AN ABSTRACT OF THE THESIS OF

Johannes K. Reinalda for the degree of Master of Science in Electrical and Computer Engineering, presented on May 20, 1991.

Title : A Controller for Internet Protocol Routing of AX.25 Packets.

Abstract approved : Redacted for Privacy

James H. Herzog

Amateur Packet Radio Networking presently uses the NET/ROM protocol to establish the network. NET/ROM is considered to be insufficient to support the expected growth of the network. This research work proposes to use the TCP/IP protocol suite instead to build the network. A comparison between features of both protocols supports this proposal.

A new and simple hardware platform is introduced. This will provide adequate support for initial experiments. Design considerations for both hardware architecture and software architecture are discussed in detail. Implementation of the IP protocol used for routing is discussed.

A Controller for Internet Protocol Routing of AX.25 Packets

by

Johannes K. Reinalda

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed May 20, 1991

Commencement June 1992

APPROVED:

Redacted for Privacy

Professor of Electrical and Computer Engineering in charge of major

Redacted for Privacy

Head of department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented May 20, 1991

Typed by Johannes K. Reinalda for Johannes K. Reinalda

ACKNOWLEDGEMENTS

It is my pleasure to acknowledge the help and guidance of Dr. James Herzog. He is my major professor and advisor for this thesis. Even with his busy schedule he could always find some time to talk to his students. I thank him for his time and the encouragements I have received from him.

I would like to thank Jim Wagner for introducing me to the world of packet radio. We have had lots of great discussions about the subject. His enthusiasm and help was wonderful.

I owe thanks to Otto Gygax. He and the support staff are a great bunch of people that keep networking in the ECE department alive.

Many thanks are going to the Dept. of ECE for supporting me with a teaching assistantship. This made it possible for me to pursue this degree. Without the help of the office staff my stay would have been much harder. I thank all of them.

Professor William Harrison, Chair of the College of Business' Department of Management Science has kindly agreed to be my minor professor. I thank him for this. I also thank Prof. Rathja, E.&C.E. and Prof. Levien, Chem. Engr, for agreeing to be in my graduating committee.

I thank all my friends here in Corvallis for all the great times we had.

Finally, i would like to thank my parents for their support, understanding, encouragements and love in all these years far away from home.

TABLE OF CONTENTS

LIST OF FIGURES

A CONTROLLER FOR INTERNET PROTOCOL ROUTING OF AX.25 PACKETS

CHAPTER 1
INTRODUCTION.

In the last two decades the use and development of computer networks has increased dramatically. Local Area Networks (LAN's) are now well known in many educational institutions, and research institutes as well as commercial enterprises. They serve a wide variety of applications. They range from simple message exchange to more extensive distributed computing applications. A less known area of use of computer networks is what is called Amateur Packet Radio. This is a form of computer networking practiced by many Radio Amateurs, or hams, around the world. Instead of expensive professional high speed equipment and Internet networks, they use relatively low speed modems and radio transmitters to establish the network.

In any network the use of efficient communications protocols is extremely important to ensure high utilization of the network capacity. An analysis of the underlying physical system is needed. Quite often, the protocols turn out to be efficient only when designed with considerable attention for the characteristics of such networks.

One of the oldest long haul networks, the ARPANET, was built by the Defense Advanced Research Project Agency (DARPA) in the late 1960s [Com88]. It has jointly participated with the National Science Foundation (NSF) to form what is commonly known as the Internet. The Internet consists of many smaller subnetworks tied together by long haul networks provided by NSF and DARPA. The set of protocols used on this network is known as the TCP/IP suite [Tan88]. These protocols have been specifically designed to handle problems arising when trying to deal with efficiency and other issues in networks.

Amateur Packet Radio uses a protocol known as NET/ROM to establish the transporting connection between two end users on separate LAN's. The protocol was first developed when there was a very limited user base. In recent years the very rapid expansion of the number of users has severely limited the capabilities of the network.

This work proposes the use of the aforementioned TCP/IP suite of protocols to establish networking in the Amateur Radio community. TCP/IP is more structured toward use over wide area packet-radio networks and has better performance than the present protocols. It not only is more robust in performance but also allows for a wider and easier use of the network facilities. A new hardware platform is proposed and developed to facilitate the user's switch to the new protocols.

This thesis is organized in 6 chapters. Chapter 2 gives an introduction to computer networking and the ISO-OSI model. The present state of Amateur Packet Radio Networking is also discussed. Chapter 3 discusses related previous work. It concludes with a statement of the research problem and the motivation for solving this problem. The hardware design issues and hardware platform are shown in Chapter 4. Details of the implementation are discussed in Chapter 5. Some performance analysis and test results are given in Chapter 6. Possible future improvements and enhancements are also given here.

CHAPTER 2

COMPUTER NETWORKS.

2.1 An Introduction to the ISO-OSI Model.

The International Standards Organization (ISO), as a first step toward international standardization of various communication protocols used in computer networks, proposed the Open Systems Interconnection Reference Model (OSI). The standard deals with systems that are open for communication with other systems [Tan88]. The model consists of several layers of abstraction to be used in the design of communication protocols. Among others, the following guidelines were used to create the different layers. Layers should perform well-defined functions and should be chosen such as to minimize the flow of information across layer-boundaries.

A short explanation of the seven layers proposed by the model is given below. (see also Figure 1)

1. **PHYSICAL LAYER**: This is the layer concerned with the transmission and reception of raw bits over a communication channel [Tan88]. All issues addressed pertain to the physical properties of the channel, such as voltage and current levels of signals, modulation and demodulation techniques, connections to the circuit, connectors etc. This layer has no knowledge of the information handled or the protocols used by higher layers.

2. **DATALINK LAYER**: The services provided by the physical layer are used by this layer to handle data in the form of a frame. A frame is a series of bytes coming from the physical layer and going to the

OSI	TCP/IP	NET/ROM
Application Layer		
Presentation Layer	SMPT, TELNET, FTP NNTP, and others	NET/ROM User Interface
Session Layer		
Transport Layer	Transmission Control Protocol	NET/ROM Level 4
Network Layer	Internet Protocol	NET/ROM Level 3
Datalink Layer	Ethernet	AX.25
Physical Layer	Ethernet	AFSK or FSK

Figure 1.
The OSI model and its relevants to the NET/ROM and TCP/IP protocols.

network layer, or vice versa. Since the physical layer does not distinguish between bits, frame distinction is handled by this layer. This can be in the form of start and end of frame sequences and checksums (or cyclic redundancy checks) to ensure correctness of the information in the frame. Some flow control is also done. This layer deals only with direct communications (e.g. point-to-point links.)

3. **NETWORK LAYER**: This layer is concerned with the workings of the subnet. Routing of packets from source to destination as well as congestion control for the subnetwork is done in this layer. Arriving frames handed over from the datalink layer will be routed and retransmitted if destined for other hosts. Available services may include change of datalink layer protocols from one subnet to another, fragmentation and reassembling of packets when dealing with subnets with differing maximum frame sizes, or packet routing decisions based upon internal routing tables and information obtained from other systems through the use of routing information protocols. The network layer, in contrast to the datalink layer, allows for communications between hosts with no direct link between them.

4. **TRANSPORT LAYER**: This layer is a true host-to-host layer. Communication is between two distinct processes. It provides an error free stream of data that ensures that all data arrives at the opposite end in the same order as transmitted. Message streams are either delivered without errors or the transport layer will report to the message source that the message cannot be delivered. End-to-end data flow control is also performed. This layer may have the ability to

multiplex several low rate sessions with the same destination into one session at the network layer. This reduces the number of sessions on the subnet and the protocol overhead [Ber87].

The physical, datalink, network, and the transport layer are illustrated in Figure 2.

5. **SESSION LAYER**: This layer permits different processes to negotiate, establish and end an error-free communication session. Logistics such as access rights, token management and process synchronization are handled by this layer.

6. **PRESENTATION LAYER**: Unlike the lower layers, this layer is interested in the syntax and semantics of the information transmitted [Tan88]. Data encoding, conversion between host formats, display manipulation for interactive jobs and data compression are some of the services provided.

7. **APPLICATION LAYER**: This layer is often programmed by end-users. Applications are often written to make the network transparent to the end user. Such application programs contain components to deal with the numerous types of terminals used by end-users, to handle file transfers etc.

When layers pass information packets to lower layers for processing, a control segment is appended. It is most often placed at the start of the frame and is called a header. The header allows the

' ______ ' the route travelled by data packet from source to destination and vice versa.

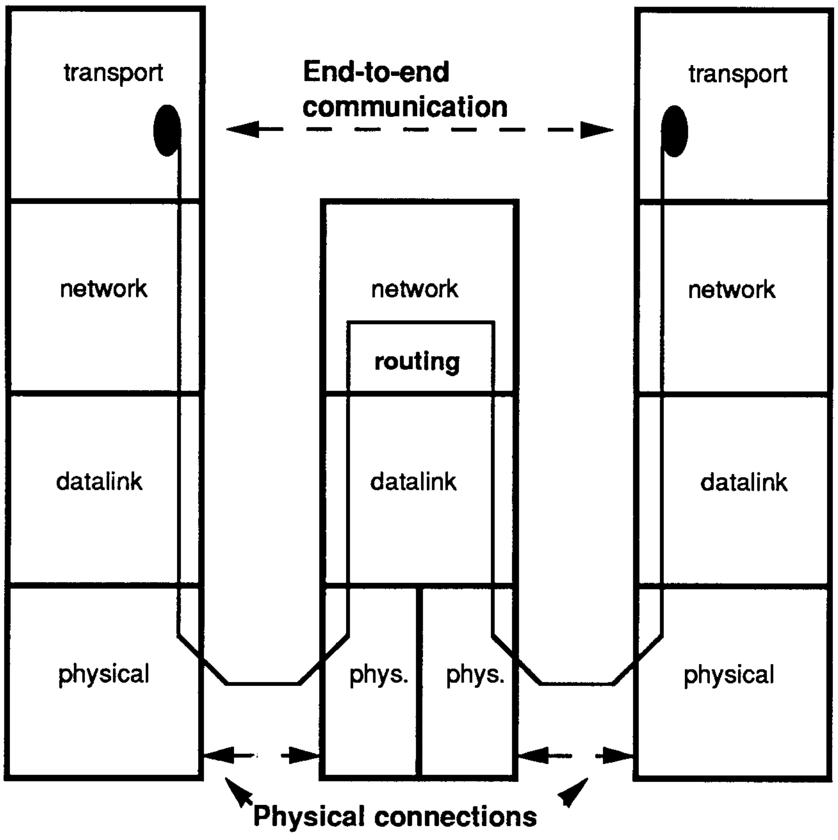


Figure 2. Routing and transporting in the ISO model.

receiving end to derive the needed information about the frame. Upon reception of a packet, and before passing it to a higher layer, each layer detaches its header from the frame for processing. The basic format of a network packet is shown in Figure 3.

2.2 The Physical Layer in Amateur Packet Radio.

The physical and datalink layers in a large number of computer networks consist of either Ethernet[1] or token ring technology. These are well known. Less well known are the standards used in Amateur Packet Radio. These will be discussed briefly.

Currently Amateur Packet Radio uses a few de-facto standards in the physical layer. All these standards are half-duplex, ranging from 300 Bd Bell 103 to 1200 Bd Bell 202 modem technology. Standards for faster modems in ranges from 9600 Bd to a scarce 56 kBd or above are being developed.

Physical layer technology is yet another multidimensional part of computer networks. The main area of this work is in the datalink and network layer. Therefor this aspect is not discussed any further.

2.3 The Datalink Layer in Amateur Packet Radio.

AX.25, or Amateur X.25, is the datalink layer protocol used in Amateur Packet Radio. It is a modified version of the commercial X.25 level 2 standard [Fox84]. X.25 was written by the International Telegraph and Telephone Consultative Committee (CCITT). In AX.25 the source and destination addresses have been extended to 7 bytes to

1 Ethernet is a registered trademark of Xerox Corporation.

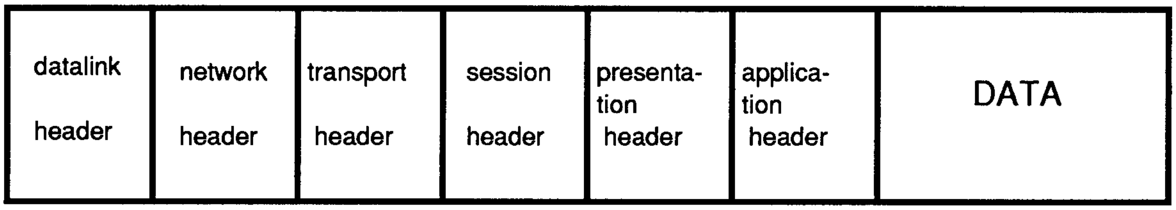


Figure 3. A general network frame format.

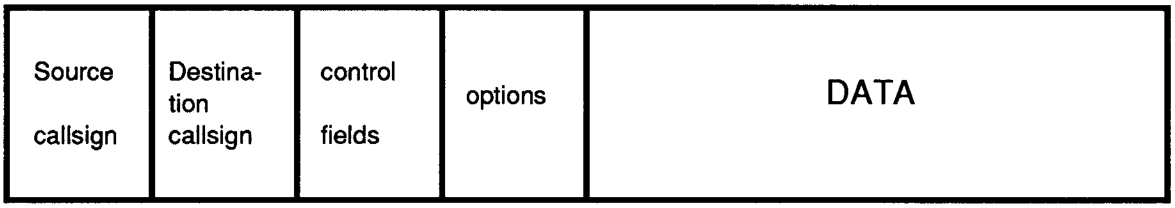


Figure 4. An example AX.25 datalink layer frame format.

allow a six character callsign and one character secondary identification. Some modifications concerning host-to-host connections have also been made. AX.25 provides for both connection oriented streams and a connectionless or datagram service. It uses a 7 frame sliding window protocol. A sample packet format is shown in Figure 4. A formal definition of AX.25 can be found in [Fox84].

2.4 An Introduction to LANs.

Local Area Networks, or LANs, consist of several machines interconnected in a local fashion. Most often the connection is through the use of coax cable or twisted pair wires. These machines can establish direct communication with each other, without intervention of another system. This often occurs at very high speed (in the order of 10 Mb/sec). LANs are interconnected through gateways. Gateways have a link to another LAN or to a WAN, a Wide Area Network. WANs often span several tens or hundreds of miles using lower speed leased lines or microwave links.

2.5 An Introduction to Amateur Packet Radio Networks.

Present packet radio networks are made up of both LAN channels and WAN channels. The LAN channel is an established frequency (or channel) that the local ham community uses to establish communication inside the local user area. In order for communication to occur between two station in the same area, they have to agree on a channel. When the channel is not temporarily occupied by a packet from another user, one station will transmit a 'connect request' to the other. If received without errors, the other station will acknowledge the request. It waits until no signal (or carrier) is heard on the channel. Next it transmits its reply, the connect acknowledge. Anytime a reply to a packet is not received in a timely fashion, a retry of the packet is sent. Following this setup phase, data can flow from either station to the other, using the same 'listen and talk' strategy. This scheme allows multiple stations access to one channel. It is called Carrier Sense Multiple Access, or CSMA. Many problems arise with the CSMA channel access method. However, this is the present de-facto standard. Many have proposed different access methods but no method has been implemented as a standard at present.

The WAN channel is better known as the 'Backbone' channel. It is a doorstep to the rest of the world: it serves as a transporting channel to establish network communications with LAN user channels in areas outside direct reach of the local user. On the Backbone channel the same CSMA channel access method is used.

The system combining LAN and Backbone channel access consists of two (sometimes more) small controller systems with modems and transmitters. Both systems have a radio interface controlling the transmitter, and a serial interface connected to the other unit(s). The controller most often used is commercially available. The controller is marketed for use by the packet radio user community. The controllers were developed to interface between a user terminal and the transmitter, thus providing AX.25 Amateur Packet Radio capabilities. The user software can be replaced by new networking software. This makes the controller capable of networking.

This setup of two or more units linked via a serial line is commonly known as a node. Each channel in a node is commonly referred to as a port. Nodes are often located at high sites like hilltops to take advantage of the improved radio channel capabilities. It allows the Backbone port to communicate directly with other nodes long distances away. In general, there are no users directly accessing this Backbone channel.

Nodes can be instructed by LAN users to establish communication to a distant node via the Backbone. When this has succeeded a connection to a local user on the remote LAN can be requested. The flow of data is as follows and can be seen in Figure 5.

1 - Data flows from the originating user over the LAN channel to the LAN port of the local node.

2 - The LAN port of the node sends the data to the Backbone port over a serial line.

3 - The data gets transmitted over the Backbone channel to the remote Backbone node.

4 - This data possibly gets retransmitted by multiple Backbone nodes to reach the remote node.

5 - The distant Backbone port sends the data over a serial line to its LAN port.

6 - Finally the remote LAN port transmits the data over its LAN channel to the remote user.

Data flows from the remote user to the local user using all the same nodes and ports in reverse order.

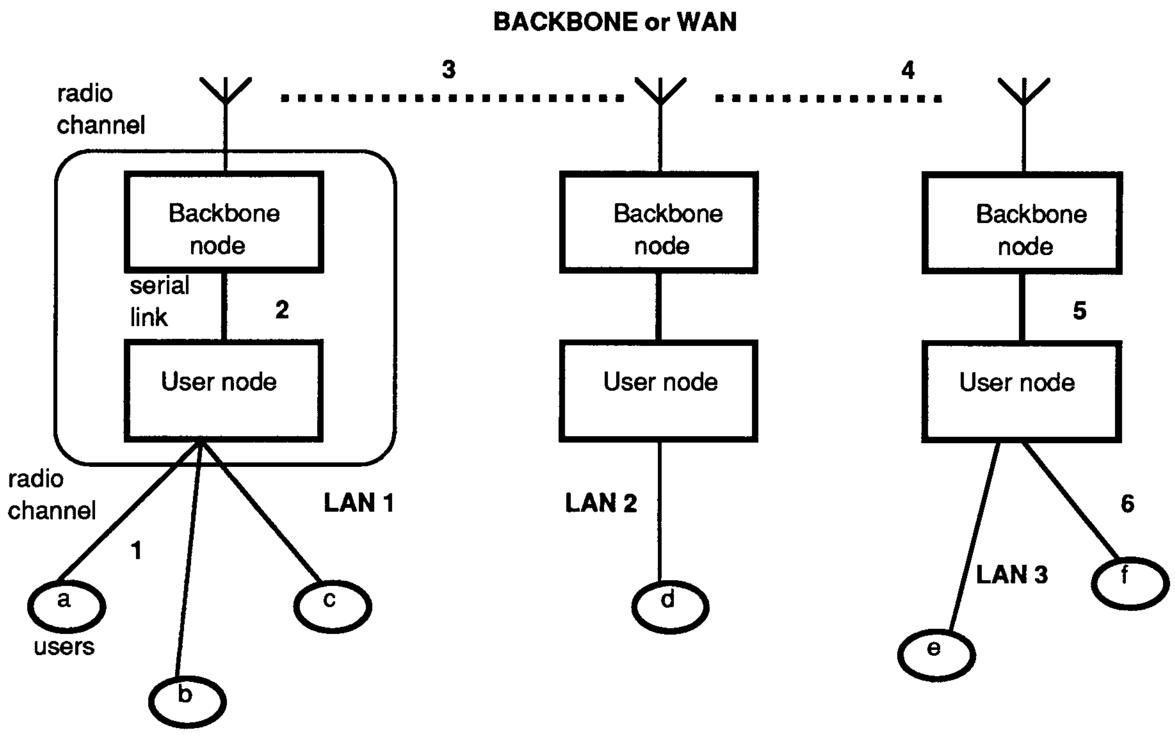


Figure 5. A common configuration of Packet Radio Networks.

A typical present system (in Western Oregon) allows for semi-usable LAN to LAN connections using five to six Backbone nodes to reach destinations up to about 125 miles away (this is with the 1200 Bd technology.) The range is limited by bad radio links which frequently drop packets. The low data rate in use creates very low throughput, which is often unacceptable to end-users.

2.6 Routing in the OSI Model.

When two subnetworks with different datalink layer protocols but identical network layer protocols are interconnected, a gateway is needed. Gateways have datalink layer translation facilities in addition to network level routing facilities. When implementing a gateway to interconnect subnetworks with identical network layers, only the three lower layers of the OSI model need to be dealt with. Everything from transport layer on up is strictly on a host-to-host basis. The gateway only has to know how to correctly communicate using the datalink layer protocol on both subnets, and how to route network layer packets correctly between the two subnets. Datalink protocols could be as different as Ethernet (10 Megabit/sec) on one subnet and current AX.25 implementations (1200 bit/sec) on another.

2.7 Link Layer Address Resolution.

The preceding discussion raises the following issue. It is assumed that hosts on the same physical network can directly communicate at the datalink layer only if they know each other's physical network address [Com88]. How then does a host (or gateway) relate a network level address to a datalink layer address ? An address resolving mechanism that allows for the determination of the datalink address of the next gateway or host to be sent a network packet will need to be derived. This dilemma is commonly known as the Address Resolution Problem.

There are several ways of solving this problem. One solution is implementing a direct relationship between network and datalink layer addresses. Use of Address Resolution Caches that save encountered network/datalink address pairs is another well-known solution. Dynamic binding is a solution that allows a host or gateway to request the datalink address of the target network address on the local subnet. The response from some host with this knowledge (very often the target system) is awaited before handling the network frame.

The most often implemented solution is a combination of the caching scheme and the dynamic binding scheme. It is known as the Address Resolution Protocol (ARP) [RFC826]. It is designed to work with a multitude of physical networks and datalink protocols, ranging from fast Ethernets to slow Packet Radio Networks.

When a gateway receives a frame it looks up the route. If it determines that this destination is on the local physical network, it checks the ARP cache for the datalink (or physical) address of the destination. If it is found, the frame is send immediately. If there is no entry for this destination, an ARP request packet is broadcast on the local network. This packet requests the physical address for the wanted destination address. If some system on the local network knows the answer, it will reply to the gateway. Most often this reply comes from the destination system itself. Following this, the gateway knows the physical address for the destination. The information is stored in the ARP cache and the frame is transmitted.

This same scheme can also be used between two users. A user wants to communicate with another user. The user knows that the other is on the local network. It also knows the network address, but not the physical address of the other user. It can broadcast an ARP request packet to find out the information. The destined user will respond and then communication can start.

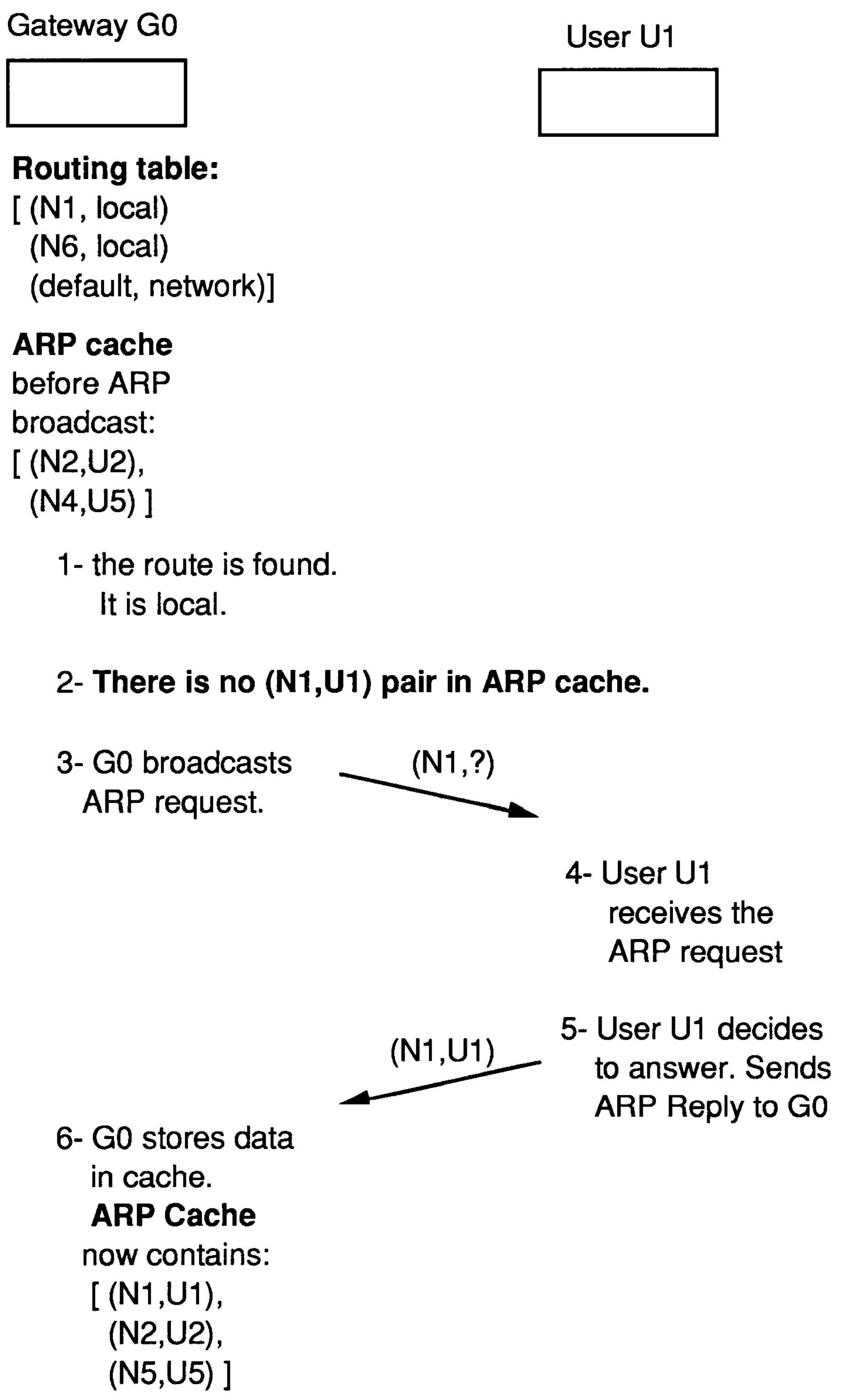
The steps in the address translation problem are as follows (See also Figure 6.) The content of the routing table and the ARP cache of the gateway are given in Figure 6.

1 - Gateway G0 gets a frame for network address N1. It checks the routing table and determines that this is a user on the local network.

2 - Gateway G0 will then try to find a mapping from network address N1 to the physical address Ux . It checks the Address Resolution Cache. If an entry (N1,U1) is found for the network address N1, the target physical address U1 is used to send the frame.

3 - If no entry is found for N1, the gateway broadcasts on the local network requesting the datalink address for N1. It sends an ARP request broadcast packet.

4 - All hosts on the subnet will receive and examine the ARP request packet.



Routing table:
[(N1, local)
(N6, local)
(default, network)]

ARP cache
before ARP
broadcast:
[(N2,U2),
(N4,U5)]

1- the route is found.
It is local.

2- **There is no (N1,U1) pair in ARP cache.**

3- G0 broadcasts
ARP request.

4- User U1
receives the
ARP request

5- User U1 decides
to answer. Sends
ARP Reply to G0

6- G0 stores data
in cache.
ARP Cache
now contains:
[(N1,U1),
(N2,U2),
(N5,U5)]

Figure 6. An example of the Address Resolution Protocol.

5 - User U1 recognizes that the requested network address N1 is its own. It sends gateway G0 a reply with its network and datalink address (ie. U1,H1). U1 sends this to gateway G0 in the form of an ARP reply packet.

6 - Gateway G0 stores the data in its ARP cache. It transmits the awaiting frame. Gateway G0 now knows the datalink address U1 for the network address N1.

2.8 An Overview of the NET/ROM Protocol.

The NET/ROM protocol adds two distinct layers on top of the AX.25 datalink layer. The network layer is represented by what is called NET/ROM Level 3. This part of the protocol does indeed handle the routing issues. It imbeds the source and destination node callsigns in its header. It presents a datagram or connectionless service to the transport layer. The latter is known as NET/ROM Level 4. It provides a transparent stream, capable of sending data, to the end user. Both layers were designed in the early eighties with the small size of then existing packet radio networks in mind. This resulted in short-comings in the protocol that will be pointed out later.

2.9 An Overview of the TCP/IP Protocol Suite.

The TCP/IP protocol suite was developed in the mid seventies. The protocols were devised for use in WANs in which hosts were connected to (sub)networks with varying characteristics, and the (sub)networks were interconnected by gateways [Lef89]. This suite is used extensively in commercial and research networks such as the Internet. These protocols do not strictly adhere to the boundaries of the OSI model. The Internet Protocol (IP) provides most of the network services. It handles the routing of packets. IP provides an unique Internet address on a per host basis (32 bits wide). This address is used in routing issues when packets arrive at the network layer. If IP cannot correctly handle a frame, a mechanism is provided that informs the packet's originator of this. These exceptions are handled by the Internet Control Message Protocol (ICMP). Serial Line IP (SLIP) provides a way to send IP datagrams between two hosts connected via a serial line. However, IP does not provide a reliable stream as the OSI standard requires.

The transport layer is predominantly represented by the Transmission Control Protocol (TCP). TCP provides reliable streams, as well as the regular services of the layer. TCP uses so-called port-numbers (16 bits wide), which allow remote processes to communicate when the destination host (network address) and port (transport address) are known. Various network services like mail and file transfers take advantage of well-known port numbers to provide services to end-users. End-users are free to use TCP ports to their own liking.

A formal definition of IP, SLIP, ICMP and TCP can be found in [RFC791], [RFC1055], [RFC792], and [RFC793]. More descriptive work is found in [Com88].

CHAPTER 3

MOTIVATION AND PREVIOUS CONTRIBUTIONS.

3.1 The Motivation.

Since the advent of Packet Radio to the spectrum of Amateur Radio activities in the late seventies, it has gained tremendous popularity. Growth has been astronomical over the past decade. Connectivity has gone from just local activity to widespread use spanning states and countries. Mailbox systems, or Bulletin Boards, allow sending of personal electronic mail as well as bulletins via store and forward techniques. Networking and store and forward techniques have tied together users across North America and most other continents.

It is expected that in the decade ahead, the use of packet radio as a means to communicate will increase dramatically. This will result in hams trying to communicate further and faster then before. In order for the network to facilitate such increases in volume of data, it has been argued for some time that the current transport service in widespread use (the NET/ROM protocol) is not suitable for this demanding job.

Several groups in different parts of the world, but mainly in the US, have experimented with different protocols. Some networks other than the NET/ROM networks have been established. One of the fore-runners in these experiments is Amateur Radio Networking based on the TCP/IP suite of protocols. This set is in widespread use in the 'professional' networking community. Using these protocols in the amateur environment promises to provide better service in the future.

It also gives immediate compatibility with existing high speed computer networks at campuses and businesses around the US and the world.

3.2 Contributions of Others.

There currently are quite a few systems running Amateur TCP/IP, ranging from IBM PC's[2] and Macintosh's[3] to UNIX[4] based systems. Several Bulletin Board systems allow for TCP/IP based communications. Some single card computer boards are becoming available offering a wide range of functionality based on TCP/IP. Most of these are derived from or based on work done by Phil R. Karn, KA9Q.

In the mid eighties Karn started to develop TCP/IP software for use with Amateur Radio. It is now commonly known as the KA9Q Internet Protocol Package, or NET.EXE (after the program's name under the MS-DOS[5] operating system) [Kar91]. In its original form this software runs on IBM PC's and compatibles. It provides a complete implementation of most of the protocols in the TCP/IP suite. It is especially aimed at, but not solely used by, the ham community. The source code for this work is freely available for noncommercial use. This has stimulated many additions and ports to other platforms by others. My experiences in the past two years with Karn's work, and

2 IBM PC is a registered trademark of International Business Machines Corporation.
3 Macintosh is registered trademark of Apple Computers, Inc.
4 UNIX is a registered trademark of AT&T Bell Laboratories.
5 MS-DOS is a registered trademark of Microsoft Corporation.

with other means of networking on the Oregon State campus, have triggered my interests in Amateur Radio TCP/IP.

3.3 The NET/ROM Network Layer versus IP.

A comparison of the features and capabilities of the NET/ROM protocol versus the TCP/IP protocols shows that indeed the latter is more suited to serve large networks. NET/ROM Level 3, the network layer in NET/ROM, is very simple. The protocol header only contains information about the source and destination nodes. Added is a packet Time-To-Live (TTL) counter. This ensures that a packet will not indefinitely flow through the network. At the originating node the TTL counter is set to an initial value. Each node handling the packet decrements the TTL value. When it reaches zero the frame is simply thrown away.

NET/ROM routing is done based on adjacencies. Every node has a list of neighbor nodes and possible destinations. A neighbor node is a node that can be directly delivered to. Each destination has a nearest neighbor node combined with it. This is the neighbor that a packet for that destination will get transmitted to. It is the next adjacent hop in the network. This routing information is stored in a routing table. The table gets updated every half hour in a default configuration via a broadcasting mechanism. When a destination is unknown, the frame is simply discarded.

The Internet Protocol header has the source and destination address and TTL counter. However, more fields are added to provide extended capabilities. IP routing is more powerful and flexible. IP, contrary to NET/ROM Level 3, is an Internetwork Layer. Its messages can be routed through multiple logically and physically dissimilar networks [Fra89].

IP routing is flexible. IP addresses are assigned in groups or subnets. This allows a gateway to know if a destination is on the local network by simply looking at the network address. If the destination network address is in the same group as the gateway, it is a destination on the local network. This means that there is no need for a specific route to each destination on the local network. IP routing also has the ability to use gateway routes and default routes. This means that destinations in certain subnets, or groups of destinations, and unknown destinations will be sent via those routes.

An integral part of IP is the Internet Control Message Protocol (ICMP). ICMP sends back messages to a source address when problems occur in trying to deliver datagrams. ICMP messages are sent in several situations. For example, one is sent when a datagram cannot reach its destination. Network overloading can result in an ICMP message being sent to a source trying to use that network. A gateway can redirect a host to send traffic over a shorter route by sending the host an ICMP message [RFC792].

IP also allows the source to force a desired route upon the network. This is known an source routing. Further provisions have been taken to allow the source to choose the type of service it expects from the network. This service can range from normal to service aimed at low delay or high reliability. When high speed networks will be used for digital voice and real time data transport these services become important.

Lastly, but very important, IP, unlike the NET/ROM network layer, has a protocol identifier. This identifies the transport protocol in use. NET/ROM does not have this. Carrying anything other than NET/ROM transport frames over a NET/ROM network layer becomes very tedious.

3.4 The NET/ROM Transport Layer versus TCP.

NET/ROM Layer 4, the NET/ROM transport layer, is basically a sequenced packet protocol. It delivers packets in sequence. It does not combine multiple packets into one to increase efficiency. It uses transport circuits to enable multiple sessions between two nodes. However, there is not an easy way for the end-user to establish multiple connections to a distant node. The layer does not react to network load, although it does allow for a choke condition to be sent when the host is impaired due to certain conditions like memory shortages.

It is important to note that the NET/ROM transport layer does not have a data error detection scheme. The NET/ROM network layer does not provide this either. The transport layer has to completely rely on the underlying Packet Radio Network to pass data reliably. It expects the AX.25 datalink layer protocol to do error detection and correction. However, the datalink layer only provides error detection and correction between two directly connected systems. In processing packets at nodes, many processing and random errors can occur. End-to-end NET/ROM transport connections often involve a multitude of nodes. Thus a large number of independent datalink connections are involved. Thus, there are also a large number of sources of uncorrected errors. There is no end-to-end error detection scheme at all.

The Transmission Control Protocol was designed to be used in unreliable WANs. It delivers an unsegmented stream of data in which each byte can be identified by a sequence number. It can handle varying sizes of data in packets. This allows for better efficiency. Provisions to allow urgent situations to be handled are implemented. In the late eighties congestion avoidance and control techniques [Jac88] were added. The technique optimizes the round trip timing and uses dynamic window sizes. This deals with end-to-end flow control problems.

TCP allows the end-users to have multiple connections
to the same remote system. It uses the previously mentioned ports for this. This allows more time consuming sessions such as mail and file transfers to go on in parallel with interactive keyboard to keyboard communications.

It should be noted that both IP and TCP headers have checksums. IP has a header-only checksum. TCP has a checksum for both the header and the data in the packet. This provides improved reliability through end-to-end error detection.

3.5 Drawbacks.

With all the added capability, the TCP/IP protocols have drawbacks as well. The most often heard argument is that the protocol overhead, the data needed to be send with each packet to control the link, has increased. The maximum size of the data portion of an AX.25 packet is 256 bytes. On current slow Packet Radio Networks packets are often not larger then one hundred bytes. NET/ROM connections need fifteen bytes for the network layer and a minimum of five bytes for the transport layer. For a packet size of one hundred bytes, the NET/ROM protocol thus requires only twenty percent of the data.

Contrary to this, the IP layer has a minimum header size of twenty bytes. TCP minimally adds another twenty bytes. TCP/IP thus requires forty percent of a one hundred byte packet. All this results in less useful data being sent per packet while using TCP/IP. Work is being done to compress the TCP header to around three bytes, thus decreasing the overhead the protocol requires. With higher data rates the packet length will increase, and the protocol overhead will decrease. The above discussion should easily make clear that these limitations are only minor, while major improvements are obtained in other areas.

3.6 The Objective.

The goal of this work is to provide the local ham community with an inexpensive and easy means of experimenting with this 'new' networking technology. This means that some of the current set of NET/ROM protocol based network nodes must be replaced with TCP/IP based network systems. The existing solutions all require a considerable amount of financial expense, in the range of $500-$1000. These newer solutions are often based on elaborate processors like the Motorola 68000 series or the Intel 8086 series. Most newer methods provide more than IP based routing for AX.25. Most of these enhancements are not needed at the introductory stage. Sometimes the complexity that comes with the multitude of options can scare users away. It is proposed to create a simple solution an order of magnitude less expensive than previous solutions.

As an initial and often introductory step with a low financial threshold, the proposed solution has a certain charm. It allows for an easy switch to TCP/IP based nodes. The end-user (the ham at home) still needs a computer to run the host software. Since most hams involved in packet radio already posses one and since the host software is freely available, this poses no problem. Once acceptance of the new protocols has been accomplished and enthusiasm has grown, upgrading to more capable systems will proceed more easily.

It is useful to be able to exploit the presence of the local Ethernets or other professional TCP/IP based networks. By doing so some aspects of a big, globe spanning network can be demonstrated. This could accentuate to the ham community the possibilities laying ahead for Amateur Packet Radio. Thus a solution is proposed that will accommodate both a stand alone TCP/IP based node system, as well as a simple but effective interface to professional networks.

The proposed system will have the following characteristics:

1) A system will be designed to provide a simple implementation of IP routing of AX.25 packets.
2) the system has a radio port as well as a serial port. The radio port will provide the AX.25 capabilities. The serial port will use SLIP to send packets to a connected unit.
3) the system interfaces to the existing base of transmitters in the same way as do the current NET/ROM based nodes. This allows for a simple replacement of the current node system.

4) the software will be written in the 'C' programming language. This will allow the code to be easily portable to other hardware platforms.
5) the total cost will be less then one hundred dollars. Regularly available parts will be used. This makes it possible for Ham operators to replicate the design.

Use of the SLIP protocol on the serial port is useful. It is a good way to send frames from one controller to another. It also provides a simple way to interface with commercial networks. Most modern Unix systems and others provide built in support for the use of the SLIP protocol. Thus no modifications to the operating system of the Unix machine are needed. It suffices to simply attach a controller system to a serial port on such a machine. Then the machine and the controller both have to be configured correctly. This then will allow users to access and exchange data with the Unix machine via the radio interface and vice versa. When the Unix host is connected to a network and has setup proper routing, targets on other networks might be reached as well.

CHAPTER 4

HARDWARE DESIGN DETAILS.

4.1 The Controller.

Several alternatives were examined. With the eye on simple and affordable hardware, the choice was a derivative of the industry standard Intel 8051 micro-controller. The 83C152 Universal Communications Controller (or C152) is an 8-bit microcontroller based on the 80C51BH architecture. Among other features, it is enhanced with a high speed multi-protocol serial communication interface. Two channels for Direct Memory Access (DMA) transfers are added [Int89]. Standard features are a serial port, an internal scratch pad memory for quick parameter access and a separate external data memory and program memory address space of 64 kilo bytes each. This made the 83C152 an attractive choice.

Very little external hardware was needed. An external driver for the RS-232 signals on the serial port was required. The communication controller needed a radio modem discussed later. These were the major peripherals needed. This resulted in a very compact system. A block diagram of the controller and the peripherals is shown in Figure 7.

4.2 The Radio Modem.

In data exchange, the modem is part of the physical layer of the OSI model. As such it is considered to be outside the scope of this work. Therefor a commonly used existing modem design was used in the design. The Texas Instruments TCM 3105 provides the current

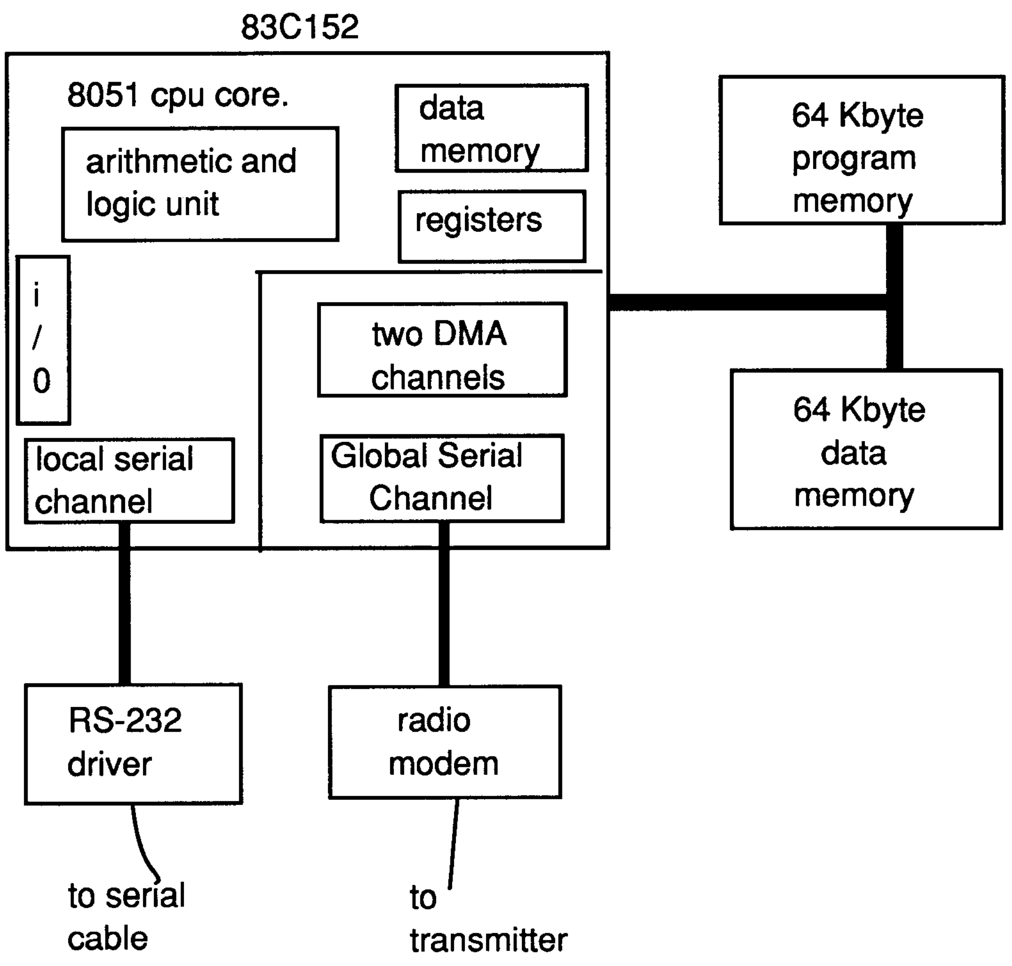


Figure 7. The system block diagram.

1200 Bd half duplex standard. It provides for simplicity and compatibility with others experimenting in the area.

The interface between controller and modem is identical to that of other existing (higher speed) packet radio modems. Change of modems requires no software change at all. The current hardware base will allow speeds up to 9600 Baud. This is considered the next step after 1200 Baud. All this requires is simply attaching a different modem design (and radio).

4.3 The Global Serial Channel.

The Universal Communication Controller of the C152 is known as the Global Serial Channel, or GSC. It provides most of the standard features used in HDLC or derived protocols (like AX.25). It does error checking on frames and provides the bit-stuffing scheme used. Automatic 1 or 2 byte address checking can be used. This allowed for a partial callsign check in hardware. Thus a large number of packets that are not addressed to the system would never get handed off to the controlling software. Doing so released the software from processing large numbers of frames that need be discarded anyway.

4.4 The DMA Channels.

The 2 DMA channels available can be programmed to service the GSC. This allowed for fast transport of data to and from memory. This took over much of the work normally done by the controller in software. Thus the controller could spend more time handling frames in the queues instead of servicing the radio channel.

4.5 The Local Serial Channel.

The RS-232 serial line was provided by the Local Serial Channel, or LSC. The LSC is a full duplex asynchronous serial transmitter and receiver. It is register mapped in the cpu internal memory space. As such it is very easy and fast to use. The LSC could be serviced by DMA channels as well. Encapsulation issues in the SLIP protocol, see [RFC1055], limited the servicing of the LSC to software however.

4.6 Additional Hardware.

Very little additional hardware was required. External code memory and external data storage memory was added. These need a few simple logic elements. Four Light Emitting Diodes, or L.e.d's, were used to indicate the status of the transmit and receive functions on both ports.

Schematics of the resulting hardware can be found in the Appendix. The wire wrapped prototype board has a six by four and a half inch footprint. A picture of the prototype board is shown in Figure 8.

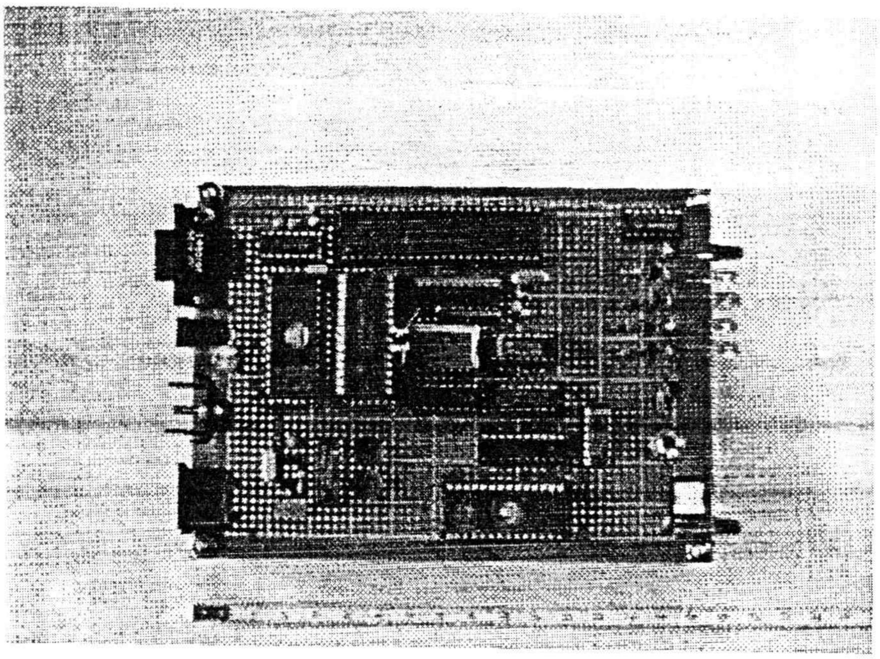


Figure 8. The System prototype.

CHAPTER 5
SOFTWARE DESIGN DETAILS.

5.1 Performance Requirements in Gateways.

Gateways are expected to accept and forward traffic from one network to another without slowing down the operation of either network [Tan88]. High volumes of traffic on both links could cause a heavy workload for the cpu. For maximum efficiency, the software must be carefully organized. The choice of operating system for the gateway software was very critical.

The tasks performed in a simple gateway are all event driven. This means that unless something external happens the system is simply idling. The arrival of a character on an interface triggers the system to react and store it in a buffer. Once a complete frame has been received, action will be taken.

5.2 Design Considerations.

On one hand, all action could be interrupt driven. With this approach, each received frame would be processed to completion in the interrupt service routine. This is very fast, but has several drawbacks. It could lead to received characters being dropped on other interfaces. This is due to the possible uninterruptable state of the presently serviced frame. Also it results in poorly structured software.

On the other extreme, all the tasks in the gateway could be implemented in different processes. Each process has its own address space and can communicate with other processes via so-called signals.

Signals can be sent to inform other processes of things such as frame-arrival, timer time-outs etc. Frames can be passed on to other processes as needed. Sophisticated operating systems can use time slices designating the maximum time a process is allow to run. This can be combined with a scheduling algorithm that decides what process gets to run next. The operating system needs to keep track of process states, stacks, signals and more. Implementing this requires a lot of overhead and complexity.

5.3 The Operating System.

Both operating system solutions mentioned had features that made them unattractive or impossible to implement on the C152. The interrupt driven version was hard to implement due to a lack of enough interrupt priority levels in the controller. This meant that the running interrupt would become virtually un-interruptable. When this happens it becomes hard to guarantee processor time to other interrupt driven processes. Other processes contending for the processor might be transmit and receive interrupts on other interfaces as well as clock-tick interrupts. This could cause errors such as loss of received characters, with obvious impact.

A full blown operating system as described in the second suggestion was impossible to implement. This would need a separate stack for each process, something not provided by the 83C152. The overhead needed to implement this in software would be a very hard performance penalty.

Instead, a mixture of the two was chosen. A simple 'round robin' design was implemented. The functionality of the gateway was split up in two logical parts. The first part is the low level processing of frames. This concernes the interrupt handling of receive and transmit frames. These service routines run independently from the main program (in the background of the controller.) There are four different functions to be serviced: transmit / receive on the serial interface and transmit / receive on the GSC interface. Each function has a queue of awaiting packets. When a complete frame has been received it is deposited on its corresponding input queue. The two output queues are periodically checked for content. If a queue is not empty a transmission is initiated.

The second part was the high level frame processing. This was the actual gateway software that makes the routing decisions. This part can also easily be divided up into functional blocks according to the protocol layers used. Each module looks at a particular input queue. If the queue is not empty, a frame is taken off and processed to completion. This can result in the frame being sent out on an interface. When an error occurs an error message might be sent back to the originator of the frame. All modules get to process frames in a round robin fashion. A graphical explanation of the operating system can be found in Figure 9.

Protocol processing consist of three modules: the AX.25 receive module, the SLIP receive module and the ARP check module. All modules check queues and act if frames are found. The IP routing, ARP resolve, and AX.25 and SLIP transmit modules get activated in the process of handling these frames. These modules and the memory management strategies used are all described in the next sections. (The interaction of the different modules is shown in Figure 11.)

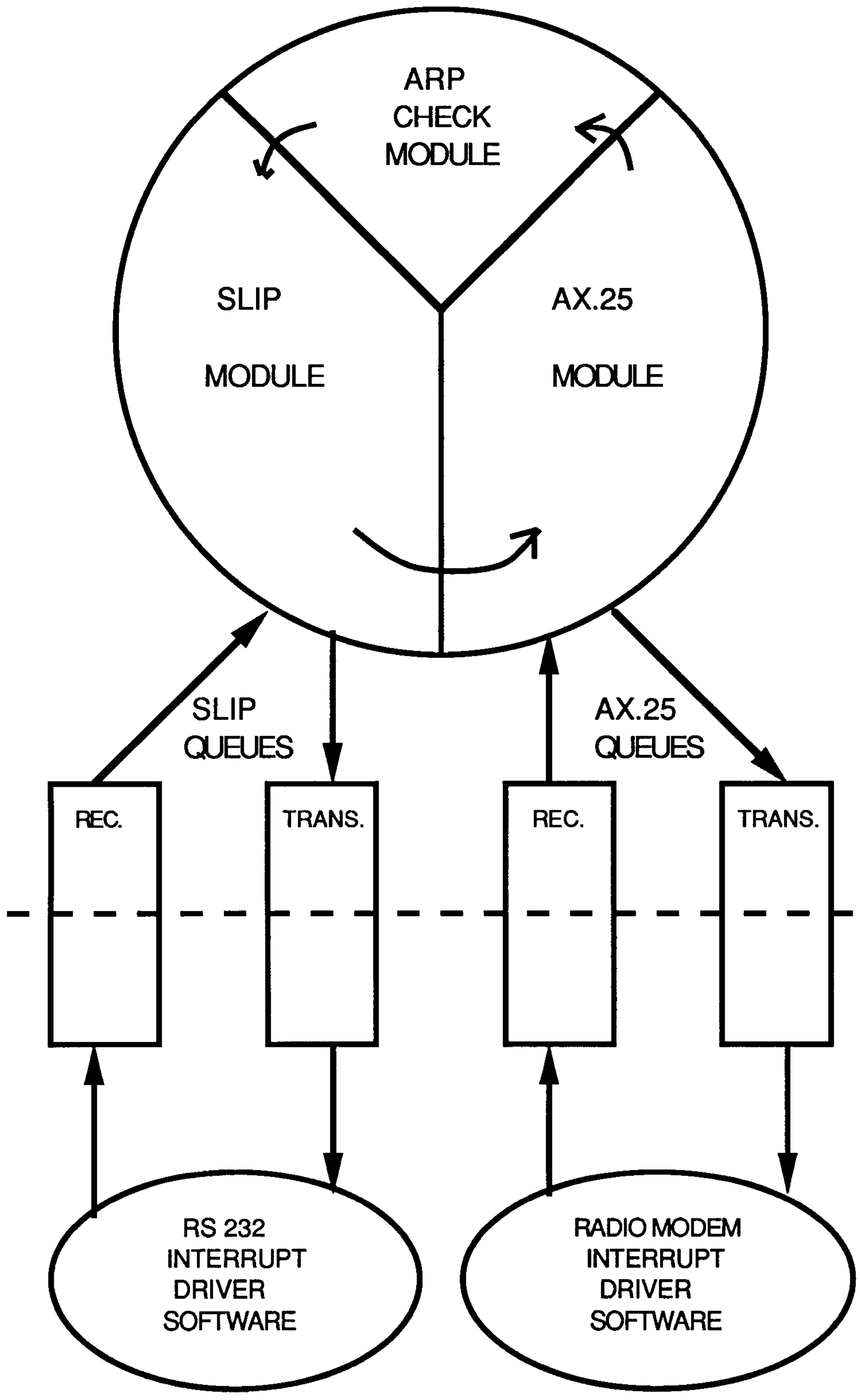


Figure 9. Software layers in the operating system.

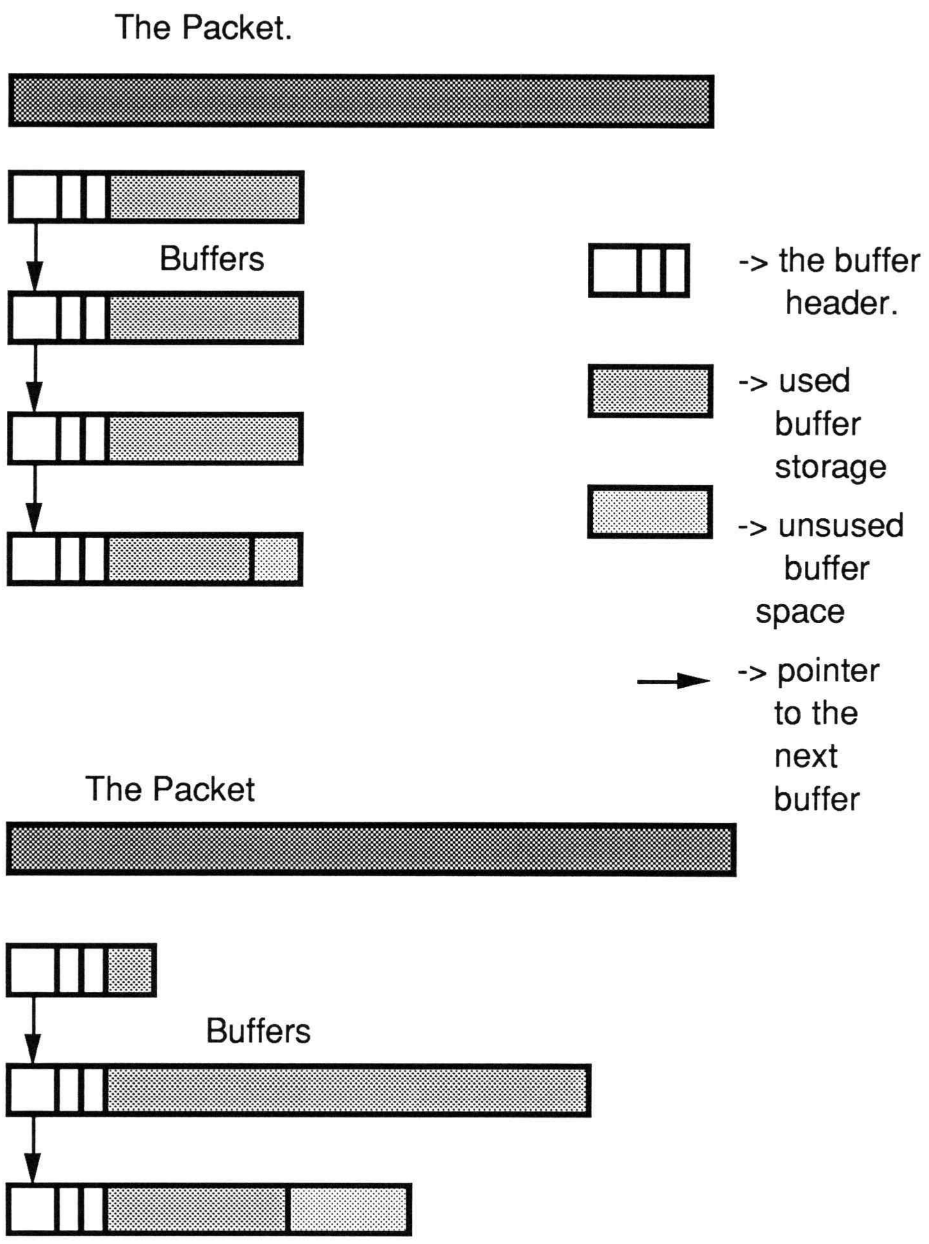
5.4 Memory Management.

Memory management strategies for network protocol operating systems tend to be quite different from those of other operating systems. Input buffers with data often are handed off to several different modules for processing. Protocol modules strip their headers off the frame and send the data part on to higher level modules. Higher layers in turn can prepare data for transmission. They can put their headers in front of the frame. Copying a buffer containing a frame each time it is handed off is very expensive. Processing time is costly and memory gets filled with multiple copies of the same information.

Received frames often vary drastically in size. Interactive terminal sessions might only send a few bytes at a time. File transfers however could send several thousands of bytes per frame. Thus if buffers are allocated according to the maximum size frame possible, lots of space will be wasted when a short frame is received. Often the maximum size of a received frame cannot be predicted. Some alternative scheme of handling buffers needs to be used.

One approach to the buffer size problem is to use a pool of small identical size buffers. These buffers can be linked together to form one large buffer. Using fixed size buffers still presents the problem of wasted space. Short frames much smaller than the buffer size waste buffer space. Another approach is to use variable size buffers. The advantage here is better memory utilization, at the price of far more complex buffer management [Tan88]. Figure 10 illustrates the use of both identical size buffers and variable size buffers to form a frame.

A) Use of fixed size buffers to store the packet below.



B) Use of varying size buffers to store the packet above.

Figure 10. Buffers and Packets.

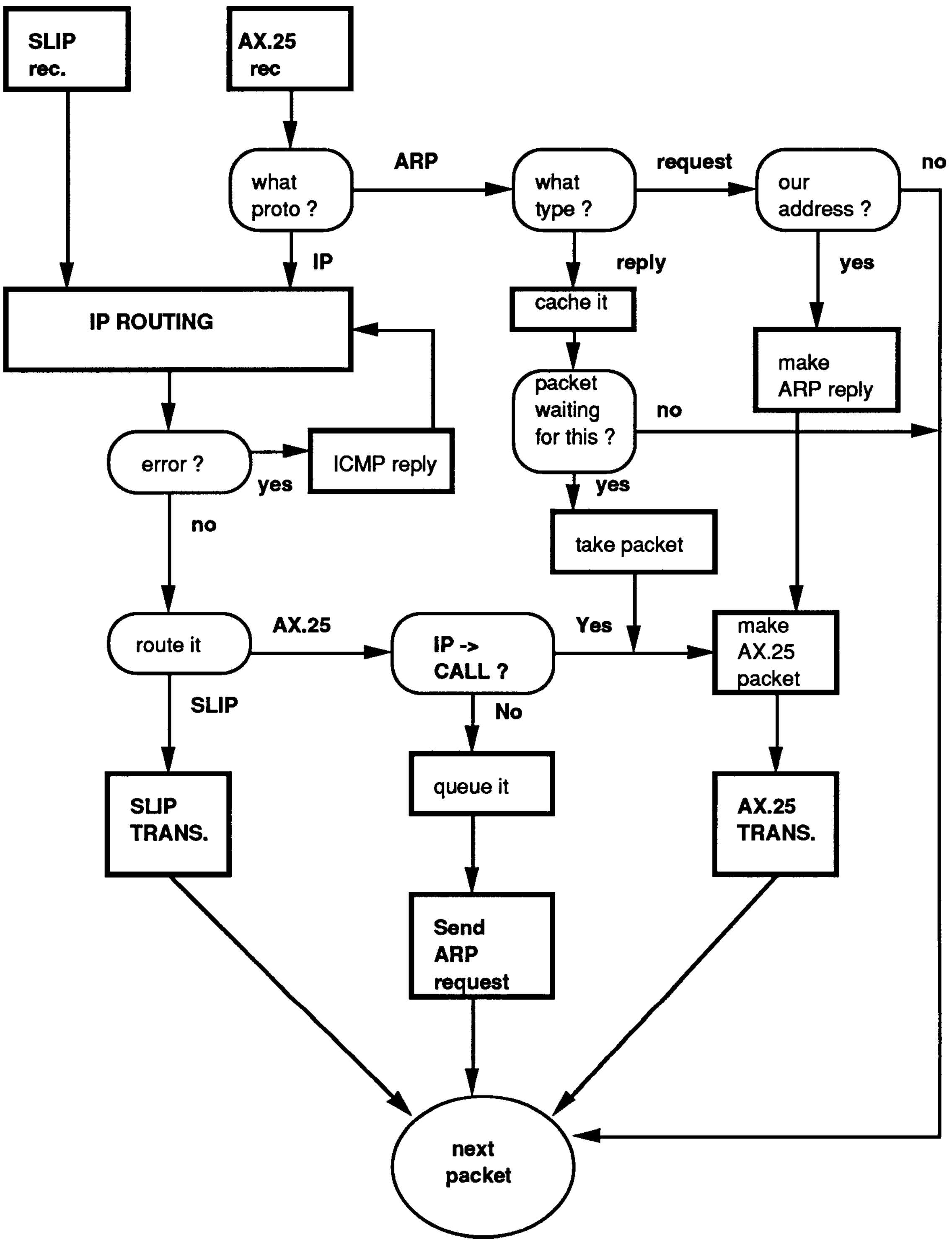


Figure 11. Flow chart of the processing of a received packet.

The BSD Unix operating system introduced so-called mbufs, or memory buffers. They are small fixed size buffers as described above. The problem of passing frames was gracefully solved. When a module needs to pass a frame (ie. a series of mbufs) to another module, no physical copy is made. Instead a pointer to the frame is handed over and a reference count for the buffers is increased. When a module is done processing a frame the reference count to the buffers is decreased. When the reference count reaches zero the buffers can be freed and used again later. Offset and length variables are introduced in each mbuf. This allows for efficient trimming of data at the start or end of the mbuf. In deletion of data at the start of the mbuf, the offset is incremented and the length is decremented. In deletion at the end, the length is simply decremented. When space is available in an mbuf, data can be added at either end. These features particularly help in implementing communication protocols efficiently.[Lef89]

The buffer management scheme chosen was a direct copy of the mbuf strategy. A fixed buffer size of 256 bytes was used aligned at the start of memory pages. This facilitated the use of some features of the 83C152 architecture regarding the addressing of external data memory. Instead of using the regular data pointer provided, the faster page-mechanism was used (see [Int89] for more details.) Most buffer management routines were written in machine code for better performance.

5.5 The AX.25 and SLIP Receive Modules.

The AX.25 receive module takes a packet from the AX.25 receive queue. The complete destination call is checked since there might be multiple stations active on the radio channel. If addressed to this system, the AX.25 datalink layer control headers is examined. If an IP network layer frame has been encapsulated, it extracts it and sends it to the IP processing module. If an ARP type packet is received, this is handed off to the ARP resolver module. Presently any other type of datalink layer frame is discarded.

The SLIP receive module takes frames from the serial line receive queue. SLIP is a point to point service and has no datalink layer. It simply passes received frames on to the IP routing module.

5.6 The IP Routing Module.

The IP module processes the protocol header of a frame. It checks for validity of the header checksum. If any errors are found during processing, the frame is discarded. Then the header destination is checked. If the frame is not destined for this system, a route will be looked up.

It was decided to implement static routing. This means that all routes had to be known at system configuration time. There were drawbacks to this approach. Network changes that cause routing changes would not be responded to automatically. In that case, new routes would have to be programmed into the system. The advantage was the ease of implementing static routes. IP allows the notion of subnet routing. This allows a group of destination addresses to be represented by one route entry in the routing table. This type route could be used for the users on the subnet reached via the radio channel. Thus no entry for each individual user was needed. Routing over radio frequencies, with their special characteristics, is very challenging. There still is development under way on routing protocols aimed specifically at Amateur Packet Radio. These arguments made static routing an acceptable choice.

The frame to be transmitted will be fragmented into smaller parts if the Maximum Transmission Unit (MTU) of the appropriate interface is smaller then the frame size. During this the IP protocol header is reconstructed. Then the fragments are handed off to the SLIP or AX.25 transmit modules.

A gateway need not implement host protocols (ie. transport layer and above.) At present the only higher level protocol implemented is the ICMP Echo reply service (see [RFC792],[RFC1009].) Other frame types destined for the gateway will cause an ICMP error message to be sent back to the source. No reassembling of fragmented frames was implemented. This was not in accordance with [RFC1060], the guideline to implementing a gateway. It did however simplify the problem significantly. For the purpose of this system not reassembling fragmented frames had little effect on functionality. It simply meant that ICMP Echo request frames had to make it to the system without being fragmented on their way. When no special options are used (as in most cases) an ICMP Echo request frame is 28 bytes long (plus possible data to be echoed back). These frames will not get fragmented by gateways.

The IP module has very little knowledge of the frames' path to it. It only knows what interface the frame originated on. This was used to prevent routing loops on the SLIP interface. If a frame came in on the SLIP interface the routing decision could be made to send it out on the same interface. It was then likely the frame would keep bouncing back and forth due to bad routing. Therefore frames with this property were discarded.

5.7 The SLIP Transmit Module.

The SLIP transmit module is not concerned about link layer addresses. It simply puts the IP frame on the SLIP transmit queue.

5.8 The AX.25 Transmit Module.

The AX.25 transmit module handles frames handed down from the IP network layer. It has to send the IP frame to the proper destination. A translation from IP address to datalink layer address (or callsign) has to be done. This is done by the previously described ARP protocol.

First the in-memory ARP cache is checked for existence of the IP destination address. If there is a hit (ie. the entry is there) the found callsign is used. A proper AX.25 packet is formatted. This is queued on the AX.25 transit queue.

If the cache check produces a miss (ie. no entry found) dynamic ARP-ing is engaged. An ARP broadcast is sent out requesting the datalink call for the IP address in question. The frame is queued at the tail of the ARP wait-queue. The IP header information, a relative timeout value and number of tries is added to the queued information. Processing will now be followed-up by the ARP check and ARP resolve modules. The module then returns as if transmission succeeded.

5.9 Timing.

There was a need for a relative time value. Gateway requirements [RFC1009] prescribe a timer resolution in milliseconds from midnight. Another short cut was taken here. A global system time value was implemented. With a tick each fifty milliseconds and a sixteen bit time value, almost an hour was needed for time to overrun and duplicate. This was considered long enough not to worry about duplicate time values. Packets queued up waiting for responses would not be around nearly this long.

Each time the timer ticks, the two transmit queues are checked in turn. If interrupt driven transmission is in progress, nothing is undertaken. If there is no activity, the queue is examined. If a queue contains frames ready for transmission, transmission of the next frame will be started. Interrupt service routines take care of the transmission of the complete frame once transmission is started.

5.10 The ARP Resolve Module.

When an ARP frame has been received the ARP resolve module is called from the AX.25 receive module. There are two reasons why this can happen. First is when a ARP request packet from some other system is received. If this is the case, the ARP request information is checked. The requested destination IP address might be one the system is using. If so, an ARP reply packet is formed. This reply packet contains the information about our IP address and our requested datalink address, ie. the callsign. This packet is sent to the requesting host by adding it to the AX.25 transmit queue.

A second type ARP packet received can be an ARP reply from some other system. This means that some information about translation from IP address to AX.25 address is available. The information about the destination IP address and AX.25 callsign is extracted from the packet. This pair is added to the ARP translation cache. If the cache is full, the oldest entry is overwritten.

Next the ARP wait-queue is checked. There could be frames waiting for this IP to AX.25 translation. These frames are dequeued and handed off to the AX.25 transmit module. Since the ARP cache has now just been filled with the needed address translation, the frame will immediately get sent as an AX.25 packet.

5.11 The ARP Check Module.

The ARP check module checks the ARP wait-queue. This contains all frames waiting for a translation of IP network address to AX.25 link address. Entries that have expired timeout values get processed. Since the entries are in sorted order, not all entries have to be checked. When the first entry with valid timeout value is found processing can stop.

An expired entry results in two possible actions. If the retry value is still positive, a new ARP broadcast will be sent. Again the IP to AX.25 address translation for the target IP address is requested. The timeout value is reset and the retry value decremented by one.

If the retry value is zero there has not been a timely reply to the broadcast requests for address translation. This means that the IP frame can not be sent using this gateway. Therefore an error message has to be sent back to the originator of this frame. The address of the originator is obtained from the queued IP header. An ICMP Destination Unreachable Message is sent. The queued frame is discarded.

5.12 Configuring the System.

The system has quite a few parameters. The most important parameters are mentioned here. At present all are set at compile time. This makes all parameters completely static. They are located in unchangeable permanent memory. They include the AX.25 callsign of the gateway. The IP address of both the radio port and the serial port are stored here. A list of local AX.25 callsigns and their IP addresses can be preloaded. These entries will permanently be in the Address Resolution Cache. This allows the address translation of active stations to be permanently known by the gateway.

The stored routes are important. They are also statically stored. They need to be known beforehand. Routing table entries consist of pairs of IP addresses and interface numbers. Routes have a network mask. The network mask is the previously discussed group identification or subnetwork id. Possibly the address of the next gateway is associated with a route. Since routing entries are static at present, the routing table has to be filled with caution. Improper entries will cause lots of problems. They might result in users not being able to use the gateway to connect to remote networks. They could cause packets unnecessarily to be re-transmitted several times by a set of gateways. This is a routing loop and should be avoided.

End-users will have to designate the gateway in their system setup. They need to have knowledge of both the AX.25 callsign and IP address of the gateway. They will have to send all packets that can not be delivered on the local network to the gateway for routing.

When all this has been properly configured, communication based on Internet Routing of AX.25 packet is possible.

CHAPTER 6

SUMMARY AND CONCLUSION.

6.1 Results.

A microcontroller system was designed and constructed. Using commonly available parts, a compact system was designed that is compatible with currently used Ham Radio equipment. The system is easily reproducable by other interested Hams

Software was written to implement the Internet Protocol based routing of AX.25 packets. Most of the software was written in the 'C' programming language. This provides for good portability to other hardware platforms. Where performance was critical, the software was written in the controller's assembly language. The Franklin C51 DOS based cross-compiler package was used.

The software implementation of the proposed system had a compact code size of thirty kilobytes. One kilobyte of external data memory was used for variable storage. Sixty-three kilobytes were usable for frame buffering.

The system was tested in several situations. A test was run using a 1200 Bd radio modem and transmitter connected to the radio port. The SLIP port of the system, running at 19.2 kBd, was connected to a PC running KA9Q's NET.EXE, also providing SLIP. The radio communicated with a transmitter on another port of the PC. The AX.25 radio modem on this port was a 'Pocket Packet'. The unit is commercially available from Heathkit, U.S.A. Inc. This setup in effect created a routing loop over the radio channel.

The system behaved well. However, the nature of well-behaving communication protocols is such that they do not try to force network throughput. Well-designed communication protocols will not overload a network. On the contrary, when they notice an congestion situation (for example, due to missed acknowledges) they slow down their rate of packets to be transmitted. When it comes to dealing with overloading, the Transmission Control / Internet Protocols are especially well-behaved. They will not overload a network. Thus in order to evaluate the performance of the system under load a simple mathematical analysis was needed.

A good measure of performance in communication protocols is the throughput under load. A simple analysis, using a worst case approach for average size packets, is performed. The results are shown in Table 1.

	packet time	packets/ second
1200 Bd radio port	950 ms	1
9600 Bd radio port	110 ms	9
19200 Bd serial port	52 ms	19
processing	14 ms	70

Table 1. Packet throughput rates.
(for 100 byte packets)

Frame processing latency times could be calculated from the source code. The time it takes for a frame to be dequeued from the SLIP receive queue to enqueuing on the AX.25 transmit queue is calculated. This particular processing was chosen because it is the most elaborate. Processing time is around fourteen milliseconds.

The calculated time assumed that the ARP resolver found the datalink address needed in the cache. If this was not the case the latency time increased drastically. Then the frame had to wait for the ARP request to be answered. At present speeds, this could be in the order of five to ten seconds. Latency time also depends on the IP options used.

Next, input rates on the two interfaces were calculated. The calculations assumed a packet length of 100 bytes. The serial interface has a maximum speed of 19200 Bd. Each byte has ten bits (eight data bits, and a start bit and stop bit). Thus a packet takes fifty two milliseconds to arrive.

The radio interface has a speed of either 1200 Bd or 9600 Bd. Apart from the time to transfer the data, there is a key-up delay time. This is the time between start of transmission and start of actual data. This delay is needed to facilitate slow receivers and modems. At 1200 Bd the key-up delay is around 350 milliseconds while at 9600 Bd it is around thirty milliseconds. These times were approximations since there are a lot of possible configurations. The total time it takes to transfer a 100 byte packet is shown in Table 6.1.

The controller can process a maximum of 70 packets per second. Worst case load will be a continuous stream of input packets on both interfaces. This situation should not occur, since this represents one way data flow with no acknowledgements. This will result in the reception of 19 packets on the SLIP port, and 9 on the 9600 Bd radio port. A total of 28 packets per second. The controller can handle this load.

At present market prices, the hardware cost for the controller is around seventy five dollars. This does not include a printed circuit board. It does include the 1200 Bd modem chip. If operation at higher baud rates up to 9600 Bd is needed, a separate modem is required.

6.2 Future improvements.

Possible future improvements could include dynamic routing. Dynamic routing would allow the routing tables in the system to dynamically change as the network changes. When new gateways are established the current system needs to be re-programmed to reflect the new status of the network. A simple implementation of the Internet Routing Information Protocol would enhance functionality. A more radio specific routing protocol such as RSPF, the Radio Shortest Path First routing protocol, would also be useful when the network size expands.

A more complete implementation of the Internet Protocol might be attempted. Improvements can be made in the IP option processing, such as source routing and route recording. Also system timing can be improved to allow for a better IP time stamp option implementation.

Remote parameter access will provide a mechanism to change the system parameters without having to re-program it. It will allow for remote changing of the route table entries, ARP translation entries, IP addresses of the system and more. When the network characteristics change, changing system parameters to reflect this will be more easy.

An implementation of PPP, the recently developed Point to Point Protocol, could also be useful. This would allow the serial port to be used for more than just SLIP. It would facilitate multiple protocols, such as IP and AX.25 , running together over the serial link.

6.3 Conclusions.

Presented here is a controller for Internet Protocol Routing of AX.25 packets. The proposed solution here is simple and effective. Some limitations are the static routing, and a limited implementation of the options provided by the Internet Protocol. Remote configurability of system parameters would be helpful.

Advantages of the controller are multiple. It provides IP based networking instead of NET/ROM based networking. The design is simple and has a low cost. The system is easy to reproduce and is simple to interface to existing radio's. It allows the ham-community to build a Packet Radio Network based on TCP/IP. When there is access to a 'professional' network, the controller will allow for an easy interface between the professional network and the Packet Radio Network. It allows hams to experiment with for them new technology.

BIBLIOGRAPHY

[Ber87] Bertekas, Dimitri; Gallager, Robert; *Data Networks*, Prentice Hall, Inc., 1987.

[Com88] Comer, Douglas; *Internetworking with TCP/IP, Principles, Protocols, and Architecture*, Prentice Hall, Inc., 1988.

[Fox84] Fox, Terry L.; "AX.25 Amateur Packet-Radio Link-Layer Protocol, Version 2.0", ARRL, Inc., 1984.

[Fra89] Frank, Daniel M.; "Transmission of IP Datagrams over NET/ROM Networks", *8th ARRL CNC Proceedings*, ARRL, Inc., 1988.

[Int89] *8-Bit Embedded Controller Handbook*, Ch. 11, Intel, Inc., 1989.

[Jac88] Jacobsen, Van; "Congestion Avoidance and Control," *Proceedings of the ACM SIGCOMM Conference*, August 1988.

[Kar91] Karn, Phil R.; *The KA9Q Internet Software Package*, Public Domain Software, 1991, available by anonymous ftp from thumper.bellcore.com

[Lef89] Leffler, Samuel J.; McKusick, Marshall K.; Karels, Michael J.; and Quarterman, John S.; *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley Publishing Company, Inc., 1989.

[RFC791] Postel, J.; ed., "Internet Protocol - DARPA Internet Program Protocol Specification," RFC 791, USC/Information Sciences Institute, September 1981.

[RFC792] Postel, J.; "Internet Control Message Protocol - DARPA Internet Program Protocol Specification," RFC 792, USC/Information Sciences Institute, September 1981.

[RFC815] Clark, D.D.; "IP Datagram Reassembly Algorithms", RFC 815, MIT Laboratory for Computer Science, July 1982

[RFC826] Plummer, D.; "An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48-bit Ethernet Addresses for Transmission on Ethernet Hardware," RFC 826, MIT Laboratory for Computer Science, November 1982.

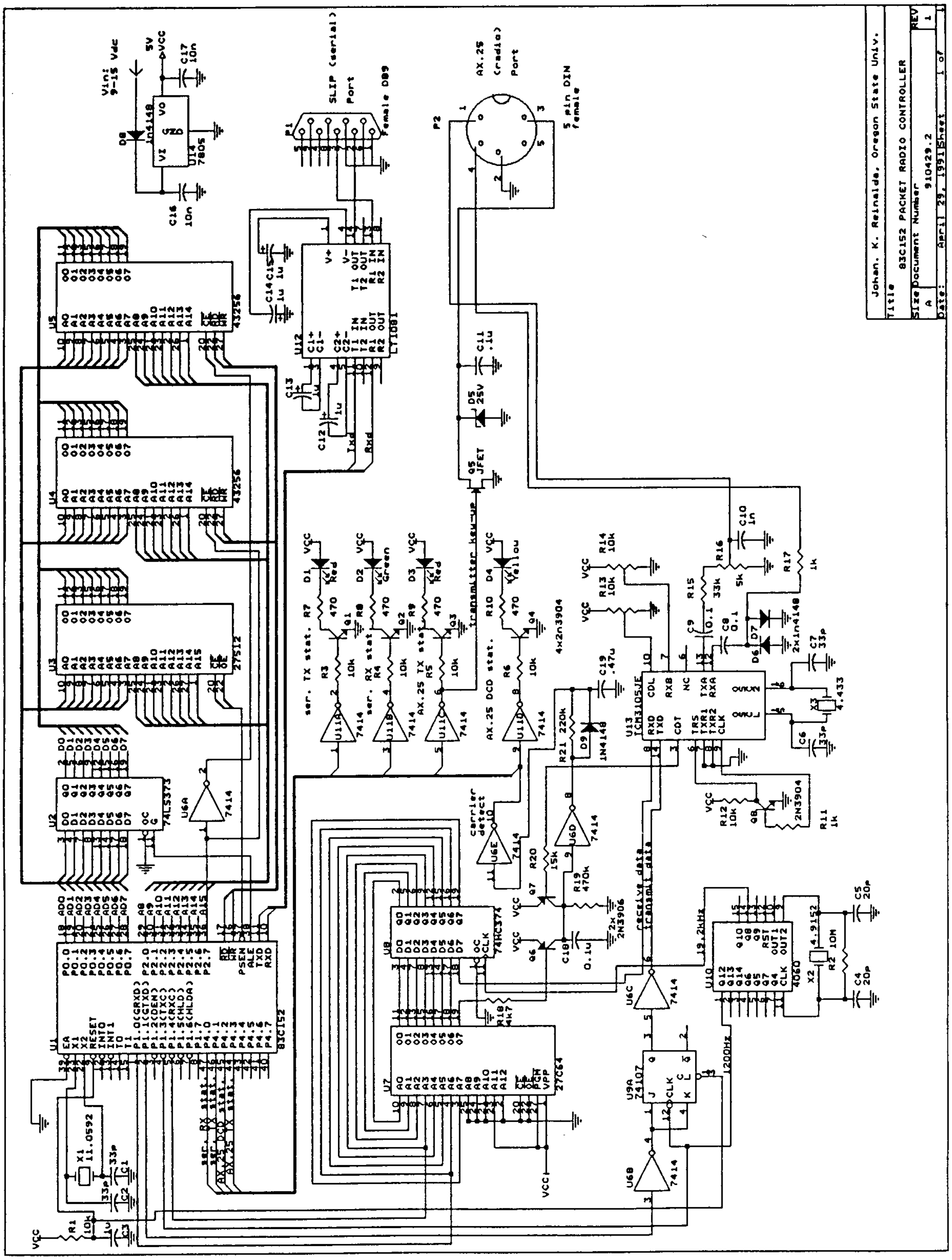
[RFC1009] Braden, R.T.; Postel, J.B.; "Requirements for Internet Gateways," RFC 1009, Internet Network Working Group, June 1987

[RFC1055] Romkey, J.L.; "Nonstandard for transmission of IP datagrams over serial lines: SLIP", RFC 1055, Internet Network Working Group, June 1988

[Tan88] Tanenbaum, Andrew S.; *Computer networks, second edition*, Prentice Hall, Inc., 1988

Appendix

APPENDIX The controller schematic diagram.


Johan. K. Reinalda, Oregon State Univ.
Title
83C152 PACKET RADIO CONTROLLER
Size Document Number REV
A 910429.2 1
Date: April 29, 1991 Sheet 1 of 1
SLIP (serial) Port
Female DB9
AX.25 (radio) Port
5 pin DIN female
Vin: 9-15 Vdc
5V VCC
U14 7805
U1 83C152
U2 74LS373
U3 27512
U4 43256
U5 43256
U7 27C64
U8 74HC374
U10 4060
U12 LT1081
U13 TCM3105JE
U9A 74107
X1 11.0592
X2 4.9152
X3 4.433
Q5 JFET
transmitter key-up
carrier detect
receive data
transmit data
19.2kHz
1200Hz
ser. TX stat.
ser. RX stat.
AX.25 TX stat.
AX.25 DCD stat.
4x2n3904
2x1n4148
2x2N3906
Txd
Rxd