Design of Home Network Architecture using ACE/TAO Real Time Event Service

Eko Henfri Binugroho [1], Young Bong Seo [2], and Jae Weon Choi [3]
[1] Electronics Engineering Polytechnic Institute of Surabaya
[2] Innovation Center for Engineering Education, Pusan National University
[3] School of Mechanical Engineering, Pusan National University
E-mail: sragen@eepis-its.edu

Abstract

This paper proposes a home network design based on publisher/subscriber architecture which is developed using ACE/TAO Real-time Event Service (RTES) as the middleware platform. This design addresses a feature to support a real-time implementation for home network application such as home automation. Home network participants have been classified into several components based on consumer and supplier implementation in the ACE/TAO RTES in order to simplify the design. To optimize the network utilization, events are filtered based on their type and source for each publisher and subscriber. To deal with heterogeneous type of home appliances, event header information has been extended to wrap more information. Each of events can be configured with a specific scheduling and priority setting to meet its quality of service (QoS) according to the requirement. Network performance in handling an increasing number of consumer or supplier has been evaluated and show an acceptable result.

Keywords: *Home Network, ACE/TAO, RTES, QoS.*

1. Introduction

Home networking development has an objective to improve the quality of human life to be more convenient by providing some services such as remote control and remote monitoring, home automation, home security management, home multimedia network, etc. Nowadays, home networking technologies and capabilities are receiving an increased attention from consumers, software developers, hardware manufacturers, and service providers. The robustness of Internet Protocol (TCP/IP) has contributed to its success in the internet environment, and the role of this kind of communication is already well established [1]. It seems that TCP/IP will become a de facto standard for connecting diverse home appliances throughout the home network. Furthermore the chip's development also became more and more aggressive and it gave a support to TCP/IP protocol. This fact makes an advantage in developing home network appliances using TCP/IP as the main protocol.

Home appliances will increase in complexity, and it needs higher connection speed to transfer its information. Many wide-ranging applications in home network development have been proposed, such as; Jini [2], LnCP [3], UPnP [4], ECHONET [5], DLNA [6], ZigBee [7], LonWork, X10, etc. However there are heterogeneous technical aspects in hardware and middleware architecture used on the home network infrastructure. They have many excellent designs and implementations, but collaboration of several architectures become difficult due the lack of the standard. A viable solution for home network platform is publisher/subscriber architecture. This architecture is chosen since it defines a communication model that can be implemented over many networks, transport protocols, and OS platforms [8]. Publisher/subscriber is already used in distributed real-time and embedded (DRE) systems which require middleware support for real-time transfer of control and data among large number of heterogeneous entities that coordinate with each other in a loosely coupled fashion [9].

In this paper we propose a home network infrastructure based on Common Object Request Broker Architecture (CORBA) middleware using ACE/TAO RTES one of the real-time publisher/subscriber based middleware. Similar CORBA based home network using IEEE-1394 network [10] has been reported, but it still need a dedicated communication wire to work. Normally, it takes several steps to accomplish the installation of a home network including pulling the connection wire, installing software, and configuring the system. The hardest task in this installation seems to be the cabling process, since a normal house typically does not have an existing network infrastructure installed on it yet. Furthermore, a modification of a network that based on dedicated connection will become an annoying problem in the future. To overcome this problem, in this paper we suggest a home network infrastructure using PLC Ethernet device as an alternative solution for the wired home network connection.

2. Overview of CORBA Event Service

2.1 CORBA COS Event Service

The CORBA Event Service provides a flexible model for asynchronous communication among objects. The standard CORBA operation invocation model supports two-way, one-way and deferred synchronous interactions between clients and servers. To alleviate the restrictions on the standard CORBA invocation models, CORBA Object Service (COS) Event Service was designed. In particular, the COS Event Service supports asynchronous message delivery and allows one or more suppliers to send messages to one or more consumers. Event data can be delivered from suppliers to consumers without requiring these participants to know each other explicitly. Suppliers use Event Channels to push the data to consumers. Likewise, consumers can explicitly pull data from suppliers.

However, COS Event Service still has limitation like no event filtering support. Most Event Service implementations deliver all events to all consumers connected to that channel. This lack of filtering eventually will increase system network utilization especially when multiple suppliers are involved. Beside that COS Event Service still has no support in configuration of different quality of service (QoS). All events will be treated equally with same priority, which make a difficulty in configuring several events that have different level of importance. COS Event Service still has difficulty in handling events that must be delivered within a specified deadline. Furthermore, COS Event Service did not address periodic task capability which supports event delivery at certain interval.

2.2 ACE TAO Real-time Event Service

The ADAPTIVE Communication Environment (ACE) is an object-oriented toolkit that implements fundamental design patterns for communication software, while THE ACE ORB (TAO) is a real-time implementation of CORBA compliant that built using the framework components and patterns provided by ACE. ACE/TAO is freely available and already used in many distributed projects and applications in diverse domains, including command and control systems, telecom, datacom, medical engineering, distributed interactive simulations, and financial services. ACE/TAO Real-time Event Service (RTES) is an enhancement of the push model of COS event service. Similar with the push model in COS Event Service like depicted in Figure 1, suppliers generate events and then push them to the Event Channel. Consumers became the target of the events, while Event Channel decouples suppliers and consumers by propagating events to consumers on behalf of suppliers.

Even though ACE/TAO RTES lacks pull of model support given by COS Event Service, it has several benefits such as prioritized dispatching within preemption classes, event data model, event filtering, event correlation, suspend/resume connection, and periodic event processing [11]. With event filtering/correlation, events can be filtered or correlated with other events based on their type or identifier. By using RTES, event channel subscriptions can supply different QoS parameters so that event delivery can be scheduled with fixed priority, earliest deadline first, least laxity first or maximum urgency first strategies [12,13]. These features will give a great beneficial in home network implementation in which heterogeneous type of suppliers and consumers can be treated in several QoS according to the design's scenario.

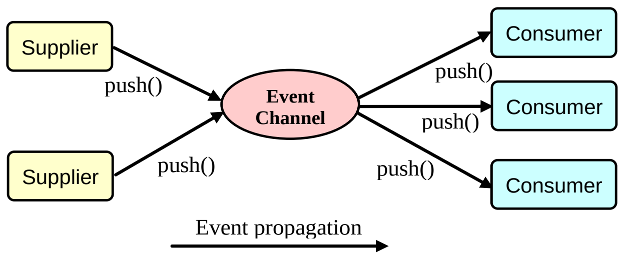

Figure 1. The communication model supported in ACE/TAO RTES

3. RTES-Based Home Network Design

3.1 Components in the Proposed Home Network

In the proposed home network, network participants are classified into several components based on the implementation of consumer and supplier program in the main Event Channel, see Figure 5. Classification of these components can be described as follow:

- *DeviceManager*: Designed to configure the connection of other components in the home network. Registration process of other component in DeviceManager can be set manually or automatically based on event source and event type registration table. It periodically checks the channel status, and if a fault occurs it will try to reconnect to the broken channel.
- *InputDevice*: Designed to publish event(s) to the Event Channel using information received from home network instrument or device connected to it such as sensors, infrared receiver, switch, etc.
- *OutputDevice*: This component receives and processes events from the channel and then uses them to drive the controllable home appliance connected to it such as lamp, fan, heater, alarm, etc.
- *IntegratedDevice*: Created in order to accommodate home appliance that not only can be controlled but also can provide data or information such as air conditioner, audio system, etc.
- *RemoteProgram*: Designed to provide an interface to the user to access home appliance remotely. By equipping it with an interface such as graphical user interface (GUI), user can manually give a remote command or to do a remote monitoring to other devices.
- *UserSpecified Application*: Configured to perform a

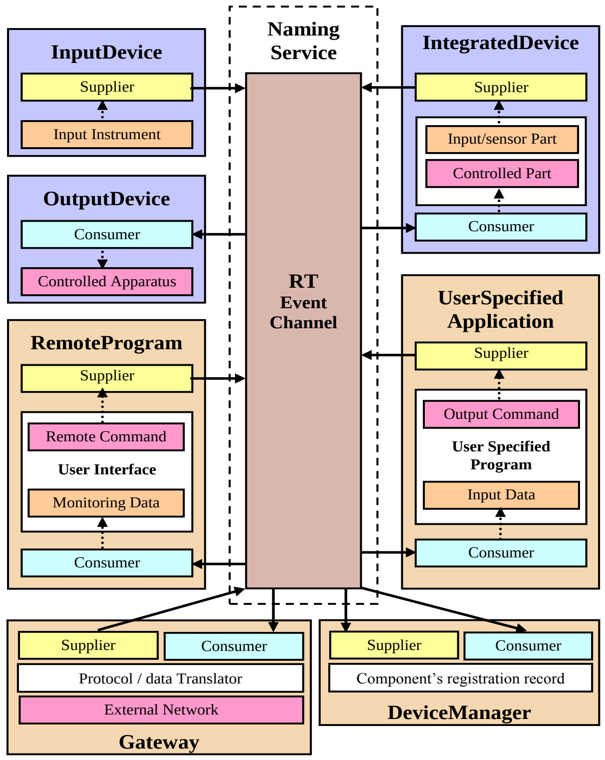

Figure 2. Components connection in to the main Event Channel.

special scenario such us home automation, security control and monitoring, environment messaging, etc. Within this component several devices can be collaborated to perform the desired scenario.

- *Gateway*: Designed to bridge the connection between Event Channel based home network with external or existing home network such as IEEE1934, ZigBee, X10, etc.

3.2 Extended Event Header Mechanism

By using RT Event Service, home network design is decoupled. It means all events to be sent from one to another will be transferred through Event Channel in which sender (supplier) and receiver (consumer) does not know the location of each other explicitly. This design makes the Event Channel's architecture flexible, but has difficulty to be implemented for dedicated connection in which the supplier wants to address the data to the desired destination. Event by default does not have information about destination address to where it should be sent. In RT Event Service, event filtering feature is used to determine event delivery instead of address routing explicitly. Each event in RT Event Service has event header that contains source and type information, thus Event will be delivered by Event Channel only to consumer(s) that already subscribe event with the same source and type. Supplier does not need to know where the consumer is, but on the other hand consumer already orders event(s) that it wants. Event Channel only uses event source and event type for filtering, which makes it becomes difficult to describe heterogeneous home appliance. Since each of them is 4 bytes in length, its content can be extended to make device description in home network implementation easier, see Figure 3.

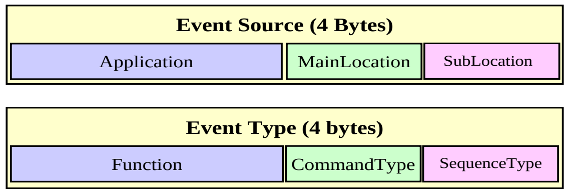

Figure 3. Information extension in the event source and event type variable.

To do this, byte manipulation can be use to insert several information into them. Event source can be expanded into two types of information, these are Application and Location. Application information is used by both of supplier and consumer to describe specific application in which it wants to be implemented. Location is used by supplier to define its location in the house, and consumer uses it to determine from which location event can be accepted. On the other hand, event type is expanded into three information, these are Function, CommandType and SequenceType. Function is used by supplier event to describe the function that it wants to command. Consumer use Function to describe the function that is implemented on it. CommandType is used to describe type of command from particular event. SequenceType describes the type of sequence which wraps the data in the event payload.

3.3 Structure of Event Data Payload

By default, RT Event Service uses sequence of octet as event payload which is described in the *RtecDefaultEventData.idl*. This payload is used more often by high-performance applications, but it is difficult in home network implementation since it does not describe any specific data type other than octet. CORBA Any is the most flexible data type for wrapping information, but it has the worst performance compared with other data type. Fixed structured event has better performance than event type Any, but it less flexible in handling event with different structure, or the same structure but with different data type. As a solution, a combination from structure and union can be used. Event is wrapped in the same structure, but the data inside the structure uses union that can wraps several data types. To support more then one data in one event, sequence of union is used. The event data structure used in the home network is shown in the Figure 4. This design is more flexible then the fixed structure and has better performance then data Any. Flexibility of this structure can be increased by adding more data type in the

```
enum DataType
{ xBOOL,xCHAR,xSHORT,xLONG,xFLOAT,xLLONG,xDOUBLE
};

union xData switch (DataType)   //union definition
{ case xBOOL:       boolean    BData;
  case xCHAR:       char       CData;
  case xSHORT:      short      SData;
  case xLONG:       long       LData;
  case xFLOAT:      float      FData;
  case xLLONG:      long long  LLData;
  case xDOUBLE:     double     DData;
};

struct RtecEventData              //payload data structure
{  long      TimeStamp;
   string    message;
   sequence <xData> HomeNetData;
}
```

Figure 4. The structure of data payload used in the file *RtecDefaultEventData.idl*.

union definition but the performance of event delivery will decrease. This is a flexible option in the trade between flexibility and performance.

Any data defined in the union's definition can be inserted to the sequence and different data type also can be inserted in the same sequence. Union discriminator will changed automatically if different data type is inserted. It enables consumer to send dynamic data type in each event delivery if needed. C++ mapping for Interface Definition Language (IDL) unions defines a class that provides accessor methods for the union discriminator and the corresponding union fields which is named _d. This accessor is used by consumer implementation program to determine the type of each data inside the sequence from the received event. SequenceType information that stored in the event header is used by supplier to mark data in the sequence as HOMOGENEOUS if data in the sequence have the same type, or as HETEROGENEOUS if its data type is various.

3.4 Registering Component to the Network

Event Channel activation will trigger the DeviceManager to be active too, so it will be ready for registration process of any other components. DeviceManager itself will subscribe events to the Event Channel without event source filtering, thus it can transfer event with all components in the network. It only subscribes several specific event types, mainly for registration and configuration process. When a component wants to connect to the Event Channel, it connects to the secondary channel first then sends its registration event including its ComponentType information to the Device Manager. Component marks the event information as blank by setting all bits in the information field as 1 to tell DeviceManager to perform configuration process.

DeviceManager by default will use manual configuration using user interface program, and let user specifies the location or application of the new component attached into the network. If user wants the automatic

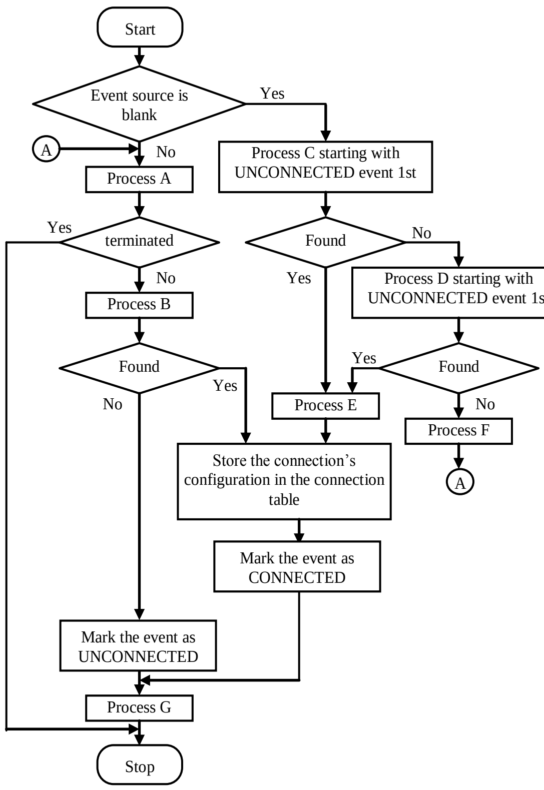
Start
Event source is blank
Yes
No
A
Process A
Process C starting with UNCONNECTED event 1st
Yes
terminated
No
Found
No
Yes
Process B
Process D starting with UNCONNECTED event 1st
Found
Yes
Yes
Found
No
Process E
No
Process F
Store the connection's configuration in the connection table
A
Mark the event as CONNECTED
Mark the event as UNCONNECTED
Process G
Stop

Figure 5. Registration process for OutputDevice, InputDevice and IntegratedDevice.

configuration to take place, then the grouping method should be chosen, based on location or application. If by mistake the user lets the setting on automatic instead of manual mode, and an undesired automatic setting already performed, user still can rollback this setting. DeviceManager will alter the undesired setting with user supplied setting, both inside its registration table and at the desired component QoS information. By default DeviceManager groups components based on the location information first, if the appropriate component is not found, it will try to connect the component based on the application information. User still can alter the order of this process if desired. If component uses multiple events in its registration, then each event will be treated using a same process sequence.

To describe the registration process which is described in flowchart in Figure 5, actions that performed by DeviceManager can be classified into:

- *Process A* : Check if any component with the same QoS dependencies and publications exist. If it exist then inform user with interface program that the same components is detected, then will terminate the automatic configuration and let the manual setting to be performed.
- *Process B* : Search event(s) with the same event

header and event type.

- *Process C* : Search event's Location for event(s) with the same event type.
- *Process D* : Search event's Application for event(s) with the same event type.
- *Process E* : Copy event source information from the event found in the searching process to the new one (component will use the new event source to connect with the main Event Channel).
- *Process F* : Inform user interface program that configuration process did not found any appropriate event to connect, give an option to user to insert information in the blank event source, or let DeviceManager to insert a default value.
- *Process G* : Mark the corresponding component as REGISTERED and search RemoteProgram that supports registered event type. If the appropriate RemoteProgram is found, then send a message about the new component's information and its status, and ask RemoteProgram whether it wants to remote the new component or not.

5. Testbed Configuration and Experiment Result

5.1 Testbed Configuration

The proposed home network is evaluated using a testbed within three computers, heater, fan and several sensors to perform a simple room temperature control application, see Figure 6. Testbed is designed using several computers and communication devices in order to show the flexibility of the home network design, since component can be distributed easily in any available computer. If the hardware configuration is changed, implementation program remains the same, except its implementation is connected to I/O device. However, only small modification is needed which is the just com port number readjustment. USB Wi-Fi and Ethernet adapter connected to PLC Ethernet in PC-1 are bridged, thus all computers logically use a same network.

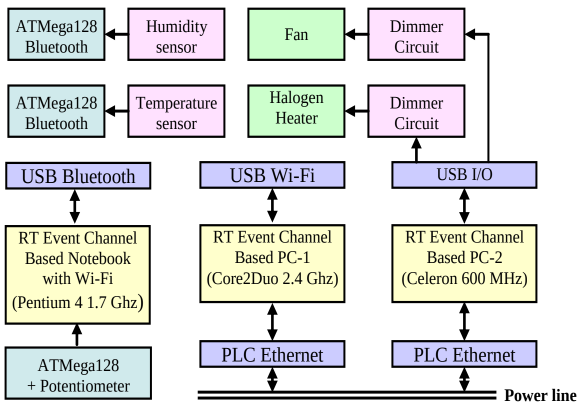

Figure 6. Block diagram of the home network testbed.

5.2 Communication Performance

In this experiment, throughput and latency from the consumer and supplier are measured. Event which is used in this measurement contains a sequence with two variables inserted and one string with ten characters. Figure 7 shows consumer's throughput and latency which is measured using single supplier. The number of consumer is increased from one to twenty with all of them have the same priority. By reversing the scenario between supplier and consumer, supplier's performance is measured and its result is shown in Figure 8. From the experiment, we found that the supplier has better throughput and latency then the consumer part. But consumer has better ability in maintaining total throughput when its number has been increased.

To measure the real-time performance of the network, two types of consumers are used. The first one uses high priority RT_Info, and the other one uses low priority. One supplier is used to send an event every 10 ms. Low priority consumer is increased from one to twenty, when the high priority consumer remains one. From the experiment's

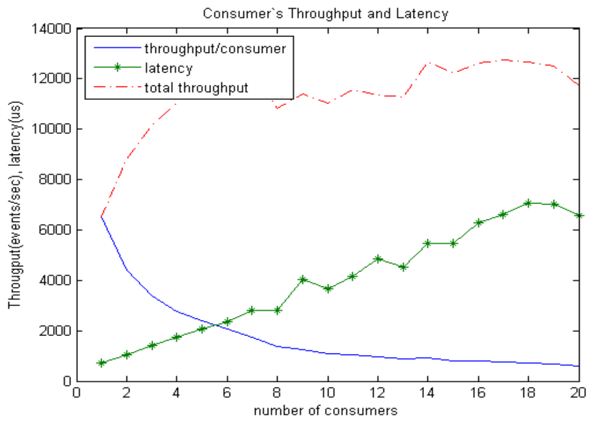

Figure 7. Consumer's performance.

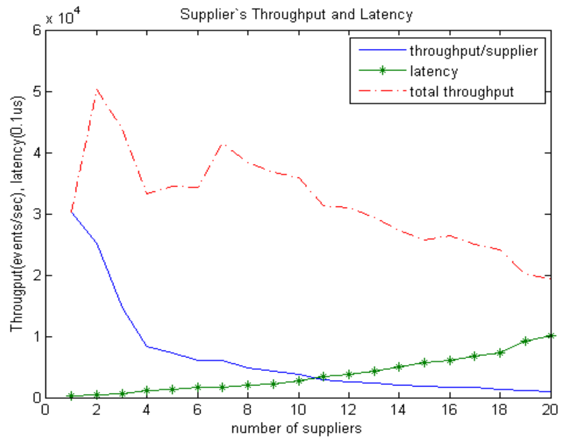

Figure 8. Supplier's performance.

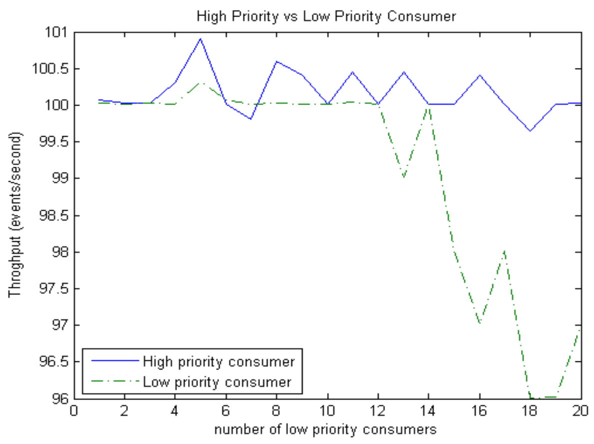

Figure 9. Comparison of high and Low priority consumer.

result shown in Figure 9, we found that the high priority consumer still maintains the throughput around 100 events/second, even though the number of low priority consumer has been increased.

6. Conclusions

RT Event Service supports a decoupling communication between supplier and consumer that makes it very flexible in developing distributed environment like home network. RT Event Service also supports real-time capability as an added value that can be adapted in home network application such as home automation. However, its hardware requirement is still high for current small embedded system that make it difficult to be implemented in an efficient way especially for handling home appliance that just requires a low speed data transfer.

In the proposed home network, two Event Channels are used to give higher reliability in terms of channel error. Default event data payload is also modified using structure of union sequence. It has less performance but with higher flexibility then the default one, but it stills a viable choice since current home network implementation does not need hard-real-time requirements yet. Event header is extended to define more specific information in describing home network event. Automatic device configuration algorithm has been designed. It can perform an automatic event source assignment, but real-time configuration still needs to be set manually and it is left as a future work.

References

[1] G. T. Edens, “Home networking and the CableHome project at CableLabs”, *IEEE Communication Magazine*, vol. 39, pp. 112-121, June 2001.

[2] R. Gupta, S. Talwar, and D. P. Agrawal, “Jini home networking: a step toward pervasive computing”, *Computer*, vol. 25, pp. 34-40, August 2002.

[3] S. C. Kim, J. A. Park, K. W. Lee, and S. W. Lim, ”Home networking digital TV based on LnCP”, *IEEE Transactions on Consumer Electronics*, vol. 48, pp. 990-996, November 2002.

[4] S.-H Rhee, S.-K. Yang, S.-J. Park, J.-H. Chun, and J.-A. Park, “UPnP home networking-based IEEE1394 digital home appliances control”, *Advanced Web Technologies and Applications*, vol. 3007, pp. 457-466, 2004.

[5] http://www.echonet.gr.jp/english/index.htm.

[6] Digital Living Networking Alliance, *DLNA Home Networked Device interoperability Guidelines Version 1.0*, June 2004.

[7] W. K. Park, I. T. Han, and K. R. Park, “ZigBee based dynamic control Scheme for multiple legacy IR controllable digital consumer devices”, *IEEE Transaction on Consumer Electronics*, vol. 53, pp.172-177, February 2007.

[8] D. C. Schmidt and C. O’Ryan, “Patterns and performance of distributed real-time and embedded publisher/subscriber architectures”, *Journal of Systems and Software*, vol. 66, no. 3, pp. 213–223, June 2003.

[9] G. Deng, M. Xiong, and A. Gokhale, “Evaluating real-time publish/subscribe service integration approaches in QoS-enabled component middleware”, *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pp. 222-227, 2007.

[10] J. Y. Oh, J. H. Park, G. H. Jung, and S. J. Kang, “CORBA based core middleware architecture supporting seamless interoperability between standard home network middlewares”, *IEEE Transactions on Consumer Electronics*, August 2003.

[11] D. C. Schmidt and S. Vinoski, “Object interconnections”, *The OMG Events Service Column 9*, February 1997.

[12] C. D. Gill, D. L. Levine, and D. C. Schmidt, “The design and performance of a real-time CORBA scheduling service”, *Journal of Real-Time System*, vol. 20, no. 2, pp. 117-154, March 2001.

[13] H. M. Huang and C. D. Gill, “Design and performance of a fault-tolerant real-time CORBA event service”, *Proceedings of the 18th Euromicro Conference on Real-Time Systems*, pp. 33-42, 2006.