ROOM-BRIDGE : Vertically Configurable Network Architecture and Real-Time Middleware for Interoperability between Ubiquitous Consumer Devices in the Home

Soon Ju Kang, Jun Ho Park, and Sung Ho Park

School of Electrical Engineering and Computer Science,
Kyungpook National University, Korea
sjkang@ee.knu.ac.kr,{zec,slblue}@palgong.ac.kr

Abstract. This research focuses on the development of a generic home network architecture and related middleware for supporting interoperability between ubiquitous consumer devices and computing devices in the home. In order to build a practical home network, design criteria are discussed, including real-time constraints and interoperability between heterogeneous protocols. As a result, a vertically configurable home network architecture and real-time middleware, called ROOM-BRIDGE, are presented that meet the design criteria. ROOM-BRIDGE is specially designed and implemented using an IEEE1394 backbone network to provide the proposed network architecture with a guarantee of seamless and reliable communication between heterogeneous sub-networks and ubiquitous devices in the home. The performance of the proposed network architecture and ROOM-BRIDGE was verified by prototype implementation and testing using a home network test bed.

1 Introduction

The home is a typical ubiquitous computing environment as it contains many types of consumer devices and computing devices. Intensive research efforts have been made to build a practical home network [1,2], however, at this point there is an overabundance of protocols and middleware systems in the home network industry, with no clear winner to cover all services in the home. This is basically because a home network requires multiple services, whereas most protocols have only been designed to support a specific service. The currently existing protocols and middleware systems can be categorized into three major services [1,2]. The first is home automation services, including the remote control of room temperature, lighting, or windows, plus the implementation of electric or gas telemetry. Since this type of network must be very reliable and safe, it is normally implemented using solutions in a control area network [4,8], originating from the building and factory automation industries. The second is home theatre services through an audio/video(A/V) network. This type of network connects various kinds of audio and video devices, including TVs, VCRs, camcorders, and so on. IEEE1394 is one of the de facto standard protocols in this A/V field and is already embedded in many audio and video devices [5,6]. The third is Internet and wireless access services [7], which enable remote reality for home devices through an Internet or wireless network. The second and third types of service can also be re-grouped into a data network category.

Because the development of such services has been pursued independently, there is currently an overabundance of protocols and middleware systems for supporting each specific type of service in the home network industry. LonWork [4,8], BACnet, and DeviceNet are examples of control area networks and support home automation services. HAVi(Home Audio/Video Interoperability) [6] and VESA [1] are the dominant middleware systems for A/V networks through IEEE1394 for supporting home theater services. CORBA [12] and JINI [9] are middleware systems for the Internet, while Bluetooth [1,2] is the main standard for wireless solutions in the home. In addition to the above protocols, there are also many more, e.g. UPnP by MicroSoft, CEBUS and CAL, in fact too many to mention in this paper.

Accordingly, due to the plethora of protocols and idiosyncrasies of each one, this creates a serious dilemma when attempting to build a practical home network. One homogeneous protocol is certainly not enough, whereas wiring a house with a sophisticated web of several independent networks to accommodate the idiosyncrasies of each protocol is equally impractical. To make matters worse, each protocol and middleware has little consideration for interoperability with other systems.

Therefore, to solve these problems and build a practical home network using currently existing protocols and the resources produced by each protocol supporting an organization, the current study adopts the concept of a vertically integrated network architecture, as first introduced by E. D. Jensen [3]. As such, the focus is on the development of a new home network architecture and related middleware that can vertically abstract low-level heterogeneous protocols without losing any of their idiosyncrasies. The basic design criteria for the proposed architecture are as follows; first, the structure of a traditional house is considered as a design criterion for the vertical integration of a home network. For example, a house can be divided into several subsections, such as rooms, the kitchen, and so on. Similarly, a home network can also be configured into several subnets and each subnet copes with the networking of ubiquitous consumer devices inside a particular subsection, such as a room or the kitchen. As a result, each sub-network consists of several local networks with homogenous protocols according to the services, e.g. a LonTalk network for home automation, IEEE1394 network for multimedia services, and so on. Second, a broadband backbone network is needed to connect all the subnets. Third, a subnet gateway server, called a ROOM-BRIDGE server, is specially designed to connect each subnet into the backbone network and accommodate un-networked ubiquitous computing devices, such as irDA devices. Fourth, a specific middleware, called ROOM-BRIDGE, and device driver architecture for embedding into each ROOM-BRIDGE server are designed and implemented to guarantee interoperability, a real-time response, and reliability in the proposed home network.

Chapter 2 introduces the basic concept and related research, along with the design criteria for the proposed architecture. Chapter 3 explains an overview of the proposed home network architecture as a target hardware system architecture and introduces the ROOM-BRIDGE server. Chapter 4 presents a brief explanation of the core components in the proposed middleware, called ROOM-BRIDGE. Chapter 5 provides a detailed description of the resource repository model, while the priority based event channel model and real-time device driver for an IEEE1394 adaptor are presented in chapters 6 and 7, respectively. Chapter 8 discusses the experimental results of the prototype home network based upon the ROOM-BRIDGE architecture. Chapter 6 presents some final conclusions along with areas for further research.

2 Related Work and Background

Although many studies continue to be conducted on developing a practical home network [1,2] [4,5,6,7,8] a de facto standard protocol and middleware that can cover all services in the home has not yet appeared. As already mentioned, the main reason is due to the idiosyncrasies related to each existing protocol. For a brief survey of these idiosyncrasies, two protocols, LonTalk [4] for home automation services and IEEE1394 [5] for home theatre services, were selected as they have already been accepted as the dominant standard in their respective areas.

2.1 LonTalk

The LonTalk protocol is a widely adopted protocol used for communication between intelligent embedded sensors, actuators, and controllers. The LonTalk protocol is based on a control area network, therefore, it is a mechanism whereby intelligent devices can exchange control and status information. LonTalk was originally developed by the Echelon Corp. and implemented as an integrated microcontroller and network communications chip (the Neuron Chip). The LonTalk protocol is able to describe using the 7-layer ISO model, as shown in Table 1, and has been standardized as EIA-709.1. As such, it is freely available for implementation in any microprocessor making it fully interoperable with Neuron Chip implementations [4].

2.2 IEEE1394 and HAVi

IEEE 1394 is an international standard, low-cost digital interface that can integrate entertainment, communication, and computing electronics into consumer multimedia. Originally designed by Apple Computer as a desktop LAN and then developed by the IEEE 1394 working group, IEEE 1394 has the following features:

- Hot pluggable - users can add or remove 1394 devices using the active bus. Scaleable architecture - can mix 100, 200, and 400 Mbps devices on a bus.
- Flexible topology – can support daisy chaining and branching for true peer-to-peer communication.

- Serial Bus Management provides overall configuration control of the serial bus in the form of optimizing arbitration timing, guarantee of adequate electrical power for all devices on the bus, the assignment for which the IEEE 1394 device is the cycle master, assignment of isochronous channel ID, and notification of errors. The bus management is built on IEEE 1212 standard register architecture

There are two types of IEEE 1394 data transfer: asynchronous and isochronous. Asynchronous transport is a traditional computer memory-mapped, load and store interface. Data requests are sent to a specific address and an acknowledgment is returned. In addition to an architecture that scales with silicon technology, IEEE 1394 features a unique isochronous data channel interface. Isochronous data channels provide guaranteed data transport at a pre-determined rate. This is especially important for time-critical multimedia data where just-in-time delivery eliminates the need for costly buffering. As shown in Table 1, the IEEE 1394 protocol layer is only defined until the transport layer, plus all network management features from the physical layer to the transport layer are integrated into a hardware-based module. Therefore, the application programmer is not required to understand any of the complex network management features. All these features of IEEE 1394 are key reasons why it has become the A/V Digital interface of choice [2,7].

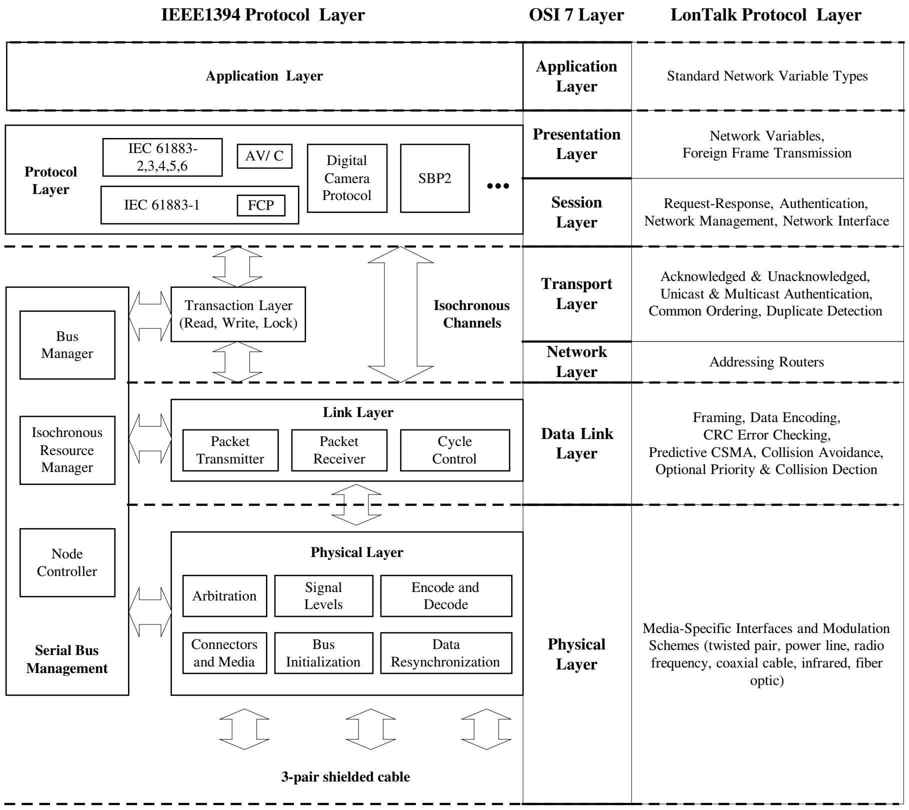
HAVi [6] is the dominant standard middleware architecture for audio / video home networks, and is intended for consumer electronic devices and computer devices supporting the IEEE Std 1394-1995 and IEC 61883 [5] interface standard. HAVi provides a set of services that facilitate interoperability and the development of distributed applications in home networks.

2.3 Comparison of Idiosyncrasies between LonTalk and IEEE1394

Table 1. shows a brief comparison of the characteristics of the two protocols relative to the ISO 7 layer [4,5] As shown in Table 1, the characteristics of each protocol are quite distinct from each other, e.g. LonTalk supports various physical communication media, such as power lines, coaxial cables, twisted pair cables, yet IEEE1394 only supports 3-pair shielded cables. Based on this property, the LonTalk supporting association can also produce a network interface module, called a Neuron Chip, for embedding into home appliances or small-scale devices, such as sensors or lights. Since a Neuron Chip has a certain level of computing power, various small-scale devices with either no or less computing power can be easily connected. In addition, LonTalk supports event-based priority scheduling and rigorous timing in order to perform real-time services.

In contrast, IEEE1394 has a casual data communication channel, referred to as the asynchronous mode, plus an additional powerful broadband communication channel, referred to as the isochronous mode, for guaranteeing multimedia communication with QoS (Quality of Service). Although the IEEE1394 supporting association also produces a communication chip [5] for embedding into devices, this chip only includes a module for communication support with no additional module to support the computing power of the embedded devices. Therefore, when compared to LonTalk, it is much more difficult to embed IEEE1394 into small-scale devices, such as lights or sensors. In addition, the asynchronous mode of IEEE1394 has no features to guarantee real-time constraints, and the isochronous mode is too different to achieve interoperability with LonTalk or traditional data communication protocols, such as TCP/IP. In the case of middleware, LonTalk has a homogenous protocol stack that covers the 7 layers, as shown in Table 1, and no middleware concept for supporting interoperability with other protocols. In the case of IEEE1394, although the IEEE1394 supporting association already suggests a well-known middleware, called HAVi, it has less consideration for none-IEEE 1394 devices, and no specific technical consideration for devices belonging to the home automation category. Accordingly, it has no definition of protocol level interoperability, such as IEEE1394 via LonTalk, and no features to guarantee real-time constraints through its asynchronous channel.

Table 1. Comparison between LonTalk and IEEE1394



OSI 7 Layer	LonTalk Protocol Layer
Application Layer	Standard Network Variable Types
Presentation Layer	Network Variables, Foreign Frame Transmission
Session Layer	Request-Response, Authentication, Network Management, Network Interface
Transport Layer	Acknowledged & Unacknowledged, Unicast & Multicast Authentication, Common Ordering, Duplicate Detection
Network Layer	Addressing Routers
Data Link Layer	Framing, Data Encoding, CRC Error Checking, Predictive CSMA, Collision Avoidance, Optional Priority & Collision Dection
Physical Layer	Media-Specific Interfaces and Modulation Schemes (twisted pair, power line, radio frequency, coaxial cable, infrared, fiber optic)

Due to these facts, LonTalk is currently widely utilized in factory automation, building automation, and even home automation, plus a variety of products supporting LonTalk, including air-conditioners, gas and electricity meters, and lighting solutions, have been produced by several companies. Conversely, IEEE 1394 has become the de facto standard for multimedia consumer devices, such as camcorders, digital VCRs, and digital cameras, thus the majority of commercial products in this area support an IEEE1394 port. However, at this point there is no possibility of interoperability without device-level protocol changes. Accordingly, since neither of these protocols can be ignored when building a practical home network, a new architecture that can simultaneously incorporate these protocols on a home network platform is necessary.

2.4 Design Requirements and Guidelines for Proposed Architecture

Based on the above study of the idiosyncratic properties of the two heterogeneous protocols, the following design requirements are defined for developing a practical home network architecture and related middleware:

- Accommodating various ubiquitous consumer and computing devices
  It is reasonable to expect that a home network should accommodate various kinds of consumer devices, whether or not they have computing power, and as many as possible in order to increase the practicality and maximize the benefit of home networking. Therefore, to achieve this goal, the proposed home network and related middleware should have an extendable and re-configurable architecture to accommodate any kind of consumer device.

- Vertically configurable network hardware architecture
  As mentioned in the previous section, a home network is totally different from a general data network in that it has many and various kinds of devices and each device has different requirements for joining a service transaction in the network, i.e. some devices require a high communication bandwidth with a QoS guarantee, while other devices require reliable and rigorous communication with a real-time guarantee. In spite of these low-level individual properties [21], high-level application service agents can still perform without considering the low-level protocol properties. In addition, agents or a home member also can control several consumer devices without considering their location and properties, such as "turn off all consumer devices in room 5" or " send a turn-on signal to the TV in room 2 using the remote control in room 3". To support these features in the proposed architecture, a vertically configurable architecture is considered, which means that the application programmer can perform unit-base device programming or group-base device programming, plus logical unit programming, for example, a room or a group of devices related with a service. For implementation, the proposed home network consists of a backbone network and several subnets. IEEE1394 is used as the backbone protocol and each subnet accommodates several homogenous local networks and un-networked ubiquitous devices. In order to connect these subnets to the backbone network, a new subnet gateway is proposed and implemented.

- Messaging system to meet real-time and QoS constraints
  As shown in Table 1, the LonTalk protocol supports priority-based messaging and rigorous timing. Plus, according to the application agent, a real-time response

based on the interaction between non-real-time devices and real-time devices is also needed. To enhance this feature in the proposed middleware, an asynchronous messaging mechanism is implemented that is a modified version of the event channel concept from Real-Time CORBA [12]. Since multimedia communications for home theater services inevitably need a mechanism for guaranteeing QoS [13], the features in IEEE1394 and HAVi are used in the proposed middleware.

- Decentralized resource repository
  The current status and computing resource information for all connected devices should be monitored and managed in a resource repository for the application agents. However, if all the agents look up the necessary information from only one centralized resource repository, as in the JINI model [9], then the traffic for looking up the resources is centralized and the network frequently becomes unreliable when too many active agents are running on the network [15]. Therefore, to prevent this situation, a replication model of a resource repository is suggested, where each subnet has it's own resource repository yet all repositories in the network always have the same content.
- Minimizing the overhead despite multiple protocols and ubiquitous device support
  To develop a practical middleware, the reuse of features in existing low-level protocols is more important than implementing from a scratch-base. Although the software architecture of the proposed middleware is not dependent on a specific low-level protocol, it was decided to use the IEEE1394 protocol as the base protocol for the backbone network, thereby enabling the use of all the features of IEEE1394, such as an addressing scheme, dynamic reconfiguration, and multimedia communication through isochronous channels. Consequently, only one device driver and a protocol adaptor module are added to the middleware layer for supporting the new protocol in the proposed middleware.
- Room-based self-management architecture to reduce complexity and increase reliability
  In a traditional house, a room is one of the important management units from several aspects, such as management complexity, security, and reliability. Therefore, a room-based self-management architecture is definitely required to accommodate traditional management customs in a house.

3 Vertically Configurable Network Architecture: Target Hardware System Model

Base on the design guidelines discussed in section 2.4, a vertically configurable network architecture (Figure 1 shows the physical topology of the proposed architecture) is presented. As shown in Figure 1, the main criterion for dividing a home network into subnets is not a homogenous subnet with a protocol, but rather the physical structure of a traditional house. For example, a house consists of several subdivisions, such as rooms, the kitchen, and so on. This physical structure then becomes the main criterion for dividing the entire home network into several logical subnets. Each subnet only has one subnet gateway, called a ROOM-BRIDGE server and supports several homogenous local networks, such as a LonTalk network, IEEE1394 network, and wireless network.

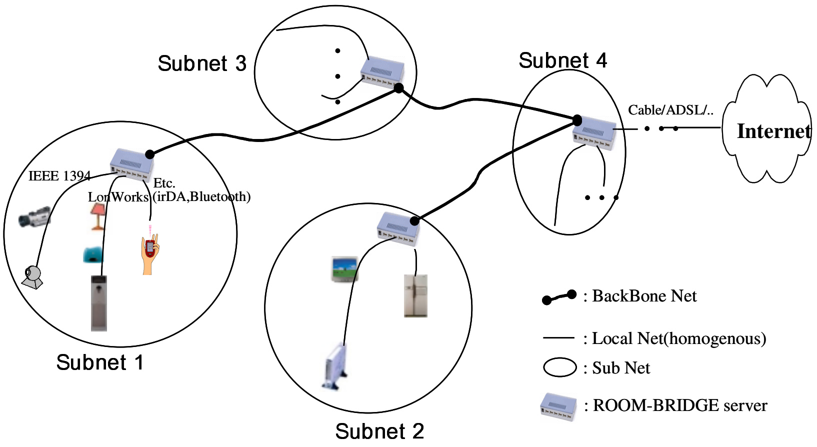


Fig. 1. Physical topology of proposed home network architecture

All subnets are connected to an IEEE1394-based backbone network via a ROOM-BRIDGE server. As such, the topology of the proposed network is dependent on the characteristics of the backbone network, i.e. flexible daisy chaining with IEEE1394. In particular, all the local homogenous networks attached to a subnet gateway need not be aware of other protocols or backbone middlewares and only need to focus on their own subnet gateway as the network management server for their local network. Accordingly, it is the ROOM-BRIDGE server that provides the interoperability between local networks and the backbone network. To fulfill this task, a middleware, called ROOM-BRIDGE, was designed and implemented for embedding in all ROOM-BRIDGE servers(the proposed middleware is explained in next section). To perform the functions of a ROOM-BRIDGE effectively, a specially designed embedded computer system was used in the current study as a ROOM-BRIDGE dedicated hardware platform, however, any computing device, such as a game machine (e.g. PlayStation 2) or PC (Personal Computer) that has enough computing power to execute the functions of the proposed middleware could be used as an alternative ROOM-BRIDGE server. Since the detailed architecture of the ROOM-BRIDGE dedicated hardware platform is outwith the scope of the current paper, it is not mentioned further.

4 Overview of ROOM-BRIDGE

The proposed middleware, called ROOM-BRIDGE, was designed and implemented to support the above design guidelines and proposed network configuration architecture, as shown in Figure 1. The following introduces the middleware components and operational principle for realizing the proposed home network. The components in the middleware layer, as shown in Figure 2, are the core components of ROOM-BRIDGE that need to be installed in all the subnet gateway servers, whether they are a ROOM-BRIDGE-dedicated hardware platform or full-powered computing devices, e.g. a game machine or PC. As such, the core components of the proposed middleware directly interact with the local networks through pairing a specific network manager with the appropriate device driver. For example, if a subnet wants to support a LonTalk network as a local network, the ROOM-BRIDGE server will pair up the LonTalk network manager with the device driver of a LonTalk adaptor board. In addition, ROOM-BRIDGE can also support several kinds of APIs(Application Program Interfaces) for programming application agents using the proposed middleware.

The middleware layer components can be divided into four categories: Device Status and Resource Registry, Device Proxies, Network Managers, and Priority-based Event Channel. Since the proposed middleware currently does not consider an isochronous communication channel and just uses the basic features in IEEE1394, isochronous applications, such as multimedia communications, only use the proposed middleware layer for sending control commands, then the IEEE 1394 network manager facilitates multimedia data communications by-passing the proposed middleware directly to an isochronous channel, as shown in Figure 2.

A brief summary of each group of components follows and more detailed descriptions are included in later sections as appropriate.

- Network Managers
  Each network manager (IEEE1394 network manager, LonTalk network manager, other network managers in Figure 2.) uses the subnet gateway as a network management server for each homogenous local network and also performs a filtered conversion of local network data to backbone network data. In particular, since the IEEE1394 network manager includes additional functions allowing other software elements to perform asynchronous and isochronous communication over 1394, command and network management data are processed by the proposed middleware components, whereas isochronous data, such as a multimedia data stream with QoS controls, are by-passed by the IEEE 1394 network manager to an isochronous channel.

- Device Manager
  Whenever a new device is connected into the home network, the Device Manager acknowledges it, initializes a device proxy for the physical device, and then registers it into the device pool.

- Device Proxy
  The Device Proxies in the device pool can be categorized into two groups: One group is real device proxies mapped one-to-one with physical devices, whereas the other is virtual device proxies mapped one-to-many or with non-physical devices, e.g. application program tasks or user program tasks, such as graphical user interface tasks, intelligent agents for instruments or the autonomous control of physical devices.

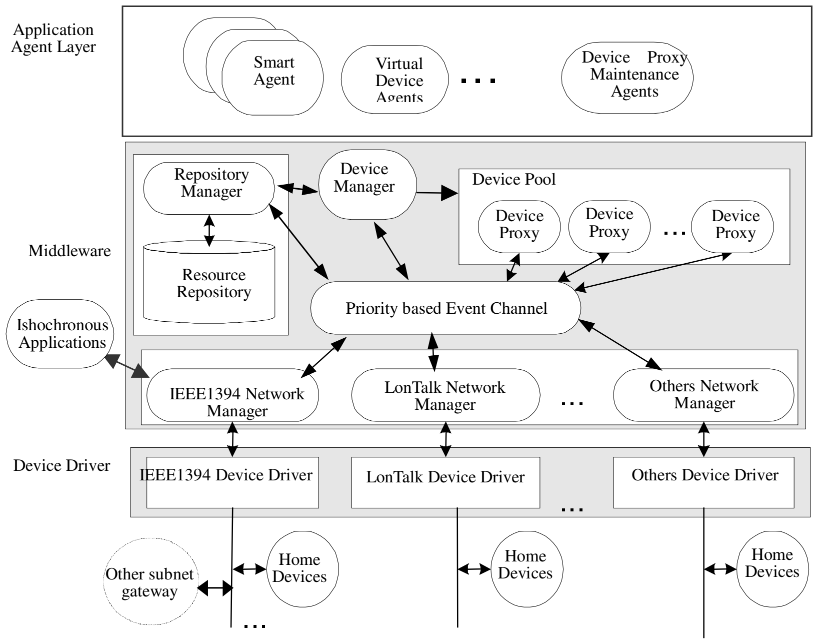


Fig. 2. Overview of ROOM-BRIDGE as middleware embedded in subnet gateways

Each device proxy mapped with a physical device in a one-to-one manner must have remotely invocable functions for monitoring and controlling the physical device without considering low level communication protocols(this protocol conversion is part of the network manager). A device proxy also has abstracted I/O variables for each active device in the home network. In addition, physical device proxies can act as a control and monitoring server to distributed clients by supporting callback functions and remote methods.

- Resource Repository and Repository Manager
  The repository manager plays the role of maintaining the status information on all the devices in the home network and computing resources for looking up remote methods. Since at least one repository must exist in each ROOM-BRIDGE server as a subnet gateway, the home network has the same number of repository managers and subnets. The device status information and related operational methods for monitoring and controlling each device are registered in the repository data base and managed by the repository manager in a subnet gateway. The resource repository database is a temporal and memory resident database, plus it always reflects real-time status information on all devices in the home network. To achieve this, the tuple - space model [10] and space programming concept in Java [11]are used. Further details are included in the following section.

- Priority-based Event Channel
  All events in the proposed home network are processed in an asynchronous manner. Therefore, an event channel with a priority queue is the basic asynchorouns message sending mechanism in the proposed concept, plus it also includes an en-

hanced mechanism for the real-time response of higher priority events. All events in a home network are categorized into three groups; the first group comprises events for maintaining the integrity of the resource repositories, the second includes events for reporting a status change from physical device proxies to resource repository managers, and the third involves events with acknowledgements for controlling remote physical devices. Based on these categories, each event has its appropriate priority value, whether it is assigned statically or dynamically, and is sent to its destination through this event channel. Section VI presents more details on the proposed event channel.

5 Resource Repository and Resource Manager

As shown in Figure 1, the proposed home network consists of logically separated subnets and each subnet a certain number of application agents and proxies for controlling or monitoring physical home devices. Since a home network is a highly sophisticated network, if a tightly coupled process model were used between the application agents and the physical device proxies or a centralized server for looking up resources in the entire network were used, such as the name server in CORBA [10], the reliable operation of the entire home network would become a critical issue as the network could easily go into a failure mode or even shutdown due to too many agents or the sudden failure of the server. Accordingly, to prevent these situations and create a more reliable and fault-tolerant network, a replicated space model for the repository is proposed based on the tuple-space model [10].

As shown in Figure 3, each subnet has its own repository space, which is shared by several agents and proxies, and all repositories in the home network are interconnected to each other through the event-messaging model outlined in the following section. Therefore, an agent can autonomously manage a specific home device, room-net, or several other agents using the repository space wherein the status of all devices can be transparently accessed regardless of the agent's location. Because of the transparency of this location, a newly attached home device or agent can easily obtain information on another agent by either reading or taking the information from the repository space. Similarly, a newly attached home device or agent can equally also announce its information to all other repositories in the home network. Therefore, agents coordinating several devices can provide a more reliable and highly intelligent service as they can easily access information on devices even though the devices may be completely unaware of each other. As a result, since all agents are loosely coupled to each other, this makes the system flexible, scalable, and reliable.

In particular, a replication model of a resource repository for maintaining device status information and resource registry is suggested, meaning that each subnet gateway has its own registry data base and all the device registries in the home network always have the same content. Therefore, whenever any changes to a physical device occur, each device proxy directly sends this information to the resource repository in the immediate subnet, thereafter the repository manager broadcasts the information to all other resource repository managers in the home network.

To synchronize the content of the replicated resource repository space in a home network, a proportion of IEEE1394's higher priority asynchronous stream channel must be allocated whenever the network is initialized. If an IEEE1394 bus is initialized(IEEE1394 buses are used as the backbone in the proposed architecture), the connection manager in the IEEE1394 network selects a channel management node, then the node allocates a channel number for broadcasting the resource repository information over the home network. After the initial broadcasting, an exchange between all device proxies then occurs, item by item, through the normal data transfer channel.

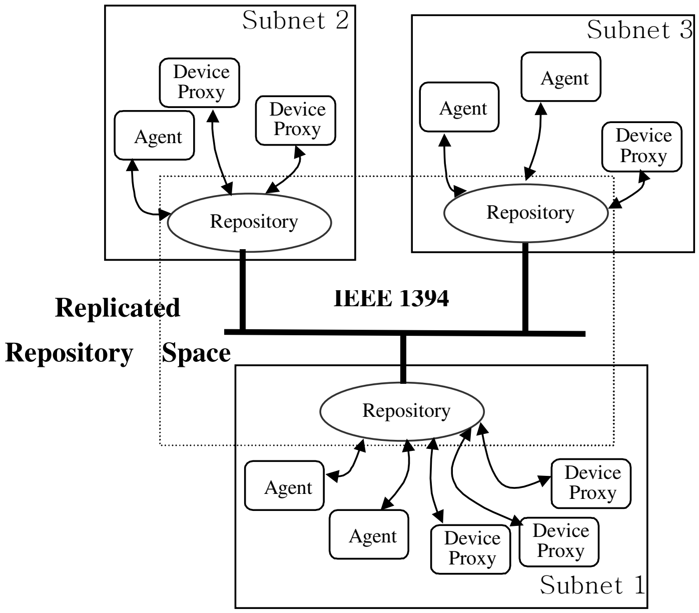


Fig. 3. Replicated Repository Space Mode

Despite the replication of the resource repository, the performance overhead caused by the task of synchronizing the resource repositories is not too high because the contents in the resource repository are highly abstracted and only include high-level information on each device. The following is an example of device status information in a resource repository:

```
("Light",Room2,5,"On",6,Time/Date)
("Light",Room2,7,"On",1,Time/Date)
("TV",Room1,5,"Off",6,Time/Date)
("AirCon",Room1,1,"On",115,Time/Date)
....
```

As shown in the above example, in the proposed resource repositories there is no local network specific information, such as IEEE 1394 bus management or LonTalk node information, only device specific information. Protocol-specific information is managed in each local subnet through specific network managers, such as an IEEE1394 network manager or LonTalk network manager, as shown in Figure 2.

Each device proxy includes some tuple-based operations [11,17] for retrieving and archiving device registry information between other agents in other subnets or device proxies inside a subnet. For example, a light management application agent can control the light status as follows: ***write("Light",Room2,5,"Off")***. To produce a more extensive operation, each operation can be used as a template-based query as follows, ***read("Light",Room2,?X,"On")***. The ?X means a variable in the query, as such, a set of items can be matched into the query, thereby facilitating a group-based query operation, for example, "turn off all lights in room 2". Further details of this resource repository architecture can be found in reference paper [17] written by the current authors. Consequently, this replication resource repository model provides all physical and virtual device proxies with transparent access to all device information in the home network, regardless of the physical location of each device.

6 Priority-Based Event Channel under IEEE1394 Asynchronous Mode

This section presents the internal architecture of the proposed event channel designed to implement an asynchronous message event handling mechanism and enhance the real-time response of events communicated using the proposed middleware. Although based on the publish-and-subscribe messaging model of CORBA and Real-Time CORBA [12,16], the concept is totally redesigned and modified for the IEEE1394 aynchronous transfer mode. As shown in Figure 4, the event channel consists of several priority queues, a dispatcher, and preprocess module. When generated, an event is transferred through a sequence of the preprocess module, priority queues, and dispatcher module.

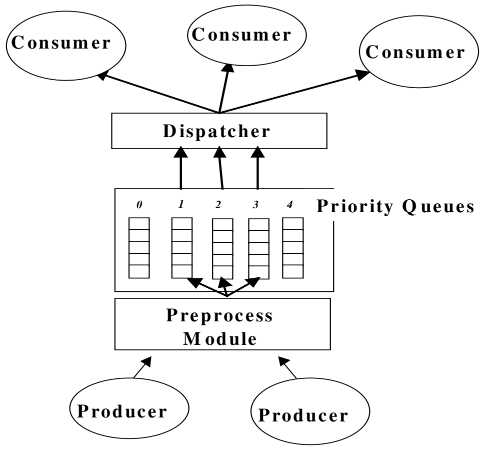


Fig. 4. Event Channel Architecture

6.1 Event Message Format, Producer, and Consumer

All software objects in the proposed home network, including the core components of the middleware and application agents, can be categorized into three object classes; event, producer, and consumer, as shown in Figure 4.

An event contains unit data for communicating messages between the software components in the home network, a producer is a software component that sends an event to others, and a consumer consumes the events that arrive. The following code example shows the class definitions of an event.

```
class Event{
        ObjectID PID;   // the reference of producer object  ID
        ObjectID CID;   // the reference of mapping consumer object ID
        EventType type; // event types such as repository, event with ACK or
                          witout ACK
        Priority value; // priority level of this event
        Int TimeSlice;  // optional, for ACK event
        Int MessageLength;
        Char* Message;
}
```

As shown in the above event definition, the producer decides all the slot values, such as the source and destination ID, event type, and priority, for an event with the help of the resource repository manager before sending the event to the event channel in the subnet. After sending, the successful delivery of the event is left to the event channel. Events can be classified into two types: events with acknowledgement(Figure 5) and events without acknowledgement(Figure 6). The former are used for controlling remote devices safely, whereas the latter are used to send newly updated device status information to all the resource repositories in the network.

6.2 Preprocess Module

The preprocess module has two functions. The first is run-time scheduling and the second is checking the timeout for events requiring ACK, as shown in Figure 5. For scheduling, the preprocess module determines the priority of an event using the event's scheduling information. The checking mechanism for events requiring ACK is also implemented in this module to assure the timing deadline of events. Figure 5 shows an event sequence diagram describing this function.

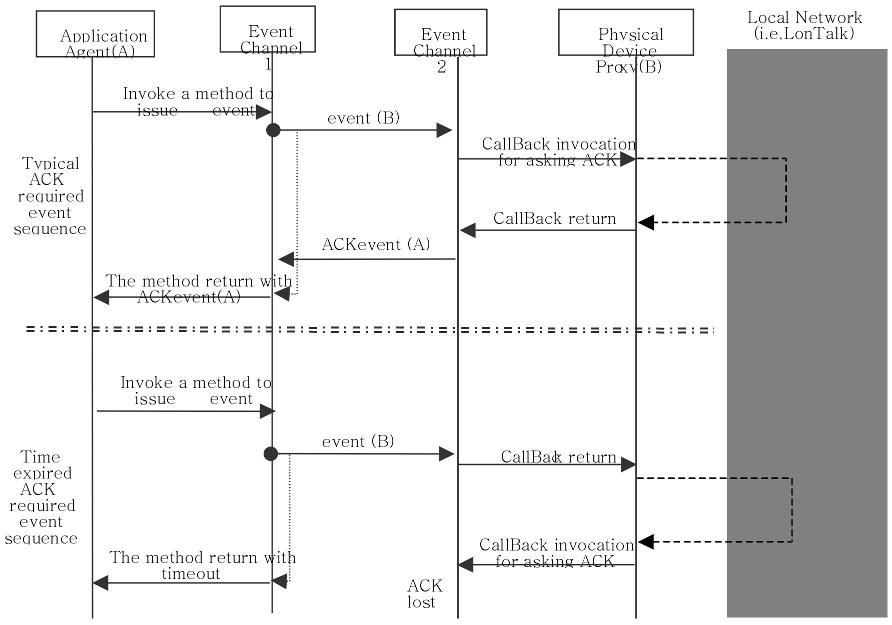

Application Agent(A)
Event Channel 1
Event Channel 2
Physical Device Proxy(B)
Local Network (i.e.LonTalk)
Invoke a method to issue event
event (B)
CallBack invocation for asking ACK
CallBack return
ACKevent (A)
The method return with ACKevent(A)
Typical ACK required event sequence
Time expired ACK required event sequence
The method return with timeout
ACK lost

Fig. 5. Event with Acknowledgement

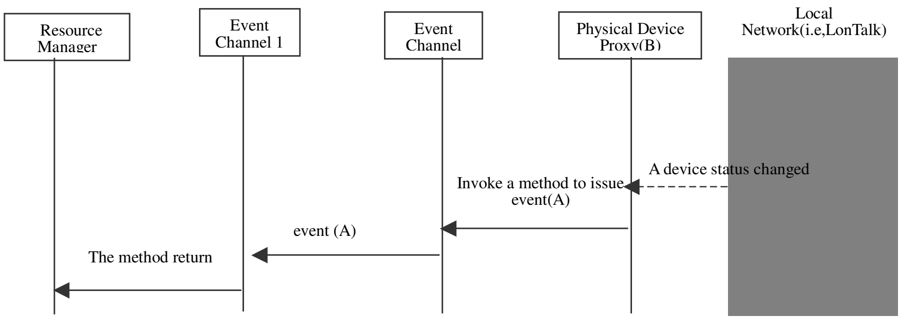


Fig. 6. Event without Acknowledgement

6.3 Priority Queues and Dispatcher

In Figure 7, each queue represents a preemption policy for each consumer priority level. The dispatcher module in Figure 7 uses threads to dispatch the events in the priority queues to the consumer. The proposed architecture allocates a thread or thread pool to each priority queue. Figure 7 shows the relation between the priority queues and the dispatcher.

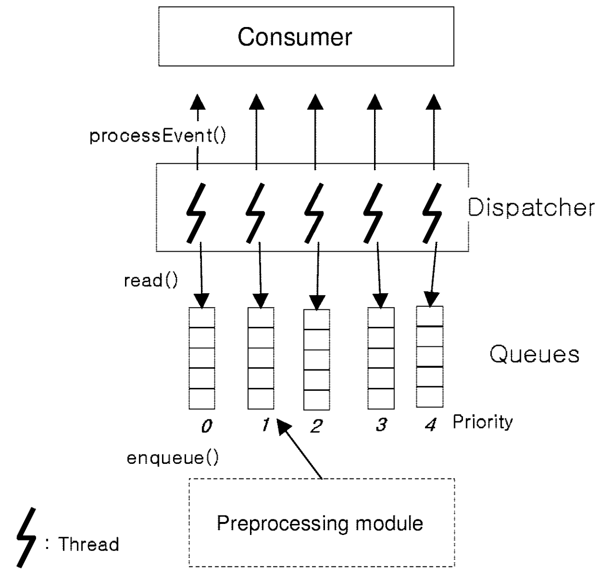


Fig. 7. Architecture of priority queue and dispatcher

In Figure 7, the preprocessing module determines the priority of an event, and then assigns the event to the appropriate priority queue. The dispatcher module consists of threads that have different priorities corresponding to the assigned priority queue. Each thread dispatches an event to a linked consumer and executes an event processing task called the `processEvent()` method of the consumer. Because each thread has associated priorities, higher priority dispatcher threads preempt lower priority dispatcher threads using OS kernel priority scheduling. By associating appropriate priorities to each thread, the dispatcher can take advantage of the OS kernel support of preemption.

7 Real-Time Device Driver for Backbone Protocol Adaptor

IEEE1394 is presently used as the backbone network in the proposed architecture, as a result, various multimedia data plus real-time event data for controlling and monitoring home devices can share an IEEE 1394 bus. Therefore, if a real-time response is not guaranteed in the device driver for the IEEE 1394 network adaptor, the middleware-level real-time features, such as the event channel, will be useless. In particular, because all event data must be sent through an asynchronous channel even though multimedia data can use an isochronous channel in an IEEE 1394 bus, the guarantee of a real-time response for asynchronous channel data is an important issue.

To solve this problem, a new real-time device driver is proposed for the IEEE1394 adaptor used as the backbone adaptor in the proposed architecture. The proposed IEEE1394 device driver architecture prevents packet-level priority inversion using priority-based queues. As such, a high priority packet processing time can be predicted and the round trip time and jitter time of a high priority packet can be reduced. Furthermore, to reduce interrupt latency, which has an important effect on the real-time guarantee for a device driver communicating with network adapters through hardware interrupts, the processing time in the interrupt context can be reduced by processing received packets in the user-level threads rather than the ISR(Interrupt Service Routine). Accordingly, the proposed IEEE1394 network device driver architecture can guarantee real-time characteristics and be applied to the device driver of a subnet gateway because IEEE1394 is used as the backbone protocol.

In the device driver, to prevent packet-level priority inversion, each unit in the link layer and transaction unit contains priority-based queues that enable packet processing according to the priority of the packet. The asynchronous link unit distinguishes the transmitted packets on the basis of their application priorities and classifies them into appropriate queues and threads. Finally, to reduce any interrupt latency, which has a significant influence on guaranteeing real-time characteristics in a device driver, the packet processing time in the ISR is reduced by processing the received packets in the user-level threads instead of the ISR.

Although the above concept is very important for achieving a real-time response in the proposed architecture, it is not included in the core software components of the proposed middleware layer. Consequently, a detailed discussion on this issue is outwith the scope of the current paper. For a more detailed description of the proposed real-time device driver refer to paper [19] also written by the current authors.

8 Implementation and Experimentation

8.1 Implementation

The core components of ROOM-BRIDGE, including the priority-based event channel [18], replication model of a resource repository [17], and real-time device driver for IEEE1394 [19] have already been developed and their performances individually evaluated. Accordingly, in this study a full-scale version was implemented in a testbed home network with the cooperative support of Kyungpook National University and ETRI(http://www.etri.re.kr/). As shown in Figure 8, three subnets were organized as the real testbed of a home network, and each subnet was connected to the IEEE1394 based backbone network through individual ROOM-BRIDGE servers as the subnet gateways. Two tiny diskless SBCs(Single Board Computers) based on X86 CPU were used as the ROOM-BRIDGE servers and dedicated hardware platform, plus a PC with a TI IEEE1394 adaptor board was used as the third ROOM-BRIDGE server and subnet gateway. The core components of ROOM-BRIDGE in subnets 1 and 2 were implemented under a VxWorks [20] real time operating system with JDK(Tornado for Java). For the backbone and subnet, MotionIO OHCI-104 IEEE1394 adaptors were installed into the ROOM-BRIDGE servers of subnets 1 and 2. In addition, a DI-10(for digital input devices) and DO-10(for digital output devices) were connected into each ROOM-BRIDGE server as the LonTalk-based control network nodes. Gesytec's LonTalk adaptor boards were used for accommodating a LonTalk network in the subnet. Two digital multimedia devices(Sony DCR-TRV510 digital camcorder and La Cie 37G Firewire Hard Drive as a digital storage for multimedia data) were used as the IEEE1394-based home devices and connected to each subnet, as shown in Figure 8.

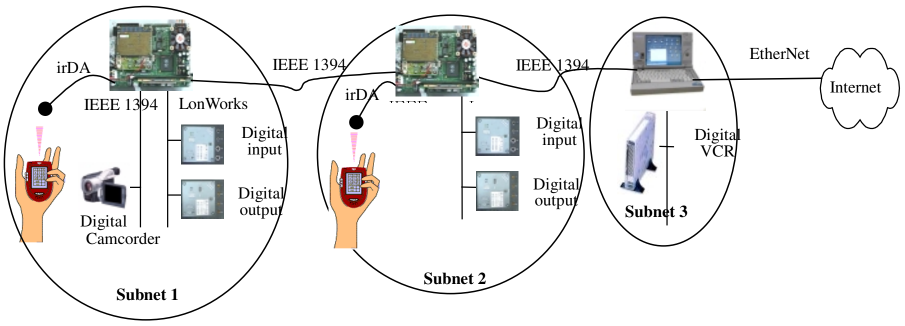


Fig. 8. Hardware configuration of prototype home network

The subnet gateway(subnet3 in Figure 8) used Windows NT with jdk1.3, which was connected to the Internet through an Ethernet adaptor. The core components of the ROOM-BRIDGE middleware, as mentioned in section 3.3, were embedded into each ROOM-BRIDGE server.

To implement the proposed concept more easily, a java-based programming environment was used for implementing the core ROOM-BRIDGE components, such as the event channel and resource repository architecture. To support the replicated repository space, the Space programming model [11] and Beans feature in Java were used. However, all the device drivers for each protocol adaptor were implemented in C and connected into Java using JNI(Java Native Interface).

9 Conclusions and Future Work

This research proposed a new design architecture and related middleware, called ROOM-BRIDGE, for a home network that needs to support ubiquitous computing and consumer devices and application-level multi-agents to meet real-time and distributed constraints. The proposed architecture was tested using a proof-of-concept prototype based on a testbed home network, and showed promise for improving home network service performance under real-time distributed environments. The full functionality of the proposed ROOM-BRIDGE architecture is still being investigated.

According to the evaluation results, the benefits of the proposed architecture can be summarized as follows:

- No extra device-level burden or consideration of consumer devices for supporting the proposed middleware: Since the core components of ROOM-BRIDGE are only installed and work on the level of subnet gateways, called ROOM-BRIDGE servers, ubiquitous devices as leaf nodes of a home network do not require any extra embedded module, except for a network interface to support the proposed middleware.

- Easy support of heterogeneous protocols and ubiquitous devices whether or not they are a networked device: Even though the necessity of supporting a new protocol occurs in a subnet, only a network manager component and device driver for the protocol adaptor are added to the ROOM-BRIDGE server, thus no modification or reconfiguration is necessary for the entire network.
- Physical and logical group-based device commands: Because a home network consists of several vertical groups of subnets, such as rooms, the kitchen, and so on, an agent or home member can issue logical group-based device controlling commands, for example “turn off all lights in room 5”
- Reliable and fault-isolated architecture: Due to the replication model of the resource repository by room in the proposed architecture, the performance of the entire home network is less affected by the overload or fault of an individual subnet. In addition, the performance of the entire network is not linearly affected by an increasing ratio of application agents.

In further studies, isochronous features, such as those of HAVi spec. [6] in the proposed middleware, will be considered for enhancing the features of multimedia data communication through the isochronous channel of IEEE1394, along with the planning and design of an agent-level middleware as an upper layer application of the ROOM-BRIDGE components.

Acknowledgement. The work reported in this paper was partially supported by ETRI(Electronics and Telecommunications Research Institute) from 1998 to 2001, and IITA(Institute of Information Technology Assessment) from 1998 to 1999.

References

1. Gerard O’ Dricoll, Essential Guide to Home Networking Technologies, Prentice Hall PTR,2000
2. Dutta-Roy.A, "Networks for Home", IEEE Spectrum, Volume 36, December 1999.
3. E. Douglas Jensen,” Scaling up Real-Time Computing to Higher Levels in the Enterprise: A Grand Challenge,” IEEE Workshop on Real-Time Mission-Critical Systems, Nov. 30, 1999
4. LonTalk Protocol Specification Version 3.0, 1994.
5. IEEE1394, Std for High Performance Serial Bus, 1995
6. Specification of the Home Audio/Video Interoperability (HAVi) Architecture Version 1.0, January 18 2000.
7. Tim Kowk,” *A vision for residential broadband services: ATM-to-the-Home*,” IEEE Network, 9(5), Sept.-Oct. 1995.
8. Richard Greenane and Simon Dobson,” Integrating LonWorks into an open systems control environment,” LonWorld’99, Echelon, 1999.
9. Jini Architecture Specification Revision 1.0, Sun microsystems, Jan. 1999.
10. N. Carriero and D. Gelernter, “Linda in Context,” Comm. ACM, vol.32, no.4, Apr. 1989.

11. E. Freeman, S. Hupfer, and K. Arnold , JavaSpaces Principles, Patterns, and Practice, (Addison-Wesley, 1999)
12. T.H. Harrison, D. L. Levine, and D. C. Schmidt, " The design and performance of a real-time CORBA event service," Int'l Conf. On Object-oriented Programming, Systems, Languages and Applications, 1997.
13. L. Bergmans, A. van Halteren, L. Ferreira Pires, M. van Sinderen, M. Aksit," A QoS-control architecture for object middleware,"7th Intl. Conf. on *Interactive Distributed Multimedia Systems and Telecommunication Services* (IDMS 2000)
    Enschede, Netherlands, October 2000 LNCS 1905, Springer, 2000, 117-131.
14. L. Bergmans, M. Aksit," Aspects and crosscutting in layered middleware systems," *Reflective Middleware* (RM 2000) workshop held in conjunction with the IFIP/ACM Intl. Conf. on Distributed System Platforms and Open Distributed Processing (Middleware 2000) New York, USA, April 2000.
15. V. Kalogeraki, P.M. Melliar-Smith, and L.E. Moser. *Soft Real-Time Resource Management in CORBA Distributed Systems*. In Proceedings of IEEE Workshop on Middleware for Distributed Real-time Systems and Services, pages 46--51. IEEE, December 1997. San Francisco, CA, USA.
16. G. Banavar, T. Chandra, R. Strom and D. Sturman, "*A case for message oriented middleware*", Proc. Of Symp. On Distributed Systems, DISC99, Bratislava, September 1999, LNCS. Vol. 1693.
17. Jae Chul Moon and Soon Ju Kang, " Multi-Agent Architecture for Intelligent Home Network Service Using Tuple Space Model," IEEE Trans. Consumer Electronics Vol 46, Aug. 2000.
18. Jae Chul Moon, Jun Ho Park, and Soon Ju Kang, "An Event Channel-Based Embedded Software Architecture for Developing Telemetric and Teleoperation Systems on the WWW," Proc. IEEE Real-Time Technology and Applications Symp., Jun. Vanquaver, Canada 1999.
19. Dong Hawn Park and Soon Ju Kang,"IEEE1394 OHCI Device Driver Architecture for Guarantee Real-Time Requirement," 7th Intl' Conf. on RTCSA 2000,Cheju, KOREA
20. VxWorks Programmer's Guide, WindRiver Systems, Mar 1997.
21. L. Wang, S. Balasubbramanian and D. H. Norrie, " Agent-based Intelligent Control System Design for Real-Time Distributed Manufacturing Environments," Agent-based Manufacturing Workshop – Autonomous Agents '98, Minneapolis/St. Paul, May 9-13, 1998