IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 4, April 2017

An Intelligent Distributed Embedded System Architecture for Real-Time traffic control system

Seketa Olika Abdena[1], Dr Vasu Pinnti[2]

Dean, College of Engineering, Wolaita Sodo University, Sodo, Ethiopia[1]

Associate Professor, ECE Dept, College of Engineering, Wolaita Sodo University, Sodo, Ethiopia[2]

Abstract: Recent advances in distributed computing, embedded systems, communication, and sensor technology are pushing the development of many new applications. This trend is especially evident in pervasive Real-time functionality in life critical situations. Current high-performance embedded cameras combine video sensing, video processing, and communication within a single device, distributed data processing with load balancing provides local safety measures if system fails and data replication across several physical systems provides Capability to continue work even if several systems fail and also resume with previous settings and data on restart and Watchdog functionality. In this paper we focused these issues and developed intelligent embedded system modules which are distributed, well organized and modeled into an intelligent system architecture to control real-time traffic system, in this each module requests data from system as input, processes it independently and provides data as output, this offers data gathering from sensors – Actuating external actuators – Data processing – Service (logging etc.),Vehicle detection in video using Number plate location in video ,Car detection using microphone arrays, magnetic sensors and other sensors and Merging data for complex vehicle detection, calling emergency services, Controlling traffic lights.

Keywords: embedded cameras, data replication, watch dog, vehicle detection, sensors, actuators.

I. INTRODUCTION

Scientific breakthroughs enable new technology whereas at the same time, technology makes it into transformative applications (see, for example, the Internet and wireless communication systems). It is evident that the advancement of smart devices with impressive sensing, computing and control capabilities makes it possible for our cities, transportation systems, factories and living environments to become more intelligent, energy-efficient, safe and secure. We are now witnessing an explosion in networked systems: everything is connected and massive amounts of devices are required to be coordinated. However, the overall system is distributed and a service should be delivered cooperatively, rather than by a unique provider that knows and owns all data. Many researchers and practitioners are interested in understanding the complexity and possible solution approaches, since distributed systems apply to different areas and applications and are anticipated to play a central role in the near future.

In applications that can be characterized as distributed systems, dynamical systems are controlled or communicate and interact with other dynamical systems via a network of sensors and actuators transmitting and receiving information over a digital communication network.

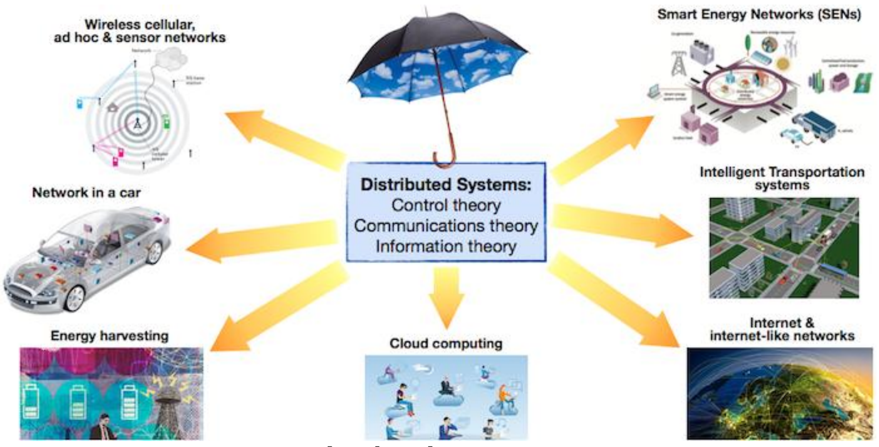
Wireless cellular, ad hoc & sensor networks
Network in a car
Energy harvesting
Cloud computing
Distributed Systems:
Control theory
Communications theory
Information theory
Smart Energy Networks (SENs)
Intelligent Transportation systems
Internet & internet-like networks

Figure 1: distributed system environment

Real communication channels suffer from various problems (such as limited capacity, bit errors, erasures and random delays) that can affect the stability and/or performance of the whole system. Given the multidisciplinary nature of distributed systems, many studies are necessarily restricted to contributions that solve problems partially, ignoring important aspects of such systems due to modular approaches to the problem. In order to be able to deal with distributed systems, we need to better understand information, and the interplay between information and decisions. However, communication and control/decision making have traditionally been treated separately whereas recent emerging applications necessitate a more holistic approach, since communication and control are no longer independent.

The main objective of this line or research is to develop the framework on how to treat the interplay between communications and control to solve distributed control and optimization problems arising in distributed embedded systems and in real-time traffic control system.

Smart cameras, one example of this innovation, are equipped with a high-performance onboard computing and communication infrastructure, combining video sensing, processing, and communications in a single embedded device. By providing access to many views through cooperation among individual cameras, networks of embedded cameras can potentially support more complex and challenging applications—including smart rooms, surveillance, tracking, and motion analysis— than a single camera. From an external perspective, users understand a system as a collection of components that correspond to system behaviors to fulfill requirements. As such, components are well understood abstractions that encapsulate domain knowledge and denote a system's behavioral units.

We designed our smart camera as a fully embedded system, focusing on power consumption, QoS management, and limited resources. The camera is a scalable, embedded, high-performance, multiprocessor platform consisting of a network processor and a variable number of digital signal processors (DSPs). Using the imple mented software framework, our embedded cameras offer system-level services such as dynamic load distribution and task reconfiguration. In addition, we combined several smart cameras to form a distributed embedded surveillance system that supports cooperation and communication among cameras. Although smart cameras have various applications we focus on traffic surveillance, which imposes demanding video-processing and compression-algorithm requirements on the camera's hardware and software. The "From Analog to Digital Smart Cameras" provides a discussion of the evolution of smart camera surveillance systems.

To meet the requirements of an innovative traffic-surveillance system—that is, the ability to autonomously monitor the traffic along a highway section, computing traffic statistics, delivering a compressed live video to the monitoring station, and performing high-level video analysis such as detecting a wrong-way driver or an accident— the distributed surveillance architecture must be scalable and flexible.

II. HARDWARE ARCHITECTURE

Our work is centered on the notion of a feature: Smart cameras are a core component of future traffic- surveillance systems. High-level computing and communication capabilities in the smart cameras' embedded platform increase surveillance system functionality, flexibility, and scalability. For example, basic video compression (MPEG-4 advanced simple profile) and video processing (such as stationary vehicle detection) require an overall computation performance of about 10 billion instructions per second (GIPS) and a transfer rate of about 1 megabyte per second (Mbps). Additional onboard video analysis increases the computation and communication requirements. We chose a commercial off-the-shelf software/ hardware architecture to support fast prototype development and achieve flexibility and performance at a reasonable price.

Figure 2 depicts the smart camera's three main parts: the sensing unit, processing unit, and communication unit.

A highly dynamic, monochrome complementary metal-oxide semiconductor (CMOS) image sensor is the sensing unit's heart. The sensing unit delivers images with VGA resolution at up to 30 frames per second, transferring the captured images via a first-in, first-out (FIFO) memory to the processing unit.
The processing unit consists of DSPs, which offer a good compromise between performance, power consumption, and flexibility. Up to 10 Texas Instruments TMS320C64x DSPs can deliver an aggregate performance of up to 80 GIPS while keeping the power consumption low. By using an adequate number of DSPs, we adapt this scalable architecture's computing performance to the requirements of the real-time video analysis and compression tasks targeted for the smart camera.

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 4, April 2017

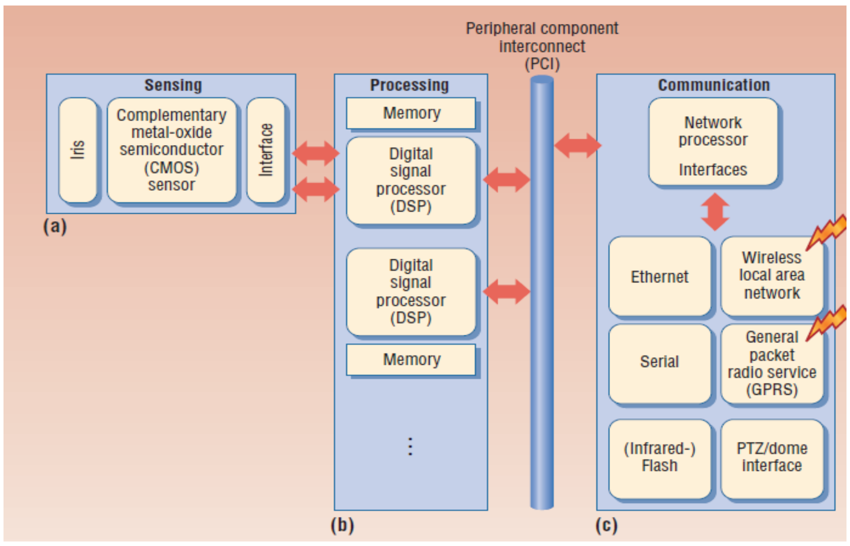
Peripheral component interconnect (PCI)
Sensing
Iris
Complementary metal-oxide semiconductor (CMOS) sensor
Interface
(a)
Processing
Memory
Digital signal processor (DSP)
Digital signal processor (DSP)
Memory
(b)
Communication
Network processor
Interfaces
Ethernet
Wireless local area network
Serial
General packet radio service (GPRS)
(Infrared-) Flash
PTZ/dome interface
(c)

Figure 2: Embedded system hardware architecture

Local peripheral components interconnect (PCI) bus couples the DSPs and connect them to the network processor (Intel XScale IXP425). The network processor establishes the connection between the processing and communication units and controls internal and external communication. We chose the XScale processor because of its communication capabilities, low power consumption, and integration of various interfaces. Internal communication between the DSPs or between the DSPs and the network processor occurs through the PCI bus. The communication unit currently supports two interfaces for IP-based external communication: wired Ethernet and wireless Global System for Mobile Communications/ general packet radio service (GSM/GPRS).

III. SOFTWARE ARCHITECHURE

We designed the smart camera's software architecture for flexibility and re-configurability. It consists of several layers, which we group into two frameworks:

- The DSP framework, running on every DSP in the system,
- The SmartCam framework, running on the network processor.

We based the architecture on the abstraction that the application logic runs on the network processor and loads and unloads the actual analysis algorithms onto the DSPs as needed. Figure 3 is an overview of our distributed embedded software architecture.

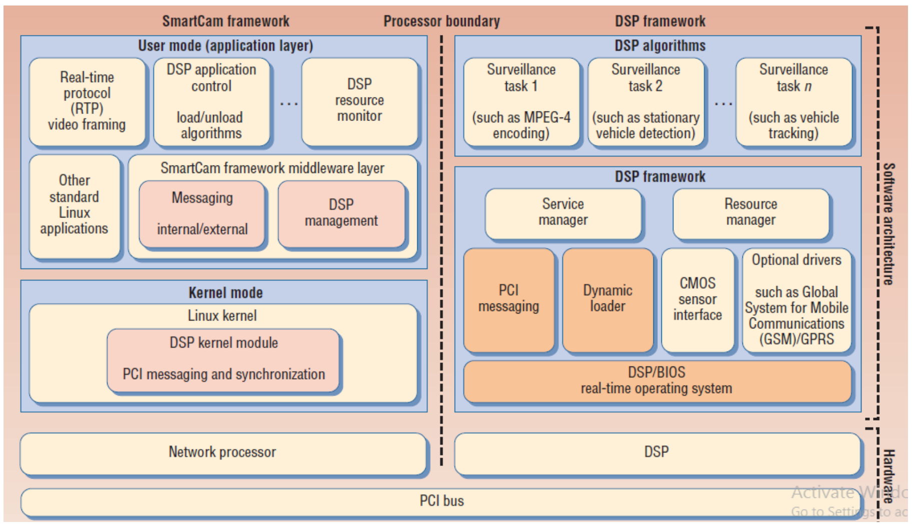
SmartCam framework
Processor boundary
DSP framework
User mode (application layer)
Real-time protocol (RTP) video framing
DSP application control
load/unload algorithms
DSP resource monitor
Other standard Linux applications
SmartCam framework middleware layer
Messaging
internal/external
DSP management
Kernel mode
Linux kernel
DSP kernel module
PCI messaging and synchronization
DSP algorithms
Surveillance task 1 (such as MPEG-4 encoding)
Surveillance task 2 (such as stationary vehicle detection)
Surveillance task n (such as vehicle tracking)
DSP framework
Service manager
Resource manager
PCI messaging
Dynamic loader
CMOS sensor interface
Optional drivers such as Global System for Mobile Communications (GSM)/GPRS
DSP/BIOS real-time operating system
Software architecture
Network processor
DSP
PCI bus
Hardware

Figure 3: distributed embedded software architecture.

DSP framework the DSP framework's main purposes are provide an abstraction of the hardware and communication channels, support dynamic loading and unloading of application tasks, and manage the DSP's on-chip and off-chip resources. Of course, only the DSP connected to the image sensor needs the sensor interface module. Algorithms on different DSPs use the service-management facilities to dynamically establish connections to each other. This is important because the dynamic loader module can load and unload all algorithms at runtime. Actually, only the shaded modules on the left in the DSP framework in Figure 3 must be available at startup. The system can dynamically load all other components at runtime. We built the DSP framework on Texas Instruments' DSP/BIOS operating system.

SmartCam framework. The SmartCam framework, illustrated in Figure 3, serves two main purposes:
• It provides an abstraction of the DSPs to ensure the application layer's platform independence.
• The application layer uses the provided communication methods—that is, internal messaging to the DSPs and external IP-based communication—to exchange information or offer data-relay services for DSP algorithms.
Modules of this part of the software architecture supports application development by providing high-level interfaces to DSP algorithms and the DSP framework's functions. The XScale processor runs standard Linux, easing application development and providing many modules for seamless internal and external communication on our smart camera. The only customization of the Linux kernel is the DSP kernel module, which the processor uses to establish the connection to the DSPs via the PCI bus.

Distributed system architecture
Developers can use our embedded smart cameras to implement a distributed intelligent video-surveillance (IVS) system consisting of dozens of cameras.
To design a scalable distributed architecture, we partition an IVS into distributed logical groups of typically collocated smart cameras, or surveillance clusters. It isn't necessary to assign most traffic-surveillance tasks—such as accident detection, vehicle classification, and computation of traffic statistics—to a specific camera. Our IVS architecture only requires an assignment to a specific cluster. The IVS then dynamically and autonomously maps surveillance tasks onto individual cameras depending on the cameras' available resources and the system's current state.
We implement this dynamic reconfiguration of tasks onto cameras using a mobile agent system (MAS) built atop the SmartCam framework. In our MAS, agents represent surveillance tasks that can migrate dynamically between the cluster's cameras. Significant changes in the observed environment or in the available resources trigger a task migration. A dedicated agent in our MAS uses a complex cost function to compute the optimal assignment of tasks to cameras.3 QoS is a major concern in a distributed IVS. In video-based surveillance, typical QoS parameters include frame rate, transfer delay, image resolution, and video-compression rate.4 The surveillance tasks might also offer several QoS levels. Furthermore, the provided QoS levels can change over time due to user interactions or changes in the monitored environment. Thus, novel IVS systems must include dedicated QoS management mechanisms.
Power awareness is another important design aspect in distributed IVS systems because they must deliver high-level QoS while using embedded devices that are partly solar or battery powered. Our smart camera supports combined power and QoS management (PoQoS)5 for distributed IVS systems. PoQoS dynamically configures the power and QoS level of the camera's hardware and software to adapt to user requests and changes in the environment.

Distributed software
We use mobile agents to support the development of our distributed surveillance system, consisting of loosely coupled smart cameras. Mobile agents are most suitable for this distributed application because we can encapsulate each surveillance task within a mobile agent, which can then migrate between cameras. An MAS supports autonomous operation of the surveillance tasks. Moreover, this approach is highly scalable and flexible. In typical MAS, individual agents migrate between hosts, which execute the surveillance tasks. We've thus extended the mobile agent to keep a DSP binary that can be downloaded from the SmartCam's network processor to one of the DSPs.
The DSP agents have three parts:
• A module that manages the agent's integration into its environment,
• A DSP binary representing the agent's functionality and
• An optional set of intermediate data.
System core services
● Configuration subsystem
● Data storage subsystem
● Inter-core network subsystem
● Watchdog subsystem
● Data replication subsystem
● Data retrieval subsystem

Configuration subsystem that keeps configuration consistent across physical systems and system restarts and also distributes external configuration changes for the active and further sessions

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 4, April 2017

Data storage subsystem stores all relevant tracking data on each physical system separately. Initially modules send data to local core only and data distribution controlled by data replication Subsystem
Inter-core network subsystem performs message distribution and routing between physical systems, System routing information and System load information
Watchdog subsystem watches heart-beat signals from all cores and Takes necessary action to restart failed system parts and/or sends notifications to appropriate personnel
In **data replication subsystem** all data packets are replicated to at least N physical systems. After receiving data packet each core determines if it should be stored and/or sent for further replication or dropped completely. Each packet contains information of cores containing it. Replication targets determined by reported system load If cores containing some data fail, watchdog initiates additional replication of the involved data
In **Data retrieval subsystem**, modules request data from local core and Local core determines which of system cores contains the freshest data of type and source requested and connects requesting module to the appropriate core Either one packet or data stream can be Requested.

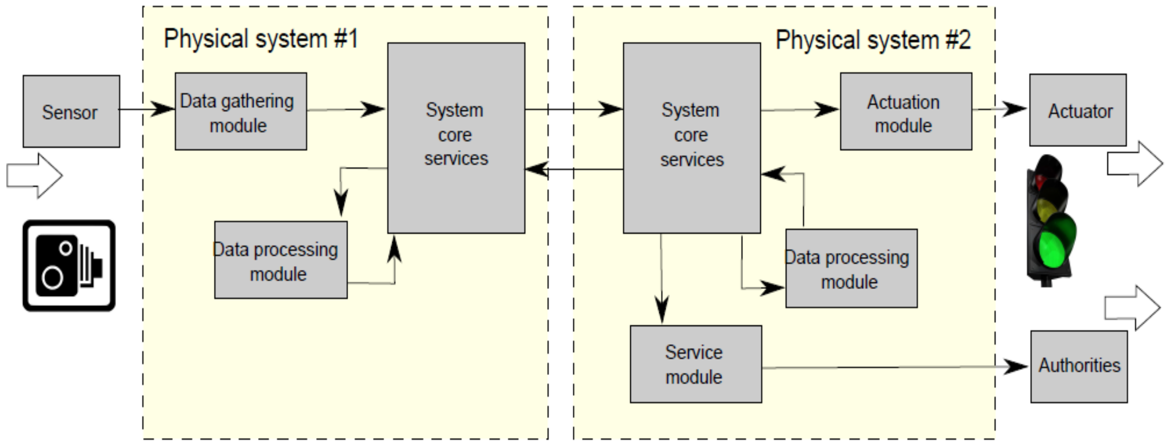

Figure 4: distributed embedded system architecture

IV. PHYSICAL TESTBED IMPLEMENTATION

Physical testbed implementation uses 3 Mini-ITX personal computers (i5, 8GB RAM), 8 Raspberry PI embedded computers, Wireless router, Sensors (cameras, magnetic sensors, microphone arrays etc.) Traffic actuators (simulated, because of birocratic reasons) Developing the SmartCam prototype has given us insight that might also be applicable to other distributed embedded systems. Our experiences show that the keys to successful deployment of smart cameras are the integration of sensing, computing, and communication in a small, power-aware embedded device; the availability of high-level image/video processing algorithms or libraries for the embedded target processors (the DSPs); a lightweight software framework supporting glueless intra- and intercamera communication; and the availability of various system-level services such as task mapping and QoS adaptation to allow autonomous and dynamic operation of the overall multicamera system.

Embedded smart cameras could potentially be deployed in applications such as smart environments, intelligent infrastructures, and pervasive computing. Augmenting the smart cameras with additional sensors could transform them into a high-performance multisensory system. By combining visual, acoustic, tactile, or location-based information, the smart cameras become more sensitive and can deliver more accurate results, making them even more widely applicable.

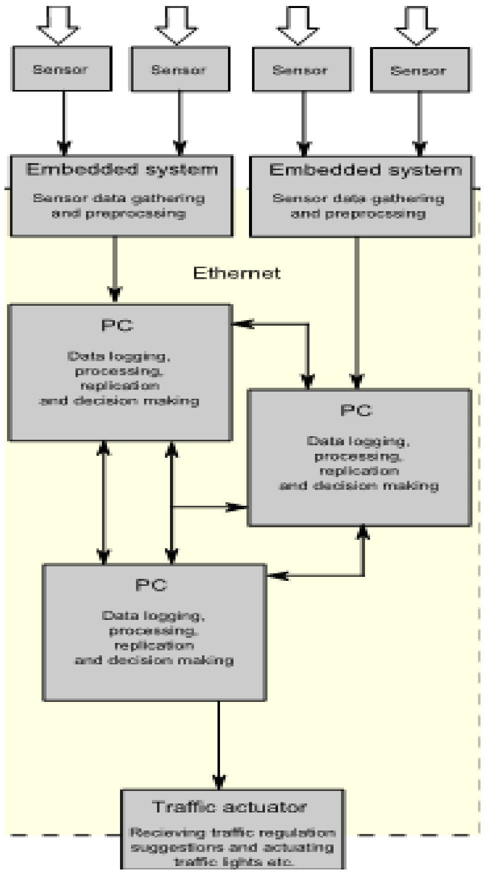
Sensor
Sensor
Sensor
Sensor
Embedded system
Sensor data gathering and preprocessing
Embedded system
Sensor data gathering and preprocessing
Ethernet
PC
Data logging, processing, replication and decision making
PC
Data logging, processing, replication and decision making
PC
Data logging, processing, replication and decision making
Traffic actuator
Recieving traffic regulation suggestions and actuating traffic lights etc.

Figure-5: logical view

V. CONCLUSION

The distributed embedded system architecture proposed and implemented in test environment. Redundancy and fault tolerance allows system to continue working even with several physical systems down, Load distribution allows additional benefits, such as capability to run identical high intensity data processing modules in parallel on several systems and Module developers do not have to concern themselves about the distributed system architecture reducing module development time and complexity. At the moment only simulated data on real physical system, will be tested in real traffic environments.

ACKNOWLEDGMENTS

We performed this work at the college of Engineering, wolaita sodo University. We acknowledge the support received from the Electrical and Computer Engineering department, WSU, ETHIOPIA.

REFERENCES

1. Arora, Sanjeev; Barak, Boaz (2009), Computational Complexity A Modern Approach, Cambridge, ISBN 978-0-521-42426-4.
2. Ghosh, Sukumar (2007), Distributed Systems – An Algorithmic Approach, Chapman & Hall/CRC, ISBN 978-1-58488-564-1.
3. W. Wolf, B. Ozer, and T. Lv, “Smart Cameras as Embedded Systems,” Computer, Sept. 2002, pp. 48-53.
4. G.L. Foresti, C. Mahonen, and C.S. Regazzoni, Multimedia Video-Based Surveillance Systems, Kluwer Academic Publishers,2000.
5. Andrews, Gregory R. (2000), Foundations of Multithreaded, Parallel, and Distributed Programming, Addison–Wesley, ISBN 0-201-357526. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. (1990),
6. M. Bramberger, B. Rinner, and H. Schwabach, “A Method for Dynamic Allocation of Tasks in Clusters of Embedded Smart Cameras,” Proc. Int’l Conf. Systems, Man and Cybernetics, IEEE Press, 2005, pp. 2595-2600.
4. R. Steinmetz and K. Nahrstedt, Multimedia Systems, Springer, 2004.
7. A. Maier, B. Rinner, and H. Schwabach, “A Hierarchical Approach for Energy-Aware Distributed Embedded Intelligent Video Surveillance,” Proc. IEEE/IFIP Int’l Workshop Parallel and Distributed Embedded Systems, IEEE Press, 2005, pp. 12-16.
8. J. Shi and C. Tomasi, “Good Features to Track,” Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition, IEEE Press, 1994, pp. 593-600.
9. Elmasri, Ramez; Navathe, Shamkant B. (2000), Fundamentals of Database Systems (3rd ed.), Addison–Wesley, ISBN 0-201-54263-3.
10. Lynch, Nancy A. (1996), Distributed Algorithms, Morgan Kaufmann, ISBN 1-55860-348-4.