# SYSTEM ARCHITECTURE AND IMPLEMENTATION OF A CEBus/INTERNET GATEWAY

Joe Desbonnet and Peter M. Corcoran
Dept. of Electronic Engineering, University College, Galway, Ireland

**Abstract - Class structures and an object-oriented software framework to facilitate Internet access to CEBus networks are described. The Internet/CEBus Gateway consists of a server application and dynamic data structures which provide a real-time representation of a CEBus network. CEBus network traffic is continually monitored and interpreted and these data structures updated accordingly.**

**In addition the creation of "virtual" CEBus devices is supported. These "virtual" devices can interact with the real CEBus network. This allows the creation of "new" device personalities whose behaviour can then be characterised through interaction with a real CEBus network.**

## 1. Introduction

The Consumer Electronic Bus (CEBus®) is a multi-media LAN standard developed for home automation applications [1]. In earlier work we discussed some of the issues involved in providing access to a CEBus network from a conventional Web browser and Java® virtual machine JVM [2].

In this paper we look at the next step in linking a local control/automation network to the Internet, or an equivalent WAN. As we shall discuss shortly, we feel it is not very practical to attempt a direct mapping of network addresses and protocols between these two distinct networks. The functionality and operation of a local control network are somewhat different from those of a wider bandwidth WAN network. Thus we shall focus on the concept of an intelligent gateway whose function is to manage, broker and integrate network traffic between these two distinct categories of networks.

The functions of a home gateway can be sumarised as follows:

- acts as ***broker*** for WAN (outside world) access to local network (home).
- provides local (home) access to WAN ***services & resources.***
- implements ***security & encryption*** services.
- provides ***secure remote access*** across the WAN (outside world) to local network (home).

One service which we see as very significant in the context of home automation networks is the ability to access and download user-interface (UI) modules for home network applicances. This issue of providing a suitable user-interface to home automation networks is covered in a companion paper [3].

Note, that although we have taken the CEBus networking standard as the basis for the implementation of our gateway technology, it is not confined to operating with CEBus networks. In fact any home automation network which offers supports for the Common Applications Language (CAL) can be supported by our gateway technology.

## 2. Design Goals for a Home Gateway.

A central theme in our implementation of an Internet/CEBus gateway is that the system architecture should be lightweight and platform independent. This is particulary important in home automation applications where "cost is king". Thus although the penetration of desktop PC technology into the home market is increasing on a daily basis we still expect that most homes will not rely on a home PC as the home gateway. Instead we expect that a set-top-box style solution with a low-cost embedded CPU will satisfy this need for most end-users.

Furthermore we see two distinctly different approaches emerging in the field of home automation: one approach is to incorporate TCP/IP networking functionality into even the simplest of consumer appliances. There are several drawbacks to this approach. The current version of the Internet Protocol (IP) allocates 32 bits for addressing. As CEBus also requires 32 bits there is not enough room to map every CEBus address to a unique IP address. Also many embedded processors used in comsumer devices are not powerful enough to implement the IP protocol.

Manuscript received June 18, 1997

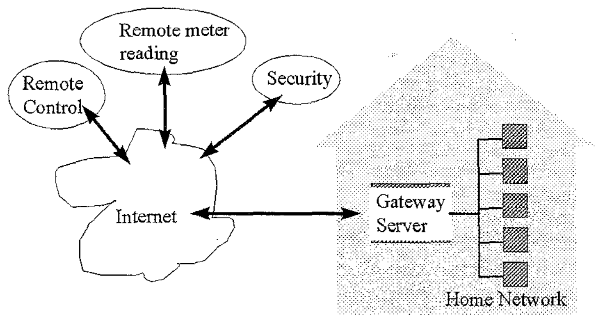


***Fig 1:** An Overview of the Home Gateway to the Internet; some typical user services are shown.*

The second approach, which we feel is more realistic, is to integrate existing established technologies, such as CEBus/CAL with the TCP/IP infrastructure. With this approach only one device - the gateway server - need be smart enough to implement the IP protocol, and only one IP number need be assigned per home. Current industry initiatives, such as the proposed *Home Plug and Play (HPnP)* standard [5], would appear to support our thinking in this matter.

### 2.1 Scalability of the Gateway Architecture

The prototype gateway has been developed on a standard desktop PC platform. However to support scalability it has been almost entirely implemented in the Java® language. This should greatly simplify the porting of the gateway to low-end embedded applications such as Set-Top-Boxes and Web-TV appliances.

### 2.2 Cross-Platform Portability

Again, as the core software components of the system are implemented in Java it should be readily portable across a broad range of desktop and embedded platforms. Some low-level interface drivers may need to be written, but as the system is built around TCP/IP network socket concepts and CEBus® devices compliant with the common application language (CAL) the necessary low-level software hooks should be quite generic.

### 2.3 Independent User Interface Modules

A further strategy to keep the gateway lightweight and scalable is the approach to user-interface issues. These are covered in more detail in a companion paper [3], but essentially the gateway interface only supports a simple device browsing service. Independent user-interface elements for each network device are loaded from external network URLs - typically in the form of home - interactive programlets, or HiPlets, as described in [3].

Thus the gateway can support a practically unlimited variety of user-interface components.

### 2.4 Internet and CEBus Standards

As was mentioned above our prototype gateway implementation is based on TCP/IP and CEBus standards. This ensures that it has a broad applicability in both industrial and home network applications. Note, however, that the gateway functionality is not confined to operating with CEBus home networks and can, in principle, support any home automation network which supports CAL product models and CAL messaging.

## 3. Software Architecture

A central theme in our implementation of an Internet/CEBus gateway is that the system software architecture should be lightweight and platform independent. To achieve these goal most of the key system components have been implemented in Java.

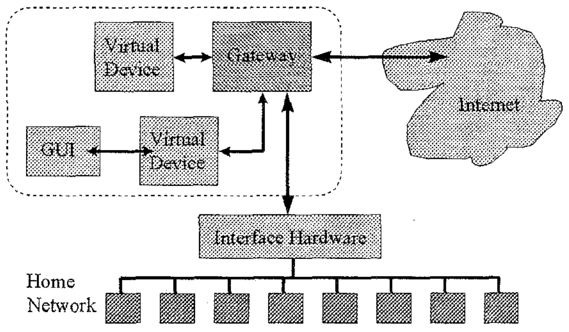


***Fig 2:** Overview of the Software and Hardware components which make up the Gateway.*

The primary components of our system are:

- The Internet/CEBus Gateway Server
- CEBus/CAL Class Library
- Virtual Devices
- Interface hardware

### 3.1 Internet/CEBus Gateway Server

The Gateway Server is a Java application that acts as an intermediatry between applications and the home automation network. The server communicates to the home automation network via a propritery interface device, somewhat akin to a conventional telephone modem. However this hardware is somewhat more flexible than a simple modem and can allow *virtual devices* to interact with the home automation network as if they were *real devices* and had direct hardware access to the network. This flexibility has some interesting implications with respect to developing new devices and/or CAL product models.

Services which are currently provided by the Gateway server to connecting applications include:

- full/filtered packet dump
- tx packet/tk_ack packet
- list network devices
- list/seek device state info
- list/seek house state info
- list/seek context state info
- select TCP/UDP broadcast
- network statistics
- MAC binding services

Note that much of the work on the gateway software is still in the development stages and that this list of services is changing quite frequently. However the above list reflects some of the more useful and important gateway services.

Applications request services from gateway by establishing a TCP (Transfer Control Protocol) socket and then exchanging a protocol similar to HTTP (Hyper Text Transfer Protocol or World Wide Web protocol) [RFC 1945, RFC 2068]. Using Secure Sockets Layer (SSL) full security can be achieved, which is vital for any remote control or remote metering applications. SSL will ensure that all transactions are encrypted. It also provides a means to authenticate the indentity of both parties by using public key cryptography certificates.

A lighter communication protocol using UDP (User Datagram Packets) has also been implemented. The UDP packets to not incurr the overhead of creating a TCP socket and thus is more suited to heavy load situations (eg routing packets). However UDP does not provide guaranteed delivery.

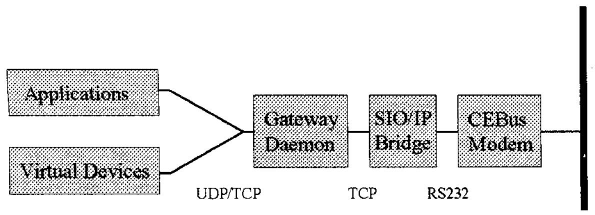


***Fig 3:*** *Structure of the Gateway S/W Components.*

As Java currently does not have a Serial I/O API communications with the CEBus modem is achieved using a simple server program written in C which channels serial I/O to a TCP socket. Java native methods could also be utilized to overcome this limitiation.

## 3.2 Virtual Devices and Networks

In the initial work to implement a prototype gateway one problem we encountered was a shortage of available CEBus compliant devices. Most available devices which employ the CEBus data transport are partly proprietary and as such are not fully compatible with the EIA standard.

To over come this problem and assist in testing and debugging our prototype gateway we set up a second desktop PC with a virtual network of CEBus devices. Network traffic from this simulated network of devices was broadcast over the powerline network, appearing to the gateway system as if a real powerline CEBus actually existed.

Virtual devices also provide a convenient mechanism to implement a GUI to a home device or sub system. For example a virtual device bound to a GUI interface running on a set-top box could act as the controller to the home security system replacing the normal keypad controller fixed to the wall.

## 3.3 Virtual CEBus Network: Class Libraries

A Java class library provides all the necessary tools to construct CEBus devices and network and implement CAL. The network can be run as a standalone simulation on a computer or bridged to other virtual or real networks.

Some of the main classes include: CEBusNetwork, CEBusDevice, CALContext, CALObject, CALIV. These are listed in Table I below with an explanation of the primary function of each class and their interrelationships as container objects.

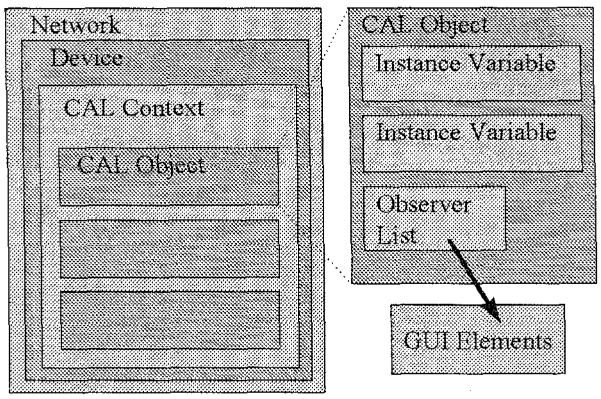


***Fig 4:** Relationships between the Main System Classes; Network, Device and Context Structures.*

| ***Class ID*** | ***Container for:*** | ***Other Functions:*** |
|---|---|---|
| CEBusNetwork | CEBusDevice | Handles routing of packets between real and virtual networks. |
| CEBusDevice | CALContext | Contains a list of CAL contexts and methods to handle a CAL packet. |
| CALContext | CALObject | |
| CALObject | CALIV | Most of the CAL parsing code resides here.<br><br>GUI components can also resister as an Observer of this object - thus when an IV changes they can be signaled to update. |
| CALIV | n/a | Data structure representing the name, label and value and other attributes of the Instance Variable. Also methods to manipulate CAL values (eg add, subtract) |

***Table I:** Main Gateway Classes*

| ***Class*** | ***Description*** |
|---|---|
| CEBusAddress | A data structure representing a CEBus Address (16 bit house code, 16 bit unit code) and methods to manipulate these addresses. |
| CEBusPacket | A data structure representing a CEBus packet |
| NetworkBrowser | The NetworkBrowser is a GUI element that allows a network to be graphically browsed. |
| DeviceBrowser | The Device browser is a GUI element that allows detailed inspection of a device. The Device Browser is normally invoked from the Network Browser. |

***Table II:** Supplemental Utility Classes*

### 3.4 Bridge/Router

The bridge/router is another Java application allowing the interconnection of two or more real or virtual CEBus networks, *eg.* linking buildings in university campus via its TCP/IP infastructure.

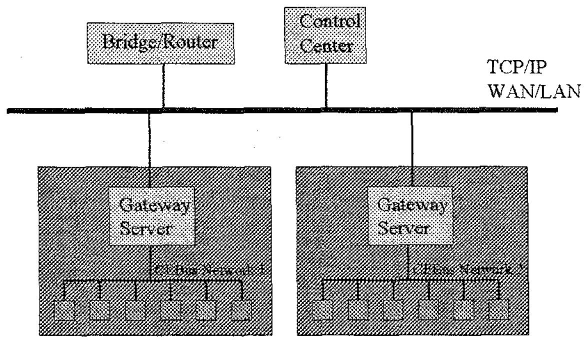


**Fig 5:** *CEBus networks internetworked using TCP/IP WAN and bridge/router application*

The bridge router application uses the lighter UDP based gateway protocol to ensure fast response times to CAL query packets. In Fig 5 two CEBus networks labeled 'Network 1' and 'Network 2' are connected to a TCP/IP WAN via the Gateway server. The current implementation of the Bridge/Router adopts a simplistic approach whereby a full packet dump is requested from both networks. A list of devices on each network is constructed and packets are forwarded accordingly.

### 3.5 Code Samples

The following Java code fragment illustrates the creation of a virtual CEBus network of two devices: a lamp and a dimmer switch. The reporting state of the switch is set so that whenever its 'current_value' changes a packet setting the lamp to the appropriate illumination value will be generated. Note: some declaration and initialization code has been omitted to improve clarity.

First a virtual CEBus network is created:

```
// Create CEBus Network
CEBusNetwork mynet = new CEBusNetwork();
```

The next block of code creates a light dimmer switch with house code 1, unit code 1.

```
// Create light  dimmer switch (house 1, unit 1)
//
CEBusDevice myswitch
                    = new CEBusDevice(1,1);
```

It is identified as device model "SW001" from a fictitious "Acme Inc." vendor.

```
// Create Universal Context  & Node Control
switch_univctx = new UniversalContext();
switch_nodectl = new NodeControl()
switch_nodectl.setIV("manuf_name","Acme
Inc.");
switch_nodectl.setIV("manuf_model","SW001");
switch_univctx.addObject(switch_nodectl);
myswitch.addContext(switch_univctx);
```

A lighting context is created with an AnalogSensor object as object #3 according to the CEBus specification.

```
// Create Lighting Context & Sensor object
switch_lightctx = new LightingContext();
sensor = new AnalogSensor();
switch_lightctx.addObject(sensor,3);
myswitch.addContext(switch_lightctx);
```

A lamp is now created with unit code 2.

```
// Create lamp  (house 1, unit 2)
//
Device mylamp = new Lamp(1,2);

// Create Universal Context & Node Control
lamp_univctx = new UniversalContext();
lamp_nodectl = new NodeControl()
lamp_nodectl.setIV("manuf_name","Acme Inc.");
lamp_nodectl.setIV("manuf_model","LAMP01");
lamp_univctx.addObject(lamp_nodectl);
mylamp.addContext(lamp_univctx);
//
// Create Lighting Context & Control object
lamp_lightctx = new LightingContext();
lightlevel = new AnalogControl();
lamp_lightctx.addObject(lightlevel,2);
mylamp.addContext(light_lightctx);
```

Next the dimmer switch is configured to send report packets to the lamp. The CAL header is constructed in byte array "calhead". The report packet is formed by appending the report value to the supplied CAL header. In this case the header instructs the lamp to set its current value (labeled "C") to the dimmer switch current value.

```
// Set reporting on switch
byte [] calhead = new byte [5];
calhead[0]=CAL.CONTEXT_LIGHTING;
calhead[1]=2;  // Send to object #2
calhead[2]=CAL.METHOD_SETVALUE;
calhead[3]='C';
calhead[4]=CAL.TOKEN_DELIMITER;
sensor.setIV("report_address",
mylamp.getAddress());
sensor.setIV("report_header",calhead);
```

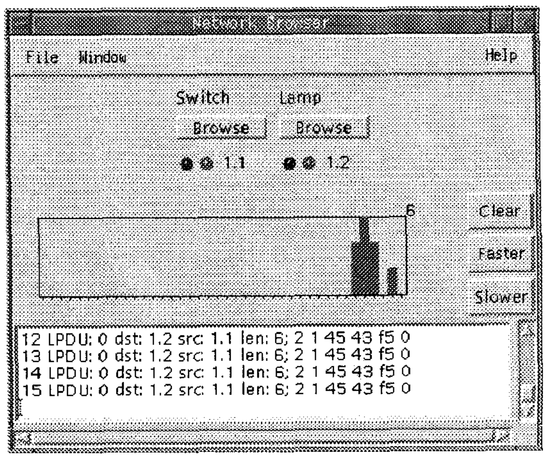
Finally register these devices with the network and launch the network browser.

```
// Register devices with network
mynet.addDevice(myswitch);
mynet.addDevice(mylamp);

// Launch network browser
Frame f = new NetworkBrowser(mynet);
f.show();
```

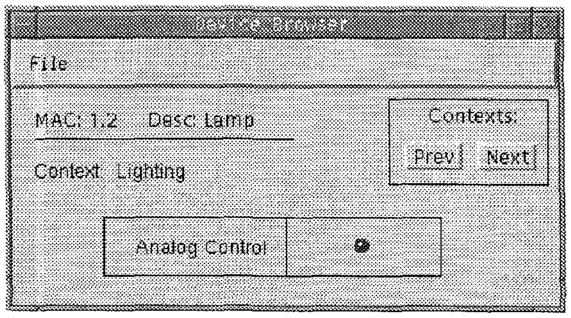
When executed the device browser window (Fig 6) appears on the desk top. A list of all registered devices appears along the top. Clicking on the 'Browse' button for a given device launches it's 'Device Browser' (Fig 7).

The device browser provides a graphical representation of all objects in the device grouped by context.



```
12 LPDU: 0 dst: 1.2 src: 1.1 len: 6; 2 1 45 43 f5 0
13 LPDU: 0 dst: 1.2 src: 1.1 len: 6; 2 1 45 43 f5 0
14 LPDU: 0 dst: 1.2 src: 1.1 len: 6; 2 1 45 43 f5 0
15 LPDU: 0 dst: 1.2 src: 1.1 len: 6; 2 1 45 43 f5 0
```

***Fig 6:*** *Network Browser Applet*



MAC: 1.2 Desc Lamp

Context Lighting

*Fig 7: Device Browser*

## 5. Conclusions and Future Work

Advanced security ... concept of groups of devices and operation classes eg some users can have 'read' access to some devices but not write etc

Home script - a secure scripting language to allow smart agents to run on the gateway server. These agents

## REFERENCES

[1] Hofmann, J., "The Consumer Electronic Bus: An Integrated Multi-Media LAN for the Home", *International Journal of Digital and Analog Communication Systems*, Vol. 4, No. 2, Apr. 1991, pp. 77-86, 1991.

[2] Desbonnet, J., Corcoran P.M., and Lusted, K., "CEBus Network Access via the World-Wide-Web", *IEEE Trans. Consumer Electronics*, Aug. 1996.

[3] Corcoran P. M. and Desbonnet, J., "Web Browser and Applet Interfaces to CEBus Networks", *ICCE '97 - IEEE Conference on Consumer Electronics, Chicago Il*, Jun. 1997.

[4] EIA Home Automation System (CEBus) Standard IS-60, "Common Application Language Specifications", EIA, Part 8, June 29th 1992.

[5] Home Plug and Play Specification, "CAL-based interoperability for Home Systems", CEBus Industry Council (CIC), April 2nd, 1997; "http://www.cebus.org/hpnp/".

## Biographies

***Peter Corcoran*** received the BAI (Electronic Engineering) and BA (Maths) degrees from Trinity College Dublin in 1984. He continued his studies at TCD and was awarded a Ph.D. in Electronic Engineering in 1987 for research work in the field of Dielectric Liquids. In 1986 he was appointed to a lecturship in University College Galway. He is involved in international Joint-Venture projects in both China and Eastern Europe. He is currently teaching on a full-time basis at University College Galway. His current research interests include home automation networks, Internet technologies, embedded computing applications, and telecommunications technologies. He is a member of the IEEE and a technical committee member of the IEEE Consumer Electronics Society.

***Joe Desbonnet*** received his B.Sc. degree in Physics and Electronics from University College Galway in 1993. He has run a professional WWW site at *http://www.wombat.ie* since 1992. He is presently working as a Research Associate at University College Galway and is a director of World-Wide-Web Marketing, a Galway based Internet services company.