DESIGN AND IMPLEMENTATION OF AN APPLETALK LOCAL AREA NETWORK BRIDGE USING PACKET RADIO

R. Ramsey and W. Kinsner, VE4WK*

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, MB, Canada R3T 2N2
E-mail: VE4WK @ VE4BBS.MB.CAN.NA
and
*Microtel Pacific Research Ltd.
Burnaby, BC, Canada V5A 4B5

ABSTRACT

This paper presents the design and successful implementation of a local area network bridge based on the link layer AX.25 packet radio protocol and the AppleTalk Personal Network on the Macintosh computer. Operated as a duplex communication channel between interconnected networks of computers, the packet radio system provides the transmission of the network layer data packets. A working prototype of the bridge was developed for slow rates. A higher-speed bridge will require a faster packet radio and faster hardware for the bridge.

1. INTRODUCTION

Local area network (LAN) technology has brought about an evolution in computer communications by sharing data and resources among a large number of users in localized areas. Now, this LAN concept is becoming too restrictive and larger (national and international) networks are required. To facilitate the creation of such large networks, internetwork connections are used to transfer data between the networks and provide data translation for different systems. An internetwork connection is called a bridge if the LANs are homogeneous, or a gateway if they are inhomogeneous. These interconnections can be made through either telephone lines, fibre optics, radio frequencies (HF, VHF and UHF), microwave terrestrial links, or satellite links. One of the affordable interconnects could include packet radio over the amateur bands. This would allow a number of interesting applications, including long distance education.

This paper presents the design and implementation of a bridge based on the AX.25 amateur radio packet protocol [1 and 2] and the AppleTalk Personal Network [3] as used on the Macintosh computer [4]. AppleTalk has been selected as an example of a practical implementation of the International Organization for Standardization (ISO) Open System Interconnect (OSI) model [5], permitting interactions between computers and peripherals either on a single network or on different nets interconnected through bridges and gateways. Firstly, the requirements of a local area network bridge are presented, and two solutions are proposed: (i) a single-connection internet bridge, and (ii) a multiple-connection bridge capable of interconnecting multiple networks simultaneously. Next, the implementation of both the bridge and the interfaces created between the bridge, together with the packet radio network and the AppleTalk network are presented. Finally, a testing procedure and preliminary results are discussed.

2. THE BRIDGE DESIGN

2.1 The AppleTalk Network to Bridge Interface

The interface between an AppleTalk network and the bridge process consists of system routines (calls) developed by Apple to perform input and output on the network, and a buffer manager to store the packets from the network in local memory. Involved in the interface specified for the AppleTalk network is a link layer communication protocol used to enable the AppleTalk hardware to transfer data over the network in an organized and error free manner. The AppleTalk Link Access Protocol (ALAI?) is used for the link layer transmission which allows the network hardware to (i) route the packets to an appropriate node, and (ii) recover the packets from the network.

The transfer of the packets from the network is carried out by first receiving the packets from the network and then storing the packets in memory. The packet receiving task is provided by a protocol handler which is an interrupt driven routine responsible for retrieving the data from the AppleTalk hardware (MPP driver) and placing the data in a buffer provided by the bridge. To facilitate the required asynchronous and rapid reception of packets, a swinging buffer mechanism [6] must be employed to reduce the possibility of packet overrun caused by slow removal of full packets by the bridge. The fist five bytes of the packet include the ALAP header (the destination node, the source node, and the type of packet), as well as the size of the Datagram Delivery Protocol (DDP) which is responsible for the internetwork communication.

The operation of the **protocol handler** is very simple and essentially involves the rapid transfer of data from the AppleTalk network hardware to main memory. Upon reception of the first five bytes of an ALAP packet, an interrupt is generated by the MPP driver and the protocol handler is invoked. The protocol handler first verifies that the packet was received correctly. If the Frame Check Sequence (FCS) is validated correctly, the first five bytes are then transferred into a buffer provided by the bridge. Next, an asynchronous read is issued which reads the remainder of the packet into the buffer as it arrives. The protocol handler finally indicates that the buffer is completed by setting the pointer to the buffer to NULL. If the protocol handler finds the buffers not empty, the pending ALAP packet is discarded, and the higher-level protocols used in the network become responsible far the retransmission of the lost data.

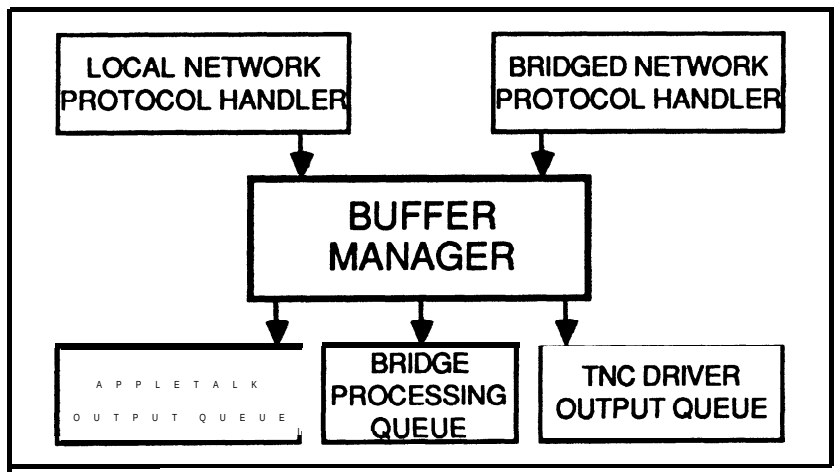

Fig. 1. The data flow path of the packets in the bridge.

As shown in Fig. 1, the sole purpose of the buffer manager is to maintain empty buffers

for the protocol handler while transferring completed packets onto one of three input queues for later processing. The three input queues are: (i) the AppleTalk queue (used for transmission of packets on the local network), (ii) the bridge processing queue, and (iii) the remote network queue (connected through the packet radio network, as described in Section 2.3).

The requirement of the bridge to execute as efficiently as possible dictates the method of transferring data from the protocol handler to the queues. Since buffer transfers require a large amount of processing time, the implementation of the buffer manager requires that only pointers to the buffers are moved to the queues while the actual packets remains intact in memory. Furthermore, new buffers are allocated to the protocol handler during noninterrupt processing periods to avoid memory manager conflicts and system inconsistencies.

2.2 The Packet Radio Network to Bridge Interface

The packet radio to bridge interface has no formal definition in any available sources. Therefore, the main guide to follow during development is to provide a functionally equivalent interfaces to both the local AppleTalk network and the remote network connected through the packet radio network. Similar to the AppleTalk interface, a protocol handler is employed to retrieve the ALAP packets from the packet radio network. The data transmitted over the radio link is organized into packets according to the AX.25 link layer protocol. As shown in Fig. 2, this system is also very amenable to the use of digipeaters in relaying packets between the network bridges to extend the minimum area of coverage by the packet radio link. Global interconnection of LANs can be achieved through this implementation since satellite repeater stations also exist on the amateur bands.

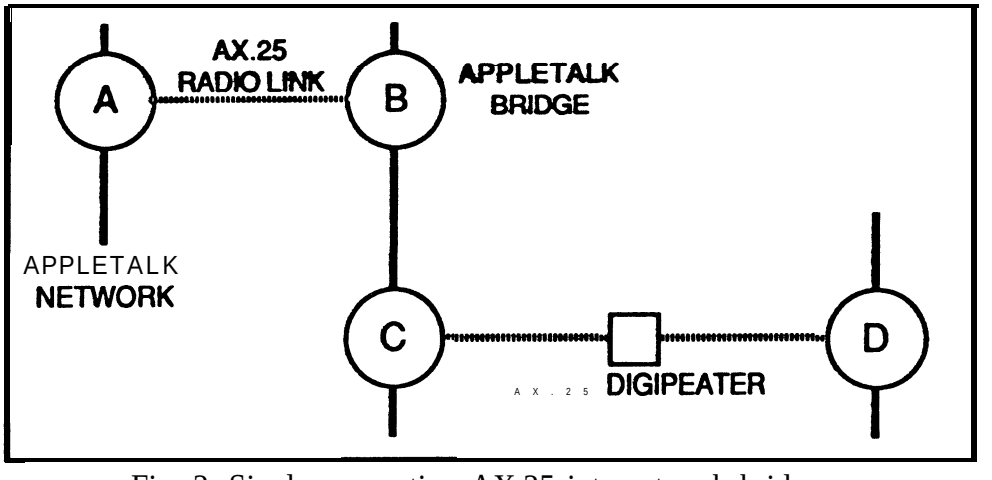

Fig. 2. Single-connection AX.25 internetwork bridges.

The AX.25 protocol operates at the link layer of the ISO-OSI model and is used to communicate the AppleTalk network layer packets between the interconnected networks without contributing any modifications to the AppleTalk network layer data. In this way, any protocol could have been employed by the packet radio link without posing any problems at the network layer, but the AX.25 is a reliable communication protocol with guaranteed delivery, thus nearly eliminating the probability of network errors, increasing the overall efficiency of the bridge through fewer AppleTalk packet transmissions.

The interface to the packet radio network to the bridge consists of the transfer of the AppleTalk packets to and from an unmodified terminal node controller (TNC), such as the all-mode KAM [7]. The TNC is responsible for the assembly and disassembly of the AX.25 packets.

Error free transmission and packet ordering is maintained by the TNC according to the protocol. This implementation removes the burden of maintaining the AX.25 protocol from the host processor, thus allowing a faster execution of the remainder of the bridge process. Using this implementation, the number of AppleTalk networks which can be connected by one bridge is limited to two (see Fig. 2). The reason for this limitation is that the TNC supports only a single connection facility at a time.

The packet radio protocol handler is also an interrupt driven process. In this case, however, the TNC is connected to the Macintosh computer through the serial port. The interrupt is created when the serial drivers receive data from the TNC over the serial port. To enable the reading of an ALAP packet, system *read* calls *are performed* on the serial ports to receive a specified amount of data. When the data is received, an interrupt is created and a completion routine of the system call is executed. The bridge reads the data from the TNC by first issuing an asynchronous *read* command to obtain the header of the ALAP packet. When the header is received, the size of the packet is known, and a subsequent call can be made to read the remainder of the packet. The two read calls are invoked through the interrupt completion facility of the Macintosh computer.

The swinging buffer mechanism is employed again for the packet radio interface. Also, the buffer manager described above for the AppleTalk network interface is responsible for the creation of new buffers for the packet radio protocol handler as well the transfer of packet buffers from the protocol handler to the three input queues.

2.3 Bridge Processing and Protocol Support

The main components of the AppleTalk network bridge involve (i) the routing of packets between AppleTalk networks, and (ii) the maintenance of the network protocols. The bridge process consists of a dispatch loop, calling each of the specific tasks that provide packet routing and protocol support.

The dispatch routine is responsible for the management of the bridge process. In simple terms, the dispatcher is the main program of the bridge process, and consists of an infinite loop executing the required operation on the packets. In each cycle of the dispatcher, the following routines are executed: (i) the buffer manager, as described above, (ii) the queue manager, and (iii) the timers and window management processes.

The queue manager routine, *CheckQueues*, monitors and processes the packets on the three input queues. There are also three output queues: AppleTalk queue, internet queue, and bridge processing queue. They are checked once through each dispatch loop, with equal processing on each queue. The packets found on the AppleTalk queue are transmitted on the AppleTalk network through a *LAPWrite* system call, while packets on the intemet (packet radio) queue are transmitted on the packet radio network by sending the ALAP packet to the TNC through *a TNCLAPWrite* call. The third queue, the bridge processing queue, contains the packets which require protocol processing or which supply routing information to the bridge through the Routing Table Maintenance Protocol (RTMP).

In order to facilitate homogeneity in the network, the bridge protocol processes must be designed according to the protocol definition outlined by Apple. The entire set of the AppleTalk protocols for OSI Layer 3 (network layer) through Layer 4 (transport layer) were implemented in the bridge. These protocols included the DDP, the AppleTalk Transaction Protocol (ATP), the RTMP, the Zone Information Protocol (ZIP) , the Echo Protocol (EP), and the Name Binding Protocol (NBP).

2.4 A **Bridge with Embedded AX.25 Protocol**

The above discussion of the packet radio bridge interface assumed the use of the packet assembler/disassembler (PAD) of the **TNC.** An alternative packet radio communication link incorporates an implementation of the AX.25 protocol as an integrated component of the network bridge software package. In this implementation, the TNC is used only as a modem and a transmitter activator for the radio. The full AX.25 protocol is implemented in the bridge package. Each half bridge would be able to create numerous connections to other half bridges on the network, reducing the amount of inter-AppleTalk network routing that would have to take place. This reduction in routing would reduce the amount of traffic on the air and throughput of the internetwork could be increased.

A dynamic routing table is employed to map the AppleTalk networks to the logical connection established between the bridge processes. One socket on each bridge implementation acts as a connection acceptor so that dynamic connections are possible. Once a connection is established, a secondary socket is opened, and the logical connection is maintained between the calling socket and the secondary socket. The ***listen*** socket automatically updates the network routing table when a connection request is obtained from the packet radio network. The AX.25 network table is interfaced to the AppleTalk routing table such that dormant nodes can be eliminated. Figure **3** shows an example of a possible connection scheme using the AX.25 embedded bridge. As in the single-connection intemet implementation, a **digipeater** system could be used to extend the transmission distance of the bridges.

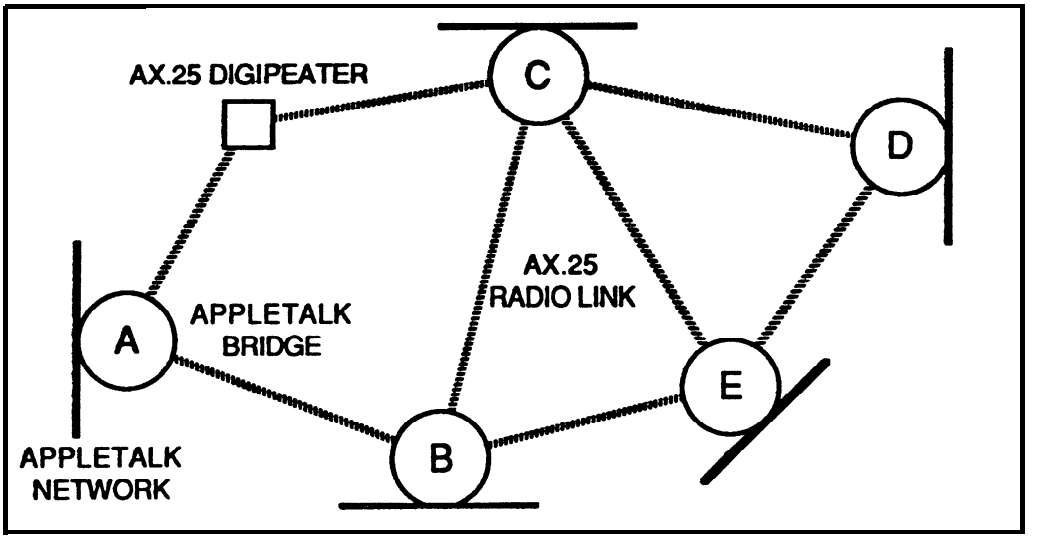

Fig. 3. Multiple-connection AX.25 intemetwork bridges.

3. THE PACKET RADIO INTERFACE IMPLEMENTATION

This section describes the implementation of the packet radio to bridge interface employed in the AppleTalk bridge. The code is in Lightspeed C and the overall listing takes 68 pages **[8].** As shown in Section 2.2, the interface is provided as the access point to the serial drivers on the Macintosh computer, using the protocol handler **and buffer** manager. Two distinct phases of operation of the packet radio are involved in the AppleTalk network bridge: (i) link establishment and (ii) packet transfer. The operation and implementation of the connect and data transfer phases

is described below.

3.1 Connection Establishment Phase

The connection establishment sequence is used to create a logical connection between the two halves of the AppleTalk network bridge. To allow for dynamic connections, two modes of connection establishment must be defined: (i) passive connection, and (ii) active connection. In active connection establishment, the bridge process initiates a connect request to another bridge process. The bridge process to which the connection establishment is addressed is operating in passive connection sequence. This process waits for connections on an AX.25 port and upon receiving a request, establishes the connection.

When the bridge is first started, the user is required to enter the amateur radio call sign of the desired network or a default call sign to place the bridge in passive connection mode. The **TNC** connect routine is then executed to open and initialize the serial drivers. Following a successful driver opening, a synchronize routine is executed. The purpose of the synchronization is to establish coordination between the bridge software commands and the execution of the 'INC.

If a connection is requested by the user, the call sign of the remote bridge process is encoded into a connect command. The request is then sent to the TNC, and the system then enters an asynchronous wait for acknowledgement phase. If the bridge has been initialized to perform a passive connect, the same routine is executed, as described above for connection acknowledgement. In order to indicate a connection has been established, the TNC returns ****Connected to* ***xxxxxx,*** where ***xxxxxx*** is the call sign of the remote AppleTalk **network** bridge.

Upon reception of the connection acknowledgement, the TNC is placed in transparent mode for packet transmission. In the transparent mode, the TNC transmits all data without modification. All interbridge communications are performed while the TNC is in transparent mode. The TNC remains in the transparent mode until a disconnect request is issued and the TNC receives the *BREAK* signal. Any error encountered during the connection process causes the bridge process to enter the passive connection mode to wait for incoming connection requests. The passive connection routine executes through the interrupt and completion routine mechanism supplied by the operating system.

The disconnect sequence is executed either prior to closing the bridge down or when it is detected that the other half of the bridge has been terminated. The disconnect routine is responsible for forcing the TNC back into command mode, disconnecting the connection to the other bridge, and closing the serial ports. The initiation for termination of a bridge connected to a dormant network is supplied by the RTMP protocol through the network aging process.

3.2 Packet Transmission Phase

The major operation of the packet radio interface involves the transmission of the **ALAP** packets between the bridged networks. For this communication, the serial drivers are employed to communicate with the TNC and packet radio network. In order to communicate with the TNC, the ports have to be configured properly. This operation is provided by the ***SerialOpen*** function which opens the input and output serial drivers and initializes the communication rate and data size.

The two read routines responsible for receiving the packets from the TNC, ***TReadSize*** and ***TReadRest,*** form the packet radio protocol handler. Operated as asynchronous read calls, the interrupt driven completion routine facility of the operating system allow for asynchronous reception of the **ALAP** packets from the packet radio network. ***TReadSize*** is responsible for

reading the size of the next ALAP packet. This routine reads the first five bytes of the packet. Upon completion, an interrupt is generated, and the *TReadRest* routine is called to read the remaining bytes of the packet. Again, on completion of reading the entire ALAI? packet, another interrupt is generated and the *TReadSize* routine is executed to read the size of the next packet coming from the bridged network. The packets received are placed in the swinging buffer mechanism described in Section 2.2, and the buffer manager stores the completed packets in memory and supplies the routines with new buffers. If no empty buffer exists when a new packet is received, the packet is discarded by reading it into a dead buffer and a *LostPacket* counter is incremented to record system performance.

The **write routine** used to transmit data to the bridged network, *TNCLAPWrite*, closely resembles the operation and calling syntax designed for *the LAPWrite* described in the Macintosh standard. An *ABusRecord* is passed to the write routine which contains the information such as the pointer to the data block, the size of the data to write, and the port reference number to access. At the end of this routine, the completion routine is executed which returns to the user supplied *ABusRecord* the actual number of bytes written and the error status created through the write request. Only a single outstanding write is allowed through *TNCLAPWrite*. If a connection is not established, the write returns success. This operation is required to allow the bridge process to continue the execution of its heart beat transmission for the RTMP. The fields of the LAP header on the packet are completed before the packet is transmitted

Two other routines, *TNCWrite* and *TNCRead*, have been created to support transmission of ASCII text to the TNC for connection establishment and termination. Furthermore, a *SerialClose* procedure is executed to terminate all communication requests over the serial port and to shut down the serial input and output drivers. In order to remove all outstanding read and writes on the operating system queues, a *KillIO* call is required. The *KillIO* call is required because outstanding reads after a driver close are not terminated and the bridge system would not be restartable without performing a warm rebooting of the computer.

4. SYSTEM OPERATION AND TESTING

4.1 Bridge System Verification

The bridge protocols were verified in a test setting consisting of a serial link interconnecting two networks A and B, as shown in Fig. 4. One computer on each network, a **bridger,** was dedicated to executing the bridge software. The serial ports of the two bridgers were connected together, while the network ports were connected to the corresponding AppleTalk network bus. Three other computers were situated on each network to run utility programs capable of testing the bridge system. One computer on each network was **used** as a **packet sender.** The packet sender computer executed a utility program called POKE packet which has the ability to format and transmit any AppleTalk packet. The third computer on each network was a **packet watcher,** in that every packet sent on the network was monitored by this program. The utility program PEEK was executed on the packet watcher node. This utility is essential in verifying that the bridges were responding with the correct data. The fourth computer ran the **throughput test** procedure described in Section 4.2. The AppleTalk protocols implemented on the bridge were tested first.

4.1.1 Verification of RTMP and ZIP

The RTMP protocol is used for maintaining the routing table. To do this, a “heart beat” packet is transmitted by the bridges at a ten-second interval, and an RTMP packet is expected from every active bridge on the intemet within the same time interval. The testing of this protocol can be best completed by first starting up one network, then starting up the other bridge a short time later.

The RTMP packet sent on the network contained only the routing tuple of the local Network B, (network address $4000). When the second bridge is started, a new routing tuple for Network A (network address $3000) is observed. The second network is then shut down. After 40 seconds, the routing tuple for Network A disappears. This is due to the aging mechanism of RTMP so that a dynamic routing table can be maintained without the need of transmission of a shut down packet. The ZIP packet is transmitted only once at the initial connections of the two packets. After this first ZIP request, both networks know the zones of each other and no further ZIP packets are required.

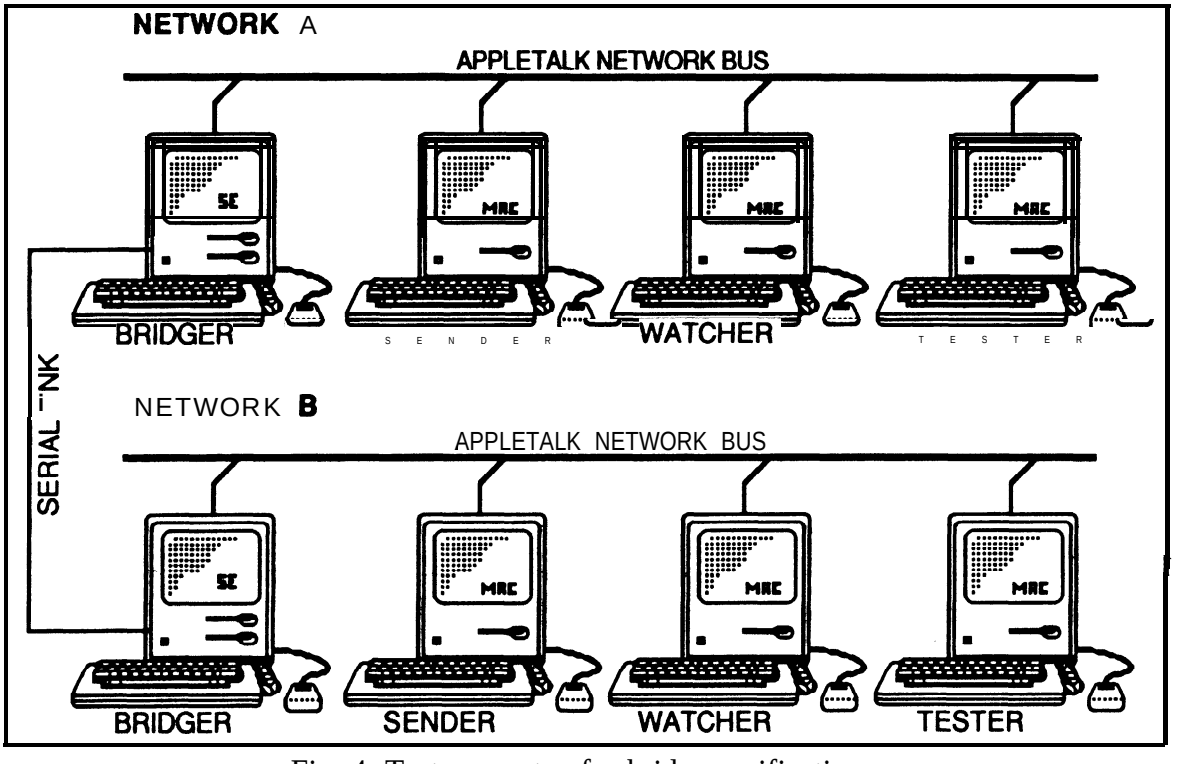

Fig. 4. Test apparatus for bridge verification.

4.1.2 Verification of **NBP**

An NBP request packet was transmitted from a node on Network B. The bridge process of Network B sends a look up packet to Network A. Network A does a broadcast on its own network to obtain the name. The node with the name being looked for then sends a directed packet to the requesting node on Network B.

4.1.3 Verification of Echo Protocol

Two types of echo protocols exist for short and long echo packets. The *short echo packet* is sent out on the local network. In the test, the bridge node was sent an echo packet from one of the packet transmitters. The first packet marked with a ZONE 1 in the data field is the initial packet transmitted. The second packet is the echoed packet. The test shows that the source and destination addresses, 1F and 83 are reversed in the echo return packet. This also occurs in the transmission of the *long echo packet* except that in this case the packet is sent across the bridge. The returned echo packet has both the source and destination node numbers and network numbers reversed.

4.2 Throughput Tests and Discussion

One of the major factors in the operation of a packet radio bridge is the expected throughput. On the packet radio link, the transmission rate is 1200 bits/sec. This, however, is a simplex communication channel and, thus, the system is limited to 1200 bits/sec in one direction at a time. Recall that the transmission rate of the AppleTalk Network is 230 kbits/sec. If we take the case when only one network is transmitting across the channel with no return traffic, we get a ratio of 230 to 1.2. Therefore, the amount of data which must be queued on a busy network could easily ovefflow the memory of the computer and cause a system crash. This large difference between queue input and output rates will cause large delays to occur in the networks operation.

Let us obtain the best-case time during a one way transmission by sending a packet on a clear network. The one-way delivery time, T, is calculated as the sum of the times of each leg of the trip and can be expressed as

$$T = BN \sum_{j=1}^{L} (1/R_j) \qquad (1)$$

where B is the number of bytes in the packet, N is the number of bits in a byte, L is the number of legs in a packet trip, and R_j is the data transmission rate on the jth leg in bits/sec. For example, the best-case time required to transmit a 100-byte packet from a node on network A to a node on network B through the packet radio bridge operating at 1200 bits/sec is

$$T = 100 * 8\ (1/230000 + 1/1200 + 1/230000)$$
$$= 674 \text{ msec} \qquad (2)$$

It is seen that the bulk of the communication time is spent in the bridge communication section of the link. This problem can be overcome by increasing the speed in the packet communication link [9 and 10]. Another method of increasing the number of bytes transmitted per second is by employing a data compression technique. The advantage of the latter technique is the bandwidth preservation.

During the initial testing, a large number of buffer overruns were observed on a busy AppleTalk network. Because of this, the dispatcher was modified to execute the buffer manager multiple times in each dispatch cycle. This modification decreased the number of discarded packets, but increased the number of packets backed up in the processing queues. This backing up of the processing queues may continue until a saturation point is reached, past which the number of discarded packets increases until the queues begin to empty again, creating available resources for the bridge.

A test has been devised to determine the average round trip transmission time required to send a packet. A test utility program outlined is executed on one of the computers, with all other computer being idle. The program then transmits a user specified number of packets using the echo protocol. The time of transmission is stamped onto the packet when it is transmitted. Upon receiving the return packet, the time is compared and the round trip delay is obtained. This test is very important to study possible improvements to the bridge.

We have transmitted text and other objects such as pictures over the bridge. Interactive graphics can be achieved between different LANs.

5. CONCLUSIONS

The implementation of a packet radio bridge for local area network is currently possible although quite limited. The major limitations occur as a result of the low speed transmission rates available on the packet radio communication link. The AX.25 Link Level protocol employed in amateur packet radio provides a strong data transmission medium in which packets transmitted are guaranteed to reach their destination, a necessity for efficient internetwork communications. Based on the present work, the following extensions are proposed: (i) Study into the throughput problems of the network bridge must be undertaken to suggest areas of improvement; (ii) For a general purpose bridge, an on-board implementation of the AX.25 protocol should be employed; and (iii) Dedicated hardware, as opposed to the general-purpose Macintosh computer, should be employed to increase processing speed while reducing hardware costs.

ACKNOWLEDGEMENTS

We thank the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Department of Electrical and Computer Engineering at the University of Manitoba, and Apple Canada for partial financial support of this work. Interactions with Rob Bueckert are also acknowledged.

REFERENCES

[1] T.L. Fox, ***AX.25 Amateur Packet-Radio Link-Layer Protocol.*** Newington (CT.): ARRL, 1984, 39 pp.

[2] T. Fox, "Proposed AX.25 Level 2 Version 2.0 changes," in *7th Computer Networking Conf.*, Newington (CT): ARRL, 1988, pp. 58-64 (Columbia, MD; 1 October 1988).

[3] Apple , ***Inside AppleTalk.*** Cupertino (CA): Apple Computers, 1987.

[4] Apple, ***Inside Macintosh.*** Volumes I to V. Cupertino (CA): Apple Computers, 1987.

[5] A.S. Tanenbaum, ***Computer Networks***. Englewood Cliffs (NJ): Prentice-Hall, 1981,517 pp.

[6] W. Kinsner. ***Microprocessor Interfacing***. Course Notes. Dept. Electrical & Comp. Eng., Univ. of Manitoba, Winnipeg, MB, Canada. 1988,337 pp.

[7] Kantronics, ***Kantronics All Mode Communicator KAM***. Version 2.85. Lawrence (KS): Kantronics, 1987 and 1989.

[8] R. Ramsey, R. Bueckert, and W. Kinsner. ***AppleTalk Bridge Using AX.25 Packet Radio: Software Listing.*** Technical Report. Dept. Electrical & Comp. Eng., Univ. of Manitoba, Winnipeg, MB, Canada. 1989, 70 pp. (available from VE4WK).

[9] M. Chepponis and P. Kam, "The KISS TNC: A simple host-to-TNC communications protocol," in *6th* ***Computer Networking Conf.***, Newington (CT): ARRL, 1987, pp. 36-43 (Redondo Beach, CA; 29 August 1987).

[10] M. Chepponis and B. Mans, "A totally awesome high-speed packet radio I/O interface for the IBM PC/XT/AT/386 and Macintosh II computers," in *7th* ***Computer Networking Conf.***, Newington (CT): ARRL, 1988, pp. 36-40 (Columbia, MD; 1 October 1988).