MAPPING HOME-NETWORK APPLIANCES TO TCP/IP SOCKETS USING A THREE-TIERED HOME GATEWAY ARCHITECTURE

Peter M. Corcoran
Dept. of Electronic Engineering, University College, Galway

Abstract - **A three-tier gateway architecture for internetworking between home automation networks and a TCP/IP based wide area network, such as the internet, is described. The architecture abstracts the functionality of any home network into a driver layer (tier one), and provides a common access layer (tier two) from any TCP/IP network application (tier three) to a local home automation network. Clients and application programs may transparently access services and resources on the home network and appliances connected to the home network may access resources and services on the TCP/IP network.**

1. Introduction

As we near the end of the century it is evident that the all-pervasive growth of the Internet will be one of the primary influences on the next generation of home applicances and consumer electronic products. Further, it is evident that a significant proportion of homes are, or will shortly be connected to the Internet. This, in turn, is leading to the growth of many new Internet-based products and services, and the convenience of these for consumers will lead, in turn, to an increasing number of homes with permanent, high-bandwidth Internet connections.

The Internet-enabled home is likely, in turn, to catalyse the market growth of home networks as manufacturers of domestic appliances and consumer electronic products seek to gain competitive advantage by adding functionality and providing new services via the developing home-Internet infrastructure. Low-cost networking for such appliances is already technically feasible using powerline communications techniques [1, 2,].

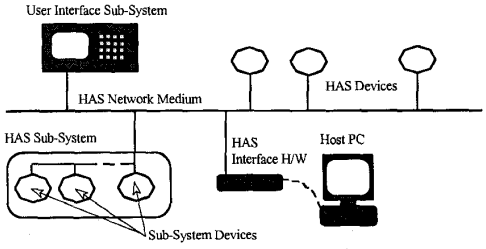

Fig 1: *Typical home network implementation - appliances and sub-systems are internetworked over the AC powerline; user access is via a modem-like device, using a desktop PC to provide the application-interface.*

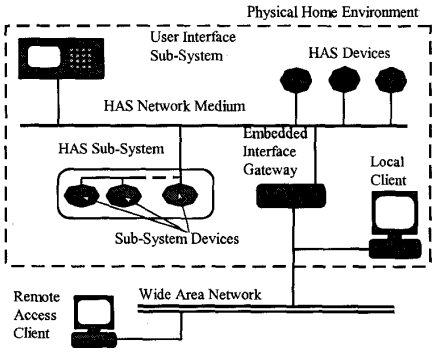

Fig 2: *A home network in an Internet-enabled household - an embedded Interface-Gateway links the powerline network to the wide area network; user access is via any PC connected to the WAN, or low-end Internet-Appliances (IAs) or set-top-box/TV combos*

In this emerging market scenario a key issue will be how appliances can access services and resources on a TCP/IP wide-area-network (WAN) from a local home powerline network. There are two key approaches to this problem:

(i) direct routing of network packets from the home network onto the WAN, and *vice-versa.*
(ii) brokering and management of accesses between the two networks by an *intelligent* gateway.

Unfortunately there are a number of key drawbacks with method **(i)**: it assumes a reasonable amount of computing intelligence at both ends of the inter-network communications link; typically a consumer applicance would be required to implement at least a micro-TCP/IP stack in order to be able to interoperate with the WAN. This is clearly beyond the current capabilities of most consumer appliances.

Furthermore, there are real-time considerations which are important for most home networking implementations - we cite, as an example, the immediate acknowledge (IACK) mechanism employed on *CEBus* networks. Given that delays of several seconds are not uncommon on TCP/IP networks, it is clear that the direct routing of home networking protocols over WAN connections is not trivial to implement. For these reasons we prefer approach (ii) - the use of an *intelligent* gateway to broker and manage access between a local home network and a WAN such as the Internet [3, 4].

Manuscript received June 17, 1998

In previous work [3, 4, 5] we have presented implementations of just such an approach employing a conventional client-server architecture. In the present paper we present some detailed refinements of the earlier work and develop a three-tiered gateway architecture which is based on open standards and is both robust and highly adaptable. An overview of the gateway architecture is given in Fig. 3 below.

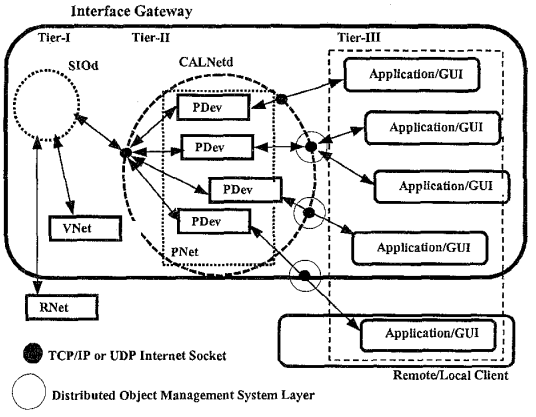

Fig 3: *An overview of the three-tier architecture: tier-I is implemented by SIOd; tier-II is implemented by CALNetd and tier-III is the TCP/IP client application layer.*

Note: RNet is a "real" home bus network; PNets and PDevs are "peer" networks and devices, *i.e.* software entities, incorporating data structures which mirror the state of the equivalent "real" network or device.

2. The Gateway Architecture

The gateway architecture described in this paper employs a three-tier model, rather than the simpler client-server implementation of earlier work [3, 4, 5]. A key goal of this architecture has been to separate the functionality of providing a basic bridge between a home automation network and a TCP/IP network from the more complex routing and management functionality required to facilitate interoperability between the two networks.

2.1 An Overview of the Three Tiers

At the physical interface between the gateway and the home automation network tier-I is implemented by a driver-agent - *SIOd*. This agent monitors and interprets traffic on the home network, performing real-time event handling, packet acknowledgements and basic filtering tasks. The raw traffic from the home network is then passed on to tier-II which we have implemented as a broker-agent - *CALNetd* - which is responsible for translating and interpreting the home network messages and maintaining a dynamic state-model of the real home network. *CALNetd* is also responsible for interacting with tier-III TCP/IP based client applications and for generating events and placing messages on to the home network via *SIOd*.

In Fig. 4 the relationships between the system software components of the home gateway, the local home network and the wide area network is illustrated. In particular we note that the software components, *SIOd*, *CALNetd* and any system peer networks or devices (*PNets or PDevs* [5, 6]) communicate with one another via TCP/IP network sockets.

In this configuration, which is employed in our prototype gateway, all of the main components run on the same *interface gateway* - in our case a standard PC. Applications may run on other *application clients* – other networked PCs, but in this implementation of the gateway architecture the key system software components run, as independent *daemon* processes on a single PC.

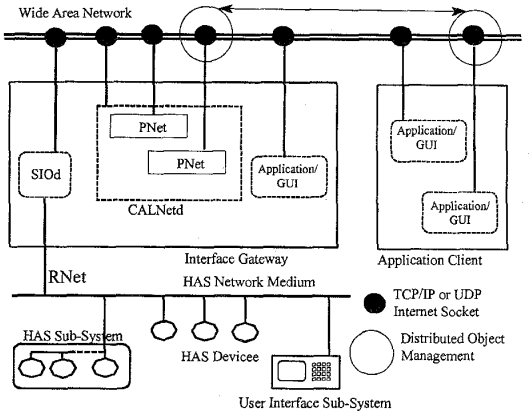

Fig 4: *Diagramatic representation of the relationships between system software components, the Wide Area Network and the Home Network.*

Each tier of the gateway architecture will be described in some detail later in this paper. First it will be useful to discuss the rationale which led to the separation of the gateway architecture into different functional layers.

2.2 Integrating the Home Network with TCP/IP

Our goal in this paper is to propose a software infrastructure which is flexible enough to integrate a variety of different home networks with the rapidly developing home-Internet infrastructure. After some consideration we adjudged that there are several key issues which have to be tackled through the architectural design, and handled by the individual components of this software.

In the first instance there must be a mapping of packets from the native home networking protocol to TCP/IP. However, as was discussed in the introduction, this is non-trivial as home networking protocols normally

implement low-level signalling and acknowledgement functionality which cannot be handled in real time by TCP/IP. Furthermore, most home networking protocols tend to generate large numbers of relatively small packets, i.e. < 40 bytes. Now, as each device on a home network is individually addressed, any packets associated with different devices should be transmitted over separate TCP/IP connections. But there is a significant overhead in establishing and maintaining each TCP/IP connection. Thus we are faced with one of two equally undesirable scenarios: either (i) a large number of semi-permanent TCP/IP connections carrying very low levels of traffic or (ii) a continuous and regular making and breaking of TCP/IP connections for each home network device.

A second issue is the heterogenous and distributed nature of home networks – consumers may add any mixture and variety of devices to a home network. At the applications layer a developer must have some means of tracking and interacting with the devices on a home network. However, due to its distributed nature home networking technology does not provide any standard form of centralized registry of information on the devices attached to a network. Thus the onus is on each application to extract and continuously update this information itself.

This approach works in a situation where a single home network is accessed from a single home PC running a single application program, as illustrated in **Fig 1** above. However, as soon as we introduce a situation where the home network is accessible by multiple clients running multiple applications then each application must continually monitor activity on the home network in order to update its state model of the home network. Naturally this will lead to a huge amount of network traffic as each message packet from the home network is transmitted to dozens of separate application programs. Clearly, it would be more practical to maintain a single state model of the home network on the main *interface gateway* and to allow applications to access this state model rather than having each application maintain its own state model.

A third issue is the nature of the client hardware and software which might be available to end-users in the near future. We tackle some of these issues in more detail in a companion paper [7], but a discussion in the context of the *interface gateway* architecture is also relevant here. Briefly, we anticipate that end-users will wish to access much of the functionality available on a home network from a variety of low-cost or portable interface devices. Such devices are very likely to provide TCP/IP connectivity, but their display and processing power will be more limited than a typical desktop PC. Thus we foresee a key requirement for applications which can up uploaded to the *interface gateway* but where the user interface is then exported to a remote client application over a TCP/IP socket connection.

In summary, all of the issues discussed in this section suggest that communications and interaction between the home network and client applications is best handled by a middleware layer or "broker-agent". This allows the number of TCP/IP connnections to be minimized; it provides a central repository for the registration of devices and for maintaining a dynamic state model of the home network.

2.3 Benefits of the Three Tier Architecture

From earlier discussion it is clear that there are a number of compelling reasons to integrate our home networking infrastructure using TCP/IP as a backbone. We might, however, have chosen to employ a conventional client-server architecture than the more complex three-tiered infrastructure we have described.

To understand better the flexibility of the three-tiered approach it is interesting to consider the gateway architecture in more general terms, particularly the potential for separating the system components physically and allowing them to run on separate hardware units, but all connected to the same WAN infrastructure. This is illustrated in **Fig 5** where the *SIOd* runs on a separate hardware unit from the *CALNetd*, which runs on an independent *Routing Host*. Application programs run on a third, *User Interface Unit*. Such an arrangement is practical because all of these software components communicate with one another over TCP/IP network sockets.

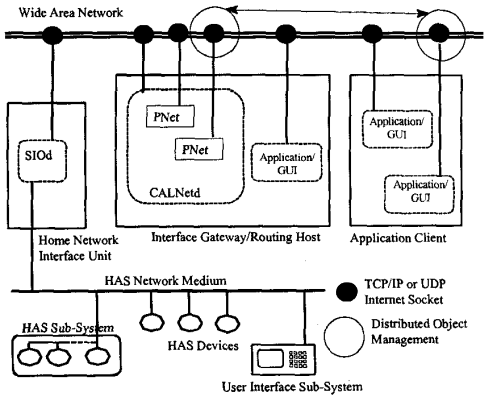

Fig 5: *As for Fig 4, but the SIOd and CALNetd agents now run on separate, independent hardware entities..*

Now consider a typical home user who has purchased incompatible home networking systems from two different vendors. However his problem is simply resolved if both vendors supply a simple low-cost network interface unit which provides a physical

interface to their home networking technology and includes a low-end CPU or dedicated hardware to implement a simple point-to-point protocol (PPP) link over low-cost twisted-pair wiring. Thus two, or more, incompatible home networking systems may be connected to the same routing host and client applications may access each home network via the same TCP/IP infrastructure and network state-model.

3. Tier I: The Network Interface Layer

It is the tier-I software layer which provides the main bridge between the home network and a TCP/IP enabled WAN. In practice there will be a dedicated hardware unit which implements the necessary physical and data-link layer interfaces to the communications medium of the home network. The role of the tier-I software is to implement a TCP/IP style network socket interface to the home automation network.

In our functional implementation of the network architecture we have used a modem-like device which interfaces with a CEBus powerline network [8] via the serial port of a PC. In section 3.1 we describe this particular implementation of a tier-I interface for a CEBus powerline network. We also describe how the the functionality of this software may be abstracted into a dedicated set of low-level hardware functionality and a separate set of software functionality. This would facilitate the design of a dedicated ASIC solution for the low-level, time-critical functionality of the tier-I software. Finally the implementation of other tier-I interfaces is discussed.

3.1 A Tier-I Agent for Powerline Networks - *SIOd*

The internal structure of this serial IO-daemon (*SIOd*) is shown in **Fig 6** below.

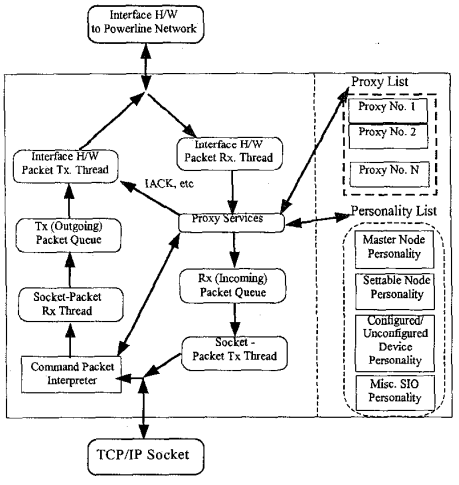

Fig 6: *The structure of SIOd, a tier-I driver agent.*

The core of this sample tier-I implementation consists of two main thread-pairs, one for Rx of packets from the TCP/IP socket and Tx of packets onto the home network, a second for Rx of packets from the home network and Tx of packets onto the TCP/IP socket. These thread pair are interlinked via incoming and outgoing packet queues.

Most packets from the home network are echoed directly to the TCP/IP socket. Packets which are written to the socket may be pure data-packets, in which case they are echoed directly onto the home network. Command-packets may also be written to the socket and these will effect changes in the proxy services implemented by *SIOd*.

However these threads are not fully independent and may interact via a *proxy services* filter. This handles redirection and real-time response generation using a table of low-level device services, for example the IACK or immediate-acknowledge response of CEBus. The exact nature of the *proxy services* offered is determined through a combination of a list of *proxy devices* and a list of *device personalities*.

To understand the function of the *proxy services* we need to recall that a tier-I agent, such as SIOd, will normally interoperate with a tier-II agent via the TCP/IP socket. The tier-II agent builds up a picture of the devices on the home network by observing and interpreting traffic on the network. It may also actively probe the network to obtain more complete information about various network devices.

Once the tier-II agent has identified a known device it can provide additional support services such as binding a TCP/IP exportable user interface to a device on the home network. From the viewpoint of the device on the home network, this TCP/IP exportable user interface also appears as a "physical" device on the home network. To achieve this *SIOd* must act as a proxy for such a pseudo-device, registering it on the home network as if it were a real device and subsequently providing time-critical responses for the device on the home network, as and when they are required. However the main functionality of such a pseudo-device is actually implemented by the tier-II agent.

The minimal set of low-level responses which are implemented by *SIOd* are what we describe as the *proxy-personality* of the device. Furthermore *SIOd* maintains a list of device addresses which it has reserved on the home network for its proxies. When the destination address of any incoming packet matches an address in its proxy list *SIOd* will check to see if it should generate an immediate response before placing the packet on the incoming queue to the TCP/IP socket.

3.2 Hardware/Software Abstraction of *SIOd*

In **Fig 7** below we show how the functionality of *SIOd* may be separated into low-level *interface hardware* functionality and a software layer implementing buffering and socket functionality.

The *interface hardware* handles redirection and real-time response generation using a table of low-level device services, for example the IACK or immediate-acknowledge response of CEBus. The software layer can reside on a separate hardware host, or be implemented on a low-resource microprocessor.

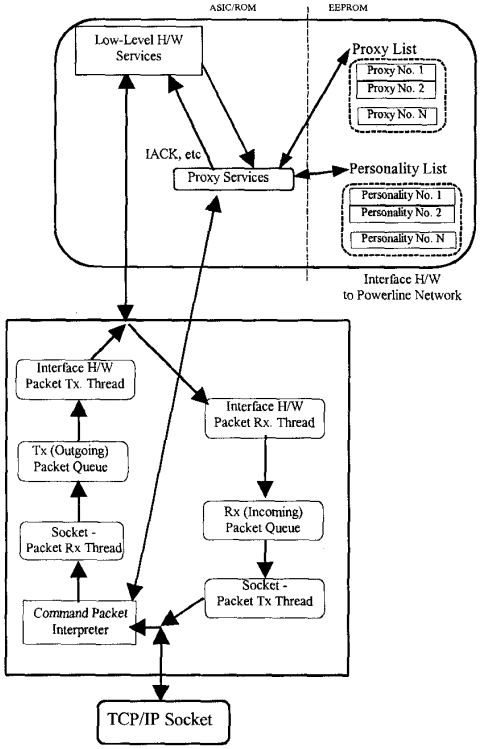

Fig 7: *Functional abstraction of SIOd, into H/W and S/W components; this allows the development of a network-specific ASIC solution for mass-market applications.*

3.3 Tier-I Implementations for USB, IEEE 1394

Both of these new consumer networking technologies offer significant improvements on existing twisted-pair wired networks. In particular they allow high bandwidth data which is synchronous in nature, such as digital audio and video streams, to co-exist with low-bandwidth asynchronous data. USB bandwidth ranges from 1.5Mbps up to 12Mbps; IEEE 1394 bandwidth can range from 200Mbps upwards.

Tier-I implementations for either of these networking technologies will feature multiple TCP/IP sockets. A single asynchronous *message socket* is required to support the transport of control messages from the home network to the tier-II agent. Additional sockets should support synchronous audio and video streams. An interesting aside is that, when combined with these new home networking technologies our three-tier architecture will provide the ability to access new on-line services such as *Internet telephony* and TCP/IP based *video-on-demand.*

3.4 Other Tier-I Implementations

There are a number of powerline home network technologies currently available on the market. The most successful and widely available home automation system is X-10. Other products are available but in most cases the cost is still prohibitive for the home consumer. Wireless networking technology is also available but has not yet reached commercial viability. IrDA controlled devices are only point-to-point links and only support proprietary command sets rather than a generic model like CAL. Thus, only powerline and wired network solutions are likely to reach the marketplace over the next 2-3 years.

As X-10 solutions were not widely avaialble at 240V until quite recently we have not yet implemented a functional tier-I agent for X-10. However this will not be a difficult task as the X-10 protocol only supports a very limited nuber of control objects and system commands. Work is currently in progress.

4. Tier II: The Middleware Layer

This system agent forms the key middleware in our three-tier architecture. Traffic from the home network is routed over a TCP/IP socket to our tier-II implementation, *CALNetd,* which interprets CAL messages and constructs and maintains a dynamic model of the current state of the home network. User applications and other tier-three services interact with the virtual model of a home network contained in *CALNetd.*

Amongst the principle services provided by the *CALNetd* broker-agent are those of packet-routing, device-registration, address binding and API/event-translation for both real and virtual devices. An overview of the internal structure & organization of *CALNetd* is given in Fig 8.

4.1 Network Models with CAL

CAL, or the "common application language" [9], is a well-know device modeling and networking language which is implemented at the application layer. It is presently being proposed as a global standard for home networking applications by the US standards authorities.

CAL provides a powerful, if somewhat arcane, method of modeling consumer devices in an object-oriented

manner. It also facilitates the routing of messages between control and sensor objects within devices, and provides a consistent means of functional description of practically any conceiveable consumer appliance. For these reasons we have chosen, in the present implementation, to base our core models for the home network and its devices on CAL constructs. Thus part of the future functionality of *CALNetd* should include the ability to interpret and translate messages from non-CAL networks into CAL network equivalents.

4.2 Core CALNetd Services

In this section we describe the main services available to devices on the home network from our middleware.

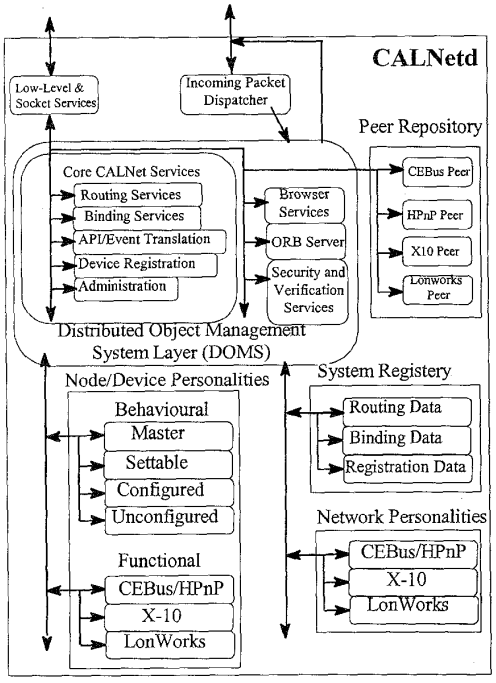

Fig 8: *Internal Structure and Organization of the CALNetd middleware or "Broker Agent".*

4.2.1 Routing Services

This is the most important function of *CALNetd.* In principle several independent home networks may be connected to a single middleware agent such as *CALNetd.* The agent may be requested to route messages between these networks as well as routing messages and data packets between these networks and client applications. It must also keep track of "virtual" devices and "user-interface" devices which it has been requested to create, or has created itself.

In most practical consumer applications many of the devices and application programs will be relatively static, but as new devices will be regularly added to a home network and as applications may be removed or disabled from time to time, routing management is a non-trivial service provided by the middleware.

4.2.2 Device Registration Services

Before home network devices can access services and resources via the system middleware they must first register with *CALNetd.* In many cases a device may not be smart enough to implement an explicit registration so that much of the onus of detecting new devices and implementing a passive registration of such devices lies with *CALNetd.* However as most home networking protocols implement fairly rigorous registration proceedures it is not too difficult to implement passive registration of nodes by continuously "listening" for the registration signatures of new devices.

4.2.3 Binding Services

When devices have registered they may require to access additional services using the capabilities of the system middleware. For example, a device may wish to upload a TCP/IP exportable user interface (UI) to be used on a variety of access terminals around the home. This user interface exists as a "virtual" device on the gateway server, and it must be registered as an extension of the "real" device which invoked its presence. We refer to this mechanism of loading and linking a "virtual" device on the gateway server to a "real" device on the home network as a *binding service.*

4.2.4 API/Event Translation

As was described in earlier work [5, 6] our implementation of *CALNetd* incorporates an object oriented implementation of CAL itself. In other words we can readily instantiate "virtual" CAL objects on the fly. These may be used to create peer objects which duplicate the state of real objects on the home network, or they may be used to create new objects which add functionality to existing devices on the home network.

In both cases an end-application can access these "virtual" CAL objects directly through API calls, or using techniques such as *remote method invocation* (RMI) to access them over a TCP/IP network.

It is the peer object whose state is manipulated by the application, but the desired end result must be propagated to the actual physical device by *CALNetd.* This requires that the API calls are translated into appropriate CAL messages, which are then passed onto the home network where they generate appropriate events on said home network.

4.3 TCP/IP Export of Tier II Object Functionality

This is an very interesting area in which there have been a range of new approaches and solutions emerging during the past 12 months. In fact several of these new technologies and their applications to the interface architecture we have described in this paper are dealt 5with in more detail in a companion paper [7].

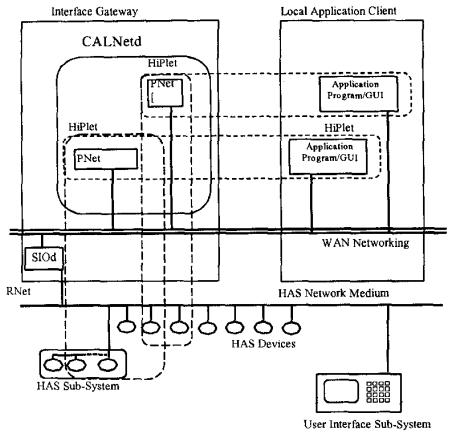

Fig 9: *The relationship between "peer" networks (PNETs), the client user interface and real devices on the home network.*

For the purposes of the present paper we will briefly summarize the principle user-interface (UI) technologies which are of greatest interest in the context of home networking:

Java-Based Techniques

Java has become very popular as a development language, but its use is somewhat limited for lightweight consumer applications as each client must implement a complete Java virtual machine (JVM). Typically this requires a minimum of 4-8M of RAM. Once the JVM is established on the client Java offers an extermely flexible and powerful means of providing a user interface to home networking applications. Current initiatives such as *Personal Java* and *Embedded Java* are intended to develop more lightweight subsets of Java which are targetted at consumer applications such as home networking [10].

Object Request Broker Techniques

Object request broker (ORB) technologies such as CORBA are somewhat similar to Java in their scope and functionality. Earlier work [11] has demonstrated that it can be practical to embed an ORB into consumer applicances. However, as with Java, there are significant resource requirements, particularly from the perspective of exporting lightweight user interfaces (UIs) from tier-II objects.

Mark-Up Languages

The success of the hypertext markup language, HTML, has led to the development of a derivative known as the handheld device markup language, HDML. This open standard was developed specifically for devices like mobile phones and handhelp personal digital assistants (PDAs) which have quite limited displays and, typically, a telephone-style keypad for user input.

By its nature HDML is very lightweight and requires only a relatively simple intepreter at the client end. It also supports an ability to export a range of user-interface options, including menus and control buttons [12].

Remote Frame Buffers

Recently placed in the public domain under the GNU Public Licence, this technology was developed by Oracle Research Labs, UK. It represents the most lightweight approach possible to client-server user interfaces for consumer applications. The application resides completely on the server, and only its graphical user interface is exported, via a TCP/IP socket, to the client. This is only updated when part of the user interface changes. The client supports basic mouse events which are relayed to the server-based application.

Perhaps the most innovative aspect of RFB technology is that it is the client that specifys the size, pixel depth and other display capabilities and the server must adapt to these. Thus it is quite practical to access the X-Windows display of a Unix workstation from a low-resolution handheld PDA such as a PalmPilot. A more detailed discussion of RFB technology is given in a companion paper [13].

5. Tier III: The Client Application Layer

Client access to the home network will be from TCP/IP enabled consumer devices. Typically most client applications will provide a relatively simple GUI interface, exported from Tier-II, to the functionality of a networked device. Thus, in many instances the client may not require more than a basic display and a low-powered CPU, especially if the user interface is to be exported using a lightweight protocol such as RFB. We can today, for example, demonstrate access to an X-Windows desktop from a device such as a Palm Pilot PDA using RFB.

More sophisticated client applications may also be developed if the hardware and software resources of, for example, a home PC are available. In the remainder of this section we will discuss briefly the potential of different levels of client hardware to provide workable user interfaces to different home applications.

5.1 The Home PC Client

This will become the de-facto home user interface by way of its ubiquity in home dwellings. We are already seeing a huge growth in the low-end PC market during the past 12 months. Further the emergence of low-cost embedded x86 CPUs can only further drive down the cost of small, low-cost, but still highly functional PC units. Such units will prove cosmetically and economically more attractive to the majority of home consumers.

These low-cost units will be functionally capable of running all of the different interface technologies discussed in this paper. We can expect that they will enable a new generation of networked consumer appliances with significant new functionality and programmability. We may also expect the emergence of new interface technologies, including speech driven interfaces to the home environment, although these may take some time to become acceptable to many consumers.

5.2 The Internet Appliance Client

Many manufacturers are about to release their version of the *internet applicance* (IA) - real soon now. However there are still very few "real" applicances available. In one sense this is understandable as the Internet infrastructure continues to develop at a tremendous pace, while the target markets for these relatively novel devices are still emerging and unknown. Thus manufacturers are reticient to "roll-out" their particular version of the IA.

Some examples of IAs include Java-telephones, Web-TV systems and even combined GSM-telephone/PDAs. Although versions of some of these products are available in the marketplace their emergence is slow compared with the adoption of home-Internet via home-PC. From the user-interface perspective these devices will range in capability from offering almost full PC functionality to very minimalist devices like today's combination of GSM-telephone/PDA offering some basic services such as E-Mail.

5.3 Wireless and HandHeld Clients

Finally, we see a new emerging category of devices which are even more lightweight than today's IAs. These will be based on todays mobile telephones and PDAs. The telephones will offer enhanced user-interfaces functionality such as larger screens; the PDA will gain wireless connectivity. Both will evolve towards a new handheld applicance which, in addition to its primary function as phone or PDA, we can expect to become a standard interface-appliance between ourselves and the electronic home of tomorrow.

6. Conclusions

We have outlined in this paper a three-tiered software architecture for integrating a home network with an external wide-area network, such as the Internet. In particular we conclude that such an approach allows much of the complexity and core functionality of an integrated home network to be concentrated in a Tier-II software layer. This in turn allows the low-level driver layer, Tier-I, and the user-interface layer, Tier-III, to be much simpler and facilitates standardization on generic interfaces between the three tiers. For example this approach allows non-CAL home networks to be modeled as CAL networks in the tier-II software. In the same way several heterogenous home networks could interact and interface with each other via the tier-II software layer, and be accessed by an end-user from a common tier-III user-interface.

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] http://www.cebus.org

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

[4] Desbonnet, J and Corcoran P. M.., "System Architecture and Implementation of an Internet/CEBus Gateway", *IEEE Transations on Consumer Electronics*, p1057-1062, Nov. 1997.

[5] Corcoran P. M. and Desbonnet, " Browser Style Interfaces to a Home Automation Network", *IEEE Transactions on Consumer Electronics,* p1063-1069, Nov. 1997.

[6] CEBus Developer's Conference, April 1997.

[7] Corcoran P. M., Ferenc P. and Zoldi A., "Portable User Interfaces for Home Systems and Networks.", *IEEE Internetional Conference on Consumer Electronics,* Los Angeles, June. 1998.

[8] EIA Home Automation System (CEBUS) Interim Standard IS-60, Vol. 1, Parts 1,7,8, EIA, June 29, 1992.

[9] EIA Home Automation System (CEBUS) Interim Standard IS-60, Vol. 2, Part 8, EIA, June 29, 1992.

[10] http://www.javasoft.com/

[11] Kitao, M *et al*, "Equipment Integrated Control Protocol Based on Distributed Object-Oriented Architecture", *IEEE Internetional Conference on Consumer Electronics,* Chicago, June 1997.

[12] http://www.uplanet.com/pub/

[13] http://www.orl.co.uk/vnc/

Biography

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 '87 for research work in the field of Dielectric Liquids. In '86 he was appointed to a lectureship in University College Galway. From '94 to '96 he worked as general manager in the electronics division of the first Sino-Irish Joint Venture, based in Beijing, PRC. During '96 he was also a visiting Professor in Telecommunications at the Tech. Univ. of Cluj-Napoca, Romania. His research interests include consumer electronics, embedded computing, networking, instrumentation and telecommunications technologies. He is a member of the IEEE.