Compact, Low-Cost, but Real-Time Distributed Computing for Computer Augmented Environments

Hiroaki Takada and Ken Sakamura

Department of Information Science,
Faculty of Science, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113, Japan

Abstract

We have been conducting a research and development project, called the TRON Project, for realizing the computer augmented environments. The final goal of the project is to realize highly functionally distributed systems (HFDS) in which every kind of objects around our daily life are augmented with embedded computers, be connected with networks, and cooperate each other to provide better living environments for human beings. One of the key technologies towards HFDS is the realization methods of compact and low-cost distributed computing systems with dependability and real-time property. In this paper, the overview of the TRON Project are introduced and various research and development activities for realizing HFDS, especially the subprojects on a standard real-time kernel specification for small-scale embedded systems called ITRON and on a low-cost real-time LAN called μITRON bus, are described. We also describe the future directions of the project centered on the TRON-concept Computer Augmented Building, which is a pilot realization of HFDS and being planned to be built.

1 Introduction

Recent advances in microprocessor technologies have made every kind of electric and electronic equipment around our daily life embedded with microcomputers and offer higher functions to the users. In the next decade, most kind of equipment, appliances, tools, and other objects making up our living environments will be augmented with embedded computers, be connected with networks, and cooperate each other to provide better living environments for human beings (Figure 1). In other words, these objects and networks constitute a large distributed computing system and support human activities on many aspects. We call this kind of system as a *highly functionally distributed system (HFDS)* and have been conducting a research and development project, called the TRON[1] Project, for its realization [1].

[1]TRON is an abbreviation of "The Real-Time Operating System Nucleus."

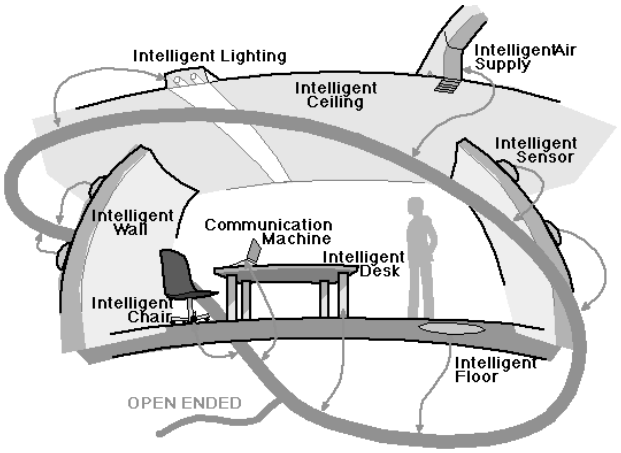

Figure 1: Computerized Living Environment

A typical example of "computing everywhere" environment is the TRON-concept Intelligent House, which was built experimentally in 1989 (Figure 2). The purpose was to create a testbed for future intensively computerized living spaces. The house with 330 m^2 of floor space was equipped with around 1000 computers, sensors, and actuators, which were loosely connected to form a distributed computing system.

This kind of "computing everywhere" environment is often called as computer augmented environments or ubiquitous computing, and is actively studied recently [2]. Most of these studies, however, stick to the notion that computers are information media providing necessary information to human users. We consider that computers should be used to provide humans with better living environments, including the provision of information.

In HFDS environments, because the number of network nodes is hundreds or thousands times larger than that of human users, the cost for a node is one of the most important issues. Dependability and real-time property is also an important issue, because HFDS is part of our real-life environments. Consequently, the realization methods of compact and low-cost distributed computing systems with dependability and real-time property is among the most important technologies to the realization of HFDS.

Figure 2: The TRON-concept Intelligent House

2 HFDS and the TRON Project

2.1 Issues towards HFDS

We have determined a number of technological issues that need to be resolved to realize HFDS, and is conducting research and development on these issues. Major features of HFDS causing the issues are as follows [1, 3].

- Huge number of nodes
  An HFDS differs from conventional networks in the number of nodes connected to it, which is hundreds or thousands times larger than the number of personal computers and workstations. About 1000 network nodes were used in the TRON-concept Intelligent House. Building will have many thousands or even millions of nodes, while cities will have billions.
- Heterogeneity
  An HFDS is characterized by the great variety of nodes connected to it. Each node varies in scale and purpose, from simple sensors gathering environmental data to supercomputers acting as computing servers.
- Open-ended network
  It must be necessary to add new nodes, remove existing ones, or move them to different locations readily. It should be also possible to install a node or remove it without turning off the power. Management method of network IDs is among the key issues.
- Dependability
  Though high dependability of individual nods is desirable, providing fault tolerance at the individual node level is difficult from cost concerns. A more realistic approach is to realize dependability of system as a whole, by making it possible to recognize a node failure and to compensate for it.
- Real-time response
  As can be seen from the full name of the project, we consider that guaranteeing real-time response will be required in all computer applications in the future. This requirement applies especially to an HFDS, which handles information used for the control of equipment in real-life environments.
- Usability
  Because network nodes are part of our living environments, they should be equipped with simple user interface that anyone can use.
- Extensibility
  The node hardware and software naturally differ depending on when they are developed. Nodes with new functions must be able to work in cooperation with older ones. It is also necessary that version management and updating are performed automatically.

2.2 Overview of the TRON Project

As a comprehensive approach to solving the problems described above, we concluded that it would be necessary to build the basic parts for HFDS in addition to making up actual application environments of HFDS, such as the TRON-concept Intelligent House and the TRON-concept Computer Augmented Building, which will be described in detail in Section 6. The results of the application subprojects are fed back to the other subprojects building the basic parts.

Subprojects we have been conducting to develop the basic parts for HFDS are as follows.

CHIP

A series of general-purpose microprocessors are necessary to realize HFDS. The TRON-specification CHIP is an original microprocessor architecture designed to support the components of HFDS. Six manufactures have developed about 10 microprocessors based on the specification.

The CHIP subproject is now at the beginning of the second stage, in which the hardware architecture and software on it are codesigned in order to achieve higher cost-performance. Research is now being conducted into automatic generation of ASIC designs, incorporating peripheral functions around a core fine-tuned for each application.

ITRON (Industrial TRON)

ITRON is a series of specifications for real-time operating systems for use in embedded computers. The ITRON subproject is described in detail in Section 4.

BTRON (Business TRON)

BTRON is an architecture for personal computers or workstations supporting smooth interaction between humans and HFDS. The important features of BTRON are its uniform human-machine interface (HMI) and data compatibility using a data exchange format called TRON Application Databus (TAD). It also supports a hypertext-based file system called the real object/virtual object model.

Three operating system specifications called BTRON1, BTRON2, and BTRON3 have been developed. A BTRON1-specification OS running on personal computers was released in 1991, and a BTRON3-specification OS running on a workstation using a TRON-specification CHIP has been implemented in 1994.

CTRON (Central/Communication TRON)

CTRON is a set of operating system interfaces for common application to switching, communication processing,

and information processing nodes in HFDS. The CTRON architecture features a two-layered design, consisting of the basic OS interfaces including kernel and I/O control, and the extended OS interfaces including data storage control, execution control, and communication control.

The initial edition of the CTRON specifications was published in 1988 and the revised edition in 1993. More than 20 products have been developed and certified as compliant with the specifications. Now, they are widely used for switching systems, PBXs, and other communication systems.

MTRON (Macro TRON)

MTRON, a key issue for realizing HFDS, is an architecture for large computer networks included in an HFDS. Research on MTRON has become active recently, because the components of MTRON have become available as the results of other subprojects. We will describe the basic architecture of MTRON in Section 3.

TAD (TRON Application Databus)

TAD is a data exchange format to provide data compatibility among the computers designed according to the TRON architecture. TAD is described in Section 3.3.

HMI (Human-Machine Interface)

One of the most important issues for realizing "computing everywhere" environments is human-machine interface (HMI) by which people can communicate with computers easily and effectively. In particular, it is essential to realize an environment in which all equipment around us is operated in a standardized way. Another requirement is that HMI standards be usable anywhere, by anyone, for any purpose, because making use of computer-embedded equipment is essential in "computing everywhere" environments.

To this end, the TRON HMI design guidelines have been published as the "TRON Human-Machine Interface Specifications", which defines a set of standard HMI guidelines for both GUI (Graphical User Interface), which is a virtual HMI on an electronic display, and SUI (Solid User Interface), which is an HMI making use of real buttons, dials, and other parts on an operating panel [4].

Importance is also laid on the adaptation of HMI for each user, We have conducting researches on multi-lingual processing and Enableware, which is an HMI guidelines for people with physical disabilities.

2.3 Open Architecture

A fundamental policy of the TRON Project is that the results of the project, in the form of published specifications and guidelines, are made available to the public. Anyone throughout the world is free to develop and market products implementing the specifications.

We regard interface specifications like those presented in the previous section as a technological infrastructure for future computerized society. They should not be monopolized by one or a few corporations or governments. Actually, ITRON, BTRON, CTRON, and CHIP specifications presented in the previous section are implemented and marketed by multiple vendors.

Another benefit of open architecture is the possibility of what we call multi-layered design diversity (MLDD), in which the design diversity approach is adopted to each system layer to archive very high dependability [5]. The fact that multiple corporations have implemented different products based on the same specification makes MLDD realistic.

3 Basic Architecture of MTRON

3.1 Two-level Network Architecture

In HFDS environments, some of the nodes are very small and have only very limited processing and memory resources. Because a general-purpose network protocol suite (such as TCP/IP and OSI) provides abundant functions necessary for various applications, relatively large computing resources are usually necessary to handle the protocols. It is often difficult to install them to small-scale nodes in HFDS, and then the approach that an HFDS is constructed with only one protocol suite is not realistic.

Consequently, we have adopted the two-level network architecture as the basic architecture of MTRON. In this architecture, a network is divided into the backbone subnetwork in which a general-purpose protocol suite (called the exterior protocol, below) is used and branch subnetworks in which protocols with limited functions (called the interior protocols) are used. A gateway node implements both the exterior protocol and an interior protocol, and exchanges information between them. Note that the usage of the protocols do not correspond with the physical topology of the network.

A widely-used protocol suite will be adopted as the exterior protocol. Though it is possible to use an appropriate interior protocol for each subnetwork, a standard protocol for HFDS must be fixed to realize an open network environment.

3.2 The Standard Interior Protocol

The standard interior protocol for HFDS environments, now under investigation, is assumed to be used in a limited area, for example, within a room or a floor of a building. The following functions will be omitted or restricted in the standard interior protocol.

- Security-related functions

  In order to support security-related functions, a concept of user (or access subject), a concept of access right, and an encryption mechanism are necessary to be incorporated in the protocol. Because they (especially an encryption mechanism) usually require large computing resources, it is difficult that every node in HFDS environments supports security-related functions.

  Instead, the security of the subnetwork using the interior protocol should be maintained with physical security. If someone breaks in a room, for example, and taps the

network line, it is unavoidable that the security of the network is also broken.

- Some protocol layers
  A much simpler layered model than the OSI reference model is adopted in the standard interior protocol suite. Specifically, four layered architecture consisting of physical layer, MAC layer, LLC layer, and application layer is a candidate [6]. The approach is also investigated that a network layer protocol with limited functions is adopted.

3.3 TAD and Programmable Interface

We specify a uniform data format, called TAD (TRON Application Databus), for various information passed between nodes in HFDS environments. TAD is intended as a universal data exchange format, handling real-time information like voice and video, and various physical information for control (such as length, volume, location, temperature, pressure, luminosity, electric power, and so on), as well as conventional text and ordinary graphics.

However, in a large-scale and open-ended network, imposing the same fixed data format will stifle the incorporation of future advances in technologies. On the other hand, if version updates are allowed, inconsistency in versions throughout the network will become a serious problem. If strict adherence to upward compatibility is made mandatory, many systems will become overweighted with non-standard and obsolete features.

The *programmable interface* concept approaches this dilemma in the following way [7]. Each interface between systems are made extensible with programs. When communication takes place between systems, one of the systems sends the interface specifications represented in program codes to the others if necessary. For this purpose, the TULS (TRON Universal Language System) specifications have been devised.

4 A Standard Real-Time Kernel Specification for Embedded Systems

In HFDS environments, a large number of small-scale embedded systems are developed and utilized. We have been investigating on standard real-time OS specifications for embedded systems since about ten years ago, and have published a series of ITRON kernel specifications as the result [8]. The reason for centering these studies on kernel specifications is that only the kernel functions are used in most deeply embedded systems.

In developing small-scale embedded systems, which tend to be manufactured in great quantities and be priced cheap, lowering the cost of the final product is usually considered more important than reducing development costs.

4.1 Requirements on a Standard Real-Time OS for Embedded Systems

In the field of small-scale embedded systems, it is common practice to use a single-chip microcontroller unit (MCU), integrating a processor core with ROM, RAM, general I/O devices, and application-specific peripheral modules. A particular problem in developing software for an MCU is the limited hardware resources. Memory size limits are especially severe, with a typical MCU having around 32 KB of ROM and 1 KB of RAM. Even larger-scale chips have no more than 128 KB of ROM and 4 KB of RAM.

Another point worth noting is that the need for high cost performance in an MCU-based system means the design is frequently optimized to the application, resulting in a large number of different processor cores.

Even in this field of small-scale embedded systems, raising software productivity is an important issue. Use of C or other high-level programming languages is one solution; another approach being adopted with increasing frequency is the use of a real-time OS.

Requirements on a standard real-time OS for embedded systems are summarized as follows.

- Being able to derive maximum performance from hardware
  Given the severe hardware resource limitations of a typical MCU-based system, the ability to derive maximum performance from the available hardware is a prerequisite for real-time OS adoption.
  In the case of small-scale embedded systems, maximizing hardware performance is considered more important than full source code-level compatibility. This is because porting to a different processor and its OS normally becomes necessary only when there are changes also in the equipment being controlled, so there is hardly ever a situation where an application program is ported to another OS without modification.
- Helping to improve software productivity
  Especially important is standardization from a training standpoint, such as adopting consistent concepts and terminology, and standardizing design methods.
- Being uniformly applicable to various processor scales and types
  The hardware used in an embedded system is normally designed optimally for its application. The processor scale, moreover, may vary widely from 8-bit MCUs to 32-bit processors depending on the kind of equipment to be controlled.

4.2 Design Principles of the ITRON Specifications

The following design principles were established in order to satisfy the requirements described above.

- Avoid excessive hardware virtualization
  Too much hardware virtualization must be avoided in order to derive the maximum performance from hardware and achieve high real-time performance. Although intended for implementing on a variety of different processors, the basic policy is to design each implementation independently for each processor.

To this end, the ITRON specification is divided clearly into aspects that are standard across all processors and implementation-dependent aspects. Standardized items include task scheduling rules, system call names and functions, parameter names, sequence and meanings, and error code names and meanings. On the other hand, those aspects that need to be decided separately for each implementation, based on runtime performance considerations, are not precisely standardized. Examples are parameter bit size, the method of invoking interrupt handlers, and exception handlings.

- Allow for optimization to application
  Optimizing to the application means modifying the kernel specification and internal implementation algorithm based on the requirements by a particular application. In the case of embedded systems, the kernel object code is generated for each application, making this optimization especially effective.
  Specifically, the specification was designed so as to make the kernel functions independent of each other to the extent possible, so that each application can use just the functions it needs. In fact, most μITRON-specification kernels are provided in the form of libraries, and are designed so that only the necessary modules are loaded when the kernel is linked to the application. Also, each system call provides a single function, making it easy to select out the necessary functions for an application.
- Allow for optimization to hardware
  Optimizing to hardware means modifying the kernel specification and internal implementation method based on the nature of the hardware and its performance, in order to raise the performance of the system as a whole.
- Emphasize ease of software engineer training
  As noted before, software compatibility and portability tend to be considered as of relatively minor importance with small-scale embedded systems. The significance of standardizing kernel specifications relates more to ease of training software engineers, and the improved communication among software personnel resulting from consistent terminology and concepts.
- Create a specification series and/or divide into levels
  Specifications are developed in series to make them applicable to a wide variety of hardware. Moreover, each specification divides functions into different levels based on their degree of necessity.
- Make available a full range of functions
  Rather than limiting the number of primitives provided by the kernel, the approach is taken of making available a wide variety of primitives with different functions. The idea is to enable implementors to raise the runtime performance and improve ease of programming by using primitives suitable to the particular application and hardware.

A concept common to many of these design principles is that of loose standardization. This means setting uniform standards only to the extent that performance will not suffer, rather than trying to force all systems into one rigid mold, and leaving room to decide matters dependent on the processor or application.

Consumer Applications
TVs, VCRs, audio components, air-conditioners, washing machines, microwave ovens, rice cookers, lighting
OA Applications
printers, copiers, image scanners, word processors, optical filing systems
Communications
answer phones, ISDN telephones, cellular phones, FAX, broadcasting equipment, wireless systems, antenna controllers, satellite controllers
FA and Other Applications
PDAs, game gear, automobiles, vending machines, electronic musical instruments, FA computers, industrial robots

Table 1: Typical ITRON-specification Kernel Applications

4.3 Current Status of the ITRON Specifications

The first ITRON specification was released in 1987 as ITRON1. Thereafter studies were carried out on a reduced-function specification called μITRON (Ver. 2.0) for smaller-scale 8-bit and 16-bit MCUs, and on the ITRON2 specification for larger-scale systems with 32-bit processors. Both of these were released in 1989.

Of these, the μITRON specification offered very realistic performance even on an MCU with only very limited processing and memory resources, and has therefore been implemented on many different MCUs. Its application has even widened to various 16-bit MCUs as well as 32-bit processors. Just counting the μITRON-specification products that have been registered officially, there are around 30 implementations for more than 20 processors. In addition to them, the μITRON-specification kernel, with its small size and relative ease of implementation, has been used in numerous developments for in-house systems. There are also several μITRON-specification kernels that have been made available as free software.

As the μITRON-specification kernel has come to be applied to a wide range of fields, a clearer picture has emerged as to the necessity of each function and the performance demands. Also, as noted above, the μITRON-specification kernel has in some instances been implemented for 32-bit processors, something we did not originally anticipate. It was therefore decided to reexamine the existing ITRON specifications, resulting in the release in 1993 of the third-generation ITRON specification, called μITRON3.0 [9].

It goes without saying that the reason for this large number of ITRON-specification kernel implementations is the wide range of application fields and numerous application examples. Table 1 lists some of the applications in which ITRON-specification kernels are used.

4.4 Network Support and Current Issues

With low-cost MCUs becoming available, multiple MCUs are increasingly being used to control one piece of equipment. A major reason is to increase ease of maintenance and reliability by configuring products of component units. Specific examples include copiers, fax machines, and automobiles.

In line with this trend, the μITRON3.0 specification incorporates additional functions, called connection functions, that support distributed systems consisting of ITRON-specification kernel-based nodes interconnected on a network. With the connection functions, objects such as tasks and semaphores on other nodes can be directly manipulated using ordinary system calls.

The connection functions in the μITRON3.0 specification at this time are intended only for controlling one piece of equipment or system, and it is necessary to decide the network configuration statically at the time the system is built. As the next step, it will be important to enable interconnection among separately designed systems. It will further be necessary to deal with dynamic changes in network configuration. The specification at that stage is called IMTRON and being studied within the framework of MTRON.

5 A Low-Cost Control LAN Standard

Another key technology to make HFDS realistic is low-cost control LAN with real-time property. In this section, requirements on a control LAN standard for HFDS are summarized, and the overview of the μITRON bus, which is a control LAN standard now under the research and development stage, is described.

5.1 Requirements on a LAN Standard for HFDS

There are following requirements on a control LAN standard for use in HFDS environments.

- Able to implement with low cost
  Because lots of cheap nodes, such as home appliances, sensors, and wall switches, are connected to the network with this LAN standard, it is essential that a LAN node can be implemented in compact and with low cost.
- Usable as a general-purpose control LAN
  Specifically, we have established the following target performance. The maximum transfer rate is 1 – 2 MBPS and more than 500 packets should be able to transferred in a second. The maximum length of a line is around 500 m and the maximum number of nodes on a line is about 250 (the physical layer may have a lower limit).
- Having real-time property
  Here, real-time property means that packets are transferred in the order of their priorities and that the time until a node finishes sending a packet with the highest priority is bounded, even when some other nodes try to send packets with the same priority.
- Interoperable in open network environments
  In the case of control LAN standards, the tuning approach is often adopted in which some communication parameters are determined to be optimal for each application. In case of HFDS environments, however, because nodes developed independently by different manufactures are interconnected with the LAN, one set (or a few sets depending on the uses) of parameters must be given.
- Supporting easy operations
  End-users should be able to easily install a node to the LAN or remove it without turning off the power. It is also necessary that the operation of the LAN can be continued when some of the nodes are turned on or off.

We have concluded that there is no existing LAN standard satisfying all of the above requirements and that the development of a new standard for HFDS is necessary. This new LAN standard is called μITRON bus.

5.2 Overview of the μITRON bus

From the cost and easy operation requirements, a simple bus topology is desirable for the μITRON bus. Considering this and the requirement on real-time property, we have decided to adopt the bitwise arbitration CSMA protocol in its MAC layer. We have extended the protocol with the following methods in order to solve the problems of the original protocol [10].

Dual-rate Bitwise Arbitration CSMA Protocol

With the bitwise arbitration CSMA protocol, there exists a trade-off between the transfer rate and the maximum length of a line, because contentions are resolved for each bit. When the transfer rate is raised, the maximum length becomes short, and vice versa. Among the existing standards adopting the protocol, the maximum length of a line is very short (typically 50 m) with CAN (Controller Area Network), while the transfer rate is quite slow (9.6 KBPS) with the Homebus standard.

In order to meet both requirements at the same time, μITRON adopts the *dual-rate protocol* in which the transfer rate for sending data packets is faster than that for arbitration. Specifically, the former will be determined to be 20 – 50 KBPS and the latter be 1.5 – 3 MBPS, resulting that the maximum bandwidth is 1 – 2 MBPS. Because the arbitration phase consists of 15 – 19 bits (1 bit for the start bit, 4 – 8 bits for the priority, 1 bit for the priority reservation bit, 8 bits for the node ID, and 1 bit for the parity), arbitration takes 1 ms at most and can be executed more than 1000 times within a second.

Priority Reservation Bit

Another fault of the original bitwise arbitration CSMA protocol is that the time until a node finishes sending a packet with the highest priority cannot be bounded, when some other nodes try to send packets with the same priority.

In the μITRON bus standard, this problem is solved with the *priority reservation bit* which follows the priority field in the arbitration phase. When a node begins to send a data packet, the priority reservation bit is cleared

to "0" (we assume that "0" represents the lower priority). When it detects that another node with a larger (or more prioritized) node ID succeeds to send a data packet with the same priority, the priority reservation bit is set to "1". With this simple method, the time until the node finishes sending a packet with the highest priority is bounded, even when some other nodes try to send packets with the same priority.

Piggy Backing

In general, when the average length of data packets is very short, the effective transfer rate tends to degrade. This problem is noteworthy with the dual-rate protocol.

We have adopted *piggy backing* to remedy this problem. When a node has multiple data packets to be sent, it sends a frame which conveys as many packets as possible, once it obtains the access right to the bus through arbitration. Packets with different destination IDs and with different priorities can be packed into a frame.

Although piggy backing is effective for improving the effective transfer rate, its realization can raise the cost of LAN controllers. Therefore, it is desirable that small-scale nodes can omit the support for piggy backing. To archive this, we have placed the restriction that a frame must not include more than one packets with the same destination ID. With this restriction, a small-scale node can assume that at most one packet is necessary to be read within a frame.

5.3 Implementation Approach

We plan to implement three types of LAN controllers for the μITRON bus and use the most suitable one for each application.

Direct I/O Type

The direct I/O type of LAN controllers are used on the nodes that can be realized without processors, such as sensors and wall switches. The LAN controller chip has a port to the LAN and a parallel I/O port that can be controlled from other nodes through the LAN.

Some functions can be omitted from the direct I/O type of LAN controllers. For example, the formats of the packets that the controller can send or receive are fixed. The priority of packets it sends can also be fixed to one value. It is not necessary to send a piggy backed frame.

ASIC Library Type

MCUs are used for more complex nodes in HFDS environments. The LAN controllers for the nodes should be provided as an ASIC library which is to incorporate in an MCU.

Some functions are also omitted from the ASIC library type of LAN controllers. For example, the packet formats that the controller may send or receive may have some restrictions.

Controller Chip Type

The LAN controllers that have all the functions defined in the μITRON bus standard is developed and supplied as an independent chip. Server machines will use this type of LAN controllers.

Figure 3: The TRON-concept Computer Augmented Building

6 TRON-concept Computer Augmented Building

The TRON-concept Computer Augmented Building, which is studied as one of the application subprojects of the TRON Project, is an experimental realization of HFDS. A pilot building is planned to be built in Tokyo and is now in the final design stage (Figure 3). This section presents the concepts and major functions of the building, and describes the information infrastructure of it.

6.1 Concepts and Major Functions

The TRON-concept Computer Augmented Building is being designed to provide better office environments for the people working in it and to enhance their productivity, through extensive uses of advanced computer technologies. It also aims to save energy through minute control of equipment in the building.

A few of the concepts and the major functions of the building is as follows.

Super ID Card System

The super ID card system keeps track of people's location in the building at all times. The location information is used for various applications, including adaptive air-conditioning and security management. It is also used to suggest improvements in the office layout.

Energy-Saving Building

Minute control of equipment can reduce energy consumption in the building. For example, when no people is in a room, the air-conditioners and lighting of the room are automatically turned off. The building is also designed to utilize natural environments: Fresh air outside the building and external light can be let in the building through computer-controlled windows and blinds.

Inter-Media Building

The building provides facilities to access various information from communication satellites and the Internet, to convert it to desirable format, and to dispatch information through these media.

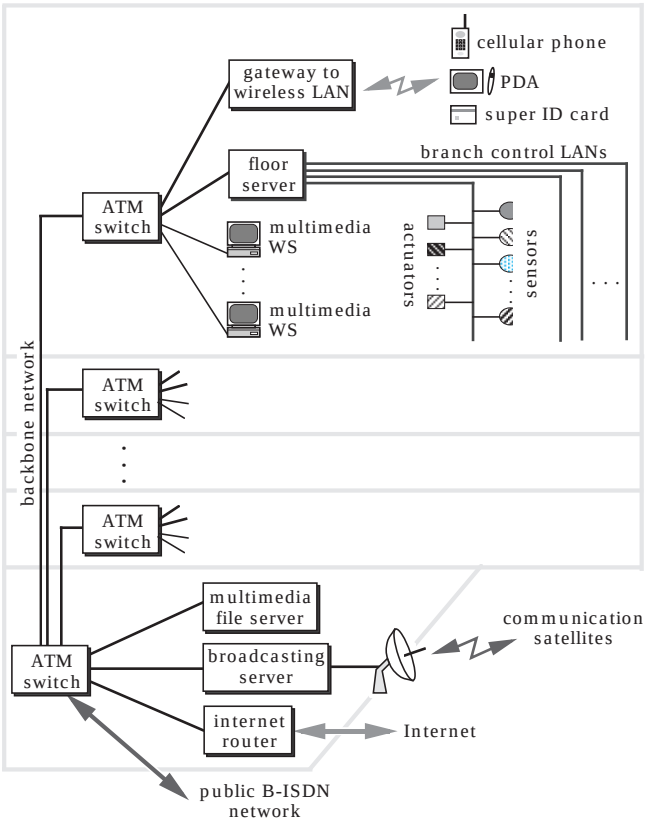

Figure 4: Information Infrastructure of the TRON-concept Computer Augmented Building (plan)

Advanced Facility Management

All equipment in the building are managed with the facility management database. When the office layout is changed, for example, the lighting control system and the air-conditioning system can change their configuration automatically reading the locations of facilities and the new layout information from the database.

6.2 Building Information Infrastructure

In most of the conventional intelligent buildings, each application subsystem (electric power management, air-conditioning, security management, and so on) has its own sensors, actuators, and communication lines connecting them, resulting in redundant investment in equipment. In the TRON-concept Computer Augmented Building, sensors and communication networks are installed and managed in a unified way as the information infrastructure of the building.

The physical layer skeleton of the information infrastructure of the building is presented in Figure 4. The ATM technology is used for the backbone network, and uniformly supports various kind of communication in the building, including communication for controlling equipment, multimedia data transfer, ordinary file transfer, and telephone applications.

A sensor connected to a branch control LAN multicasts the collected data to the LAN. An actuator that needs the information from the sensor reads the data from the LAN and determines its behavior. A *building information server* running on the floor server machines on the LAN reads all data multicasted on the branch control LAN and provides them to each application subsystem. TAD is used as the data exchange format for all of the communications described here.

Other important functions of the building information infrastructure include the facility management and the infrastructure management. The latter provides diagnostic function for the system.

7 Conclusion

In this paper, the overview of the TRON Project is introduced and its research and development activities towards the realization of HFDS, especially issues on compact and low-cost distributed computing with dependability and real-time property, are described in detail.

Though there is much work remaining to be done to realize an HFDS, we will continue the effort to solve problems with eyes on the long-term consequences.

Acknowledgments

We wish to thank all the people participating in the various activities of the TRON Project.

References

[1] K. Sakamura, "After a decade of TRON, what comes next?," in *Proc. of the Eleventh TRON Project International Symposium*, pp. 2–16, IEEE CS Press, Dec. 1994.

[2] Special Issue: Computer Augmented Environments: Back to the Real World, *Comm. ACM*, vol. 36, July 1993.

[3] K. Sakamura, "Infrastructures for an age of computerized environments," in *Proc. of the Tenth TRON Project Symposium*, pp. 2–14, IEEE CS Press, Dec. 1993.

[4] K. Sakamura, "Human interface with computers in everyday life," in *Proc. of the Ninth TRON Project Symposium*, pp. 2–13, IEEE CS Press, 1992.

[5] A. Watanabe and K. Sakamura, "Design fault tolerance in operating systems based on a standardization project," in *Proc. Int'l Symposium on Fault-Tolerant Computing*, June 1995. to appear.

[6] K. Arvind, K. Ramamritham, and J. A. Stankovic, "A local area network architecture for communication in distributed real-time systems," *Real-Time Systems*, vol. 3, no. 2, pp. 115–147, 1991.

[7] K. Sakamura, "Programmable interface design in HFDS," in *TRON Project 1990*, pp. 3–22, Springer-Verlag, 1990.

[8] H. Takada and K. Sakamura, "Advances in the ITRON specifications – supporting multiprocessor and distributed systems," in *Proc. of the Ninth TRON Project Symposium*, pp. 89–95, IEEE CS Press, 1992.

[9] K. Sakamura, ed., *μITRON 3.0 Specification*. Tokyo: TRON Association, 1994. (can be obtained from http://tron.is.s.u-tokyo.ac.jp/TRON/ITRON/SPEC/mitron3.txt.Z).

[10] Y. Mano, H. Mori, H. Takada, and K. Sakamura, "Dual-rate bitwise arbitration CSMA protocol," in preparation.