International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 4, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

By Using Internet Integration Technique Implementation of Embedded Systems

Prof.P.Rama Bayapa Reddy
Professor
Dept. of CSE, Dhruva Institute of Engineering and Technology, Hyderabad, India

Dr.K.Soundararajan
Professor
Department of ECE, JNTUACE, of Jawaharlal Nehru Technological University College of Engineering, Anantapur, A.P, India

Dr.M.H.M.Krishna Prasad
[3]Associate Professor of CSE & Head,
Dept. of Information Technology
University College of Engineering
JNTUK – Vizianagaram, A.P, India

Abstract - **Embedded devices have been recently connected to the Internet, enabling their management and monitoring from the Web. Although Web interfaces can bring several advantages to embedded systems, from designers to end-users, re-designing legacy systems in most cases may not be the most effective way to upgrade a system to the Internet. In this paper, a hardware/software architecture is presented for an embedded proxy agent to add Internet-awareness to an existing design. An approach to add graphical user interfaces through the Web is presented as well. This paper deals with different approaches and solutions for enabling network connectivity in the embedded systems domain. Furthermore an RTOS based open software architecture is introduced in order to realize flexible applications with functional and non functional requirements on network and internet connectivity. The main advantage of this "open solution" in comparison to plug & play solutions is that the desired network functionality can be customized, even if it is necessary to extend or violate some network protocol standards, which is not unusual in embedded system applications due to specific constraints and limitations.In the last few years there has been an increasing demand for the connection of embedded systems to the Internet for the purposes of monitoring and management of these devices. Embedded systems are systems implementing specific functions, such as printers, household devices, communication devices, avionics, medical instruments, and process control systems. The connection of embedded systems to the network enables the replacement of traditional and proprietary user interfaces to standard Internet interfaces, such as the use of familiar Web browsers from any desktop.**

I. INTRODUCTION

The technology and innovation in embedded systems now follows the line of progress of distributed business systems as shown on Figure 1. The Web Services Architecture (WSA) provides this reliability, scalability and flexibility needed for distributed embedded systems. The experiments in this paper are part of National Science Fund of Bulgaria project – "BY-906", 2005, entitled "Web Services and Data Integration in Distributed Automation and Information Systems in Internet Environment". Today, Internet accessibility in one form or another, if not an a priori requirement, is at least a highly desirable option in many embedded applications. Internet connectivity – or in more general words – network connectivity for embedded systems can be seen as an enabling technology for both current and future applications. Even today's low cost microcontrollers are strong enough to handle a LAN controller (on-chip, off-chip), a communication stack and usually multithreaded applications.

II. IMPLEMENTATION

Computer science is currently built on a foundation that largely assumes the existence of a perfect infrastructure. That is why even a single failure can ruin the whole systems. Among these complex systems are Distributed Embedded Systems (or Networked Embedded Systems - NEST), composed of many different nodes in need to communicate with each other. The tasks for these systems are dynamically assignable, and require reliable and predictable performance even though the nodes are individually unreliable. This approach is now feasible because Moore's law continues to reduce cost of computation and memory, new technologies are present for reducing size of the controllers, and the adaptation of well-known business standards in embedded systems is successful. Coordination, control and programming of embedded systems is currently an unsolved problem [2, 6]. Actually there are two types of network applications distinguished by their requirements for network capability. The first category of applications requires network connectivity as a core feature (functional requirement) to establish the desired functionality, e.g. all kinds of mobile computing, health care applications (patient monitoring), home automation and home security, as well as several environment monitoring tasks. Additional features, e.g. system maintenance, remote configuration, remote system control and monitoring, and software upload, are the motivations for the second class of applications. These so called non-functional requirements are primarily not required for the device's main tasks, but lead to increasing system flexibility.

Introducing Web Services Architecture (WSA) The WSA combines the best aspects of component-based development and the World Wide Web. According to [1] a service is: "a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols." Applications access web services via common Web protocols and data formats. The most common used protocol for transferring data in the Internet is HTTP (Hypertext Transfer Protocol) and it is the key transport protocol in the WSA. The universal schema in the Internet that codes the data is XML (Extensible Mark-up Language). The current structure of Internet is based on program-to-user interaction and the WSA is based on

program-to-program interaction [7, 8]. The WSA is based on some key standards: XML for data representation; SOAP for accessing services; WSDL for describing services; UDDI – registering and discovery of services [1, 7]. SOAP (Simple Object Access Protocol) is XML-based, lightweight protocol for data exchange in a decentralized, distributed environment. SOAP request packets call methods available on the SOAP server, and the server sends a response packet, structured in a similar manner [1, 7]. WSDL (Web Services Description Language) is a XML grammar for specifying properties of a Web Service, such as what it does, where it is located, and how to invoke methods on a particular service. WSDL is an interface definition language similar to those of CORBA and DCOM [1, 7].in fig 1 UDDI (Universal Description, Discovery and Integration) is a group of web-based registries that expose information about a business or other entity and its technical interfaces (or API's). By accessing any of the public UDDI Operator Sites, anyone can search for information about web services that are made available by or on behalf of a business. The information that a business can register includes several kinds of simple data that help others determine the answers to the questions "who, what, where and how" [1, 7, 8]. Figure

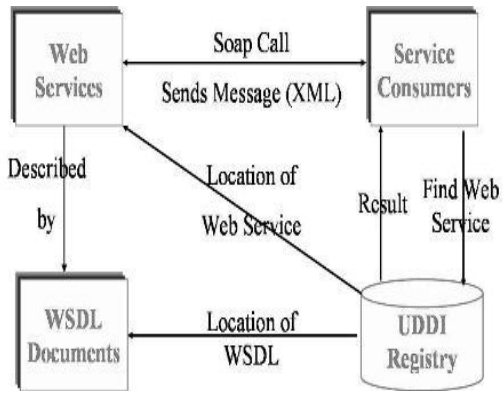


Fig.1. Web services. Components interaction.

III. SOLUTION

In the following a few out of the box hardware-/software-solutions are introduced:

3.1 LANTRONIX XPort

The XPort is connected to a serial interface of a microcontroller and enables web- and other network-access by means of a fully developed TCP/IP network stack and OS running on a high performance microprocessor with a 10/100Mbit Ethernet interface. Mechanically the XPort is integrated in a 33,9x16,25x13,5 mm3 RJ45-Package. The microcontroller software application communicates via high level commands (e.g. AT-commands) to the XPort.

3.2 Beck IPC@CHIP

The IPC@CHIP is an embedded controller designed to WEB- or LAN-enable products implemented in a 32 pin 600mil package. The hardware consists of a 16 bit 186 CPU, RAM, Flash, Ethernet, Watchdog and power-fail detection. The preinstalled software consists of a Real Time Operating System (RTOS) with file system, TCP/IP stack, web server, FTP server, Telnet server and Hardware interface layer, providing a high level API to the application programmer. The goal is to put the whole application into this chip.

3.3 High end Microcontroller systems

An embedded system based upon a high end 32-Bit Microcontroller – usually equipped with an on-chip LAN-unit –, such as the Coldfire (Freescale Semiconductors), has enough power to run an operating system like µCLinux together with inet-demon supporting full internet connectivity. The software API for network applications is the BSD Socket Interface [3].

The ARM architecture is particularly well suited to the real-time and deterministic requirements of embedded systems, and the implementation of RTOS running on those systems:

Small code footprint, allowing an RTOS to run from onboard memory

Fast interrupt response to reduce context switching overhead

Semaphore support via exclusive load-and-store instructions

SVC instructions and exceptions to support privileged state operation for the RTOS kernel

Separate stack pointers for each processor mode for easier memory management

Sleep modes to minimize power consumption

3.4 High end Microprocessor systems

An embedded system based upon a high end 32-Bit Microprocessor, such as a PowerPC, has enough power to run an operating system like a full Linux or RT-Linux. The design environment for networking applications is quite the same as for platforms, e.g. Unix- Workstations and PCs.

3.5 Low cost Microcontroller systems

Widespread used 8- and 16-Bit Microcontrollers are generally not equipped with dedicated LAN interfaces but with a variety of serial and parallel IO units. To obtain network and internet connectivity additional hardware (and software, of course) has to be added.

IV. PRINCIPLE APPROACHES

Basically there are three suitable approaches embedded systems engineers can apply in order to achieve internet connectivity for the embedded device:

4.1 Internet connectivity via a modem facility

By means of a standard serial interface (e.g. RS-232) a communication path to a modem chip or modem-device (fig.2) is established. In fig.3. The other side of the modem is connected either to a POT-, ISDN-, or xDSL-line. A TCP/IP-stack with a serial network layer (SLIP, PPP) has to be implemented into the systems software or firmware of the embedded device.

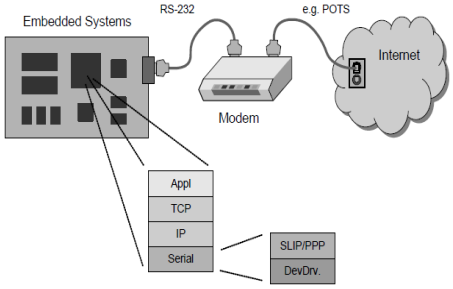


Fig.2. internet connectivity via modem

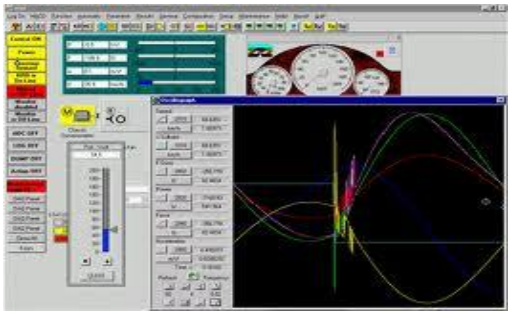


Fig.3:http connectivity

4.2 Using a Gateway-Computer

By means of a standard serial interface (e.g. RS-232) a communication to a gateway computer is established. In the fig 4 the gateway computer itself is equipped with a standard LAN/Internet interface; the entire internet related software (TCP/IP-stack, Web server with CGI-functionality [2]) runs on this gateway computer. There's no need for any standard protocol between the embedded system and the gateway.

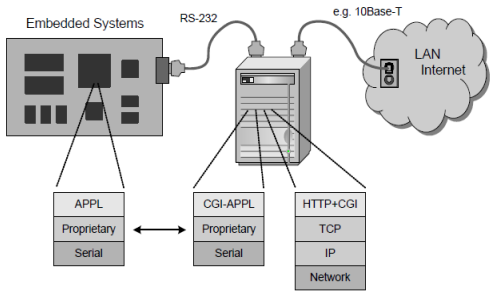


Fig.4. Internet connectivity via a gateway computer

4.3 Direct LAN access

The embedded system hardware is equipped with a LAN controller to access a network directly. The entire networking and internet related software (LAN-driver, TCP/IP-stack including all application modules like web-, ftp-, and telnet-server), as well as the application software run on the embedded system

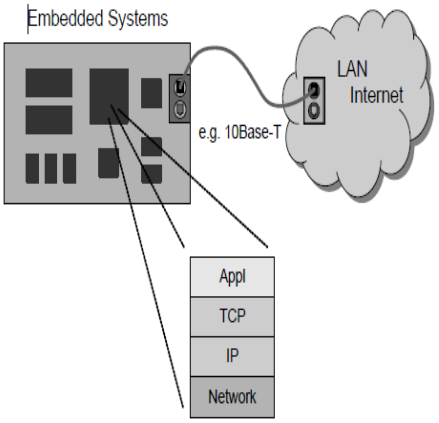


Fig. 5: Direct network / internet connectivity via LAN interface Load that program! Send me that file! The efficiency of a modern university hinges on moving information quickly and conveniently. If personal computers are the houses where information lives, Local Area Networks (LANs) are the avenues over which it moves.

Back in the days of prehistory, some 20 years ago, we didn't need networks, we had The Mainframe. We could share our files because they all resided in the same place; we only had to give other people permission to use them. We could share data, because the database was on the mainframe and everyone could log in and use it. We could share programs, because each program was stored in one place and everyone got a copy when needed. And we could share printers, because all printers were attached to the one computer that we all used.

Mainframes have worked well for us, but they have disadvantages. The two most significant are their lack of flexibility and their high cost. As personal computers came of age, many people started using them for their flexibility and economy. Everyone could choose his or her own word processor, own operating system, own printer, own database, and his or her own headaches.

The only problem was that some important things that mainframes did well were missing -- convenient communication and coordination among its users. (Carrying a floppy disk from one machine to the next, aka sneakernet, does not count as convenient communication.) LANs were invented to replace these missing functions, without sacrificing the economy and individuality of the small machines.

V. TYPES OF LANS

A Local Area Network, as the name suggests, connects machines in close geographical proximity, although exactly what "proximity" means can be stretched. The term "Wide Area Network" (WAN) is used for networks that expand beyond the campus or office; the Internet is the best known example of a WAN. (The Internet is also an "internet", a collection of networks acting as one.) Usually a WAN will have one or more slow links, perhaps over telephone lines between cities, whereas all the links in a LAN will be fast. This difference in speed is important for optimizing the overall network performance.

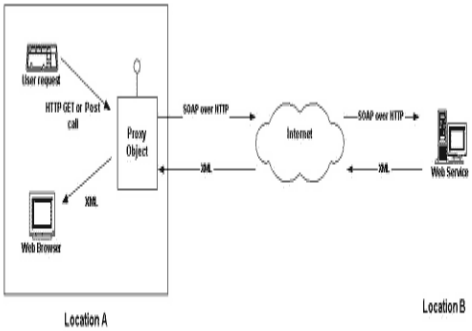


Fig.6. Providing web services through Proxy Object

The architecture is still three-tier and the middle tier is a proxy.

The usual use of the of term "LAN," however, implies more services than simply making connections between local machines. On a LAN, we expect to share files, programs, or printers, all without being particularly aware of where the physical resources we're using are actually located. LANs providing these types of services are typically set up either as "peer-to-peer" or "client-server" LANs, or perhaps as a combination of the two.

1) Peer-to-peer LANs

All the machines on a peer-to-peer LAN are equal. Provided that the file's owners give permission, a file on machine A can be accessed from machine B, and vice versa. Peer-to-peer LANs do not require any one machine to be a dedicated, high-performance server; service by a peer-to- peer LAN is often cheaper for this reason. Peer-to-peer LANs work well when only a small number of machines are connected to it. But as the size of the LAN grows, peer-to-peer services can become quite disorganized, and because each machine on the LAN must be powerful enough to serve all of its peers, the cost increases. For larger LANs, the dedicated client-server LAN architecture becomes more cost effective.

2) Client-server LANs

A client-server LAN consists of one or more server machines on which shared files and programs reside and many client machines where people do their work. The LAN server machines are usually big and fast because they must serve many users, while the client machines need only be fast enough for one person to use at a time. Shared printers are either attached directly to a server, or to a print server (a specialized computer attached to the network), or to a personal computer on network that acts as a print server. Connecting a LAN: The Physical Connection Regardless of type, all LANs require special hardware. The usual parallel and serial ports that come with personal computers are not fast enough for most uses on a LAN. At UIC, and probably most universities, each desktop computer that will be networked on a LAN must have an Ethernet card, which gives the desktop computer a third, very fast fig.7. novell network communications port.

Fig.7. In addition to LANs based on Ethernet hardware, in which data is broken up into small "packets" for transmission, including Twisted Pair Ethernet, Fast Ethernet (100 Mbps vs only 10 Mbps for regular Ethernets), and Gigabit Ethernet, there are "token passing" networks such as Token Ring and FDDI (Fiber Distributed Data Interface), and "cell relay" networks such as ATM (Asynchronous Transfer Model). Ethernet, Token Ring, and FDDI are all "baseband" networks -- the wires carry only one signal (aka channel) at a time. ATM, which is part of B-ISDN (Broadband Integrated Services Digital Network), on the other hand, is "broadband", in which a single wire can carry multiple channels simultaneously.

All types of networks differ, of course, in cost and speed, and in necessary software and hardware. And all of these types of network are, or have been, used at UIC. (For more information about ATM, see Building the Data Highway and the other articles in the November/December 1996 ADN Connection newsletter.) Once a desktop computer has an appropriate network card installed, it can be connected to other computers through cables, hubs, and routers, so that the information can flow from one to another as quickly as possible. This is precisely how computers at UIC are connected to the UIC campus network and ultimately to the Internet. Commonly, "LAN hardware" also includes dedicated machines like file or print servers, which provide LAN services to really make the network a LAN.

Connecting a LAN: The Logical Connection Desktop machines on a LAN also require special software, which breaks a file or other information into packet-sized chunks, adds information about where the packet should be sent (this is called an envelope), and tells the network card to send the packet. This software allows the machines on a LAN to make use of the various services available on the LAN. There are several logical protocols used for network traffic. Fig 8 At UIC, most network traffic uses IP (Internet Protocol); Novell Netware's IPX (Internetwork Packet Exchange) and ATM are also quite important. Apple LocalTalk networks use DDP (Datagram Delivery Protocol). An important point is that the same network card can handle many different software protocols and can do so simultaneously. In particular, your computer can be connected to the UIC campus network and to a departmental LAN through the same Ethernet card and cable. This situation already exists in the ACCC public PC labs. The lab PCs are part

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 4, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

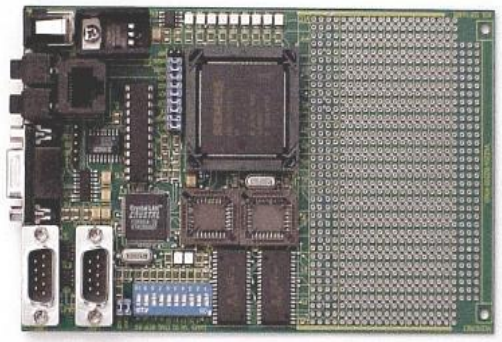


Fig.8. Keil MCBnet-167 evaluation board

of a Novell LAN and also are attached to the UIC campus network and the Internet; you use IP on these machines when you logon to a workstation account or use a Web browser to explore the Internet, but you use IPX when you do a dir your H: home directory provided for you by the LAN server.

Behavior of a LAN Client What changes can you expect when your personal computer is added to a LAN? There will naturally be a few differences in the way it acts. First, you will need to logon to the LAN. After you logon to the LAN, remote files will appear to be directly on your machine. You'll have new drives on your PC, with new drive letters. For example, in the ACCC PC labs and with LANS Active Directory Desktop Applications, you'll have a drive that is a directory on the server that provides you with private disk space. There might be other drives holding software and data that the machines on the LAN can use (provided that suitable accesses permissions are set). The file permissions associated with your PC's LAN account will specify which of the LAN services you can use, which disks you can access, and whether you can write on or otherwise modify the disks or just read their contents. Often, special licensing and installation procedures are needed when installing software on LAN servers, but once it's installed, you can use software on LAN drives exactly as you'd use it if it were on your own drive.

VI. Open Software Architecture for Embedded Networking Applications

In our case study we use an embedded system equipped with Infineon's C167 low cost microcontroller together with the Cirrus Logic CS8900 Ethernet controller chip (see fig. 4) in conjunction with the µC/OS-II real time operating system kernel [1] as an open design platform for multithreaded applications with network and internet connectivity. The main advantage of this solution in comparison to plug & play solutions like the LANTRONIX XPort and the BECK IPC@CHIP is that the desired network functionality can be customized, even if it is necessary to extend or violate some network protocol standards, which is not unusual in embedded system applications due to specific constraints and limitations. he Keil MCB167-NET single-board computer is an evaluation board for the Infineon C167CR Microcontroller. The MCB167 allows you to investigate the capabilities of the C167 and embedded internet applications and create real working programs with the Keil development tools. The board provides a capable prototyping platform for your C167 projects.

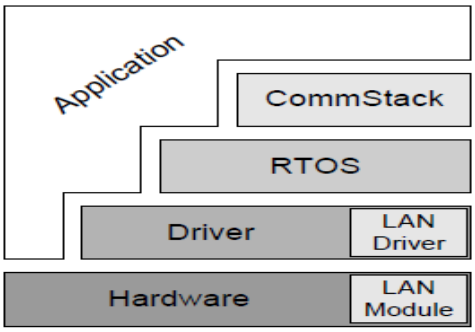
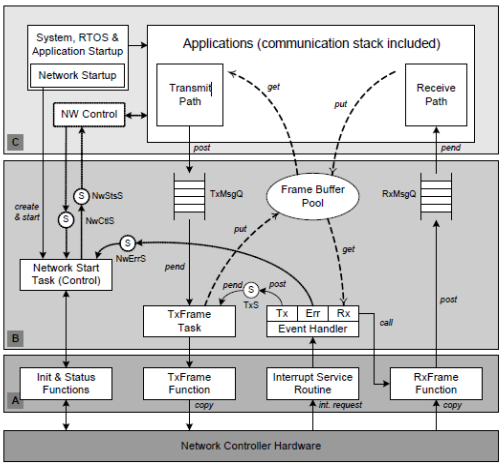


Fig 9: Basic software architecture

From fig.9 the application's point of view there are several abstraction levels as shown in figure 5. A typical network based application doesn't access directly the LAN-chip hardware or the functions at driver level. Usually a communication stack (e.g. TCP/IP, UDP/IP) provides an appropriate application programming interface (API) to the network tasks of the application. Furthermore, both the application and the communication stack make use of the features of the underlying multitasking- or real time operating system.

6.1 RTOS-based software architecture

The goal of our approach is a transparent integration of the network functionality by means of inter-process communication (IPC) objects offered by almost all multitasking- or real time kernels used in the embedded systems domain. Figure 6 shows the software architecture in detail. The basic hardware abstraction is done by a set of driver functions located in layer A. Layer B realizes an event oriented interface at network frame level to the application (with or without a dedicated communication stack) in layer C. Both layers B and C need operating system (OS) support.



International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 4, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

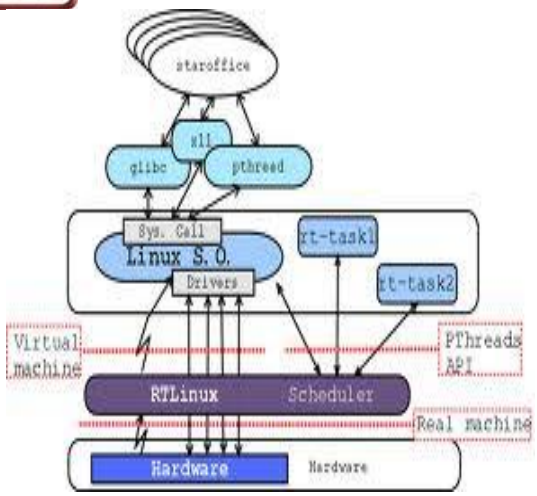


Fig.10 &11. RTOS-based networking software architecture All major vendors of Real-Time Operating Systems (RTOS) support ARM architecture.

Fig. 10 Many embedded systems require software to respond to inputs and events within a defined short period. Such systems can be categorized as hard real-time, where missing a response deadline is unacceptable (for example an anti-lock braking system), and soft real-time, where hitting a deadline is desirable but not critical. In both types of system, a degree of determinism is important.

RTOS is designed to control an embedded system and deliver the real-time responsiveness and determinism required by the controlled device. Applications run under the control of the RTOS, which schedules allocated CPU time.

In modern systems, a RTOS consists not only of a real-time kernel, but also higher-level functions such as device management (USB, UART, Ethernet, LCD, etc.), file systems, protocol stacks (CAN, TCP/IP, HTTP, etc.) and graphical user interfaces (GUI).

See the "RTOS vendors" tab below to see a table of ARM Connected Community Partners whose RTOS support the ARM architecture.

Fig.12. The communication paths

From Fig. 12 After a successful startup of the networking hardware and the objects of layer B a frame based communication can take place. Let's first consider the receive path. After the LAN controller has received a complete MAC frame the receiver part (Rx) of the Event Handler is invoked by the Interrupt Service Routine which is triggered by a hardware interrupt. A buffer is allocated from the Frame Buffer Pool and the Rx Frame function is called to copy the whole frame from the LAN-controller to the buffer. After that, a pointer to this buffer is posted to the received frame message queue (RxMsgQ). This signaling of an event makes a waiting task at layer C ready to run. Depending on the communication stack the frame-payload is processed and delivered to the associated application task. The uppermost level of the communication stack – or maybe the application task itself – is responsible for returning the buffer to the buffer pool when the frame isn't needed anymore. The transmit path works very similar; an application task or the top layer of the used communication stack allocates a buffer and put in the payload data at an adequate position. The buffer is passed down the communication stack and specific header information is put into the buffer at each level. Finally the buffer's address is posted to the transmit frame message queue (TxMsgQ). The TxFrameTask is scheduled and the frame is copied to the LAN-controller by means of the TxFrameFunction. After that, the buffer is put back to the buffer pool. The transmit semaphore (TxS) is necessary to synchronize the TxFrameTask to the LAN-controller in order to lock the controller's frame memory unless the previously copied frame is transmitted over the network.

6.3 Buffer management and the zero-copy approach

The Frame Buffer Pool is a memory partition with a sufficient number of fix-sized memory blocks large enough to hold the maximum size network MAC frame together with the management header MH (Extended Frame, see figure 13a). The management header is not standardized in any way and holds additional information needed for the levels of the communication stack (see figure 13b). Usually the MH holds a set of pointers to the corresponding protocol headers and the start of the payload.

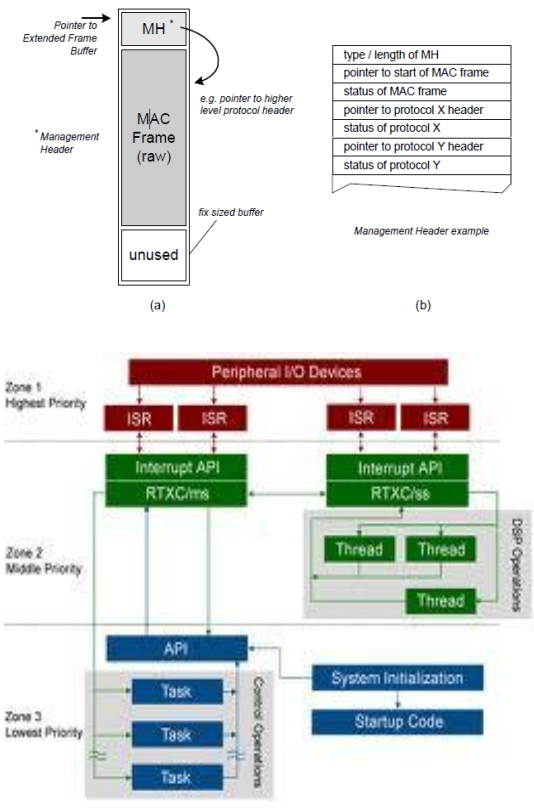


Fig.13. a, b Extended Frame architecture

All the way up and down the levels of the communication stack there is no need to copy any data due to the zero-copy approach. Information is passed from one level to another only by handing over the pointer to the Extended Frame Buffer. A certain level knows the position of its relevant fields in the frame by means of the pointers supported in the management header. Each level, of course, has to know the structure of the management header, as it knows the organization of its protocol header as well.

6.4 Initializing, control, status and error handling

To initialize the network the application has to create and start the Network Start Task. This is usually done during normal system startup, right after starting the operating system. The Network Start Task itself creates all necessary tasks and inter-process communication (IPC) objects and calls the driver level initialization functions (layer A). After successful completion of the Network Start Task's main function the event oriented network interface is up and running. Further control operations and status requests can be handled also by the Network Start Task by means of IPC-objects such as semaphores (shown in figure 6) in conjunction with shared memory regions or additional message queues. Layer B might be extended by a dedicated Network Control Task to implement a more elaborate network management functionality, e.g. to provide a basis for SNMP. Networking errors are usually signaled via hardware interrupts by the LAN-controller. The associated Interrupt Service Routine (see figure 6) calls the Err part of the Event Handler, where the network error semaphore NwErrS is posted and the Network Start Task (or a dedicated Network Error Task, not shown in figure 6) is scheduled in order to perform an escalation procedure to inform the afflicted application task.

6.5 OS requirements

As mentioned earlier we use the real time kernel µC/OS-II; currently in the release V2.51 ported for C167 microcontrollers and the Keil-IDE. For implementing the layer B functionality there are some requirements to operating systems in addition to normal task-management:

- semaphores
- message queues
- simple memory management

Basically almost all operating systems fulfill these requirements. If memory management isn't part of the OS, this feature can be realized very easily using the standard functions malloc() and free(). If the OS doesn't provide message queues, this feature can be realized using circular buffers in conjunction with general semaphores to synchronize the conditions empty and full. If the OS doesn't support semaphores (or semaphore like synchronization objects) it would be a good idea to think about choosing another OS.

VII. CONCLUSION

Embedded Linux Systems and their importance in the control system have been addressed in this report. Portable Linux is being used most widely in the control field. Many vendors are designing control cards based on Linux. These systems are being used to measure and control certain parameters in the accelerator. An embedded Linux prototype card was used to measure the different parameters of the accelerator and control them. These results were compared with the results taken by some other data bus solutions to check the performance of this control card and the results were almost identical. These kinds of cards are useful in small to medium level machines but are not feasible for large The open software architecture introduced in this paper is suitable for almost all microcontroller platforms equipped with a LAN circuitry and a ported RTOS to implement a software layer with basic functionality necessary for both communication stacks (e.g. a TCP/IP stack) and flat networking applications. Machines and machines where there is shortage of manpower. These solutions can be used in research field and lab assignments also.

REFERENCES

[1] Labrosse, Jean J., Micro C/OS-II The Real-Time Kernel, R&D-Books
[2] Jan Axelson, Embedded Ethernet and Internet Complete, Lakeview Research
[3] W. Richard Stevens, UNIX Network Programming - Volume 1, Networking APIs: Sockets and XTI), Prentice Hall
[4] RFC791 Internet Protocol http://www.faqs.org/rfcs/rfc791.html
[5] RFC793 Transmission Control Protcol http://www.faqs.org/rfcs/rfc793.html
[6] RFC821 Simple Mail Transfer Protocol, http://www.faqs.org/rfcs/rfc821.html
[7] RFC826 Ethernet Address Resolution Protocol: Or Converting Ne http://www.faqs.org/rfcs/rfc826.html.
[8] Robert Karban, Rudlof Hauber and Tim Weilkiens. MBSE in Telescope modeling
[9] Lotos, ISO international standard IS8807 , Feb 1989.
[10] Miroslav Popovic, Communication protocol engineering , CRC press 2006.
[11] Omg, Systems Modeling Language (SysML) Specification, 2005
[12] Omg, http://www.sysmlforum.com/FAQ.htm , june 2010
[13] Pete Loshin, TCP/IP Clearly explained 4th ed. Elsevier 2003
[14] Austin, D., A. Barbir, C. Ferris, S. Garg. Web services architecture requirements, http://www.w3c.org/TR/wsa-reqs.
[15] Borriello, G., R. Want, Embedded Computation Meets the World Wide Web,Communications of ACM, Vol. 43 №5, May 2000, pp. 59-66.
[16] Channabasavaiah, K., K.Holley, M. Edward, Jr. Tuggle, Migrating to a service oriented architecture, Part 1(2003), library/ws-migratesoa/index.html.
[17] Davis, D., M. Parashar. Latency performance of SOAP implementations. In 2ndIEEE International Symposium on Cluster Computing and the Grid, 2002, pp. 407-412,
ISBN: 0-7695-1582-7.
[18] Engelen, R., Code Generation Techniques for Developing Light-weight XML Web Services for Embedded Devices, ACM SAC'04, March 14-17, 2004, Nicosia, Cyprus, pp.854-861, ISBN:1-58113-812-1.
[19] Estrin, D., G. Borriello, R. Colwell, J. Fiddler, M. Horowitz, W. Kaiser, N. Leveson, B. Liskov, P. Lucas, D. Maher, P. Mankiewich, R. Taylor, J. Waldo, Embedded, Everywhere. A Research Agenda for Networked Systems of Embedded Computers, NAP Washinton, D.C. 2001, ISBN 0-309-07568-8.
[20] Kreger H., "Web Services Conceptual Architecture (WSCA 1.0)", IBM Software Group, May 2001, www.redbooks.ibm.com.
[21] Stoilov T., K. Stoilova, Integration of Web Services in Internet, 18th International Conference "SAER-2004", 24-26 September, St. Konstantin resort, Varna, Bulgaria.

[22] Spasov G., N. Kakanakov, N. Lupanov, Three-tier distributed applications, Computer Science'2004, 6-8 Dec 2004, Technical University Sofia, Bulgaria, pp. 172-177.

AUTHOR'S PROFILE

Prof. P. Rama Bayapa Reddy
completed BE in computers from Marathwada University, Aurangabad and M.Tech. from JNTU, Anantapur. Now presently pursuing Ph.D from JNTUA, Anantapur. My interested Research area is internet controlled embedded systems. I conducted two national level seminars. I also conducted two workshops on Real Time Embedded Systems. Also I attended workshops conducted by JNTUK. I have total of 15 years of teaching experience. Right now I am working as Professor in Computer Science and Engineering in Dhruva institute of Engineering and Technology, Hyderabad. putluru.ramreddy@gmail.com

Dr. K. Soundararajan
received the B.E Degree in Electronics and Communications from S.V.U, Tirupati and M.Tech. Degree in Instrumentation and control Engg. from J.N.T.U, Kakinada. He received Ph.D. from University of Roorkee, Roorkee. He is having 32 years of teaching experience. He got the Best teacher award for the year 2006 from Andhra Pradesh Govt. and from Lead India2020 in 2005, President of India Award in Bharat Scouts & Guides in 1968, Best Paper Award in 1990–91 and 2000 from Institution of Engineers (India) for best technical paper published in Journal. He is an Expert Committee member, for different central Govt. organizations, affiliation committee member for several colleges and different academic boards. He published 21 International Journals/Conferences and 27 national journals/conferences and he participated in 12 National seminars. He guided 9 Ph.Ds and 10 research scholars are working to obtain their PhD. He worked as principal of JNT university college of engineering Anantapur, A.P.) and Rector of JNTU ANANTAPUR, Anantapur. Presently, he is working as Professor, Department of ECE, JNTUACE, of Jawaharlal Nehru Technological University College of Engineering, Anantapur, A.P., India. soundararajan_jntucea@yahoo.com

Dr. M.H.M. Krishna Prasad
has completed B.Tech, M.Tech. (CSE), JNTU, Hyderabad Ph.D., in Computer Science & Engineering from JNTU Hyderabad with specialization as Data Mining and successfully completed a two year MIUR fellowship at University of Udine, Italy. He has published no of papers in International and National Journals. Under his guidance multiple research scholars are doing research. Presently he is working as Associate Professor of CSE & Head, Dept. of Information Technology University College of Engineering JNTUK-Vizianagaram Campus, VIZIANAGARAM, A.P, India. krishnaprasad_mhm@yahoo.co.in