, 205
 Protective zone, 67, 78, 79, 101–103, 108–112, 116

Protocol, 52, 63, 94, 102, 108–113, 116,
120–123, 125, 131, 133, 137, 138, 144,
146, 227, 228, 249

R

Random variable, 159, 244, 245
Remote Method Invocation (RMI), 58, 127,
141, 143, 145
Remote Procedure Call (RPC), 53, 59, 127,
141–147, 187, 189
Resource, 1, 8, 11, 56–59, 66, 67, 69, 70, 75,
78, 102, 108, 119, 142–144

S

Scalability, 56, 61, 131
Semaphore, 8, 11, 12, 66, 67, 116, 120
Sequential machine, 2
Series of real numbers, 242
Socket, 53, 121, 126, 127, 145
Starvation, 65, 75, 112, 119, 162, 215
Stub, 144–146
Synchronization, 2, 19, 57, 58, 65, 74, 78,
87–91, 94, 95, 119, 121, 138, 142, 161,
162, 173, 189

T

Thread, 50, 124–126, 147
Time compensation, 94, 99, 100, 102
Timeout, 74, 128, 131, 138, 146
Timesharing, 7
Timestamp, 99, 100, 102, 108, 110, 112, 160
Timestamps vectors, 111
Token ring, 66, 78, 79, 82, 83
Transactions, 58, 65–67, 69–71, 73, 74, 87,
119, 143, 164
Transparency, 56, 60, 141, 145, 160, 200, 221
Two Army problem, 162, 164, 165

U

Uniform Memory Access (UMA), 62, 63
UNIX 4BSD, 53
Unlock, 67, 70, 72
Unmarshalling, 120, 123, 133, 141, 142

V

Vector systems, 43

W

Wait graph, 73, 74
Weak precedence, 96