Real time ATM for remote access to home automation and digital home AN networks

REAL TIME ATM FOR REMOTE ACCESS TO HOME AUTOMATION AND DIGITAL HOME A/V NETWORKS

Alexandru Cucos and Peter M. Corcoran
Department of Electronic Engineering, University College, Galway

Abstract - **Remote access to a home automation network via an Asynchronous Transfer Mode (ATM) network is described. Real-time access to such networks is important in applications where the inherent latency of TCP/IP does not guarantee a response to real-time events on the home network.**

1. Introduction

Remote access to a home automation network from a computer with an Internet connection, using a Internet/Home Bus gateway has previously been demonstrated [1, 2]. 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.

Fig1 shows a Home Network in an Internet-enabled houshold. An embedded *interface-gateway* links the powerline network to the wide area network. User access is via any PC connected to the *wide area network* (WAN), or low-end *internet-appliances* (IAs) or set-top-box/TV combos. Note that most of today's end-user WAN implementations are based on TCP/IP although the main internet backbones are more likely to be ATM based.

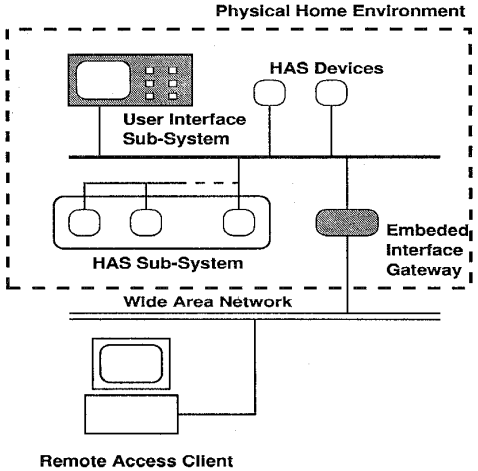


Fig 1: *Home-Network/Internet Infrastructure.*

We also note that an inherent limitation of the TCP/IP protocol is its inability to provide an end-to-end delay guarantee in the case of applications which must provide real-time responses to events on the home network. An example of such an application might be a "network diagnosis service" [3] which would remotely analyze and test the behavior of a home network and the devices connected to it. Such a service will be needed by modern consumers as more and more domestic appliances become Home Bus enabled. ATM has emerged as the most promising technology in suporting such real-time communication services.

Thus, in this paper, emphasis is placed on the potential to implement real-time access to the home network. Other potential benefits of using ATM over conventional TCP/IP wide area networking are also disscused including the potential for ATM to provide internetworking with *universal serial bus* (USB) and *IEEE 1394* home networks.

2. System Overview

In this section we describe the prototype experimental setup used in our work. We also describe what we mean by real-time ATM and how an ATM wide-area network can be physically integrated with a home network.

2.1 Experimental Set-Up

Our experimental setup is shown in **Fig 2** and has the following components:

1.- *CEBus Monitors* modem like devices which interface the power line network with the serial port of a PC.
2.- *Pentium 133Mhz, 32MB RAM*
3.- *ATM Switch* high-performance switch that enables to connect 25Mbps ATM end-stations both to each other. 12ports 25Mbps, suports ATM Forum-compliant UNI 3.0 or 3.1 singalling, UTP cabling.

Manuscript received June 17, 1998

4.- *25Mbps ATM network adapters* that have 128kbytes of on-board SRAM, require one interrupt channel (which can be shared).

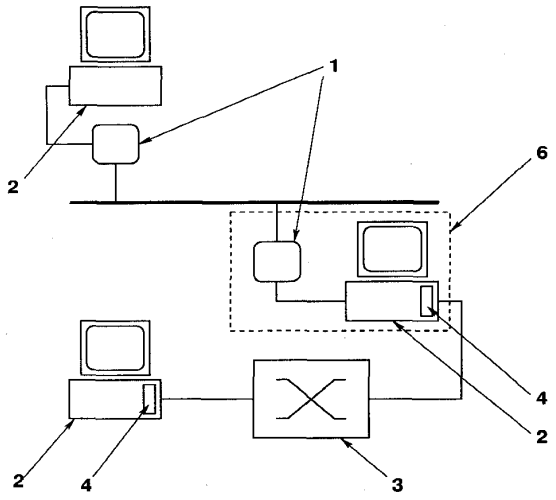


Fig 2: *Experimental Setup*

2.2 Real-Time ATM

The key advantage of an ATM network over a TCP/IP network is its ability to handle delay-sensitive applications. It boasts a comprehensive set of network protocols to specifically address the requirements of real-time traffic. These include a range of service classes for transporting various types of data, as well as circuit-emulation capabilities. ATM specialized service classes for real-time support include, *Constant Bit Rate* (CBR) and *Variable Bit Rate - Real-Time* (VBR-RT) are two service classes applicable to real-time applications.

There has been a lot of research addressing the issues involved in providing real-time end-to-end delay guarantees. End-to-end in a networked environment can mean different things to an application. We classify end- to-end into three principle categories:

(i) Application-to-application (AtA),
(ii) memory-to-memory (MtM), and
(iii) network interface-to- network interface (NtN).

Each of these different real-time implementations is summarized briefly below:

AtA is where the guarantee is provided from the moment the sending application generates the data to the moment the receiving application retrieves the data.

MtM is where the guarantee is provided from the moment when the data is taken from the sending host memory to the moment when the data is deposited into the receiving host memory, regardless of when the data is generated by the sending application and when the receiving application actually retrieves the data.

NtN is simply the network guarantee from when the data is transmitted from the sending network interface to when data is entirely received by the receiving network interface.

Different application scenarios require different levels of end-to-end guarantee. As the bandwidth of most Home Bus applications is relatively small the application can use NtN without significant problems.

2.3 Integrating ATM with the Home Network

In this paper we show the implementation of an example home service application, programmed using the BSD network socket model. The integration of an typical home networking protocol with ATM transport layer is also described.

In a companion paper [4] we introduce the hypothesis that 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. In this emerging market scenario a key issue will be how consumers can access networked home appliances, and equally how these applicances can access services and resources on a TCP/IP wide-area-network (WAN) from a local home network. The former is desirable to improve the accessibility and utility of applicances to end-users in a networked home environment; the latter allows appliances to gain added-value functionality by providing networked based services and by accessing distributed applications software, including systems-level upgrades.

There are two key approaches to these problems:

(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.

The gateway architecture described in a companion paper [4] assumes that approach (ii) will prevail and descirbes a three-tier software architecture for implementing the home gateway. In this paper we are mainly concerned with a scenario which overcomes the inherent limitations of TCP/IP and allows a working solution to approach (i) to be realized.

In **Fig 3** we show the basic elements that enable remote access of an Home Automation Network. The dynamic data structures which provide a real-time representation of a Home Automation Network can be relocated to the remote end-system, or "application client" due the high speed connection-oriented nature of the ATM Network. Home Automation Network traffic is continually monitored and interpreted and these data structures updated accordingly.

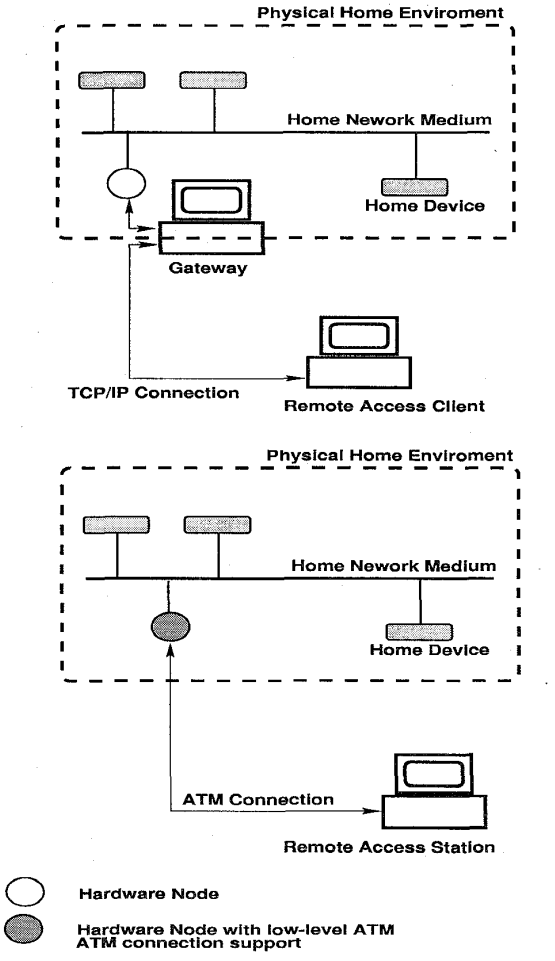


Fig 3: *Real-Time ATM Vs TCP/IP*

At the physical interface between the ATM network and the home automation network, the hardware node with low-level ATM connection support is implemented by a driver-agent - *SIOd.*

3. The Network Interface Layer

In our functional implementation of the network architecture we have used a modem-like device which interfaces with a CEBus powerline network via the serial port of a PC. In this section we describe our particular implementation of the interface between the home network and the ATM wide area network. A detailed discussion on related ATM issues is also given.

3.1 Agent for Powerline Networks - SIOd

It is the serial IO-daemon (SIOd) which provides the main bridge between the home automation network and the ATM wide area network. In practice there will be a dedicated hardware unit which implements the necesary physical and data-link layer interfaces to the communication medium of the home network. The role of the SIOd is to implement a ATM style network socket interface to the Home Automation Network.

We have two posibilities in designing our daemon:

(i) multitasking SIOd

(ii) single-tasking SIOd

The core of the multitasking SIOd consist of two main processes, one for Rx of packets from the ATM 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 ATM socket.

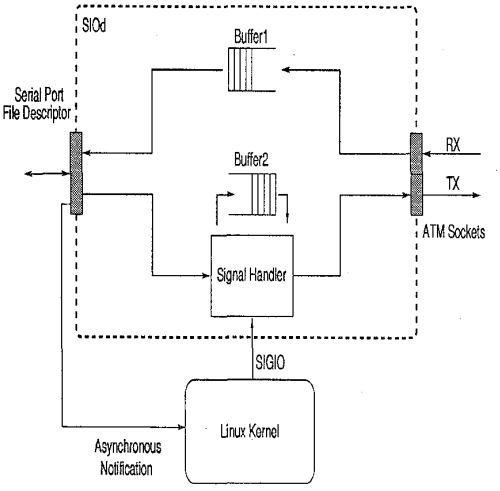


Fig 4: *SIOd*

The single-tasking SIOd consists only of one process for Rx of packets from the ATM socket and Tx of packets onto the home network. Rx of packets from the home network and Tx onto the ATM socket is made asynchronously using the signalling capabilities of the Linux operating system. The internal structure of this serial IO-daemon (SIOd) is shown in **Fig 4**. The daemon is continously polling the ATM socket for incomming packets. Whenever new data is avaiable on the serial port the Linux Kernel generates a SIGIO

signal which is caught by SIOd. The associated signall handler reads the available packet from the serial port and puts it onto the ATM socket.

3.2 ATM Considerations

VBR class allows us to send data at a variable rate. Statistical multiplexing is used and so then may be small nonzero random loss. Depending upon whether or not the application is sensitive to *Cell Delay Variation* (CDV), this class is subdivided into two categories: *Real-Time* Variable Bit Rate (VBR-RT) and *Non Real -Time* Variable Bit Rate (VBR-NRT). While cell transfer delay (CTD) is specified for both categoires, CDV is specified only for VBR-RT.

CTD is the delay experienced by a cell between network entry and exit points. It includes propagation delays, queueing delays at various intermediate switches, and service times at queuening points.

CDV is a measure of variance of CTD. High variation implies larger buffering for delay sensitive traffic.

Most home networking protocols tend to generate large numbers of relatively small packets i.e. <40bytes, the number of packets per time unit is also variable so we consider that the best ATM service class for our purpose is the VBR-RT class.

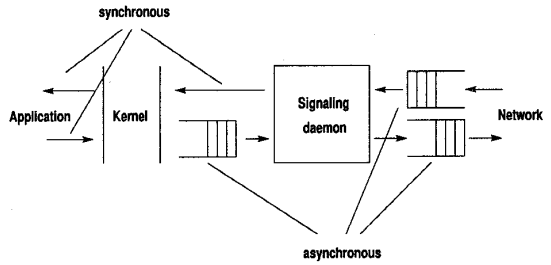


Fig 5: *Linux ATM signaling concept*

For our experimental setup we use a Unix clone operating system (OS). The OS kernel implements a simple protocol to support ATM signalling. All the main complexity of ATM signalling is delegated to a user-mode daemon process [6]. This configuration is shown in **Fig 5**. Communication between the kernel and the signaling daemon is generally assumed to be reliable and to preserve the sequence of the messages. Furthermore, it is assumed that there is a queue when sending to the daemon so communication is asynchronous. Communication is synchronous from the signaling daemon to the kernel. There is only a single sequential communication path in either direction. Messages related to different sockets may be interleaved.

The signaling daemon is not directly involved in opening a *virtual connection* (VC) for data traffic - that task is handled entirely by the kernel.

The protocol is designed for the use with Berkley-style sockets. All the general operations such as binding to a local address, requesting an outgoing call, accepting incoming connection, etc, are suported. **Fig 6** shows the socket descriptors in the kernel and in the signaling daemon. The socket descriptors in the kernel consist of a general socket structure and the ATM specific VC structure.

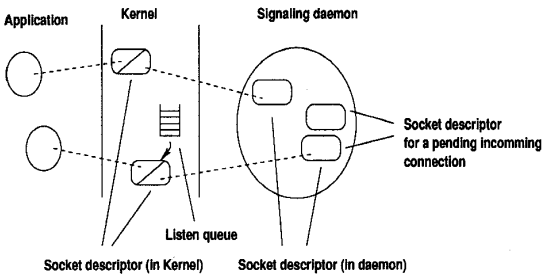


Fig 6: *Socket descriptors in the kernel and in the signaling daemon*

In ATM, the fundamental communication paradigm is the connection. In our current implementation we are using *permanent virtual connections* (PVC). Opening a PVC socket implies attaching an endpoint to a connection that already exists in the network. The state diagram shown in **Fig 7** illustrates the life cycle of PVC sockets. PVC sockets are created with the "socket" system call. Then the QOS requirements are indicated, an address descriptor is set up, and a bind or connect system call is called to open a virtual channel (VC). During connection preparation parameters are set and general local resources, *e.g.* socket descriptors, are allocated. During connection setup local networking resources, *e.g.* bandwidth, connection identifiers, buffers, etc are allocated.

After these two steps are completed then real-time data exchange between Home Automation Network and remote end-system is possible. Finally, during connection tear-down communication is stopped and resources are de-allocated. The reader is referred to [4] for further details.

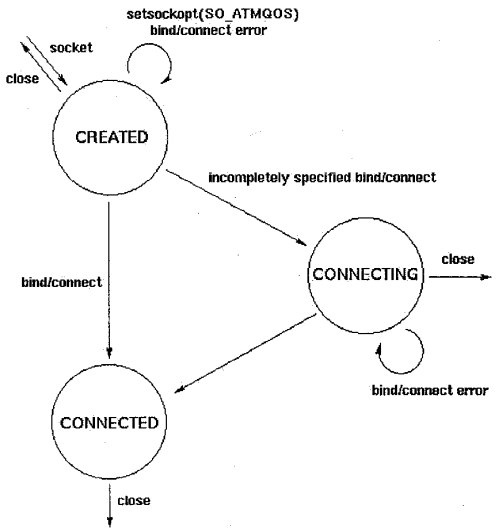


Fig 7: *ATM State Diagram*

4. Sample Applications

There are several existing and emerging home networking technologies. One of them is the Consumer Electronic Bus (CEBus), a multi-media LAN standard introduced by the Electronics Industries Association. Other emerging standards in the Home Network arena include *universal serial bus* (USB) and *IEEE 1394.* In this section we consider some potential applications for practical Home Networks which can take advantage of the characteristics of our *real-time* ATM implementation.

4.1 Virtual CEBus Network Devices

Using our RT-ATM implementation a *virutal* CEBus network, with *virtual* devices can be created on the application client. This takes advantage of earlier work we have done on TCP/IP networks, again using the BSD socket model as our internetworking metaphor.

These *virtual* devices, or VDevs can interact with the real CEBus network in real-time over the ATM link. This allows the creation of "new" device personalities whose behaviour can then be characterized through interaction with a real CEBus network. It also opens up the possibility to bridge local CEBus networks at remote locations. Another key application for this technology will be in remote network diagnostics [3].

In **Fig 8** we show a simple Home Automation Network. One of the "real" devices sends a packet to our *virtual device* which is simulated by a software entity which is running on the *application client.* The "real" CEBus device is waiting for an *immediate acknowledgement* (IACK) message. The IACK mechanism enables the transmitting node to determine the succes or otherwise of its messages across a single medium - the IACK scheme does not work across routers, brouters, etc. When the *virtual device*, which is the receiving node in this instance, determines that a frame has been properly received and that an acknowledgement is required, it must form an IACK and sent it out over the channel within 2 *Unit Symbol Time*, or UST's, of receiving the *End_Of_Packet* (EOP) symbol. At this point, all other nodes are in the minimum wait period of 10 UST's so the receiving node is assured of gaining immediate channel access. The originating node requires the IACK preamble within 6 USTs of transmitting the EOP symbol for the frame.

If the IACK is correctly received, the originating frame can discard the original packet, else it must prepare to retransmit. As we can see time constraints are critical, an ATM connection with required QOS can guarantee us a correct functioning of the home automation network.

4.2 Adding a Java Interface

Java has become very popular as a development language. There already exists a Java class library that provides all the necessary tools to construct virtual CEBus devices [2]. For this reason we wrote some C native methods to allow these Java classes to access the Linux ATM capability using the JDK1.1.3 native method interface. We have created an "endpoint" object called *AtmConnection* whose main methods are described in Table1. These native methods are used in

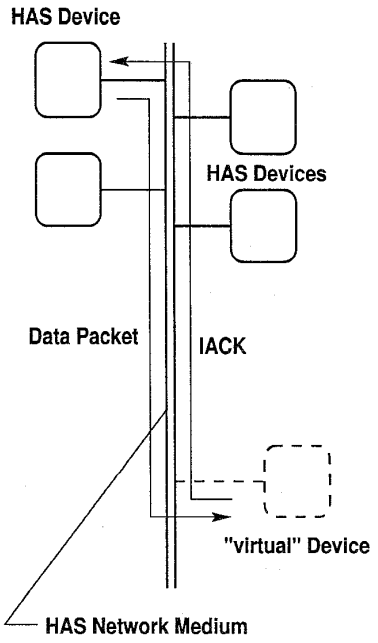


Fig 8: *Sample Application*

two main threads running on the remote system. One thread is used for receiveing packets from the ATM network and to update the virtual devices accordingly. The second thread is used to send packets, generated by the virtual devices, onto the ATM connection.

Method:	Description:
Open()	*opens the ATM connection*
Close()	*closes the ATM connection*
Send()	*sends some data on the ATM socket*
Receive()	*receives incoming data*

Table 1: *Native Methods of the AtmConnection Class*

The following Java code fragment illustrates the creation of such an object used by the receiving thread.

```
AtmConnection atm = new AtmConnection(ConnectIdent, Qos);
Packet packet = new Packet ();

atm.Open();
atm.Receive(packet);
atm.Close();
```

4.3 Network Testing and Analysis

The CEBus standard is a subset of the OSI seven-layer model. In particular it conforms with the OSI model at the three lower protocol levels: Physical, Data Link and Network layers. The implementation of these lower layers is quite robust and a practical CEBus network is fault-tolerant. Despite this robustness, communications errors will still impact on network performace generating increasing traffic, incomplete packets and excessive usage of IACK call-backs.

In worst case situations it is conceivable that a network may fail, or be rendered dysfunctional, by a single faulty node. The fault diagnosis system consists of a CEBus node with a physical interface to the AC power-line [7]. This *active node* [3] is connected to a remote system via an ATM network and controlled by software running on the remote system.

The fault diagnosis system has the possibility to capture packets from all other nodes in the home network. Test packets can also be generated to allow testing of the network. This feature allows periodic or continous remote testing of a suspect node and would be particulary useful for the detection of intermittent fault conditions. The remote system controls the analysis of the network monitoring activities and presents data summaries and control functionality via a GUI interface [3]. Real-time response from the remote system is extremely useful in detecting and analysing errors that imply precise time measuring. Below we describe two possible errors types:

Failure to send IACK packets - if a CEBus node transmits a packet and the packet data indicates that an ACK packet is expected, then the next packet received should be an ACK packet from the receiving node. The fault diagnosis system must examine all packets to see if an ACK packet is requested, and then check the next packet to see if an ACK packet was sent. Failure to do so suggests that the receiving node is faulty and can be checked by transmission of packets to the suspected faulty node.

Incorrect Channel Access Method - all CEBus nodes must adhere to a set of rules for obtaining access to the AC powerline medium. These rules, known as the *Channel Access Method*, stipulate certain delay times which must be satisfied:

i) attempting channel access after the last communication,

ii) deferral to nodes already communicating on the network,

iii) allowing time for ACK packets before attempting new communications, etc.

Thus, most real-time timing errors can be detected by examining the inter-packet timings sent from an *active node* [3] to the remote system. These interpacket timings consist of the number of UST elapsed since the last CEBus channel activity, and precede the packet information.

Such a fault diagnosis system must not only examine the inter-packet timing, but also the packet type to see if the *Channel Access Method* is being adhered to. The *Channel Access Method* also specifies additional delays for packet priority, packet queueing, and randomisation. By examining the inter-packet timings and the packet data containing priority information, *etc.*, the fault diagnosis system can detect more subtle errors in the Channell Access Method.

4.4 Interfacing with USB and IEEE 1394 Networks

Universal Serial Bus (USB) and IEEE 1394 are two other emerging home automation technologies. As USB is more likely to reach market before 1394 we will focus on it in the discussion that follows. Note, however, that much of the discussion applies equally to 1394. This is as a consequence of our software architecture which is oriented around the BSD network sockets interface.

USB supports functional data and control exchange between *host* and a *device* as a set of either uni-directional or bi-directional transfers. Data transfers take place between host software and a particular endpoint on a *device*. A given USB device may support multiple data transfer endpoints. The USB host treats communications with any endpoint of a

device independenly from any other endpoint. Such associations between the host software and a *device* endpoint are called *pipes*.

As an example, a given *USB device* could have an endpoint which would support a *pipe* for transporting data *to* the *device* and another endpoint which would support a pipe for transporting data *from* the *device*. *Pipes* have associations of data bandwidth, transfer service type, and endpoint characteristics like directionallity and buffer size. *Pipes* come into existence when a USB device is configured.

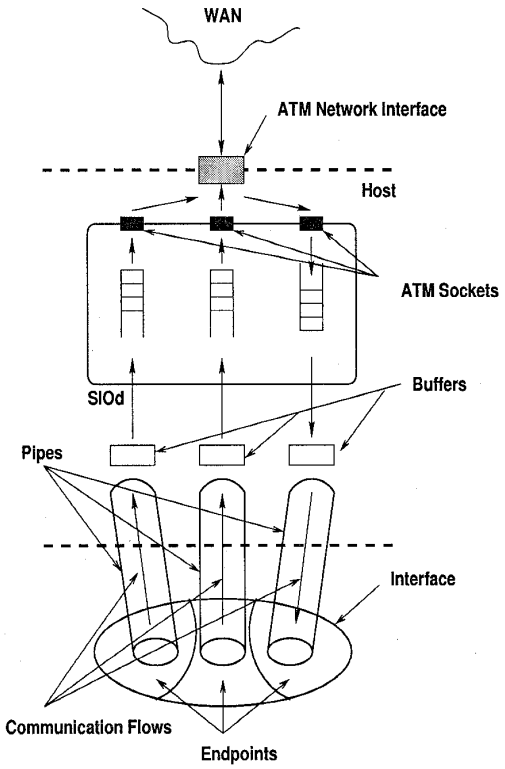


Fig 9: *Interfaceing USB & ATM*

The USB architecture comprehends four basic types of data transfers:

- *Control transfers* that are used to configure a device at attach time and can be used for other device specific purposes
- *Bulk data transfers* which are generated or consumed in relatively large and bursty quantities and has wide dynamic latitude in transmission constrains.
- *Interrupt data transfers* such as characters or coordinates with human perceptible echo or feedback response characteristics.
- *Isochronous or streaming real time data transfers* which occupy a prenegotiated amount of USB bandwidth with a prenegotiated delivery latency.

Any given *pipe* supports exactly one of the types of transfers described above.

The idea is to create a ATM connection (between the USB host and the remote client) with required traffic parameters for every USB pipe existing between the USB host and the USB devices. This implementation can be readily achieved using a multi-socket extension of our *daemon* program *SIOd*, as shown in **Fig 9**. The same applies to 1394 networks.

5. Conclusions & Findings

In this paper we show the potential advantages of accessing a home automation network via an ATM connection. Such a system can provide real-time access and control services to the home from remote locations over an ATM based wide area network (WAN).

Significant potential exists to apply such technology to the development of added-value services for the next generation of consumer electronic products for the home.

References

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

[2] Corcoran P.M., and Desbonnet, J. "A CEBus/Internet Gateway.", IEEE Intl. Conf. Consumer Electronics, Chicago, June. 1997

[3] Corcoran, P.M., Lusted, K., Humbourg, K, and Nolan, P., "An Active-Node Fault Diagnosis System for CEBus Networks", IEEE International Conference on Consumer Electronics, Chicago, June 1995.

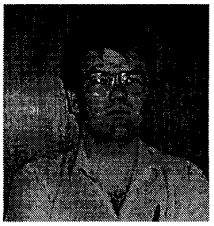
[4] Almesberger, W., "Linux ATM API", Internet Draft ftp://lrcftp/epfl.ch/pub/linux/atm/api, july1996.

[5] Almesberger, W., "Linux ATM internal signaling protocol", Internet Draft, july 1996.

[6] Almesberger, W., "ATM on Linux", Internet Draft ftp://lrcftp.epfl.ch/pub/linux/atm/papers/atm_on_linux.ps.gz, March 1996.

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

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 '87 for research work in the field of Dielectric Liquids. In '86 he was appointed to a lecturship 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, Rommania. His research interests include consumer electronics, embedded computing, networking, instrumentation and telecommunications technologies. He is a member of the IEEE.

Alexandru Cucos received his B.S. degree in Electronic Engineering from "Transil-vania" University Brasov, Romania, in1997. Currently he is completing the M.S. degree in Electronic Engineering at University College Galway, Ireland. His major interests include networking, Internet technologies, operating systems, VLSI design.