(12)

Filaments: Lightweight Network Interfaces for Embedded Devices

by

Pehr C. Anderson

Submitted to the Department of Electrical Engineering and Computer Science
on August 24th, 1998 in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

August 24th, 1998

Author_____
Department of Electrical Engineering and Computer Science
August 24th, 1998

Certified by_____
Professor Neil Gershenfeld
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Filaments: Lightweight Network Interfaces for Embedded Devices

by

Pehr C. Anderson

Submitted to the

Department of Electrical Engineering and Computer Science

August 24th, 1998

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

Abstract

Filaments are low-cost interfaces for attaching devices to an Ethernet network. While most networking research pushes for faster high-end systems, the filament project targets low-end devices with the goal of making networking easy. Filaments wrap the complexity of the network into a simple and convenient package. The vastmajority of device communications require only a tiny thread or filament of connectivity. Before filaments, one could either tie each device to a desktop PC or to deploy a specialized device network. Filaments allow you to leverage the same network used by desktop computers without making devices dependent on them.

Contents

List of Figures

List of Tables

Dedication

This thesis is dedicated to the Free Software Foundation and the Open Source philosophy. Freedom of expression is a fundamental human right whether spoken, written, or encoded as bits and set down a wire. When great works are shared with humanity, we all benefit. Many thanks to the developers of GNU[1] and their campaign against software hoarding. Without open source software, this thesis would not have been possible. Special thanks go to the LyX team[2] for developing the WYSIWYM front-end for LaTeX, used in the preparation of this thesis.

[1] Homepage of the GNU effort: http://www.gnu.org
[2] Homepage of the LyX development project: http://www.lyx.org

Introduction

As more people depend on digital information for their daily lives, the paradigm of the desktop computer is being stretched to its limit. In order for the the next generation of computers to be more usable, they must be less invasive. The desktop paradigm works well for isolated tasks, but it does not integrate well with the physical world. We need to take computers off the desk and hide them in the environment. The Things That Think Consortium works to move information technology beyond clunky desktop interfaces. By moving computers into the background, people should be able to live without doing all the bookkeeping by hand.

Things That Think

The MIT Media Lab has several consortiums which aim to improve different technologies. The goal of the Things That Think consortium is to push computers off the desktop and into our environment. By giving ordinary objects the ability to do book-keeping and information processing we make it possible to build interactive systems without having to type in data or fill in forms every step along the way.

This technology may embedded in a variety of everyday objects. For example, the shoes you wear might contain a digital assistant which acts as your wallet, keycard, and a personal information archive. A locked door could recognize you by a code from your shoe, opening automatically when you turn the doorknob. The shoe might also be able to power its self, stealing power from your step as you walk.

To make intelligent things, you need to be able to put intelligence into objects. Embedded computers have been an integral part of our lives for many years, running microwaves, car engines, airplanes, TV sets, and even children's toys. For truly intelligent behaviors, these embedded devices will need to talk to each other, or to the Internet.

Imagine a refrigerator that knows how cold you like your milk, or a drink machine that tunes the beverage to match your tastes. A refrigerator could keep track of what's inside. It could let you know when your foods have expired or order replacement items automatically.

It may be hard to imagine deploying intelligent refrigerators, but consider an intelligent medicine cabinet built out of the same technology. The cabinet provide a detailed record of dosage and consumption for patient monitoring purposes. Accurate medical information can be the difference between life and death when an outpatient mismanages their own medication. There is a strong interest in the medical profession to find ways of reducing the chances for error. One approach is to develop intelligent devices which interact with the patient's database records.

By taking ordinary things and adding new intelligence, people can live, work, and play with new freedom, secure in the knowledge that things are taking care of themselves.

A Pragmatic View

There are many ways to build intelligence into ordinary objects, but to focus on what is practical today, one has to look at the current state of computing. The key to taking advantage of intelligent objects is to have an information infrastructure that can handle and process the added information. Most consumers don't have a network in their home beyond the telephone line, cable TV, and wiring for their stereo system. As devices benefit from being able to talk to eachother, more devices will want to join a digital network in the home. A pragmatic approach to technology development would start with the customers who have the most incentive to upgrade their information systems.

Corporations spend large amounts of money on re-engineering their core businesses processes. An efficient business wants to optimize the flow of information through the organization. If a customer fills in a web form instead of calling an operator to make a purchase, the business as a whole will be able to spend less money and make a greater profit.

On-line commerce can be very efficient. Customers can track a product as it is being shipped via FedEx. Wholesalers can offer detailed catalogs that provide as much or more service than the traditional retail vendors. With efficient shipping, even a small store can keep itself from running out of toothpaste or paper towels. If replacement goods are automatically ordered, then the store will only need to stock a few days' supply of each item.

The key to controlling the cost of doing business is controlling the cost of collecting and managing information. Great strides are being made in information management, but there are still inefficiencies in the area of data collection. FedEx has achieved tremendous success from tightly controlling the flow of packages through their organization, but what about smaller companies with less stringent requirements?

There are two fundamental barriers to the widespread collection of physical data: cost and complexity. Some can afford to overcome these barriers today, but many will wait for these barriers to fall. A pragmatic organization does not change its information infrastructure lightly. This document

proposes a device for reducing both cost and complexity to an acceptable level, allowing new innovations in data collection and information management to be used by even the most pragmatic of corporations.

The Network

In order for embedded computers to behave intelligently, they need to be able to share information. There are many technologies for embedded networking available today. The most important requirement is that the network be usable. The ideal network would be simple, well known, understood by system administrators, low in cost, robust, fast, and non-proprietary. Fortunately there is already a network that does all this. Ethernet[3] is the workhorse of the computer networking world. Ethernet connects desktop computers from every vendor as well as file servers, printers and and internet routers. Additional protocol may be transported inside Ethernet messages, helping to glue the disparate systems together.

Every device needs a connection to the network. Your shoes aren't going to help you get through any doors if the door isn't wired. There are many options available for building high-end networked devices, but in the world of Things That Think most things will only need the tiniest thread of connectivity. Filaments are intended to be the thinnest possible connection for attaching devices to a standard Ethernet network.

Local Area Networks are present in a large number of businesses in the US and around the world. These networks are mostly used for print sharing, file sharing, and web access. Increasingly, people are looking at this network and wondering why they have to plug devices into a PC when the network is right there. Printers are the most popular embedded network devices, however many shared resources could be readily attached to the same network. Filaments will greatly simplify the development of hardware with embedded Ethernet, making it possible to wire up just about anything.

Examples

Throughout this document, examples will be given to express the ideas of ubiquitous networking. It is important to use something familiar to illustrate how things can be made easier. An airport is a classic large system. Many components of its infrastructure must work reliably every day.

There are many types of information processing at an airport, and many different pieces of information that have to work together. Heterogeneous systems must share information and coordinate in order to operate smoothly and avert disaster. Everything from the computer terminals at the check-in lines to the baggage handling equipment can be thought of a s part of a single network infrastructure.

---

[3] *Ethernet* is a registered trademark of Xerox Corporation.

The Filament is an essential piece for extending the network into the realm of objects and things. Today most systems are either completely isolated or have desktop computers on both ends. With inexpensive, lightweight filaments new interfaces can be dispersed without affecting the wiring plan. New security features can be implemented without installing new equipment, and all transactions can be logged for later study without any additional human intervention or cost.

Warehouse Example. Most organizations beyond a certain size need to maintain a warehouse. A warehouse can be big or small, structured or ad hoc. Different systems will have different needs, but all need to do the basic functions of tracking inventory and managing the storage area.

In an ideal world, the warehouse would keep track of its own inventory, without someone typing into a computer every time a box enters or leaves. Shelf space for new parts could be allocated automatically. Traffic flow within the warehouse could be analyzed off-line, allowing popular items to be moved closer to the loading dock for faster availability. Efficiency is gained by maximizing throughput with the fixed number of resources.

Work schedules could also be adjusted to fit the load profile based on usage data. With a detailed knowledge of the system load and performance capcity, workers could be scheduled to handle peak loads while performing other duties when the warehouse is idle.

Security System Example. Filaments are ideal for simple interactions with the real world. Very little data is involved in knowing whether a switch is on or off, whether a door is open or closed, or whether there is anybody moving around in different parts of the building. It should be easy to put together a system that will measure these things.

A security keypad has little need for a high bandwidth network interface. A user types an access code, causing an electronic lock to be released. Existing network components are tuned for high bandwidth services. The need for a simple low-cost interface is clear.

A more comprehensive system might use a magnetic card reader or a RF ID scanner, logging the identity and arrival time of each person, possibly storing motion tracking data as well. These types of application require that the system be readily interfaced to a networked server. Software can be added on to provide database service, remote access, and administration. Once installed, the system can be as simple or as comprehensive as the user wants. The hardware is simple and static. All the important functionality is software.

Introducing Filaments. While Ethernet works on the desktop, it is still challenging and expensive to design an embedded product that works with Ethernet. There are a few embedded devices with Ethernet functionality on the market. Printers, routers, print servers, and some high-end custom equipment. Ethernet-based serial ports are available from several vendors priced between $250-$500/port depending on configuration. Unfortunately those prices are roughly equivalent to a

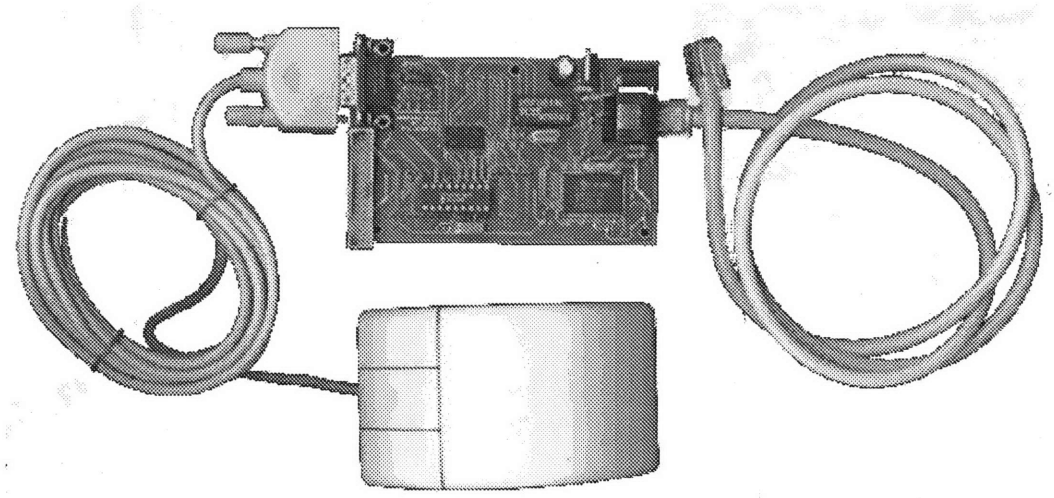


FIGURE 0.0.1. Ethernet Mouse based on the first generation Filament

stripped down PC, which costs approximately $500 with two serial ports and network functionality built in.

What is missing is a way to connect low-end devices directly to the network without cost or complexity. What is needed is a light weight filament of connectivity. What is needed is a network interface that works for everything from mice to toasters.

This document introduces filaments as a solution for networking low-cost embedded devices. This network interface is physically small and has minimal complexity. An important goal is to make filaments easy to use. An engineer should be able to add and remote access to an existing project with very little work.

Chapter one introduces the concepts involved in embedded networking and the tools for creating embedded network devices.

Chapter two describes the problems involved in various embedded network strategies. The solutions of specific vendors are discussed and evaluated.

Chapter three describes the architecture of the filament chip. Real application scenarios are used to illustrate the design tradeoffs.

Chapter four documents the software interface for communicating with the filament chip. This reference provides information necessary for writing new applications to communicate with the filament chip.

Chapter five explores the world of pervasive networking, extrapolating price/performance trends and envisioning new scenarios of ubiquitous communication.

CHAPTER 1

Networking Beyond the Desktop

Early computer networks were used for moving files. Large mainframe computers transfered data from one machine to another using UUCP (UNIX[1] to UNIX copy) over a network. Creative users added services on top of the basic operations to facilitate communication. Terminals allowed people to communicate more fluently with the machine. A terminal provided an inexpensive window through which a user could manipulate the machine.

Modern networks are still used for sharing files, but also provide client/server services like shared access to databases. A huge industry has developed around providing middleware, software that sits between clients and servers and makes both work more effectively.

A server is a high speed machine for serving files, running databases, running middleware, and providing access to other networks through routing and protocol translation.

A client is a desktop PC with a monitor, keyboard and mouse. A client runs programs locally and also has a local file systems, but the most important use of a client is for viewing and interacting with the information stored on servers.

Middleware does things like serve web pages, forward email, process payroll records, and just about any other information processing application.

Traditionally, a user logged on to a single computer with a terminal or terminal emulator. The server provided access to local services and information. This is known as a two-tier system. Modern information technology (IT) solutions ares still built with clients and servers.The limitations of a two-tier architecture have become apparent. Newer systems are three-tier or 'N'-tier with multiple layers of middleware moving information between users, databases, and transaction servers.

The web has ushered in a new era of client/server computing. The uniform resource locator is an important innovation. URLs provide a simple way of jumping from one server to another without the previous headaches of logging off, connecting to a new site, and logging on again. Whereas traditional client/server computing scales up to thousands of simultaneous users, the web scales up to millions of users. This isn't just a matter of faster computers, it has to do with the nature of the protocols themselves. The telnet and TN3270 protocols require that the host maintain state information for each user. HTTP (the protocol used for requesting and receiving web pages) is a

[1] UNIX is a trademark of X/Open

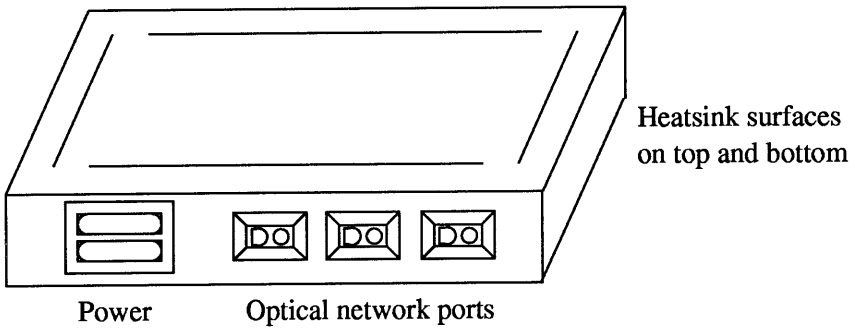


FIGURE 1.1.1. A simplified CPU package: power, network, and heat-sink

stateless protocol, which means that the server does not have to keep track of the clients. Hosts just deal with requests, as they arrive.

While the network is installed in the workplace it has not permeated silicon. Most devices require external components for any sort of network connectivity. A limited size bus can be the half-way point for meeting of networkable and raw components, but to truly explore the possibilities of ubiquitous networking we must be able to link up even the simplest of devices.

1.1. Hardware

A CPU needs three primary interfaces: power supply, communications bus, and heat sink. Today CPUs have several hundred of pins for connecting to many different of parallel busses. These pins take up space in both the physical package and the on the silicon die. Pads must be placed on the silicon such that tiny wires can connect to the external pins of the package.

While parallel busses are extremely effective at moving large volumes of data across short distances, on-chip integration and high-speed network technology are pushing back. With more components implemented on-chip, the need for an in-system bus is reduced. With high speed serial buses for in-case communications, there will be little need for high pin counts on many components. An audio amplifier will need both a network interface and the interface to the necessary analog support components. Intellectual property in the form of ready-to-use networked components will be the currency of the electronics industry.

Current computer architectures run synchronously, requiring a uniform clock to span the breadth of the motherboard. As clock rates continue to rise, it becomes less and less possible to carry signals that distance while retaining coherence. The effects of skew creep in from minute differences in trace lengths and capacitance. Switching to asynchronous architectures is the easiest way to overcome such a barrier. Manchester encoding used in Ethernet has the clock embedded in the signal, allowing data to be send on a single channel, eliminating the problems of skew and network length.

Ideally every IC would look something like the above. Every part of a complex system needs to be connected to other parts of the system. Currently, the connections between components must be wires, with all the associated disadvantages. The capacitance of the wire requires a large current for bit transitions. A typical IC pin has several picoFarads of capacitance which add to the delay. Higher resistance in the wire means that more power must be dissipated for a transition to occur. Although optics are still delicate and power hungry, it is possible that they will eventually outperform a parallel busses, even inside the case of a single desktop machine.

CHAPTER 2

Existing Solutions for Embedded Networking

2.1. Serial Communications

Bus technologies have evolved separately from network technologies. While both buses and networks are designed to move data between nodes, the design parameters are very different.

2.1.1. Synchronous Serial Communications. Synchronous protocols are simpler than asynchronous protocols when the devices are share a common clock. Typical communications involve sending a train of clock pulses to signify data, while raising a frame sync line to wake up each device in turn. Examples are SPI Microwire, 2 wire, and 3 wire protocols.

As networks scale up, it becomes difficult to control clock skew. Telephone switches have traditionally been synchronous, shoveling data and clock signals across long distances. A significant part of their complexity comes from synchronizing the clocks from multiple sources and dealing appropriately with skew. Asynchronous transfer mode (ATM) is one way to deal with such problems. Another way is to build the network on a fundamentally asynchronous protocol, like gigabit Ethernet. Ethernet is Manchester encoded, which means that the clock signal is embedded in the data, completely eliminating the problem of skew.

2.1.2. Asynchronous Serial Communications. Communication between devices can be accomplished in many ways. Over the years, electrical engineers have developed serial communications protocols to solve various problems and make different pieces of hardware work together.

The RS-232C specification was frozen in 1969. While computers have gotten faster every year, many new devices are tied to this historic protocol. The original specification allowed speeds of up to 19,200 baud and cabling of up to 50 feet. High speed UARTs became the de facto standard on the PC platform, giving a new maximum speed of 115,200 baud. Some systems can go faster but the hardware requires special device drivers. Most software is written for the lowest common denominator, defaulting to 19,200 baud and allowing the user to reconfigure the speed up to 115,200 baud.

The original application for RS-232 was for Data Terminal Equipment (DTE) to talk to Data Communication Equipment (DCE). Using RS-232 one could attach a communications device to a computer (DTE).

While the data signalling part of RS-232 is nearly trivial, the out-of-band signalling represented by the other pins has caused many man-years of frustration. The complexity of the extra signal lines comes from the wide variety of devices they are intended to connect. Accordingly, 22 pins of a 25 pin connector have special function defined in the RS-232C specification. Most implementations only use 9 pin connectors, and some devices that use RS-232 only make use of three pins: transmit, receive, and ground. The additional lines are reserved for use in handshaking, or flow control.

To send a byte, the transmit line is raised to signify a start bit, then each bit is sent in turn: high for one, low for zero. After all seven or eight bits have been sent, the transmit line is lowered to signify either one or two stop bits. Some devices prefer that there be a parity bit as well, but almost all serial devices communicate with N81 (no parity bit, 8 bit words, and 1 stop bit). Flow control is used to signal when the device is ready to receive new data, or to request permission to transmit. Confusion of the implementation of cables and connectors makes life complicated for the end user, since mapping between the different approach on non-standard equipment can be difficult.

Another problem with RS-232 is that there is no standard way for devices to agree on a data rate. Some devices analyze the first byte, decoding it to derive the data rate. Simple devices might support a single data rate. Their output looks like garbage if the receiver is not set appropriately and the receiver can try different rates in an attempt to match the device. If a protocol for negotiating line parameters were specified in the original standard, RS-232 might not be such a problem.

Another problem is that, there is no safe way to automatically probe for the presence of a device. Manual configuration is required. Even if you know that a piece of hardware is attached, it is dangerous to check all serial ports looking for it. There might be a device on one of the ports that self-destructs when it receives a bad message. Without a specification fro probing, there is no way to protect such a device!

The RS-232 ports in modern PC are still implemented as they were on the ISA bus. This makes them susceptible to all the problems associated with configuring ISA devices. Add a plug-in ISA card, and you could easily disable your serial ports by introducing an IRQ or memory mapping conflict. The PCI bus has the necessary features of probing and auto configuration, but PC architecture systems don't take advantage of this to improve the implementation of RS-232 serial ports.

The RS-232 specification requires (comparatively) high voltages for signaling. An output of +15 volts sends one and -15 volts sends a zero. For many years, 5V logic was the universal standard. In the last decade, digital systems have reduced the logic voltage to allow higher clock speeds and increase efficiency. There are several RS-232 driver ICs available which use a voltage multiplier to generate the necessary levels from a single 5V supply. The MAX233 series is particularly popular as it has integrated the capacitors into the IC package. Building a

2.1.3. RS-422 and RS-485. As the problems with RS-232 became more evident, superior standards were developed which made use of differential signaling. RS-422 requires only a single 5 volt

power supply. Extended functionality was added for multiple transmitters, creating RS-485. By driving a differential pair, these protocols are able to run faster, using less power, and operating over a longer stretch of wire. Electrical isolation is crucial for all scalable networks.

Serial protocols like RS232 require that all devices share a common ground. When many devices must connect to another, it can be a real nightmare debuggin the system. Many serial and parallel busses do not need to scale beyond a few local systems and thus do never have to worry about isolation. A network differs from a bus in that it must explicitly deal with issues such as isolation that inherently limit scalability.

Systems connected on a parallel bus often share a common power supply, making the grounding issue nearly moot. This is particularly true in the PC architecture where cards and drives inside the case share a single switching power supply. Care must be taken when designing external SCSI and IDE hardware to insure that conflicting grounds will not be an issue.

Fiber optics are particularly attractive because they provide a complete solution to the issues of electrical isolation and interference. Even lightning will not have an adverse effect on an optical fiber since there is no conductor. Early generations of fiber required power for amplification stages, but advances in amplification through Erbium doped fibers allow for an all-optical pumping mechanism. By not requiring a conductor in the signal, fiber is an extremely safe mechanism for distributing signals across very large distances.

2.1.4. Serial Networks. Formerly called Appletalk, the LocalTalk protocol is based on RS-422 with transformer coupling allowing 32 nodes but restricting distance to 1000 feet. LocalTalk has some of the benefits of Ethernet. Devices listen before sending to avoid most collisions. However, LocalTalk does not implement hardware collision detection. Instead, higher level algorithms are responsible for retransmission and acknowledgement. Localtalk has a maximum data rate of 230kbps. Because of speed and because it requires more processing in the host computer for retransmission, LocalTalk has been almost completely replaced by Ethernet. Shiva developed and sold a LocalTalk bridge to allow the two networks to coexist. Support for LocalTalk protocol over Ethernet is still provided by default in Linux, allowing legacy networks to take advantage of modern server features.

2.1.5. Other Serial Protocols. I^2C is a protocol for networking devices with very few wires. It is supported can be supported in software on some microcontrollers like the PIC series from Microchip.

A newcomer to the serial world is the OneWire protocol. Dallas semiconductor has been having a great deal of success with their One Wire technology used in the iButton and Java Ring. Power and signal are communicated on the same two wire bus, allowing the unit to be packaged in a robust

Protocol	bits/sec	Distance	drivers	receivers
RS-232C/D	20K	15m	1	1
RS-423A	100K	1200m	1	10
RS-422A	10M	1200m	1	10
RS-485	10M	1200m	32	32
LocalTalk(AppleTalk)	230K	350m	32	32
USB	12M	5m	1 (128)	1 (128)
FireWire, IEEE 1394	200M,400M,800M		1	1

TABLE 1. A comparison of serial communications standards

metal can with no leads. So far, Dallas is the only source for devices implementing their protocol, but there might be a broad base of applications for this kind of communication bus in the future.

USB and FireWire have recently been introduced.

2.2. A High Speed Bus

2.2.1. Parallel Buses. When the PC architecture started to rise, the ISA bus achieved a nearly complete dominance. While ISA has many limitations, a simple implementation is relatively straightforward. Many vendors developed ISA cards to extend and enhance IBM compatible PCs.

Several critical features of a modern bus are missing the ISA bus. There is no specification for autoprobing, so a computer cannot tell which cards have been inserted except in a haphazard way. Also in ISA there are no per-slot signal lines. All lines are shared. This leads to great confusion for resources that can only be used by one cared at a time, specifically interrupt request lines (IRQs). These deficiency have resulted in millions of headaches around the world. Each time someone adds a device to an ISA bus, they must manually identify and configure such that it doesn't conflict with their existing hardware.

One advantages of the ISA bus is its simplicity. Because it has few features, chips designed for an ISA bus have a relatively straightforward electrical interface. This allows a general purpose microcontroller to emulate the signal lines of the bus in a technique known as 'bit-banging'. The signal lines are explicitly pulled high and low with instructions. While it may be slower than a single cycle bus, it eliminates the need for glue logic and reduces the required pins for interfacing to an IC. This dirty trick lets the filament implementation use only 8 data lines, 5 address lines and a handful of control lines, where normally we would need about twice as many pins. Conserving pins on the microcontroller can be a very useful way to sacrifice speed to reduce cost.

2.3. Networks

There are many networking technologies for embedded networking. Proprietary systems tend to be frozen and do not allow improvements to be integrated while open systems adapt and grow with

Product	Description
Neuron IC	Integrated circuit fabricated by Motorola
LonWorks Node	Neuron IC on a board with isolation transformer
Developer's Workbench	Package containing NodeBuilder and LonBuilder
NodeBuilder	Development software for nodes
LonBuilder	Installation software for networks

TABLE 2. Pieces of the LonWorks embedded network from Echelon

market needs and new applications. The proprietary systems that are popular today have a number of striking limitations that are unlikely to be rectified. These systems have typically been frozen to conserve developer resources, and they remain so in order to inspire consumer confidence. Open technologies may be adapted by anyone to his or her particular needs. Although this may require a larger initial investment, over time the contributions of multiple vendors can be amortized and make substantially greater technology than any single vendor could provide.

Using a proprietary network for device control can be a good idea for simple projects or projects where time-to-market and industry partnering are more important than the long term state of the project itself. The real disadvantage from proprietary networks comes when you try to connect multiple networks together.

Few vendors consider the their network to be anything more than a terminal network. This means that in order to connect multiple proprietary networks together, you must devise your own internetwork solution, develop a system that understands the addressing and transport issues of both proprietary networks, and deploy both networks in addition to the standard network linking the two together.

2.3.1. LonWorks from Echelon. The LonWorks product line from Echelon Corporation[1] may seem like an ideal platform for embedded networking. The nodes are inexpensive, a complete development suite is provided at a fixed cost. Bridges are available fro interfacing their network to other networks. Software tools are provided for installation and management of nodes.

The Neuron IC is the core of the LonWorks network. Each node contains a neuron chip and an isolation transformer or other physical interface. Neuron chips are available from Motorola for approximately $6.30 in 1000 piece quantities. Echelon sells Nodes which contain isolation transformers and support components for approximately $65 each.

[1] Echelon Corporation
4015 Miranda Ave
Palo Alto, CA 94304
+1-650-855-7400 / 800-258-4LON +1-650-856-6153(fax)
LonWorks@echelon.com
http://www.echelon.com

Echelon sells a development kit called the Developer's Workbench. There are two components to the LonWorks solution: NodeBuilder and LonBuilder. The NodeBuilder tool allows the development of software to run on isolated nodes.

In 1996, I was involved in a development project at the Charles Stark Draper Laboratories. We were building a modular robot with three wheeled segments. In wanted to keep they systems flexible and re-usable, we decided to use a distributed architecture to control various subsystems within the robot. We purchased a complete development kit for approximately $20,000 and attempted to develop our control system using Neuron nodes.

The NodeBuilder software worked quite well. Applications are written in a version of C with extensions to support the specialized LonWorks environment. Programmers familiar with C with have not trouble working with the hardware. Neuron ICs operate at speeds up to 20MHz, 2K of SRAM, 2K of PROM and 10K of ROM for the LonWorks system software. Because the developer has to fit his code into only 2K, the LonWorks nodes can only hold very simple programs.

While the NodeBuilder software worked quite well, every network also needs a configuration step for installation. LonBuilder is a DOS program that is required for performing this step. This tool did not perform as expected and was not able to meet our needs. After designing nodes, the installation and management processes are done by separate tools. The installation tool was difficult to use and inflexible. It did not recognize the nodes on the network. The lack of scripting facilities meant that there was little hope of automating the process of installation.

To limit the potential for future problems, the nodes were used in isolation. An RS232 protocol was used for control and communications within the vehicle to guarantee flexability in the future. RS232 had the advantage that the project was no longer locked in to a LonWorks solution. We could easily upgrade to other microcontroller systems without being limited by the small code space and slow execution speed of the LonWorks nodes.

Based on an examination of LonWorks' most recent promotional material, it would appear that LonBuilder application has not been radically improved.

2.4. Wireless Communications

2.4.1. Radio Networks. Aloha was one of the first packet radio systems. It was developed for communication between the islands at the University of Honolulu. This radio network was an effective testbed for implementing algorithms for communicating over a shared channel. A lot of early packet collision algorithms were developed in the context of the Aloha radio network.

Packet radio is alive today in both amateur and commercial capacities. Rooftop Communications[2] is researching solutions for community-based radio Internet access. They define a system where a service organization within a community would provide a public radio Internet gateway

[2]http://www.rooftop.com

called an AirHead. Most activity centers around AX.25. This mechanism for packet radio is supported natively in the Linux operating system.

There are numerous amateur radio satelites in operation. Apparently there were several occasions where a dummy load was needed for testing in launches involving "not quite mature" rocket technology. The Radio Amateur Satellite Corporation [3] Amateurs stepped up to the plate with fully functional packet radio transceivers and were able to hitch a ride to orbit without the usual $10,000 per pound expense.

Amateur radio operators have been building packet radio networks for many years. DARPA established a packet radio research program in 1972. Since then, researchers and government contractors worked to develop a spread-spectrum communications system that was robust, rapidly deployable, and adaptive.

Progress is being made in several areas. Cellular networks are adding limited packet data to their services. Pager broadcast networks are evolving into two-way paging services where a receiver can acknowledge the page and even send a short reply.

Ricochet wireless service is available from Metricom in a few cities on the west cost. It offers continuous access to the IP network at approximately 19.2kbps.

2.5. Ethernet

Bob Metcalfe designed the Ethernet protocol and proved its stability as his Ph.D. research at M.I.T. He later joined Xerox park where the system was implemented to connect computers together. Like other innovations at Xerox PARC, the technology was not commercialized internally. Metcalfe left and founded 3COM to build saleable components.

As Ethernet became a de-facto standard, efforts were made to establish an official standard through the IEEE. When people refer to Ethernet today, they are usually referring to IEEE 802.3[?]. In this document, the word *Ethernet* refers to the IEEE standard rather than the original ad-hoc standard.

Over a third of all small businesses in the US use Ethernet internally to share computers and printers. Many vendors provide products which conform to the IEEE standard, making it possible for devices to compete solely on the basis of price and reliability. Because competition has driven prices down, Ethernet has become a commodity solution for networking.

2.5.1. Auto-configuration. Ethernet is particularly effective because it allows many computers to talk to each other without prior configuration. This is a key capacity not built into all networks. For example, when you attach two modern phones to the same piece of phone wire, they can't call each other. The phones aren't capable of broadcasting information. In the early days of telephony,

[3] http://www.amsat.org/

phones had a hand crank and by turning the crank, energy was broadcast onto the cable, ringing a bell at the other end.

Having phones that can call each other is very important when you only have two phones and no telephone company. Such is the situation with most LANs. We pay the phone company to configure our phones for us, but we configure the computers ourselves.

2.5.2. Components. The two main pieces of Ethernet infrastructure are hubs and controllers. The controller, sometimes called a media access controller (MAC) is usually put on a network interface card (NIC) and installed in the expansion bus of a computer. ISA and PCI are the common bus formats today. The ISA bus is being phased out, but with the complexity of PCI, it seems unlikely that hardware vendors will completely abandon the ISA bus anytime soon.

Two topologies are available for deploying Ethernet: 10Base2 and 10BaseT. Originally Ethernet was implemented as a series of taps into a single coaxial cable that snaked throughout the organization. The current implementations of this is known as 10Base2. A transceiver converts the signals from the tap and feeds them into an AUI cable. The maximum length of the AUI cable determines how far machines can be from the cable. To prevent echos on the wire from interfering, the coaxial cable must be terminated at both ends. If any part of the cable is damaged or a break occurs, the entire network goes down and there is no ready indicator of where to look for the failure. A special class of devices were developed for diagnosing such network problems. By sending a pulse down the cable and looking at the echo, a specially designed time domain reflectometer (TDR) can show the distance to line shorts or breaks.

Since maintenance is a big expense in most organizations, a much preferred way of linking systems is to use 10BaseT, which relies on Ethernet hubs and Category 5 unshielded twisted pair (UTP) wiring. This topology is implemented like a star or hierarchical network. Every device must connect to a port on a hub. Hubs can connect to other hubs as well, making the traffic flow across the entire network.

If an organization needs more collective bandwith than a single Ethernet, the computers are split up onto separate Ethernet networks, which are connected together with either an Ethernet switch or a router. Switches only need to understand Ethernet protocol, which makes them simple and easy to deploy. Routers move packets around based on higher level protocols and must be configured appropriately. Depending on the complexity of your network, setting up a router can be non-trivial.

Strong competition between suppliers has driven prices down to $10 per port for hubs and $20 each for low-end network cards. Networking software is built into all major operating systems allowing for easy setup and deployment. Midrange office computers cost on the order of $1500, which means that the network only adds another 2% (not including the cable).

After Ethernet became the de facto standard for computer networks, the IEEE worked to formalize the specification and create an official standard. The IEEE committee was pulled in several

directions by the existing users of Ethernet, including General Motors, and IBM. GM was unsatisfied with the probabilistic nature of Ethernet and wanted a token bus architecture where each node would speak in turn.

LAN type	IEEE Standard	Backing Companies	Purpose
Ethernet	802.3	DEC, Xerox, Intel	office automation
Token Ring	802.4	IBM	serviceability
Token Bus	802.5	GM	factory automation

GM and its suppliers needed a protocol for factory automation. The the MAP (Manufacturing Automation Protocol) was developed to run on the token bus and was quickly adopted by manufacturers.

In its original implementation, Ethernet used coaxial cable, but most installations use UTP (unshielded twisted pair). There are two performance ratings for UTP cable, Category 3 and Category 5. T

Media	Cable	Configuration
10Base2	Coaxial Cable	multi-point,
10BaseT	UTP (cat 3 or cat 5)	point-to-point
10BaseFX	Fiberoptic cable	point-to-point
100BaseT	UTP (cat 5)	point-to-point

2.5.3. Device Networking with TCP/IP. Several companies have products for attaching devices to the Ethernet. Lantronix[4] makes a remote access device for RS232 over TCP/IP called the MSS1-T Micro Serial Server.[5] This component could be used for some of the same applications as filaments, however the device is currently sold for $499 USD. While this device is physically small, measuring only 5.75 x 4.25 x 0.8 inches, the Micro Serial Server does not meet the needs of ubiquitous networking since its cost rivals that of a stripped-down PC.

Sun Microsystems has recently released the specification for Jini, a protocol designed for networking all types of devices from televisions to cellular phones. The main features of this protocol are autoconfiguration, remote access, remote administration, and mobile code. The goal is ambitious and has the design strengths of such industry legends as Bill Joy, author of vi, and one of the original developers of UNIX.

Protocols like BOOTP and DHCP may automate the configuration process, but they do not completely fix the problem. IP Addresses must still be allocated from a central authority in the

[4] http://www.lantronix.com

[5] http://www.lantronix.com/htmfiles/prodinfo/datashts/mss1-tz_id.ltx

Protocol	Use
Ethernet	Packet deliver, broadcasting, & CRC integrity
SNAP	(optional) Can be used to identify custom protocols not specified for Ethernet
ARP	Translates an IP address into a MAC address
IP	Routing outside the local network
ICMP	Error reporting when problems occur
UDP	directs packets to the correct application
TCP	directs packets to the correct application, adds connections, reliability, timeouts

TABLE 3. Protocol Layers

Language	Overview
C	Incompatible APIs on different platforms, will be cross platform with extra work
C++	Same problems as C with greater complexity
Java	Binaries work on many platforms but lack robustness
VisualBasic	Proprietary, Only available on Microsoft platforms
Scheme / Lisp	Many find it challenging to think recursively, few networking tools available
Perl	Widespread use, and supported on many platforms
Python	User base is small, but extensive functionality is available

TABLE 4. Languages for implementing network features

installation process. Ethernet MAC addresses are assigned by the OEM (original equipment manufacturer) and are guaranteed to be unique without any action from the installer.

centered around putting every device directly on the internet. Device networks that rely on IP addresses and the Internet Protocol still require a fair amount of manual configuration. Each device must be assigned a unique IP address from a limited supply.

2.5.4. Host Implementation Languages. Languages and APIs are desperately seeking compatibility. C suffers from incompatible APIs available on different platforms. Cross platform toolkits are available. With extra work and attention to detail, applications can be written to compile on for multiple operating systems

Software tools were developed to make it easy to configure and test the filament chips. A cross-platform strategy was necessary due to the variety of machines in use within the lab. Within the M.I.T. Media lab: Machines are used as illustrated in the following table.

Concept	Status
Berkeley sockets	Unix: C, Perl, Python, Scheme
Winsock & Winsock II	Microsoft's Windows API essentially Berkeley sockets with extensions
Java.net	(UDP, TCP) consistent across platforms
Java.RMI	migrating toward CORBA compatibility
CORBA	universal object model for distributed systems

TABLE 5. Network APIs

Task	Operating systems
Word Processing	WindowsNT, Windows95
Document Processing (LaTeX)	Linux, HPUX, Solaris (Athena)
Software Development	WindowsNT, Windows95
Matlab Computation	HPUX, Digital Unix, WindowsNT
File Serving	Digital Unix, WindowsNT
Demonstration Tasks	Windows95, WindowsNT
Embedded Development	Windows95
3D-Modeling	WindowsNT
Administrative Tasks	MacOS

TABLE 6. Tasks performed on various operating systems in the lab

Network API	Java	C (requires porting)	Perl	Python	VisualBasic
Windows 95/NT	x	x	x	x	x
Linux	x	x	x	x	
HPUX	x	x	x		
Solaris	x	x	x	x	
Digital Unix	x	x	x	x	
MacOS	x	x			

TABLE 7. Availability of Network APIs on Various Platforms

Within the Physics And Media group, several operating systems are in use for different tasks. Each system is needed for a particular attribute. Unix machines are used for file-serving and document processing.

:

In other groups within the building SGI and Sun workstations are used. A large number of Apple computers are used for administrative work. The main file servers for the lab are DEC Alpha workstations.

Given the availability of hardware and the use of machines within the lab, Java was selected as the platform of choice for host software development. The availability of the solution makes it practical to develop once and reuse on multiple architectures without hassle.

CHAPTER 3

Filament Implementation

> Everything should be made as simple as possible, but not simpler.
>
> – Albert Einstein

The ideal scenario for building "things that think" is to have network access already built in to the parts. Filaments combine a microcontroller with a network interface in one simple package. With a filament-based device, the only external pins are for power and network.

There are hundreds of viable microcontrollers and dozens of different network standards. In the previous chapters we narrowed the network down to what is already installed: Ethernet.

All Ethernet controllers were designed for network interface cards. These ICs interface directly with the system bus. Ethernet controllers are available for ISA, PCI, and PCMCIA[1] busses.

Nearly all Fast Ethernet controllers use the PCI bus. That makes sense because ISA and PCMCIA can't keep up with the 100Mbps data rate. Unfortunately, building a PCI interface is a huge endeavor. Current high-end processors like the Pentium and Pentium-II require an external bridge to connect to the PCI bus. For our inexpensive filaments, we cannot afford PCI. National Semiconductor is producing the only known exception. 100Mbit Ethernet controller.

3.1. NE2000

Early Ethernet implementations had a fair number of components. Chip companies started to integrate the functions into ICs to sell chips at higher margins. National Semiconductor[2] produced one of the first of such chips, the 8390 Media Access Controller. This was very useful because they also released a reference implementation. Many vendors built cards based on their reference board, including Novell. The Novell NE1000 (8bit) and later NE2000 (16 bit) cards had the most software support. Novell's board was nearly identical to the reference design, making it easy to copy.

The cheapest network interface cards are based around the NE2000 architecture. Price competition between software-compatible cards drove Asian chip makers to produce them in large quantities

[1] PCMCIA is sometimes known as PC Card, but that is too confusing

[2] National Semiconductor may be reached on the web: http://www.national.com
Documentation on their line of Ethernet products is currently available at the following URL:
http://www.national.com/catalog/LocalAreaNetworksLANs.html

for incredibly low prices. Unfortunately the NE2000 architecture has some flaws which make these cards less desireable for use as a PC network interface card.

NE2000 cards do not support DMA[3] and must be accessed through polling the IO ports. DMA access is considerably faster and less CPU intensive than explicitly reading and writing to the IO ports. In our application, IO ports are just fine. Our simple microcontroller would be unable to take advantage of any DMA features.

WinBond is one vendor that has ceased making ISA based Ethernet chips. They adapted their design for PCI and have not looked back. It is important to consider lifetime availability when choosing an IC supplier. Fortunately, there are others who also have NE2000 devices.

UMC is one of the largest IC fabriction companies, and used to also sell components produced in their spare cycles. To make that process more efficient they spun off a separate company to hold the intellectual property of those parts. The company is called Davicom and was established in 1996. The part we are interested in is the DM9008, formerly the UMC9008. Being independent from UMC, Davicom is free to use cycles at any fab, and does not limit the success of their parts to the idleness of UMC's fabs. Given their special relationship within the IC manufacturing industry, it should not be surprising that Davicom can deliver pricing unavailable from other vendors. While ISA compatible Ethernet controllers are available from AMD, Intel, Fujitsu, SMC, Crystal Semiconductor, and Lucent Technologies, none of them will sell the components for less than $9/per unit, even in quantity. Davicom shipped

3.2. Microcontroller

The following diagram illustrates the minial hardware architecture of the filament. The goal is to use as few building blocks as possible, keeping the system small and simple.

RS232 is the most common interface used in the Physics and Media Group. Asynchronous serial data can be unreliable at high speeds but the protocol is easy to learn and simple to implement. Desktop and laptop PCs have built-in serial ports that are used for debugging. Since people can learn to use RS232 in a matter of hours it will continue to be a standard protocol in the lab.

RS232 is especially useful when using low-end microcontrollers. By calculating the appropriate software delays, outputs can be explicitly controlled and inputs can be explicitly polled in a process known as bit-banging. No external components are required (except for voltage conversion) for even the simplest microcontrollers. More advanced processors have an integrated UART to offload the task of serial processing.

In the filament implementation, the RS232 interface is handled by an internal UART while the interface to the Ethernet controller is implemented explicitly (bit-banged). The control lines are

[3]Direct Memory Access. With DMA support, a driver can move blocks of data to and from a device without having to execute separtate instructions for each word.

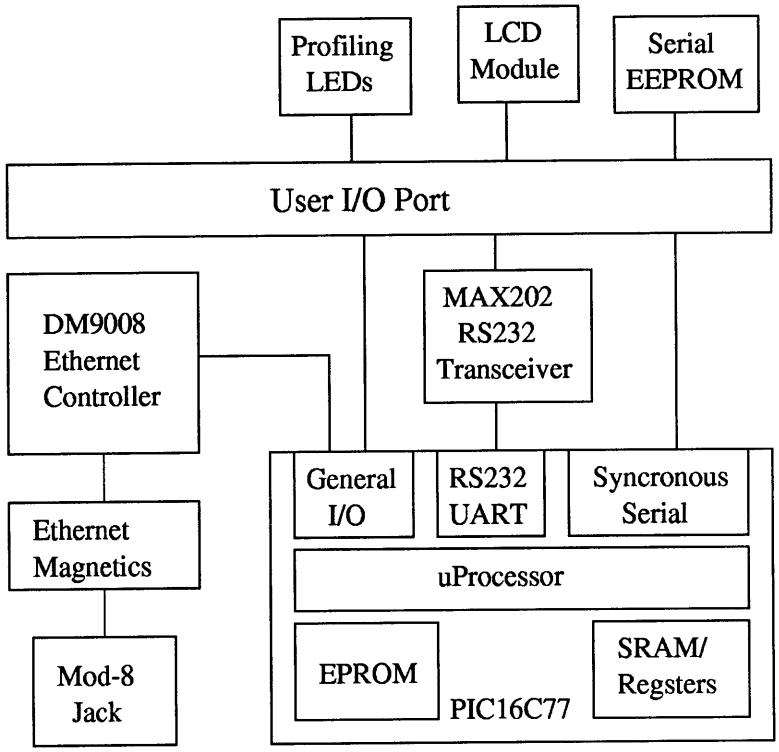


FIGURE 3.2.1. Filament Hardware Architecture

hardwired to the voltages required by the power-stealing circuitry in a Logitek or MouseSystems mouse.

The first versions were designed to use either the PIC16F84 or the PIC16C77. The PIC16F84 is available as an electrically erasable part and was selected to accelerate early development. Because the '84 does not have enough IO lines to control the DM9008, an extra component was needed to latch the address lines, adding extra steps to the bus cycle and cluttering the design. When the software began to push the limits of the PIC16F84, that form factor was abandoned in favor of the PIC16C77 which has enough spare IO lines without needing the additional latch.

The ISA bus interface to the DM9008 is controlled by explicitly raising and lowering the control lines. Not all of the signalling lines in the ISA bus specification are relevant. A quick way to narrow down which ones are really needed is to look closely at actual ethernet controllers built out of the DM9008. For signalling we will need IOR, IOW, ISA_RESET, and IOCHRDY. A significant pin-count optimization can be obtained from the fact that we will never have to address more than one device on our bus. In fact, we only need to address the narrow range of addresses to which the card is normally reset.

The IOCHRDY line has an interesting function in the ISA bus. It is essentially a handbrake for the bus. ISA devices have the option of pulling this line in the middle of a bus transaction if they cannot keep up. IOCHRDY halts the ISA bus until the device in question is finished and releases the line. Most good ISA cards can keep up with normal goings on. Other Ethernet controllers including the components from Fujitsu and Crystal Semiconductor never pull IOCHRDY. Unfortunately the DM9008 isn't quite so advanced, requiring us to monitor this pin as part of our normal bus transaction cycle. The BALE line may be left active since we will always want to latch on an IOR or IOW clock.

Speaking of clocks, you might have noticed how theassemblingre is no clock on our ISA bus. This is not a bug! The unpredictable nature of ISA slots forces manufacturers to develop cards that work in static operation with IOR and IOW as the only reliable signal for latching. Ethernet controllers need an internal clock for implementing the wire protocol and we steal this signal for reuse as the clock for our microcontroller.

CHAPTER 4

Filament Software Interface

The software on the filaments is intended to run very close to the wire, having a very simple protocol for communications. For most applications, a user should not need to write their own routines for assembling packets, the interface in the filament administrator should be sufficient.

A simple protocol is here defined for loading code and controlling filament behavior. The normal BOOTP protocol was inadequate for several reasons. First, in the majority of cases, the filaments will not be assigned a unique IP address. Addressing beyond the local subnet is handled by a filament administrator, which can be configured to make the filaments globally available, or to make the access or for local access only.

The default action for filament chips is to wake up in an orphan state. Being helpless on its own, the filament broadcasts a plea for help, crying out for attention.

The equivalent behavior in the IP protocol is for an unconfigured device to broadcast an ARP request for itself, hoping that some device on the network will assign a valid IP address. DHCP and BOOTP are two protocols that handle IP assignment in slightly different ways. DHCP is the currently accepted way of doing this.

This section has three purposes. Its first purpose is to provide a quick overview so that new users may start talking to filament chips right away. Its second purpose is to provide an in-depth discussion of routing packets. Its third purpose is to show people how to write advanced applications that may use the filament chip across networks, so that they may write their own filament chips which are compatible with the models displayed here.

4.1. Ports and Portability

When engineers design hardware that requires computer control, they usually pick an operating system and and only develop one set of control software. After the project is finished it is often very difficult to port the software to other platforms. It is generally agreed that device drivers are difficult to write. As a result of this, few vendors port their drivers to more than one architecture. With only one operating system targeted per device driver, a user must juggle several operating systems if he or she wants to use customized hardware from multiple vendors.

Using filaments as the interface to custom hardware makes it easier to develop cross-platform software for interfacing with the rest of the world. All computers deal with the Internet Protocol (IP) in relatively the same way. The software can speak at the IP level to a filament administrator which translates in to the simpler protocol of filaments. This gives the engineer the ability to design a system once and run it on multiple platforms without redesign.

Several cross platform tools are available. The Java run-time environment has been ported to many architectures. Perl and Tcl can be used to build prototype applications quickly. If the author is careful about designing for portability, the application can be written in C and compiled for multiple architectures. In general, designing for portability means not doing anything that is implemented differently on different architectures, like trying to access the io ports.

Without filaments it is difficult to extend a desktop's hardware interface beyond the built in serial and parallel ports. For over a year, all new PC motherboards have shipped with a USB interface which theoretically makes it possible to connect up to 128 devices to a single PC. The limitations of USB result from a short maximum cable length (5meters) and the lack of operating system support. Writing a USB device driver that coexists with other drivers is still a risky thing. If one had access to the internals of the operating system, one could guarantee the eventual success of such a development project. Since the majority of operating systems in use today do not provide such access, the developer would be completely dependent on the whims of the OS vendor. In many cases that is an untenable risk.

This situation can change. Microsoft is slowly migrating users to the device driver model of Windows NT. Apple has recently introduced the iMac which depends on USB for basic peripherals like mouse and keyboard. Many systems are still not compatible with other operating systems. In the electronics design world, it is difficult to find reasonably priced development tools that work on any platform other than Widows 95. With filaments, a vendor can build an Ethernet interface into their product instead of a serial, parallel, or USB interface. Ethernet is well tested and well understood from a device driver perspective. Building off of the network insures that your IO port won't disappear from the marketplace.

The modern operating system grew out of the frustration of managing multiple programs that each had their own collection of device drivers. Device drivers have been causing engineers to pull out their hair in frustration ever since the very first computer was created. Hardware is not usable until the device drivers exist. The operating system is the center of control for a computer. Device drivers add new behaviors to this base functionality. By providing a common user interface and including a wide range of high-level application programming interfaces (APIs), operating systems have grown in size and functionality.

4.2. Protocol

A person may wonder why he or she should use TCP sockets when the local Filaments use UDP? When a chain of unreliable retransmissions are required in order to deliver a single message, the probability of delivery can shrink substantially. While TCP allows connections to be linked reliably end-to-end, each station that hands off a UDP packet introduces a non-zero chance for failure.

There are three software components defined in this architecture: filaments, the filament administrator, and client applications. The filament administrator relays data between local filaments and client applications. Filaments have a very simple interface and only have enough on-board intelligence to keep from self-destructing.

Client applications typically involve a stand-alone interface for controlling or communicating with the filaments. Another form of client is a middleware application that interfaces between the physical world of filaments and the virtual world of the database. Data is logged, spreadsheets are updated, and other processing is performed.

With filaments linking into middleware, a corporation can have high volumes of sensory information flowing through it. A body has nerves for sensing pain, temperature and motion,. Filaments allow a business to know the physical goings-on of its self, regulating everything form the physical parameters of temperature and humidity, down sensing and responding the the level of stress pheremones in the air. If it is inexpensive to deploy such sensors, the benefits can certainly outweigh the costs.

The filament protocol is designed for maximum compatibility across different hardware and software environments. Using a special protocol, filament are able to communicate with software at the IP level. Even though filaments do not have IP addresses themselves, messages can be sent and received from cross-platform implementations like Java hierarchically relying on IP broadcast.

4.3. Network Naming & Routing

Naming and Routing are two problems that plague every network. An internetworking protocol must solve these completely in order to scale beyond a limited size. The current Internet solves both issues with a centralized authority. Routing tables are updated in the root servers and percolated down to the last router in a tree. Naming is handled in two stages, first a hierarchical numbering system is used to assign IP addresses, second a lookup table for more descriptive host names is added on top of this using the domain name service or DNS protocol. Each country is alloted a root domain and the authority to grant domain names with their country extension. The only exception to that is in the US where Network Solutions has been selling ".com" and ".org"domains for $100 for 2 years, now reduced to $70 for 2 years.

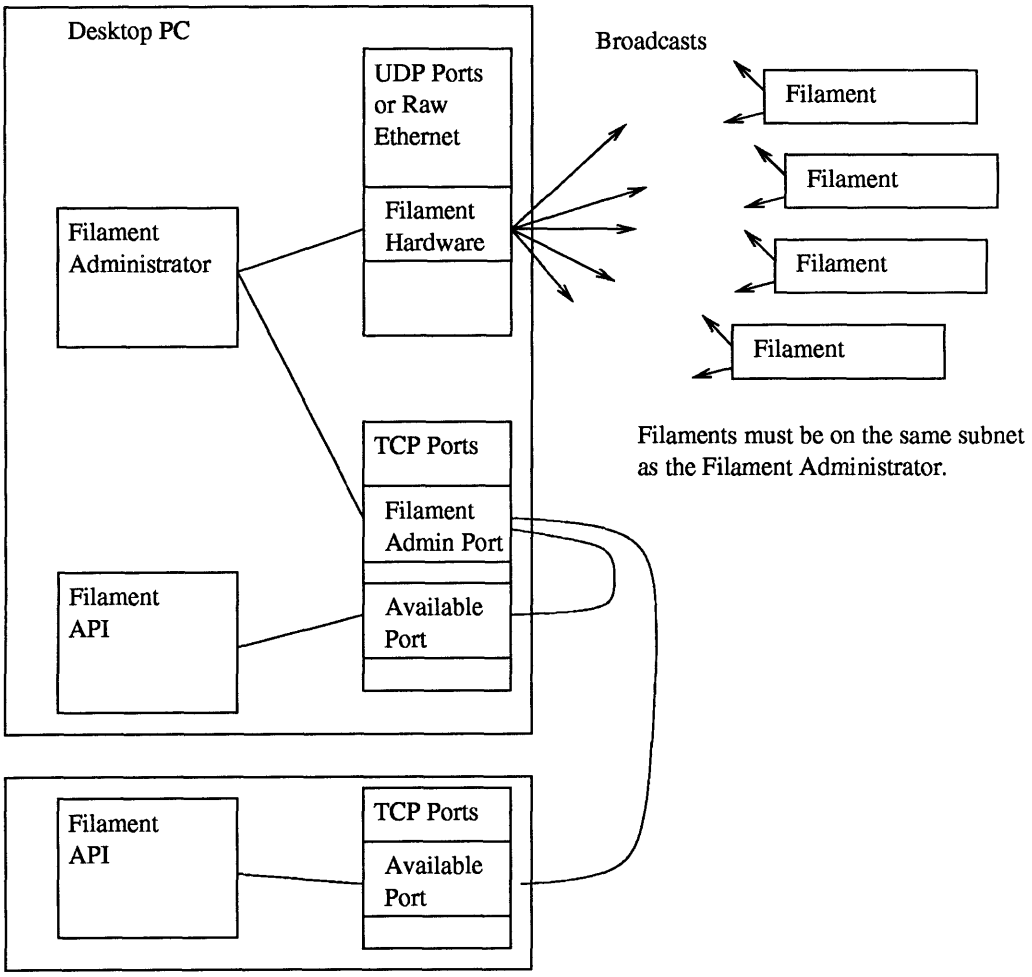


FIGURE 4.2.1. Protocol for communicating with Filaments

Without assigning a unique address to each host, one can't route packets at all. There must be a way to distinguish one host from another. With a flat address model, one must know how to route to every single address. This model works fine for small networks, but would require far too much information to quickly route packets on the scale of the Internet.

The mechanism for assigning Ethernet MAC addresses is for each vendor to obtain a range of addresses from the IEEE. For a $1000 fee, the IEEE will assign a three byte MAC prefix. The 802.3 MAC address consists of six bytes. The last three bytes are up to the vendor and can be parcelled out in any way, providing unique numbers for over 16 million hosts. The IEEE places a restriction on getting additional address ranges, asking that the purchaser show records which will prove that

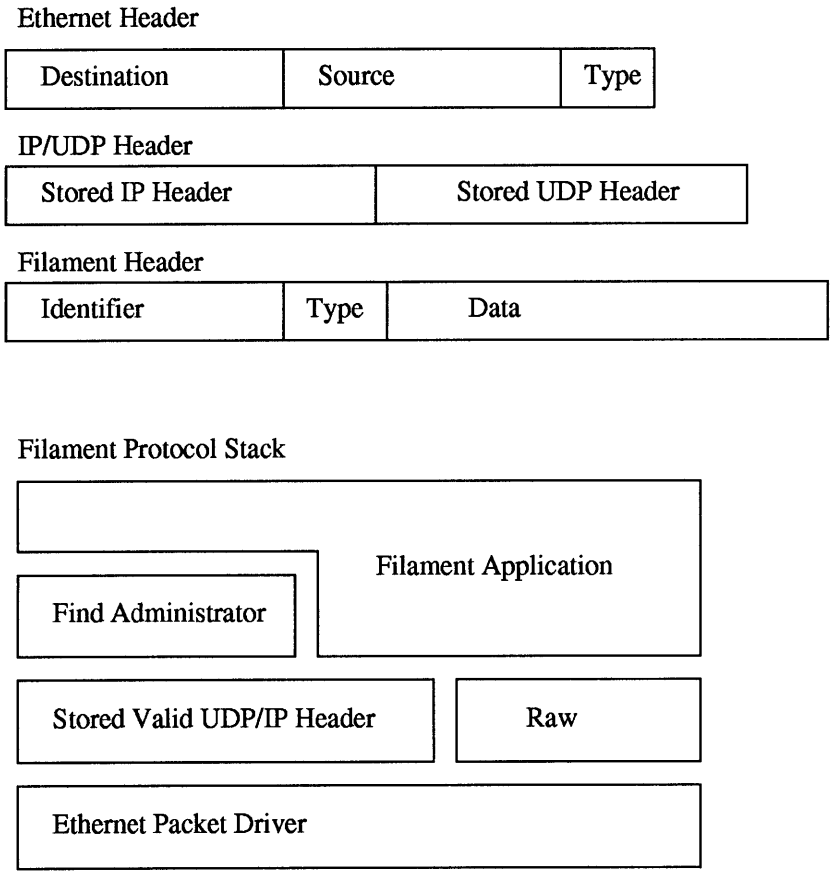


FIGURE 4.2.2. Components of the Filament Protocol

over 90% of the previously assigned addresses have been put into use. This keeps the address space from being wasted by poor management.

Even with those address-conservation mechanisms in place, the IEEE forsees the day when six byte addresses will run out. A newer 64-bit unique number range can be purchased for use with other technologies that require guaranteed uniqueness. The 64 bit standard is recommended for future technologies.

An Ethernet switch must keep track of every Ethernet MAC address and how it should be routed. This need to learn and remember information about every host limits the scale of a switched network. The advantage of a switch is that it is simple to deploy and doesn't require a change in the use of the network.

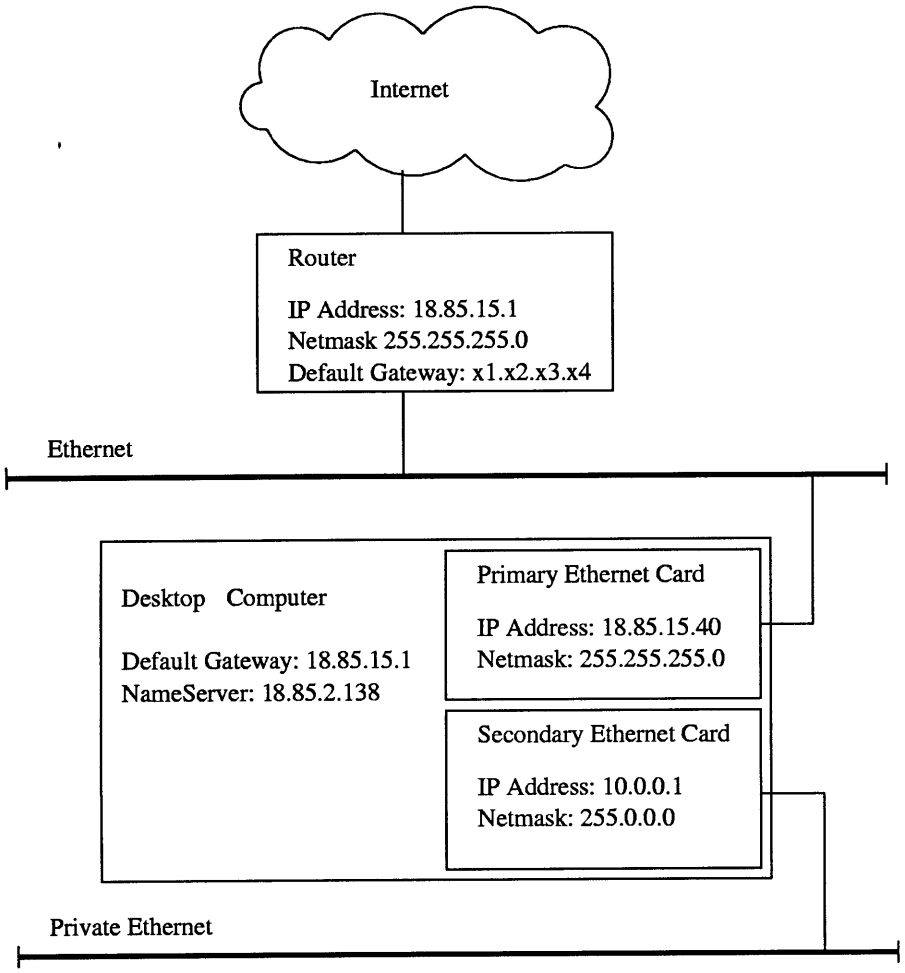


FIGURE 4.3.1. Private Subdomain

Once the number of hosts grows beyond the scale where a switch can know keep track of all the nodes, there need to be some way to simplify the problem. The Internet Protocol (IP) does this by assigning in a hierarchical manner. Small ISPs buy numbers from bigger ISP which obtain a numbers from the backbone providers. Over the years, this strict hierarchy has eroded, requiring routers to keep track of all the exceptions to the normal routing procedures. Internet Service Providers (ISPs) worry about these tables becoming unmanageably large. Routers must look up exceptions when forwarding packets to the correct locations, and more exceptions means more expensive routers.

There are a huge number of IP addresses available on a partitioned network. Devices can use the full range of addresses under a reserved Class A domain (10.xx.xx.xx). Commonly referred

to as NET-10. Cisco recommends the use of NET-10 and uses this technique internally. It is for common for subnets behind firewalls for to take advantage of this mechanism. The only problem is that NET-10 devices cannot be addressed directly from the outside world.

Network address translation (NAT) is a technique for mapping internal NET-10 IP addresses to specific ports on a machine with a valid Internet Protocol address. NAT can allow internal hosts to be accessed through a firewall as if they were directly available. A significant amount of planning, configuration, and management is required to maintain most implementations of NAT.

NAT is effective for allowing local computers to access proxied remote services. It is still difficult to use in a reciprocal way. Mappings must be specified explicitly to prevent conflicts

4.4. Security

Security of the network is very important when physical interfaces are directly available. For example, if your security system was wired and vulnerable, a competitor could literally lock you out of your own building. Real-worlseparated equipment should not be globally available by default.

Total security is achieved with filaments by partitioning the networks. It is important to understand that any other solution will at the very least result in the possibility of a denial of service attack. Partitioning the network is the only way to provide total security.

Since filaments do not have a flexible software implementation, a flaw in the security mechanisms of the filament protocol could not be easily patched and fixed. It is nearly impossible to guarantee that a single solution will stay secure with time. The filaments don't even try. They do not implement any real security. Applications that require security either operate in complete isolation, or are made accessible to the outside world through a filament administrator. The software on the filament administrator can be easily upgraded, making it possible to easily fix security problems as they are discovered. No other approach can be effective in the long run.

4.5. The Filament Administrator

The filament administrator has not been completely implemented, however a skeleton application was implemented for testing and development purposes. This utility allows the user to transmit a message to a range of filaments, and displays other filament traffic in a variety of formats.

The skeleton interface consists shown in the figure above consists of several components. A canvas (occupied by the splat) is used for graphical output. A selector just below the canvas selects the major operating mode.

A scrolling text field is used to print incoming messages and diagnostic information. Below the text field, a message area is available for composing new messages.

The *Broadcast* button causes the message to be transmitted on the specified IP broadcast address. Normally the IP broadcast address would be 255.255.255.255 but if there are multiple internet

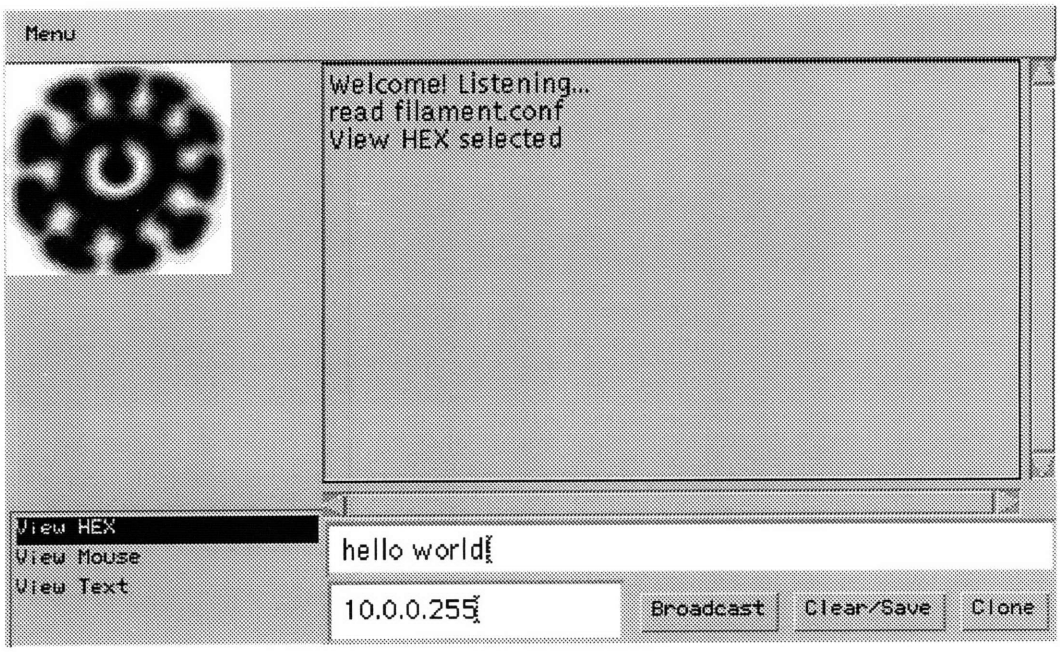


FIGURE 4.5.1. The Filament Administrator

interfaces, you need to be able to specify which one gets the broadcast. IP routers will not forward these broadcast packets so they stay within the local subnet.

The *Clear/Send* button clears the text field and stores the current settings. Right now the only settings are the broadcast destination address, but the underlying code is very flexable for use in loading and saving initialization files.

The *Clone* button simply clones the window, opening up a duplicate on the screen.

The three major operating modes are relatively straightforward. In *View HEX* and *View Text* mode, messages are correspondingly decoded into hexadecimal or shown as text strings. The *View Mouse* mode parses messages as a serial stream from a Logitek or Mouse Systems compatible

mouse[1]. A square is drawn in the canvas and is moved according to the movements of the mouse. For more than one Ethernet mouse on-line additional would be necessary to sort out the streams.

The Ethernet mouse was implemented as a demonstration of the basic functionality of a filament. It shows the spirit of lightweight network access finding simple ways to do complex things. The mouse steals power from the RS232 port and the filament board transmits packets as bytes are received. This code was originally written for the first generation filament based on th PIC16F84 (as pictured), but the application has not yet been completely migrated to the new filaments based on the PIC16C77.

[1]**Mousesystems protocol (quoted from the Linux manpage for "mouse")**
The Mousesystems protocol uses 1 start bit, 8 data bits, no parity and two stop bits at the speed of 1200 bits/sec. Data is sent to RxD in 5-byte packets. dx is sent as the sum of the two two's-complement values, dy is send as negated sum of the two two's-complement values. lb (mb, rb) are cleared when the left (middle, right) button is pressed:

```
byte d7 _d6_ _d5_ _d4_ _d3_ _d2_ _d1_ _d0_
__1_ 1_ __?_ __?_ __?_ __?_ _lb_ _mb_ _rb_
__2_ 0_ dxa6 dxa5 dxa4 dxa3 dxa2 dxa1 dxa0
__3_ 0_ dxb6 dxb5 dxb4 dxb3 dxb2 dxb1 dxb0
__4_ 0_ dya6 dya5 dya4 dya3 dya2 dya1 dya0
__5_ 0_ dyb6 dyb5 dyb4 dyb3 dyb2 dyb1 dyb0
```

CHAPTER 5

Future Directions

While this implementation of the filament has potential, it needs to cost much less to become a truly universal interface. The proposal for a filament chip is listed in the appendix as a way of addressing the shortcomings of cost, space, power, and complexity. Vendors such as Osicom are rapidly approaching the filament ideal, however there are still a few steps away from the simple, lightweight implementation inherenet in the filament architecture. Doubtless, people will continue to use existing bus technologies and will continue to depend on proprietary networks some applications. To penetrate the device market with a compelling solution will require winning vendors away from their existing commitments.

The network will gradually extend to cover more and more of the Earth's surface. People demand more information and allocate irrationally large resources for access of any kind. More places can be reached by through television than through roads or highways. As people are exposed to it they slowly realize how the Internet increases the quality of life. We now share the ability to organize and plan en-mass without a budget, needing only a single person, a single point of focus, and a single place on the web.

Networks built out of optical fiber have an electrically robust nature. They are immune to electrical interference. Fiber provides the ultimate reliability when the pipes are physically secure, but spread spectrum radio communications are extremely useful in a hostile environment.

Wireless technologies are still unreliable and unpredictable compared to the robust nature of fiber, but fixing a fiber split can take days, while radio interference can be routed around with a flexible network of transmitters. Advances in Wideband CDMA (code division multiple access) make it easier to share the same airwaves with other transmitters. This technology shows great promise for eventually making wires obsolete. As great as that potential is, it will take years to build up the kind of institutional acceptance that Ethernet has earned. We will eventually build up a complete industry of support tools and infrastructure management for the new wireless domain. Until then, Ethernet is our best alternative for getting the job done.

5.1. Wired Things, Unwired People

The difficulty involved in evaluating a reliable communications link makes it impractical to connect two devices that need a high data rate but have an unknown physical layout. When deploying a wireless LAN in a business, you may need to put computers in places where radio frequencies are blocked, making it difficult to predict the exact configuration necessary for a reliable network. Diagnosing network problems becomes more difficult with a wireless network. You need to account for signal strength, interference, and other parameters that can vary with time. For example, if people on other floors of your building deployed their own wireless network, it might throw off your calculations for maximum network throughput. One might have problems if the two systems fight eachother for bandwidth.

5.1.1. New Wired Networks. An initiative is underway with a consortium of vendors to develop a wire protocol for home networking. This protocol would use the existing already-installed telephone wiring without disrupting low-frequency voice signals on the line. Bob Metcalfe refers to this technology as a giga-node network since he estimates that more than a billion devices are currently attached to phone lines around the world.

5.1.2. New Wireless Networks. An industry consortium has been formed to develop a low-cost spread spectrum radio network for linking personal area devices like cell-phones and laptops. Bluetooth[1] is designed to implement major features like security and robust transmission in hardware, making it easy to integrate the small footprint silicon into existing products. With all the wires hanging out of the back of computers today, Bluetooth could go a long way towards reducing the cost of owning and maintaining a PC.

The system relies on a concept of private networks, called PicoNets, where devices can communicate and share resources in relative safety and security. Multiple PicoNets could coexist and overlap, acting as if devices in the network were wired explicitly.

This technology will fill-in a critical missing link in existing radio solutions. High and low bandwidth long-range systems have been available for a long time, but a robust way of implementing low-power local wireless systems has not been available in an inexpensive package.

5.2. Ubiquitous Networking Applications

5.2.1. Networked Tag Readers. Computers aren't very good at identifying thing just by looking at them. It would be a major engineering task to build a scanner that could identify loaf of bread at a scanner at a cash register. Barcodes are used in many places to attach a digital identity to objects. Unfortunately barcodes require a clear line-of-site and require the object to be oriented

[1] http://www.bluetooth.org

correctly. Radio frequency identity (RF ID) tags are one way to fix the problem. Unfortunately, silicon chips are unlikely to ever cost less than ten cents per chip.

An initiative at the M.I.T. Media Lab is working to develop ID tags that can be made less expensively than RF ID. Using materials with a natural ringing property, one can make It would be possible to hide a penny tag inside of every manufactured item. With a penny chip installed inside each device, a person could hold any piece of equipment up to a terminal's tag reader and the reader could tell them what they were holding, who made it, when it was made, and how to use it. The tag reader could pass along information about some of the common problems that other people have experienced with that particular device. It could provide a list of places to buy more devices, indicate whether the particular device is still under warrantee, relate the estimated resale value of that particular device, and provide a pointer to a newly developed alternative. The tiny chip that provided all of this information would cost only $.01 per device.

5.2.2. The Universal Network. There are many constraints on how an integrated circuit is designed such that it can be reliably made in volume. The most obvious requirement is that the IC must have an interface for automated testing. Clean rooms are incredibly pure but occasional particles will damage some ICs.

A configuration interface is required for sufficiently complicated chips. If some regions of a chip have faults, it is beneficial to be able to disable them by blowing on-chip fuses, rather than throwing away the entire part. As die sizes increase, the risk of bad chips increases rapidly. If a fuse can select an alternative path for a bad row or block, SRAMand DRAM can be enlarged with reduced risk.

Another possible use for such configuration is for parts to be branded with a unique identifier by blowing on-chip fuses.

Traditionally the boundary-scan interface and user interface have been physically separate. The pads for testing weren't necessarily broken out in the final plastic or ceramic chip package.The circuitry for boundary scanning was not expected to be used after the part was shipped to the customer.

The IEEE[2] has issued a standard for boundary-scan interfaces in 1990. Many IC manufacturers use the Joint Test Action Group (JTAG) specifications. The IEEE Std 1149.1-1990 specification defines a this high-speed synchronous serial protocol. An important part of this standard is the Boundary Scan Description Language (BDSL) IEEE Std 1149.1b

There are many types of synchronous serial buses for communications between chips on a circuit board for inter-chip communication. These serial buses are designed for extremely simple hardware implementations such as requiring a separate chip select line for each component, or having a low maximum data rate.

[2]A table mapping standards to the buzzwords we use is available at http://standards.ieee.org/faqs/buzzwords.html

JTAG signals	Function
JTDI	serial data in
JTDO	serial data out
JTMS	test mode select
JTCK	serial clock input

TABLE 1. The JTAG boundary-scan interface

The boundary scan is usually performed by specialized hardware, manufactured by Teredyne or Caydence. Fabrication is not guaranteed to be flawless since the tiniest particle or impurity can disrupt a transistor or break a circuit trace. All ICs must pass a rigorous boundary scan test before they may be shipped to a customer. The fabrication process is tightly controlled to maximize the yield of good chips.

A proposed application for filaments is to turn the boundary scan interface into a general purpose network. The circuitry for boundary scan needs to exist in every implementation, so why should it be wasted? Implement a general network interface instead an have a remotely maintainable component without added cost. For this application, the filament would be a VHDL description resulting in a square of gates near the network interface. Chip designers would use the filament as the default interface for all communications, adding additional high-speed busses or custom purpose I/O only where required. The filament port would be the essential interface for boundary scanning, burn-in testing, configuration, monitoring, and normal use. It would be a necessary component in the corner of every silicon die, and would inherently simplify the processes of implementing complex systems by moving all medium-bandwidth control functionality to a network interface rather than a collection of disparate busses.

Everything must link into a network, from test equipment on the assembly line, all the way down to each silicon die of each wafer. With a filament in every die, linked into the boundary scan, it makes it easy to control, test, and configure devices as before they have been cut from the wafer. If the components on the die are computational elements, they might needed no connection other than power and network. For building a supercomputer with such components, there would be no need to break the wafer into chips. The entire wafer could be packaged and used whole, routing around dead . This might be an effective way to build a powerful array of computing elements at very little cost.

5.3. Physical Interfaces for the Distributed Object Web

The ultimate challenge with filaments is to make the software completely transparent. We have had the technology to build networked systems for many years. The massive changes sweeping our

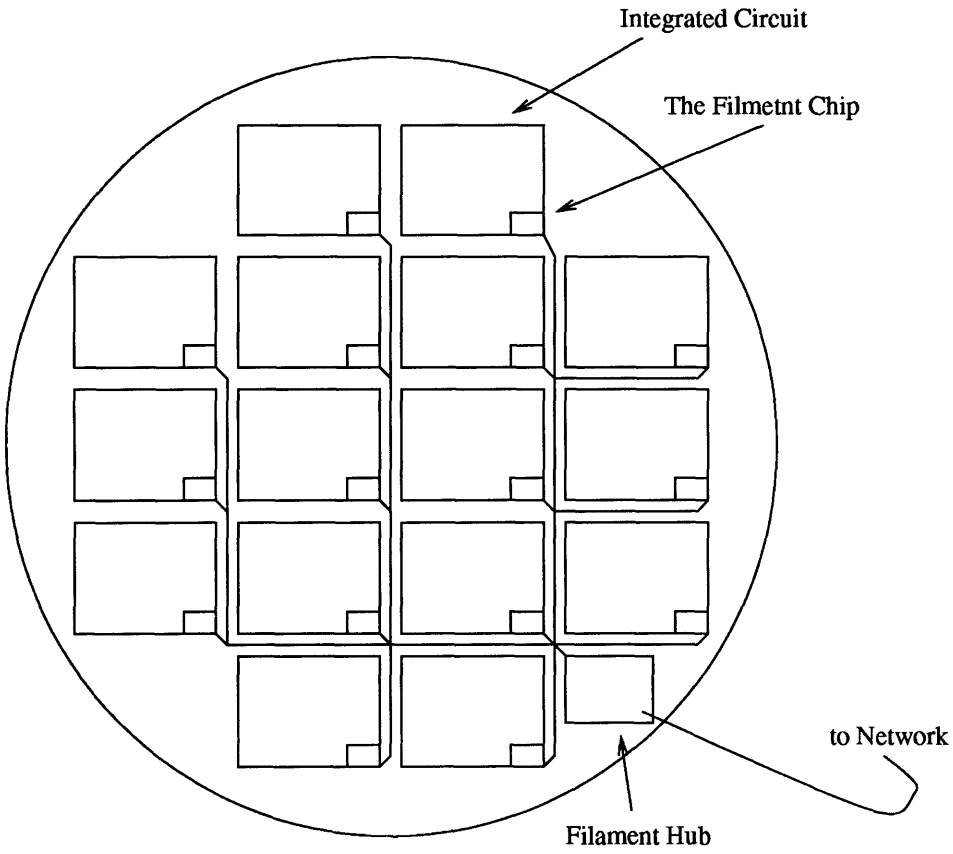


FIGURE 5.2.1. Filaments on every silicon die

world are coming from using information in new ways. In 1990 no one would have believed that search engines could index a resource as enormous as the web, much less do it as a free service based on advertising revenue. The effects of the web are seeping into our culture and having an effect on how we design information systems within businesses and also at home. We currently have a crude back-end for delivering these services.

HTTP has limitations that make it impossible to do advanced operations like sharing consistent distributed objects. Systems can go beyond CGI functionality by using an object model technology. Existing standards such as CORBA and COM/DCOM are growing, but have not reached the scale of the web. To jump-start the "object web" the W3 Consortium (W3C) is proposing a next-generation HTTP that builds on these fundamental concepts.

Whatever object model becomes dominant, one thing is certain: with distributed objects we will need new ways of thinking about problems, or rather, we will need new ways of thinking about how to solve problems with information technology.

This is a very young field and has parallels to the early days of of the industrial revolution. We started manufacturing parts by hand, but then learned how to build machines with other machines. This system took off when we began to design and build devices with of interchangeable parts.

Today in software we lose a lot of design work every day. The proprietary software model keeps innovation from proliferating but instead hoards it in a company coffers. Whenever a software program is slightly less than successful it is yanked away from the public. The source code is secret, never revealed to anyone. Thus when the product stops, the code is effectively gone, erased, destroyed. We do not show the same respect to software libraries as we do to those with books. We burn down the libraries of software, never giving most programs a chance to mature. It takes years to get the last bugs out of a program but the makers of software have no patience and push rushed implementations of new features instead of polishing an existing product into solid stability.

The object web offers hope to have the same effect for the information revolution as as interchangeable parts had on the industrial revolution. Filaments of connectivity will need to extend into the physical world from this object web. It will be done many ways, but the solution we can live with will give us direct access to our filaments in a secure, ubiquitous web of information. That is an information web in which people can live.

Appendix A

There are two primary sources of standards information for embedded networking: http://www.IETF.org and http://www.IEEE.org.

The Internet Engineering Task Force (IETF) has a document policy based on requests for comment (RFCs). These documents are the mechanism for creating public internet standards in a process managed by the IETF. The full text of the RFCs are are freely downloadable form several archives around the world. Check the IETF homepage for information on recent RFCs.

RFC1317 Definitions of Managed Objects for RS-232-like Hardware Devices

RFC1369 Implementation Notes and Experience for the Internet Ethernet MIB

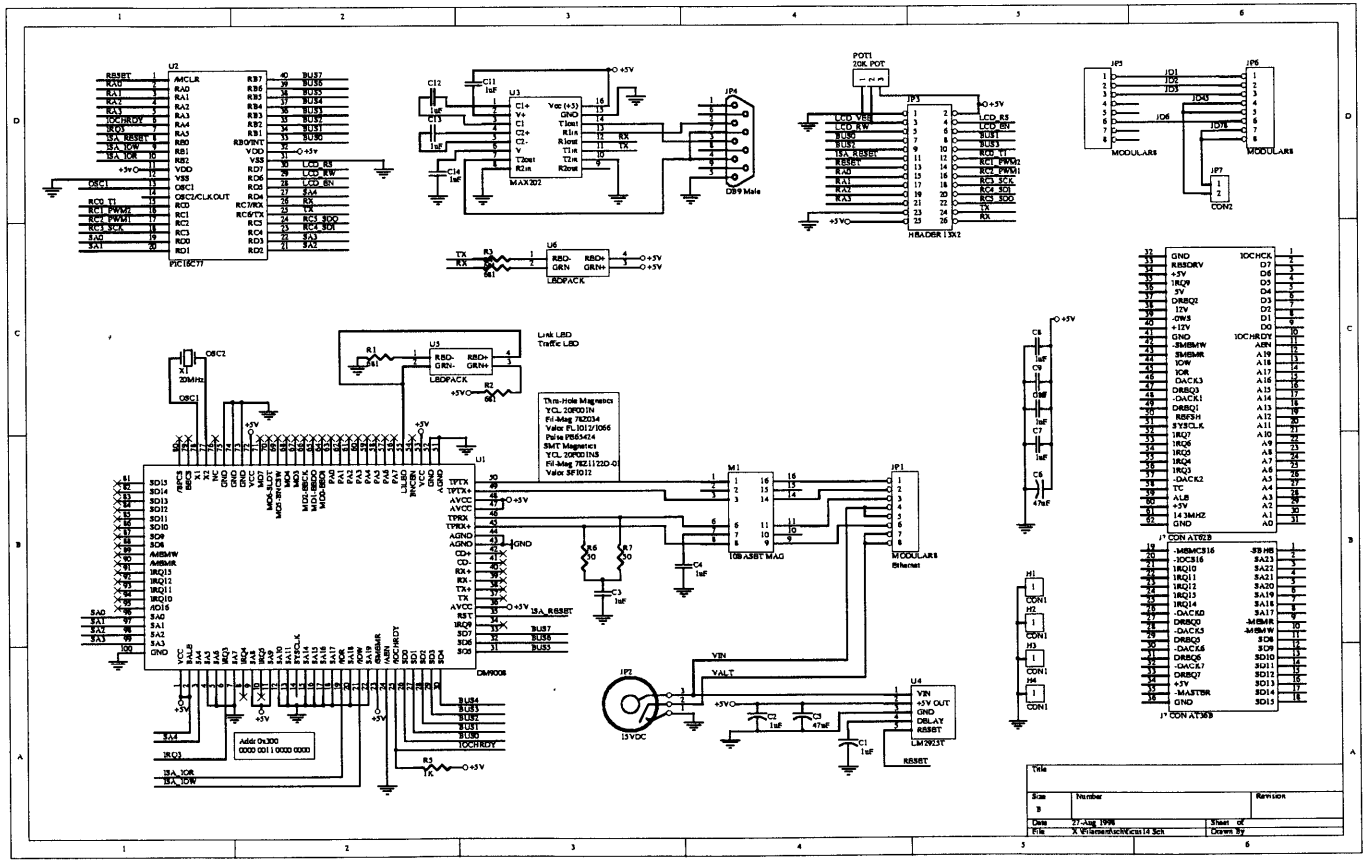
RFC1541 Dynamic Host Configuration Protocol

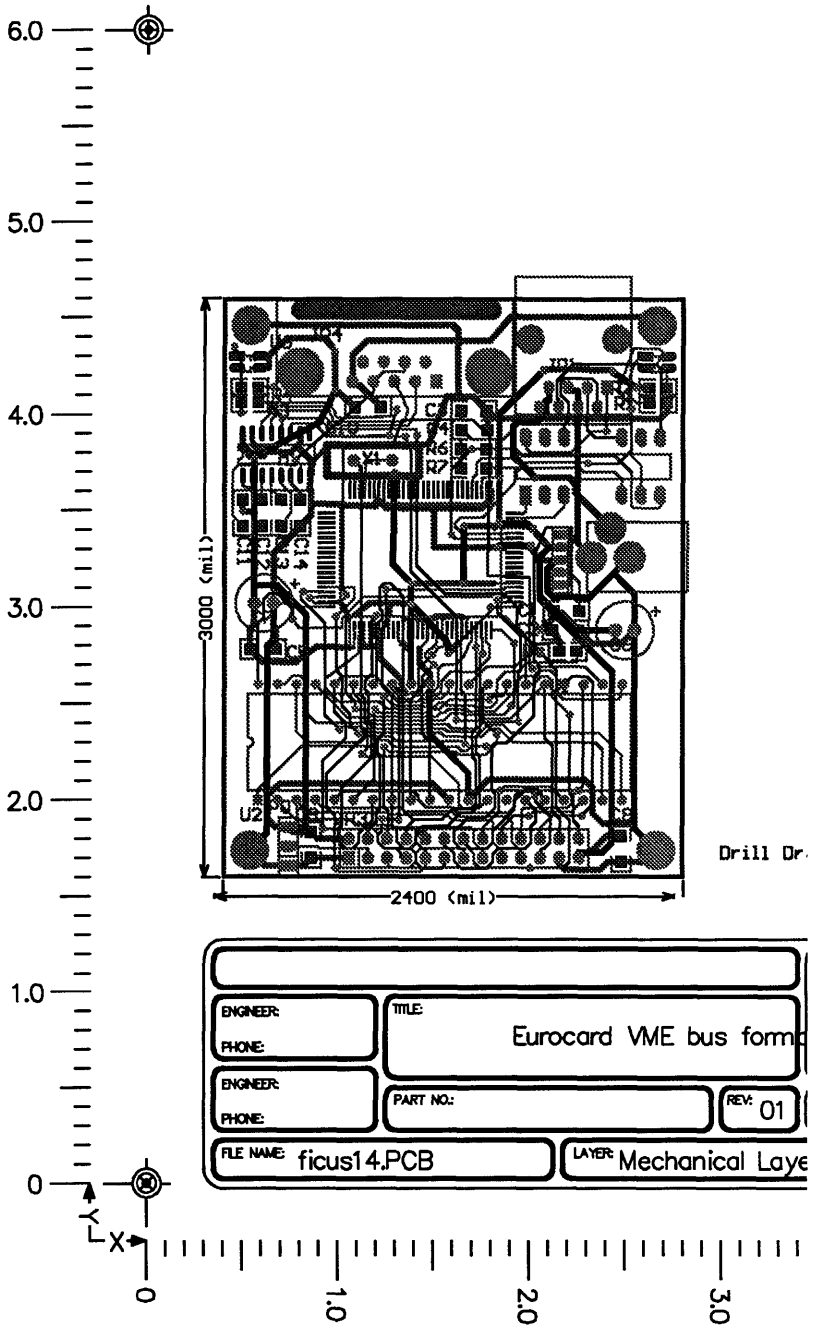
Name	IEEE Standard
VHDL	IEEE Std 1076-1993
Verilog	IEEE Std 1364-1995
Wireless LAN	IEEE 802.11-1997
PCI Bus	IEEE P1386.1
JTAG	IEEE Std 1149.1-1990
Fire Wire	IEEE Std 1394-1995
USB	Not an IEEE standard, see Intel Corp.
Ethernet LAN	ISO/IEC 8802-3 (ANSI/IEEE Std 802.3)
10BaseT	IEEE Std 1802.3d-1993 (UTP Cat 3 cable)
Fast Ethernet LAN 100BaseT	IEEE Std 802.3u-1995 (UTP Cat 5 cable)
Token-Passing Bus (Manufacturing)	ISO/IEC 8802-4 (ANSI/IEEE Std 802.4)
Token Ring LAN	ISO/IEC 8802-5 (ANSI/IEEE Std 802.5)

TABLE 2. Common IEEE Standards and their numbers

Network	Media	Contact
Appletalk	2 wire	http://www.apple.com
Ethernet (10Mbps)	Cat3/Cat5/fiber	http://www.ieee.org
Fast Ethernet (100Mbps)	Cat5/fiber	http://www.ieee.org
Gigabit Ethernet	fiber/soon Cat5	http://www.ieee.org
Sonet+ATM	fiber	http://www.t1.org
DeviceNet	2 wire	http://www.odva.org
ARCNET		http://www.arcnet.com
X10	AC power lines	http://www.x10.org
CANbus	2 wire	http://www.can-cia.de
USB	4 wire (includes power)	http://www.usb.org
FireWire/ IEEE 1394	Cat5/fiber	http://www.ieee.org
LonWorks	2 wire	http://www.echelon.com

TABLE 3. Sources of additional information for Network Standards


U2
PIC16C77
U3
MAX202
JP4
DB9 Male
POT1
20K POT
JP3
HEADER 13X2
JP5
MODULARS
JP6
JP7
CON2
U6
LEDPACK
Link LED
Traffic LED
OSC2
OSC1
20MHz
U1
DM9008
Thru-Hole Magnetics
YCL 20F001N
Fil-Mag 782034
Valor FL1012/1066
Pulse PE65424
SMT Magnetics
YCL 20F001NS
Fil-Mag 78Z1122D-0
Valor SF1012
M1
10B ASBT MAG
JP1
MODULARS
Ethernet
VIN
VALT
15VDC
U4
LM2931Z
RESET
CON AT92B
CON AT36B
Title
Size
B
Number
Revision
Date
27-Aug-1998
Sheet of
File
X:\Manuals\Ch14\ch14.Sch
Drawn By


6.0
5.0
4.0
3.0
2.0
1.0
0
Y
X
0
1.0
2.0
3.0
3000 (mil)
2400 (mil)
Drill Dr
ENGINEER:
PHONE:
TITLE:
Eurocard VME bus form
ENGINEER:
PHONE:
PART NO.:
REV: 01
FILE NAME: ficus14.PCB
LAYER: Mechanical Laye

Filamentors

Building Communications

FL00401

IEEE 802.3/Ethernet Filament

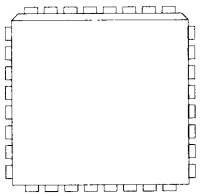
FEATURES

- Remote configuration and control of all interfaces and behaviors over the Ethernet
- Interconnects directly with IEEE 802.3/ Ethernet networks
- Open protocol allows easy development of new host software
- Interconnects directly to RS-232
- Asynchronous communications are handled by internal UART
- Interconnects directly to EEPROMs, CODECs, and other synchronous serial peripherals
- Option to boot from EEPROM
- Software implementation of I^2C protocol is available
- Single +5V supply
- Pre-configured with a valid MAC address in EPROM, guaranteed to be globally unique
- General purpose IO
- All I/O lines can source & sink 25ma
- Interfaces directly to standard LCD modules
- Unprogrammed OTP and windowed EPROM versions are available for applications needing direct control

DESCRIPTION

The FL00401 Ethernet Filament is an Ethernet controller merged with a general purpose microcontroller. The chip can be controlled remotely over the Ethernet using the filament protocol. Other protocols may be supported with custom developed software. The chip may be used without any additional embedded software for many programmed I/O operations.

PIN ASSIGNMENT



1 digital GND
2 digital +5V
3 TX-RS232
4 RX-RS232
5 SDO
6 SDI
7 SDCK
8 FRAME
9 /RESET
10 TX+
11 TX-
12 RX+
13 RX-
14 OSC1 in
15 OSC2 out
16 EtherStatusLED
17-32 D0-D15

OVERVIEW

The FL00401 contains three major components: a 10BaseT Ethernet Controller, a microcontroller, and peripheral I/O. The microcontroller is intended to be used with provided code which implements the filament protocol. For peripheral I/O, synchronous and asynchronous serial ports are available as well as tri-state logic.

HARDWARE RESET

When running the filament code, the microcontroller goes through a series of steps on reset.

1. Attempt to make contact with a filament administrator
2. Look for a serial EEPROM on the synchronous serial port
3. If present, check for a valid checksum on the contents.
4. If valid, begin interpreting the stored program.
5. Main loop: listen for new instructions from the filament administrator while executing the stored program (if available).

All clocking is derived from a 20MHz crystal. Future implementations may use a PLL to run the microcontroller at higher frequencies without needing an additional crystal.

SOFTWARE MODES

There are two operating modes for a filament: *command mode* and *execution mode*. On reset the filament looks for available code. If it finds a serial EEPROM with valid code, it then enters execution mode. While stepping through the interpreted instructions, the filament continues to have orthogonal communications with the filament administrator.

While in execution mode, the filament administrator can tell a filament to switch into command mode. In command mode, it is a simple operation to load a new program into the serial EEPROM. On completion, the filament administrator may restart execution mode to begin running the new program.

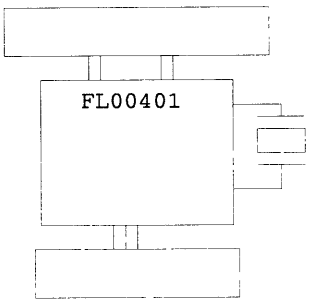


FIGURE 1.1 Block Diagram of Circuit

The figure above illustrates the basic use of the FL00401 to build a filament board. The upper block is the 10BaseT isolation transformer,. An Ethernet port interfaces to the transformer.

The lower block is the user application. The application may include building blocks such as a serial EEPROM for storing interpreted code, an LCD display, or an RS-232 interface.

Appendix C

Components

Qty	Part	Description	$each	$total
	DIGIKEY	**http://www.digikey.com**		
1	NJM78M05FA-ND	Voltage Regulator +5V @ 0.75A	0.36	0.36
1	X438-ND	20.000 MHz Crystal	1.17	1.17
1	T405-ND	Wall Transformer 9V @ 0.50 A	4.62	4.62
1	CP-202A-ND	Connector: Power Jack 2.0mm PCB	0.25	0.25
1	PIC16C77/JW-ND	IC Microcontroller 8K EPROM 40pin	22.85	22.85
1	A9033-ND	Connector: Mod 8-8 Jack, low profile		
1	ED3740-ND	IC Socket 40pin tin/tin 0.600 spacing	1.12	1.12
1	MAX202	RS232 Transceiver/voltage multiplier	2.42	2.42
3	8714K-ND	50 Ohm 1206 SMT resistors	-	-
3	8401K-ND	681 Ohm 0805 SMT resistors	-	-
2	LU603715CT-ND	LED RED/GRN WATER CLR 1210	0.57	1.13
2	###	47uF Radial Cap @ 25V Low Impudence	0.15	0.30
	MOUSER	**http://www.mouser.com**		
12	140-cc502z104/REEL	.1uF 1206 SMT Chip Caps (4000pcs)	0.03	0.33
	DAVICOM	**http://www.davicom8.com**		
1	DM9008	NE2000 Compatible Ethernet Controller	5.10	5.10
	YCL	**http://www.ycl.com**		
1	20F001N	10BaseT Magnetics	2.00	2.00
	Alberta Printed Circuits	**http://www.apc.com**		
	###	Printed Circuit Board	7.63	763

TABLE 4. Filaments Parts List

Appendix D

Digikey

Known as the best parts source in the world, Digikey has on-line order forms with pricing information and part descriptions. With same day shipping, having a Digikey catalog on hand is often the difference between success and failure. Though not the least expensive reseller, Digikey is definitely the most organized. Their parts stock is not complete but is still quite broad.

1-800-DIGIKEY

http://www.digikey.com

Davicom

Davicom makes extremely inexpensive Ethernet controllers. Winbond stopped selling their buggy ISA bus chip, but RealTek and other still sell competing components.

http://www.davicom8.com

YCL Magnetics

One of many sources of 10BaseT magnetics. While I purchased 100pcs for $2.00 each, one should be able to find an equivalent part from other vendors for around $1.00. Pulse and Valor compete directly with YCL.

http://www.ycl.com

Custom Computer Services, Inc

CCS makes a small C-like compiler for Microchip They support the PIC16Cxx and most other parts, however they have not released a version that supports the PIC17Cxx at this time. Since the PIC17Cxx doesn't have tiny pages like the PIC16Cxx, it should be just as easy to work without a compiler on those chips.

The CCS compiler is not the best available and is not full ANSI C. We recommend looking at the Microchip webpage for information on which compiler is recommrecommendedoned.

http://www.ccsinfo.com

http://www.microchip.com

RF Solutions

When building a complex software application, it is very useful to be able to do incremental testing. The PIC emulator from RF Solutions has proven to be a valuable tool for developing non-trivial embedded software. The emulator is available from Digikey for approximately $800.

http://www.rfsolutions.co.uk/

Corel Computer Company

The Filament Administrator requires a robust server with two Ethernet ports. Fortunately such a device is available off-the-shelf from Corel Computer. The Netwinder DM (development machine) has all the functionality necessary to act as a small, low-power, reliable filament administrator. For approximately $700 USD they provide a complete system with a 275MHz StrongArm and RedHat Linux. The unit has both 10BaseT and 100BaseT Ethernet, and as such makes an ideal bridge between networks.

http://www.corelcomputer.com

http://www.netwinder.org

Osicom

Osicom has developed a midrange system that may evolve into an ideal solution for building filaments. Called the NET+ARM, they have integrated an ARM microcontroller core with a 100BaseT Ethernet controller. This is bundled with PSOS, a real-time operating system with a network stack, and sold for $27-$32 in quantity. Currently it is a four-chip solution as it requires external RAM, ROM, and physical interface transceiver for Ethernet. They intend to integrate the physical interface in the near term and migrate towards higher-end and lower-end applications. One goal is to achieve a $10 pricepoint on the low-end by 2000. If they make this goal, their system will be an extremely compelling way to integrate Ethernet functionality. For applications which require 100 Mbit, the Net+Arm is the most viable embedded option, since existing single-chip 100Mbit controllers only offer a PCI bus interface. Designing PCI into an embedded application is not impossible, but is potentially rather expensive. In the future, the NET+ARM may be the platform of choice for embedded networking with Ethernet.

http://www.osicom.com

http://www.net+arm.com

Bibliography

[1] Beyer, D., Vestrich, M., and Garcia-Luna-Aceves, J.J. *The Rooftop Community Network: Free, High-speed Network Access for Communities*, to be published by Harvard University's Information Infrastructure Project, Spring 1997.

[1] Flanagan, David. (1996). *Java in a Nutshell.* O'Reilly & Associates.

[2] Garcia-Luna-Aceves, J.J., Fullmer, C.L., Madruga, E., Beyer, D. and Frivold, T., "Wireless Internet Gateways (WINGS)", Proc. IEEE MILCOM'97, Monterey, California, November 2–5, 1997.

[3] Horowitz, Paul, and Hill, Winfield, The Art of Electronics Second Edition, Cambridge University Press, 1980, 1989.

[4] Huitema, Christiann. (1998). *IPv6 The New Internet Protocol (Second Edition).* Uppper Saddle River: Prentice Hall PTR.

[5] Murphy, Eaton. Hayes, Steve. Enders, Matthias. *(1995) TCP/IP Tutorial and Technical Overview Fifth Edition.* Upper Saddle River: Prentice Hall PTR.

[6] National Semiconductor. (1993). *DP8390 Network Interface Controller: An Introductory Guide.* http://www.national.com/an/AN/AN-475.pdf

[7] Orfali, Robert, Hrkey, Dan, and Edwards, Jeri. (1996) *The Essential Distributed Objects Survival Guide.*John Wiley and Sons.

[8] Shanley, Tom and Anderson, Don. (1995). *PCI System Architecture*, MindShare Inc, Addison Wesley