Copyright © 1985, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

A GENERAL ARCHITECTURE FOR DATA LINK LAYER CONTROLLERS

by

Randy Cieslak and Ayman Fawaz

Memorandum No. UCB/ERL M85/103

(Protocol Workroom Document No. 85-4)

12 December 1985

A GENERAL ARCHITECTURE FOR DATA LINK LAYER CONTROLLERS

by

Randy Cieslak and Ayman Fawaz

Memorandum No. UCB/ERL M85/103

(Protocol Workroom Document No. 85-4)

12 December 1985

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A GENERAL ARCHITECTURE FOR DATA LINK LAYER CONTROLLERS

Randy Cieslak and Ayman Fawaz

Protocol Workroom Document No. 85-4

ABSTRACT

The Protocol Workroom is a facility used for studying the performance of communication protocols. It has a number of node emulators, each consisting of a 68010-based single board computer and a 68020-based board that functions as a data link layer controller. For experimental purposes, the boards communicate over a channel emulator capable of emulating various network topologies such as a broadcast channel or a ring. The node emulators are also connected to a Multibus along with a SUN computer and a file server. Experiments for the facility are controlled from the SUN.

An important part of the design is the design of the link layer controller board. It must be capable of implementing various link layer protocols. In this report, we study a number of commercially available link layer controllers and evaluate the Protocol Workroom design based on its ability to implement the functions of the commercially available boards. The board can implement their functions, and its flexibility allows experimentation with many different protocols.

A GENERAL ARCHITECTURE FOR DATA LINK LAYER CONTROLLERS

by

Randy Cieslak and Ayman Fawaz

Memorandum No. UCB/ERL M85/103

(Protocol Workroom Document No. 85-4)

12 December 1985

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

A GENERAL ARCHITECTURE FOR DATA LINK LAYER CONTROLLERS

Randy Cieslak and Ayman Fawaz

1. INTRODUCTION

In the past decade a great deal of work has been done in the area of communication networks. The use of local area networks such as Ethernet and larger networks such as the ARPA network has increased. New technologies such as optical fibers are being applied to the telephone network, and there is an effort to expand the services offered over the telephone network; cable television is an example of such a service. An important aspect of this work is the development of protocols.

It is often difficult to analyze the performance of protocols in a network. It is expensive to reserve a network and computers soley for the purpose of experimentation. Simulation of a network with many nodes can be very slow because of the complexity of the simulator. Analytical models can ignore many important details.

Another approach is to build a network for the purpose of experimentation that can emulate a network under study and is flexible enough for use in a variety of experiments. This is the approach used in the Protocol Workroom (PW) facility [1]. A diagram of the facility is shown in Figure 1. It consists of a number of node emulators, a channel emulator, a SUN computer, and a file server.

Each node emulator consists of two parts: an MC68010-based single board computer made by Pacific Micro Systems and a custom board based on the

MC68020 with programmable special purpose state machines. The MC68020-based board performs the functions of the data link layer [2]. The Pacific board performs the functions of higher layers, often lumped together as the client layer. The Pacific board can be used to emulate various digital devices such as a computer or a printer.

The channel emulator can be programmed to emulate the behavior of various network topologies. It is implemented in logic, presenting digital signals at the interface with each node, thereby eliminating the need for signal decoding and encoding. A sequence of commands from the SUN programs its configuration. The configuration can be changed in real time to model the effect of faults or noise. Nodes can communicate by means of point to point connections or on a bus, with programmable delays placed arbitrarily in the connections. More details can be found in [3].

Experiments are controlled from the SUN, and data collected from an experiment is stored on the file server. The architecture of the monitoring system is similar to that of the Metric system developed at Xerox [4]. At each node emulator, protocol-related events from both the data link controller and the Pacific board are recorded and stored in the Pacific board's memory. The SUN reads the memory of the Pacific boards through the Multibus in a way that does not interfere with the processing done on the Pacific board. Intermediate processing may be done at the SUN, and then the data is stored on the file server. Data analysis programs on the SUN access the data and summarize the results.

An important part of the design is the design of of the data link controller. It must be flexible enough to implement a variety of data link protocols. In this report, we study various comercially available link layer controllers and evaluate the PW design based on its ability to emulate the commercially available boards. Section 2 gives a description of the board. Section 3 looks at Ethernet

controllers made by 3COM and Excelan. Section 4 considers the Pronet controller made by Proteon. Section 5 looks at the DMR11 controller made by Digital Equipment Corporation. Section 6 highlights the strengths and weaknesses of the PW board and discusses future work.

2. DESCRIPTION OF THE PROTOCOL WORKROOM LINK LAYER CONTROLLER

The data link layer controller is a custom designed board based on the MC 68020 and programmable state machines. It serves as an interface between the Pacific board and the channel emulator. The 68020 is the executive on board. It supervises the operation of the other components on the board, and it performs computations associated with the protocol being implemented. It is also responsible for communication with the host and the collection of monitoring data.

Besides the 68020 microprocessor, the board consists of programmable state machines used for pattern recognition and fast decision making, and of dedicated pieces of hardware for packet transmission, packet reception, and event recording. The main idea behind the link layer controller architecture is the use of dedicated hardware to relieve the board processor from real time constraints. Figure 2 gives a general overview of the board architecture.

2.1. Communication Between the Pacific Board and the Data Link Controller

This operation is controlled by interrupt signals used to inform the destination processor about the presence of valid data. All control message and data exchanges between the two boards take place through mailbox memory realized with dual-port RAM. Any processor willing to send information to the other one stores the data in the dual-port RAM and sends an interrupt signal to the other processor to notify it.

The dual-port RAM is divided into different mailboxes. Two are used to buffer packets: the first is dedicated to the packets to be transmitted, and the second to the received ones. Two others contain monitoring information: one is associated with the transmitted packets, and the other with the received ones. The buffer mailboxes are also used to exchange control messages. This partition is taken into consideration by the interrupts service routines and will prevent any consistency problem in the shared memory.

Two memory locations in the dual-port RAM are used to indicate the type of interrupt request to the receiver of an interrupt. One of these locations is associated with each processor.

The 68010 controls the reset signal of the 68020. This feature is implemented because during the initialization phase, the 68010 modifies the configuration of the link layer controller board and therefore must control it.

The transfer of information takes place through a private bus connecting the two boards. This bus is similar to a system bus connecting the CPU to its peripherals.

2.2. Channel Emulator Interface

The controller board is designed to interface the channel emulator. The interface consists of several lines in parallel. Two of them are data in/out lines, and the rest are used for control and timing signals. Among these, there are:

- signals to indicate valid data on the data line
- clock signals that are used by the controller hardware for transmission and reception
- global clock signals used for a global time stamp used for monitoring.

2.3. Programmable Finite-State Machines

There are programmable state machines to control transmission, reception, and event recording. They monitor the channel and perform pattern recognition and fast decision making. All these functions are highly dependent on the communication protocol implemented in the network and are user definable. These functions regulate the access to the channel by following the protocol. They consist of looking at signals from the channel emulator, identifying them, and, depending on the state of the node, making decisions regarding packet transmission and reception.

The rest of the board communicates with the state machines by setting latches that are used to define the state of node. The state machines send interrupt signals to the other subsystems with some encoded information describing the events that are occurring, and based on which some specific actions are taken.

There are two state machines. The first one identifies the control signals or the patterns sent on the channel, while the second one makes the correct decision associated with the current state of the node and the result of the pattern recognition performed by the first one.

2.4. Transmission

Transmission is controlled by the 68020 and the state machines. A transmission occurs in the following way. A packet is generated and stored in mailbox memory by the 68010 on the Pacific board. An interrupt signal from the 68010 to the 68020 informs the latter about the presence of the packet. A dedicated piece of hardware is used to transfer data from the memory to the channel. This piece of hardware is controlled by the 68020 and the state machines, and it starts the transfer when it receives a signal from the state machines.

After the transfer of the last byte in the packet, the data transfer hardware sends a signal to the state machines to notify them. The state machines put the required signals on the channel, and inform the 68020 about the successful transmission attempt. In case of an error signal, like a collision in an Ethernet, for example, the state machines stop the data transfer, take the necessary actions, and inform the 68020. The 68020 resets the dedicated hardware to start a new transmission attempt. The 68010 is notified of the outcome of a transmission attempt and can collect monitoring data associated with it.

A CRC calculation can be performed in hardware while transmitting the packet, and the checksum is appended to it.

2.5. Reception

Packet reception is performed in a very similar way to packet transmission. Whenever the state machines detect an incoming packet, they send a signal to a dedicated piece of hardware that is used to transfer data from the channel to the buffer memory. After storing the received packet in memory, the state machines send a signal to the 68020 informing it about the event that took place. The 68020 checks the packet, collects the monitoring information associated with it, and sends an interrupt signal to the 68010, so that the latter reads the packet and the monitoring data.

A CRC check can be done in hardware while reading the packet from the channel, and the result is stored with the other monitoring information.

The use of a separate piece of hardware to perform packet reception is essential, so that a node can transmit and receive simultaneously at 10 Mb/s. This feature is required for some network architectures.

2.6. Experimental Data Collection

Since the channel is operating at 10 Mb/s, and to relieve the 68020 from functions with critical time constants, many subsystem blocks cooperate with the processor to perform the event recording in our monitoring architecture. The state machines are the first to identify the occurrence of an event, and, therefore, inform a hardware unit, made up of a FIFO and a time stamp circuit, about this fact. This hardware unit associates with each event a code and a time stamp, and stores the pair in the FIFO to preserve the order of occurrence. After the completion of a packet transmission or of a packet reception, the 68020 collects the events information from the FIFO and stores them in the mailbox memory, later to be read by the 68010 on the Pacific board.

2.7. Other Protocol Functions

Many protocol functions with no critical real time constraints can also be implemented on the PW board. Some of these functions belong to the data link layer, and other higher layer functions can also be implemented to relieve the 68010 CPU board of its burdens.

An example of data link layer functions that can be performed is the truncated binary exponential backoff retransmission control policy used in Ethernet whenever a collision occurs. Examples of higher layer functions that can be implemented are packet formation, packet reception acknowledgement, and even some routing functions in the case of a gateway. All these additional functions are realized by procedures executed by the 68020 processor.

3. ETHERNET

In Ethernet, all nodes are connected to a broadcast medium. Ethernet uses a media access scheme based on carrier-sense multiple-access with collision detection (CSMA/CD). With this scheme, a node with a packet to transmit monitors the transmission medium and waits until there are no transmitting nodes. Then the station begins transmitting. Several nodes may begin transmitting simultaneously and cause a collision. Then both nodes cease transmission and retry after a random period of time. A more detailed description of Ethernet is given below. The protocol is completely described in [5].

3.1. The Ethernet Protocol

Ethernet specifies the protocol for both the data link and physical layers. We are only interested in the data link layer. The Ethernet data link layer performs the following functions:

Data Encapsulation/Decapsulation

- framing (frame boundary delimitation)
- addressing (handling of source and destination addresses)
- error detection (detection of physical channel transmission errors)

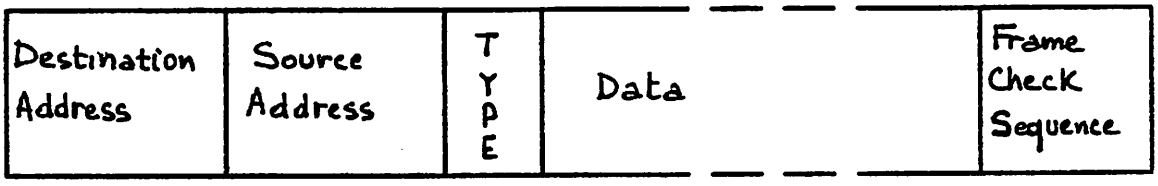
Link Management

- channel allocation (collision avoidance)
- contention resolution (collision handling).

It is most convenient to describe the operation of Ethernet by ignoring the possibility of collision at first and then describing the handling of collisions later.

The client may request the transmission of only one packet at a time. When the client layer makes such a request, it passes the destination address, the

source address, the type (used for higher level protocols), and the data to the data link layer. The data link layer then constructs a packet with the format shown below The frame check sequence is a cyclic redundancy code computed by the data link layer for the client layer-supplied data.



After the packet is constructed, the data link layer monitors the carrier sense signal from the physical layer and defers while there is another node transmitting. When the channel becomes clear, the data link layer provides a serial stream of bits to the physical layer. At the same time, the physical layer monitors the medium and generates a collision detect signal if there is a difference between transmitted and received data.

At a receiving node, the physical layer provides the link management component of the data link layer with a carrier sense signal and a stream of bits that makes up the packet. When the carrier sense signal goes off, this indicates the end of a packet, and the data decapsulation component of the data link layer checks the destination field to see if the frame should be accepted. If so, it passes the contents to the client layer with a status code. The status code is generated by checking the frame check sequence and making sure that the packet consists of an integral number of octets.

If a node senses that the channel is clear, it may begin to transmit. Because of delay over the transmission medium, it takes time for the signal to propagate to the other nodes. In the meantime, another node may still sense that the channel is clear, and it may begin to transmit. In this case, a collision occurs, and it takes time for the collision to propagate to the first node that transmitted. The maximum time to elapse during this sequence of events is equal to twice the transmission time between the two most distant nodes of the network. This time interval is called the collision window. If a node transmits for the duration of the collision window without detecting a collision, the other nodes, if they are operating properly, will have the carrier sense signal and no longer attempt to transmit. At this point, the node is said to have acquired the channel.

In a collision, the physical layer notices the difference between the transmitted and received data and sets the collision detect signal. The transmit link management component of the data link layer notices this and enforces the collision by transmitting a jam sequence. This ensures that the duration of the collision is sufficient for the other transmitting station(s) to detect it. After the jam is sent, transmit link management terminates the transmission and schedules a retransmission for some time in the future.

The time is random and is computed according to a binary exponential backoff algorithm. Repeated collisions indicate a busy channel, so the algorithm increases the range of the defer time. After many tries, if the transmission does not succeed, the attempt is abandoned, assuming that the channel has failed or become overloaded. At the receiving end, the bits of a collision are decoded just like those of a valid packet. Packets that experience a collision will either not contain an integral number of octets or they will be smaller than the minimum frame size. This will be detected by the data link's receive link management

component. These frames are discarded.

3.2. The 3COM Ethernet Controller

The 3COM Multibus Ethernet controller board provides a connection to Ethernet for any Multibus compatible system processor [6]. This controller performs part of both the physical and the link layers services; we are only interested in the link layer functions. The two main ones are data encapsulation with error detection and link management --channel allocation and contention resolution.

The controller board communicates with the host board through buffer memory and a control/status register. The buffer memory is used to store up to two received packets and one to be transmitted. It is accessible to both the controller and the host CPU. The CPU sends a command to the controller by setting a certain bit in the control/status register to one. This action initiates the associated function in the board. The control/status register is also written by the controller to set a status flag. The setting of this flag will interrupt the host CPU requesting a specific service or will indicate to the host CPU, when the latter reads it, the status of the controller.

To implement the media access mechanism, two signals from the physical layer are used. The first one is the collision detection signal, based on which the binary exponential backoff retransmission policy is implemented. The second one is the carrier signal which determines if the channel is quiet or not, and therefore allows a node to transmit. When a collision occurs, the controller puts the jamming sequence on the channel and sets a special bit in the status register to one. The CPU then responds by computing the retransmission backoff time and writing it into another control location the retransmission backoff time. This stored value is used to load a counter that initiates another packet

transmission attempt when the written value is decremented to zero.

To transmit, the host loads the packet to be sent into the transmit buffer, and sets the transmit bit in the control register to one. This buffer is 2K bytes large and can store only one packet. When the channel is clear, the controller reads the bytes from memory and sends them to the serializer section where they are converted to a serial bit stream. The serialized data is fed to a CRC generation section where the result of the computation is automatically appended to the end of the data packet. When transmission is completed the transmit bit in the control register is reset to zero by the controller to inform the CPU that the task was accomplished, and that it can use the buffer for a new packet. If a collision occurs, the bit corresponding to collision is set, and the necessary action is taken.

The controller board provides two buffers to receive packets. Each one of them is 2K bytes large and can contain one packet. A control bit is associated with each one of them. This bit tells which one, among the host processor and the controller, can access the buffer. Upon receiving a packet, the controller stores it in the buffer with the corresponding control bit set to one. When done, it resets this control bit to zero to indicate to the processor that the buffer is full and ready to be read. If both buffers are full a third control bit will indicate to the processor which packet is the oldest one. In this case, the controller does not read packets from the channel. While reading the packet from the channel, a CRC check is performed by the hardware section. If an error has occurred, a flag is set in the first byte of the received packets memory. In this same byte more information related to different classes of received packets is stored. In the case of an alignment error, corresponding flags are set to notify the processor. The controller is capable of recognizing the packet's destination address and see if it matches the station address. In the case of a multicast address the

controller requires the help of the processor to perform the comparison. The processor can configure the controller to accept a selected classes of packets. Among these classes, there are all packets, all packets - packets with errors, station packets + multicast address packets - packets with errors, and station packets + broadcast address packets.

The 3COM controller board requires the help of the processor to implement all the Ethernet protocol functions, mainly part of the binary exponential backoff algorithm. This could cause overhead for the host CPU, but studies of even heavily used Ethernets indicate that this overhead can be considered negligible [7].

3.3. The Excelan Ethernet Controller

The Excelan EXOS 204 board connects a Unibus based system to an Ethernet [8]. It implements the complete Ethernet data link level interface, and in addition can support high level network protocols. These protocols can be standard, like DARPA TCP/IP, or user defined for a specific application.

The board is managed by an Intel 80186 CPU running a real time operating system. The Ethernet data link protocol is implemented by the Intel 82586 LAN coprocessor. A large (128K bytes) dual-port memory is used to allow concurrent memory access by the two on-board processors. This RAM is also accessible to the host CPU in the Unibus system. In a similar way, the host memory is also accessible to the controller processor, the 80186.

The NX 200 is the real time operating system. It resides in EPROM and runs on the 80186. This operating system provides a multitasking environment for the implementation of higher level protocols, in addition to the procedures used to communicate with the host CPU and the Ethernet controller. A running process makes system calls through a hardware interrupt to the processor. A

priority-based preemptive round robin scheduling algorithm is used for CPU time allocation. Up to 256 priority levels are supported, and any process can examine and change the priority of any other one. A process can also suspend, delay or resume the execution of any other process. A process lock mechanism is used when running critical sections. This mechanism postpones scheduling decisions until a corresponding unlock is executed. Interprocess communication is realized by the use of mailboxes and the exchange of messages.

All the link level Ethernet functions are performed by the 82586 LAN coprocessor. These include:

- serial/parallel and parallel/serial conversion
- address recognition
- framing and unframing of messages
- carrier sense and deference
- collision detection and enforcement, including jamming, backoff, timing, and retry
- CRC generation and verification
- error detection and handling

For the received packets, the recognition of the physical, broadcast and multicast addresses is fully supported. This function is mainly implemented in the Intel 82586, except for the recognition of the multicast addresses, in which case the Intel 80186, with some special routines, helps achieving the required filtering scheme. Up to 252 multicast address can be assigned to a node.

The main feature of this board is the communication scheme between the two pairs of processors: the host and 80186 pair and the 80186 and Ethernet coprocessor pair. These two functions are based on a request/reply message exchange scheme, using dedicated mailboxes. For a transmission, the host

sends a transmit request message to the board processor. This message contains a pointer to the packet in host memory that is to be transmitted. The processor reads the packet from the host memory, buffers it in the on board dual-port RAM and sends a message to the coprocessor, requesting the transmission of the packet on the Ethernet. By doing so, the processor gives the coprocessor full control over the buffer containing the packet. When the coprocessor finishes transmission, it replies to the board processor to confirm the packet transmission, and the buffer is freed. The 80186 then replies to the host CPU and informs it of the transmission. Only then can the host CPU access the memory space that contains the packet and modify its contents. This scheme insures the consistency of transmitted data in the host memory.

In the case of the received packets, a similar scheme is followed. The host CPU sends a receive request message to the board processor. This message contains a pointer to buffer space in the host memory. The 80186 does the same with the Ethernet coprocessor and upon receiving a packet from it, stores it in the host buffer and sends a reply message.

Several requests can be pipelined. Two cyclic queues are used to store messages, one is used for the host to controller messages, and the other for the controller to host messages. In the case of transmit request, up to four requests, initiated by the host, can be pending. On the other hand, in the case of received packets, up to 32 packets can be buffered in the board memory, while waiting for receive requests.

In addition to the transmit and the receive requests, many command requests are initiated by the host CPU. Among them is the command telling the board to respond only to certain address classes.

The board also collects monitoring information and the host has access to it through a specific command request message. This information consists mainly

of the number of packets transmitted, aborted, received intact, and received with error. These last ones are classified by error types.

3.4. The Protocol Workroom Implementation of Ethernet

The Ethernet protocol is easy to implement on the PW controller. The channel emulator provides the controller with several control signals. When the carrier sense signal is asserted, the data on the channel is valid. If both the carrier sense and the collision detect signals are asserted, a collision has occurred. Another control signal indicates the end of the packet. These signals are received by the state machines, and are used in making decisions in the protocol.

The binary exponential backoff algorithm is executed by the board processor, the MC 68020. Its execution is initiated by an interrupt from the state machines whenever they detect a collision. The retry is under the control of the 68020 and is very similar to a new packet transmission attempt.

Address recognition is done by the state machines. The pattern recognizer can be programmed to recognize several addresses. When one of these is detected, the state machines initiate the input data transfer hardware, which copies the incoming data into the buffer.

The transmission and reception functions, including CRC generation and verification, are realized by the data transfer hardware that is controlled by the 68020 and the state machines.

The board handles one packet to be transmitted at a time. After receiving a transmit request from the host 68010, the board undertakes the packet transmission and does not reply until the request is accomplished. Meanwhile, the 68010 can store a second packet to be transmitted in the board buffer, but will not request its transmission unless it has received a reply from the 68020

concerning the previous packet. Storing a packet while another one is being transmitted reduces the delay introduced by the hardware between packet generation and packet transmission. All the other packet generation requests are queued in the 68010 board. This method also reduces the size of the memory in the controller board used to buffer the packets to be transmitted.

In the case of the received packets, the state machines control the associated buffer and initiate all packet reception. In the current design, the buffer is capable of storing four packets. Whenever a packet is received, the 68020 is informed by an interrupt signal initiated by the state machines, and subsequently the 68020 interrupts the Pacific board's 68010 to notify it. If the received packets buffer is full, a special interrupt is generated to notify the 68010 and the 68020.

Any event occurring on the board is associated with an interrupt signal initiated by one of the subsystems. These interrupts are the key elements in the implementation of the monitoring function, since they are used to control the hardware responsible for storing the events codes and timestamps in the FIFO. These same interrupt signals, if sent to the 68020, can start the execution of the monitoring data collection routine.

3.5. Ethernet Implementation Comparison

All three boards implement Ethernet data link layer services, although the architectures are different from each other.

Looking first at the 3COM board, we see two main deficiencies: its small buffer size for received packets and its high interaction with the host CPU. We suspect that under heavy load, the performance of the station connected to the network would deteriorate because of the host CPU overhead created by the Ethernet link layer functions. This overhead is mainly due to the fact that the

CPU has to empty the buffers quickly and assist the controller whenever a collision occurs. In the other two boards the buffers are larger and the interaction with the host CPU is reduced tremendously. In fact the other controllers interact with the host only to make a transmission request or to accept a packet.

Both the Excelan and the PW boards perform the data link layer functions without interrupting the host processor. In the Excelan board, all Ethernet functions are performed by a single chip. This frees the 80186 for data transfer to and from host main memory and the implementation of higher level protocols. In the PW board, the 68020 is needed to perform some of the data link layer functions and functions related to monitoring data collection, and so it has a higher load than the 80186 of the Excelan controller. On the other hand, it does not have to do the data transfer function, and it is a more powerful processor. Therefore, it is difficult to determine which controller has a processor with less load and is thus better suited to implement higher level protocols. Another point is that the data transfer function performed by the Excelan board reduces the load on the host CPU.

In the case of a small network and under moderate load condition, the three boards would perform equally well. However, in the case of a large network with many stations creating a heavy load, the more sophisticated controllers would relieve the host CPU and therefore the overall station performance would be better.

4. THE PRONET TOKEN RING NETWORK

In the Pronet token ring network, nodes are connected in a ring, and messages pass in one direction around the ring. Access to the transmission medium is coordinated by having a special sequence called a token circulate around the

ring. When a station wants to transmit a frame, it waits for the token to pass. When this happens, the station removes the token from the network and begins to transmit the frame. This avoids the problem of collision encountered in Ethernet. The frame is repeated at each station, even the destination station, and circulates around the ring, eventually returning to the originating station, where it is removed from the network. After transmission is complete, the token is placed on the network so that other stations can transmit. A complete description of the Pronet protocol can be found in [9].

4.1. The Pronet Protocol

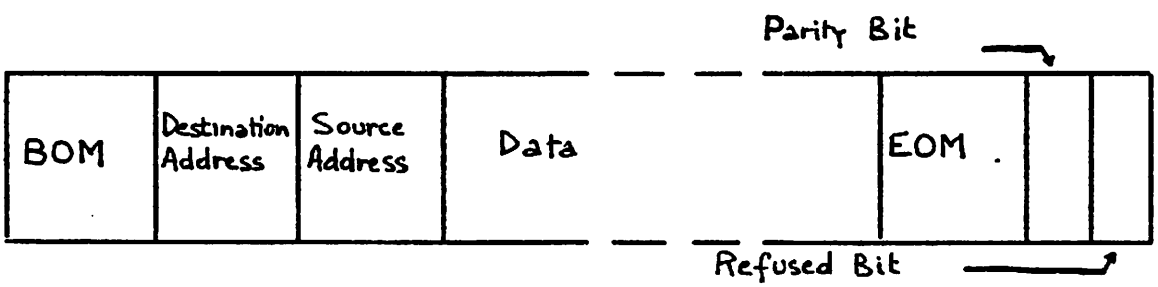
The Pronet data link layer performs the following functions:

Encapsulation/Decapsulation

- Framing
- Source Addressing
- Bit Stuffing

Network Management and Error Recovery

- Transmission Queueing
- Control Sequence Detection
- Ring Initialization



The format for a frame is illustrated above. The destination address, source address, and data fields are surrounded by the Beginning Of Message (BOM) bit sequence and the End Of Message (EOM) bit sequence. After the EOM sequence there are two status bits. The first of these is a parity bit, and the second is a refused bit that is set by the destination station if it does not copy the frame into its buffer.

There are three control bit sequences used in Pronet: the BOM and EOM sequences mentioned above and the token sequence. These sequences are shown in Figure 3. Each one consists of a flag followed by one or two bits. A flag is a sequence of seven consecutive one's preceded by a zero. If this sequence existed in the sequence formed by the address fields and the data field, it would be interpreted as the beginning of a control sequence. To avoid this confusion, the address and data fields are bit stuffed before transmission by inserting a zero bit after six consecutive one's that follow a zero. This action insures that seven consecutive ones indicate a control sequence. At the receiving station, this operation must be undone by removing a zero bit that follows a zero and six one's found in the address and data fields.

As in the Ethernet protocol, the client layer requests the transmission of only one packet at a time; the next transmission request can only take place after the first is completed. When a request is made, the client layer passes the destination address and the data to the data link layer. The data link layer adds the source address and does bit stuffing on the address and data fields. Then it waits for a token to circulate on the ring. When a token is detected, the last bit is changed from a 1 to a 0, turning the token into a BOM sequence. Then the bit-stuffed address and data fields are transmitted, followed by the EOM sequence, the parity bit, and the refused bit set to zero.

When a station transmits, it does not repeat data coming around the ring until it takes its frame off the ring. When the message comes back, it is removed from the ring. The station checks the refused bit, and if it is set, the client layer is notified. After transmission, the station puts a token on the ring.

When a station is not transmitting, it repeats messages on the ring. When a new packet is received, the station checks the destination address field of the packet and compares it with its own address. If there is a match, then it copies the frame into its buffer unless the buffer is full. If the packet is not copied, the refused bit of the packet is set to one.

There are various things that can happen to a network that can cause faulty transmission. In this case, the network needs a recovery mechanism. Pronet uses three timers to assist in recovery: a token timer, a flag timer, and a message lost timer.

During normal operation, a token circulates around the ring. Whenever a token is recognized at a station, the token timer is reset. A token is considered lost if the token timer shows more time that the amount of time it takes for each station to send the longest allowed frame. If a token is lost, the client layer is notified. The flag timer is used in a similar way. Its maximum value is the time it takes to send the longest allowed frame. The flag timer has a shorter time constant that allows certain error conditions to be reported before the loss of a token is detected.

The network recovers from a lost token in the following way: the first station that has a message to transmit that has detected a lost token transmits the message. With this scheme, it is possible that two stations will transmit a message at about the same time. In this case, since transmitting stations do not repeat bits coming around the ring, the message of each station will be lost.

The message lost timer is reset when a node starts transmitting a packet. A lost message is detected when the message lost timer value exceeds the amount of time it takes for a packet to travel around a ring with the maximum number of nodes. When a message lost timeout occurs at a node, the node goes into repeat mode. If a message is lost because of the situation described above, the token will be lost, and the next station desiring to transmit a message puts it on the ring and reinitializes the network.

4.2. The Pronet Data Link Controller

The Pronet data link controller consists of two hardware modules: the Ring Control Board (CTL) and the Host Specific Interface Board (HSB). The CTL performs the functions of the data link and physical layers, and the HSB provides an interface between the host processor and the data link layer.

4.2.1. The CTL

The CTL is divided into three blocks: the input machine, accepting data from the ring, the output machine, putting data on the ring, and the interface to the physical layer.

The input machine is organized around a state machine that controls the operation of address comparators, counters, the bit destuffer, and the lost token and flag timers mentioned above. Data is destuffed as it comes from the physical layer. Counters are used to track the various fields of the incoming frame. Node address and broadcast address comparators are used to recognize addresses in the address fields. If there is enough buffer space available on the HSB, the data is copied.

The input machine also performs other functions. It detects format and parity errors in frames copied from the ring. If a station is transmitting, the

input machine checks the format of the messages originated at the output machine as they are taken off the ring. It also detects the presence of flags and tokens and resets the appropriate timers. If an error is found, the input machine reports it to the HSB.

The output machine, too, is organized around a state machine. The state machine controls the bit stuffer, counters, and the message lost timer. Data comes to the output machine either in serial from the input machine when the station is repeating or from the HSB in 8-bit byte parallel format when the station is transmitting. It is bit stuffed before going to the physical layer. The source address data is obtained from a hardware switch on the CTL during transmission. During transmission, the transmitting node does not repeat data on the network.

The lost message timer and initialization of the ring is also handled by the output machine. Upon request of the HSB, a packet is transmitted if a packet is ready to be transmitted and the token timer times out.

4.2.2. The HSB

The HSB consists of three parts: the UNIBUS DMA interface, the transmit/receive control and status registers, and the transmit/receive packet buffers.

The DMA interface controls the transfer of data, address, and control signals between the host and the HSB. Each one can interrupt the other.

The transmit/receive control and status registers are used to pass information between the HSB and the host. Control signals include transmission and network initialization. Status conditions include received data present, message refused, and data link errors. These include output time-out, parity error, bad format, and token missing.

The HSB has separate buffers for transmission and reception. The buffers are each 2K bytes long. This limits the maximum frame length.

4.3. The Protocol Workroom Implementation of Pronet

The Pronet token ring protocol is implemented on the PW data link controller with only one modification. Instead of using bit stuffing to ensure the uniqueness of control bit sequences, other signals from the channel emulator are used in parallel. It was thought that adding bit-stuffing capabilities to the board would be unnecessarily complex.

For a transmission, the Pacific board's 68010 first writes the destination address and data in the buffer of the controller and notifies the 68020. The 68020 provides the source address and completes the packet, adding the EOM sequence and the parity bit at the end. It then prepares the state machines to start a data transmission when a token arrives. When the state machines detect a token, the last bit is flipped, and the data transmission hardware is initiated.

At this point, the controller board stops repeating data on the ring until the state machines recognize the packet sent. Then it puts a token on the ring. The parity is computed by the 68020 and compared with the parity bit of the packet. The result of this comparison and the refuse bit are reported to the Pacific board.

When a node is not transmitting, the state machines repeat the data around the ring while waiting for a BOM sequence. If the address of the packet matches that of the node, the packet is copied into the buffer provided there is space. If the buffer is full, the refuse bit is set. After a packet is copied, the 68020 is notified. It then checks the parity and informs the 68010 of the packet reception.

The timers used for error recovery are controlled by the 68020. They are reset when the state machines notify the 68020 of the detection of a flag or token or the beginning of a message transmission. When a time-out occurs, the 68020 initiates the proper action.

4.4. Pronet Implementation Comparison

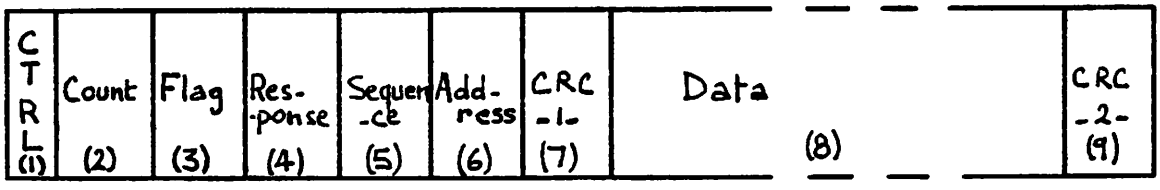
Both data link controllers execute the token ring protocol, but each one has advantages. The PW controller is much more powerful in that it is capable of implementing higher level protocols. Furthermore, it has more buffer space. On the other hand, the Proteon controller has DMA hardware and thus does not burden the host CPU with data transfer between host memory and the controller; the PW controller does not have this feature.

5. THE DIGITAL DATA COMMUNICATIONS MESSAGE PROTOCOL (DDCMP)

The DDCMP is a point-to-point protocol that supports multipoint connections. Packets are received in the order they are presented to the data link controller by the client layer at the transmitting end. Receptions are acknowledged, and one message from the receiver can acknowledge up to 255 packets. The data transfer may be full-duplex or half-duplex. A full description of the protocol is found in [10].

5.1. The Protocol Description

The DDCMP packet format is as follows.



(1) This control character differentiates between the packet classes; there are data packet and control message.

(2) This field indicates the number of bytes transmitted in the data field if the packet contains data. In the case of a control message, it contains a code corresponding to the control message sent. Furthermore, if the control message is a negative acknowledgement (NAK), this field also contains a code indicating the reason for the error.

(3) The flag field indicates to the receiving station if the packet is followed by a sync character or not. Therefore it tells the destination if this is the last packet or not. This flag is useful in half-duplex mode since it permits to the receiving station to start transmission when the sending end is done.

(4) The response field indicates the sequence number of the last packet received intact. This field can be part of a control message or part of a data message. This scheme reduces control overhead.

(5) This field indicates the sequence number of the packet being sent.

(6) In case of a multipoint connection, this address field is used to differentiate between the different stations. If a point-to-point connection is used, the receiving end discards the contents of this field.

(7) The CRC-1 field is the CRC check of the header block. The designers felt that this field was necessary because of the importance of the header information.

(8) This is the data field. It does not exist in control messages.

(9) The CRC-2 field is the CRC check of the transmitted data block.

Data exchange between protocol users follows a handshaking procedure. First of all, the transmitted station sends a START control message to inform the receiving end about its intention to transmit data. The destination station

returns a START ACK control message, to tell the sending end that it is ready. Upon the reception of the START ACK message, the transmitting end starts sending data packets with sequence numbers. The receiving end periodically sends a message containing a positive or a negative acknowledgement, depending on the situation. This message can be an independent control message or part of a data message. A negative acknowledgement arises if an error has occurred in a received packet or if the packet with the right sequence number has not reached the destination. Upon receiving the NACK message, the sending end retransmits the requested packet and all the subsequent ones, because whenever an error occurs, the receiving end discards all messages with sequence numbers issued later. The sending end expects an acknowledgement message from the destination, and for this reason a time-out is used. If a time-out occurs before receiving any acknowledgement message, the sending end sends a control message containing the sequence number of the last packet transmitted. Upon receiving this control message, the destination responds by sending the sequence number of the last packet received intact. This information is used by the sending station to retransmit the missing packets, and resume its normal operation. The last packet to be transmitted is marked by a special value in the flag field.

5.2. The DMR11 Controller

The DMR11 controller is used in a network link for high performance interconnection of VAX-11/780 or PDP-11 computers [10]. It consists of a microprocessor module and a channel interface module. The controller can operate at speeds ranging from 2.4 Kb/s to 1 Mb/s to accommodate different channel bandwidths.

The DMR11 is controlled by a user program residing in the host CPU memory. After requesting its use, the user program waits for a reply message from the controller to confirm the granting of the request. The exchange of messages is done through a control/status register. Once the user program is granted the access to the board, it initializes it. After the initialization stage, the board is ready to transmit and to receive packets.

The transmit and receive commands are sent to the DMR11 through a four register mailbox. The first two indicate the command requested. the third one is a pointer to the buffer space in the CPU memory that is used when executing this command. The last register indicates the length of the buffer. It is written by the host CPU when the command is a packet transmission, and by the controller when a packet is received. The controller can access the CPU memory, and the buffers allocated to the packets are part of this memory. Therefore, all packet data is written into or read from the buffers in main memory by the controller. After writing the correct sequence in these registers, the CPU sets a bit in the control/status register to notify the controller. Upon accomplishing the requested command, the controller sets another bit in the same control/status register to inform the CPU.

When implementing the DDCMP, the DMR11 provides the following data link layer functions:

- It creates an error-free data path and transfers data between protocol users over a physical link, while maintaining data integrity.
- It transfers messages in proper sequence. Messages will be delivered from one user to the other in the same order as they are presented to the DMR11 from the host, even though the controller may require the use of retransmission for error recovery.

- It provides notification of the channel error. Such errors might be a high bit error rate or a modem failure.

All of the functions mentioned above, are implemented by the board microprocessor. The different commands executed are microcoded to improve the performance of the hardware. When implementing DDCMP, the microprocessor needs two counters to keep track of the sequencing of the incoming and outgoing packets. A timer, controlled by the microprocessor, is also needed for the time-out calculations. It interrupts the microprocessor whenever a time-out occurs.

5.3. The Protocol Workroom Implementation of the DDCMP

In the PW implementation of the DDCMP, the only function performed by the state machines is the control of the hardware transferring data to and from the channel. As with the other protocols discussed earlier, the Pacific board must transfer packets to and from the data link controller buffers. All of the other functions except the computation of the CRC are performed in software by the 68020. The state machines can be programmed to perform the address recognition if multipoint connections are used.

5.4. The DDCMP Implementation Comparison

Since the PW board has no access to the host main memory, the host CPU must perform the data transfer function. This causes a load on the host CPU that is not present with the use of the DMR11.

6. CONCLUSIONS AND FUTURE WORK

The architecture of the PW controller is general in the sense that it can be programmed to implement different protocols and emulate several commercial controllers. Its major characteristic is the use of separate blocks to perform the different tasks ordinarily associated with the data link layer. This is the key to the architecture's flexibility. The existence of a powerful microprocessor on board permits the implementation of higher level protocols and functions. This reduces the load on the host CPU.

The monitoring function performed by the controller in hardware and software is essential for the performance analysis. It can also help in debugging and evaluating the hardware and software. This feature is missing in all the commercial controllers that we studied.

The function that is performed by almost all the other controllers but not by the PW board is the data transfer between the host memory and the controller buffers. This limitation is due to the architecture of the Pacific board and the interface it provides to the controller. However, this relieves the 68020 from the data transfer function and frees it to do other tasks. On the other hand, it adds a burden to the host. A controller capable of performing the data transfer to and from host memory would be better for network performance than the PW board.

The strength of the PW controller lies in its flexibility. Parameters of a protocol, such as the maximum packet size for all protocols and the backoff algorithm for retransmission used in Ethernet, can be modified easily. Experiments performed by generating network traffic can be run to examine the effects of parameter changes on performance.

In our opinion, the best architecture for a data link controller is an architecture similar to the PW one enhanced by the addition of a DMA controller chip

capable of accessing the host memory. This architecture would have all the advantages of the PW and the other controllers, and by having this additional feature, both the controller processor and the host CPU would be relieved from data exchange and therefore free to perform other tasks. A controller with this architecture could also take on the task of implementing higher level protocols as well. We suspect that this would improve the overall network performance since it is believed that the processing bottleneck in networking occurs at the layers higher than the data link layer.

The PW board can be used to implement more complicated protocols than those studied above, such as X.25. The addition of a communication path between several PW controllers connected to the same Pacific board is under investigation. This feature would permit the study of circuit switching and store-and-forward networks. Higher level protocols could also be studied with the facility, once a reliable link layer is implemented. The facility's use could be extended even further to encompass the study of distributed computing.

7. ACKNOWLEDGEMENTS

The Protocol Workroom project is supervised by Professors Pravin Varaiya and Jean Walrand and is supported in part by NSF grants ECS-8118213/85-06337, ONR contract N00014-80-C-0507, JSEP contract F49620-79-C-0178, and grants from Bell Communications Research and the MICRO program.

8. REFERENCES

[1] A. Fawaz, D. Giralt, and L. Ludwig, "The Protocol Workroom: An Experimental Protocol and Distributed System Research Facility for UC. Berkeley," Protocol Workroom Document No. 84-1, June 1985, EECS Dept.

[2] A. Fawaz, L. Ludwig, and M. Peck, "Node Emulator Architecture For The UC. Berkeley Protocol Workroom Facility," Protocol Workroom Document No. 85-3, June 1985, EECS Dept.

[3] L. Ludwig, "Channel Emulator for the UC. Berkeley Protocol Workroom Facility," Protocol Workroom Document No. 85-1, February 1985, EECS Dept.

[4] G. McDaniel, "Metric: A Kernel Instrumentation System for Distributed Environments," Proceedings of the sixth SOSP, Operating Systems Review, Vol. 11, No. 5, November 1977, pp. 93-99.

[5] XEROX Corp., The Ethernet A Local Area Network: Specifications, November 1982.

[6] The 3COM 3C400 Multibus Ethernet Controller Reference Manual.

[7] J.F. Shoch and J.A. Hupp, "Measured Performance of an Ethernet Local Network," Communications of the ACM, Vol. 23, No. 12, December 1980, pp. 711-721.

[8] Excelan Inc., EXOS 204 Ethernet Front-End Processor for Unibus Systems Reference Manual.

[9] Proteon Associates Inc., The Operation and Maintenance Manual for the Pronet Local Network Interface.

[10] DEC, DMR11 Synchronous Controller Technical Manual.

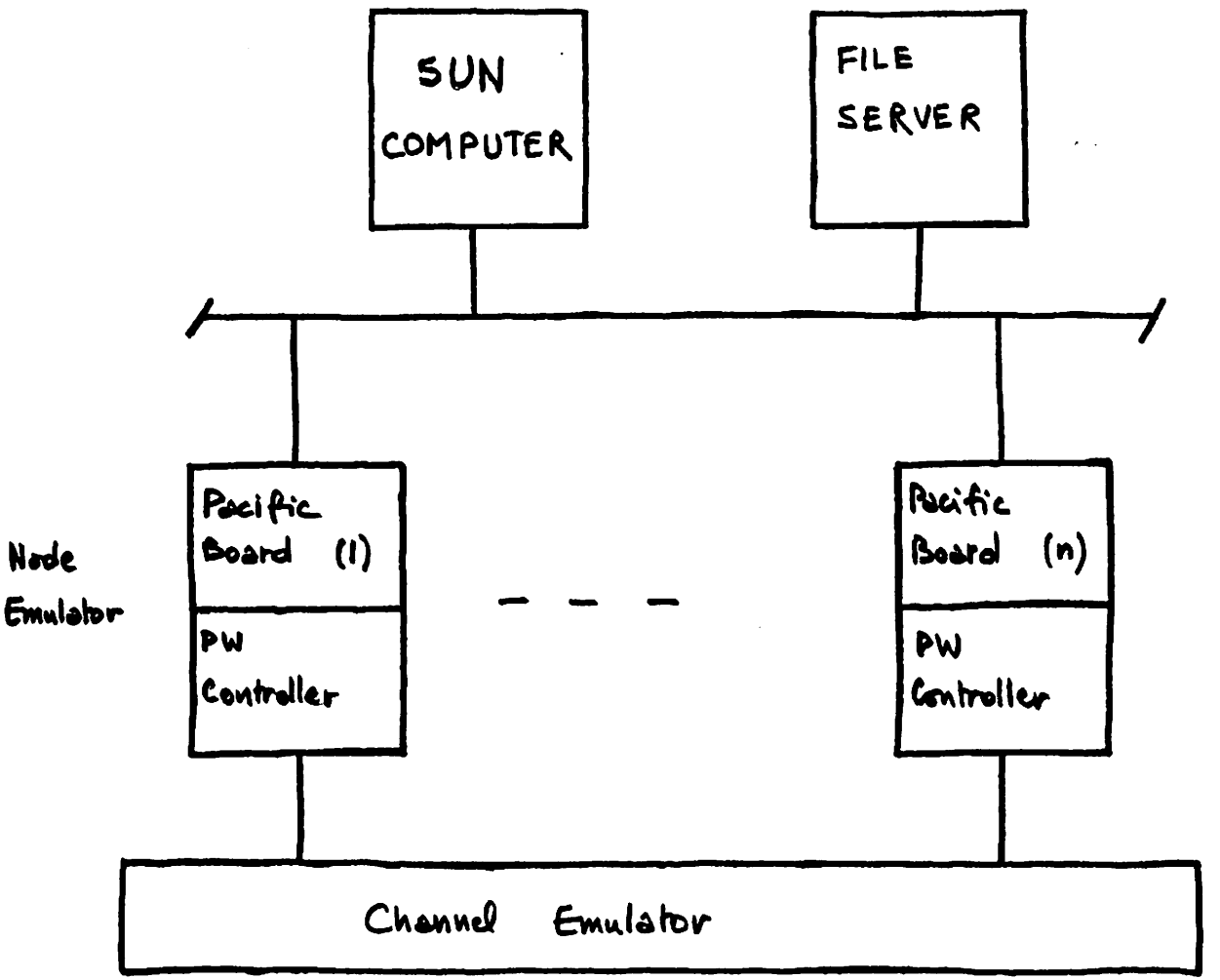


Figure 1. Overview of the Protocol Workroom Facility

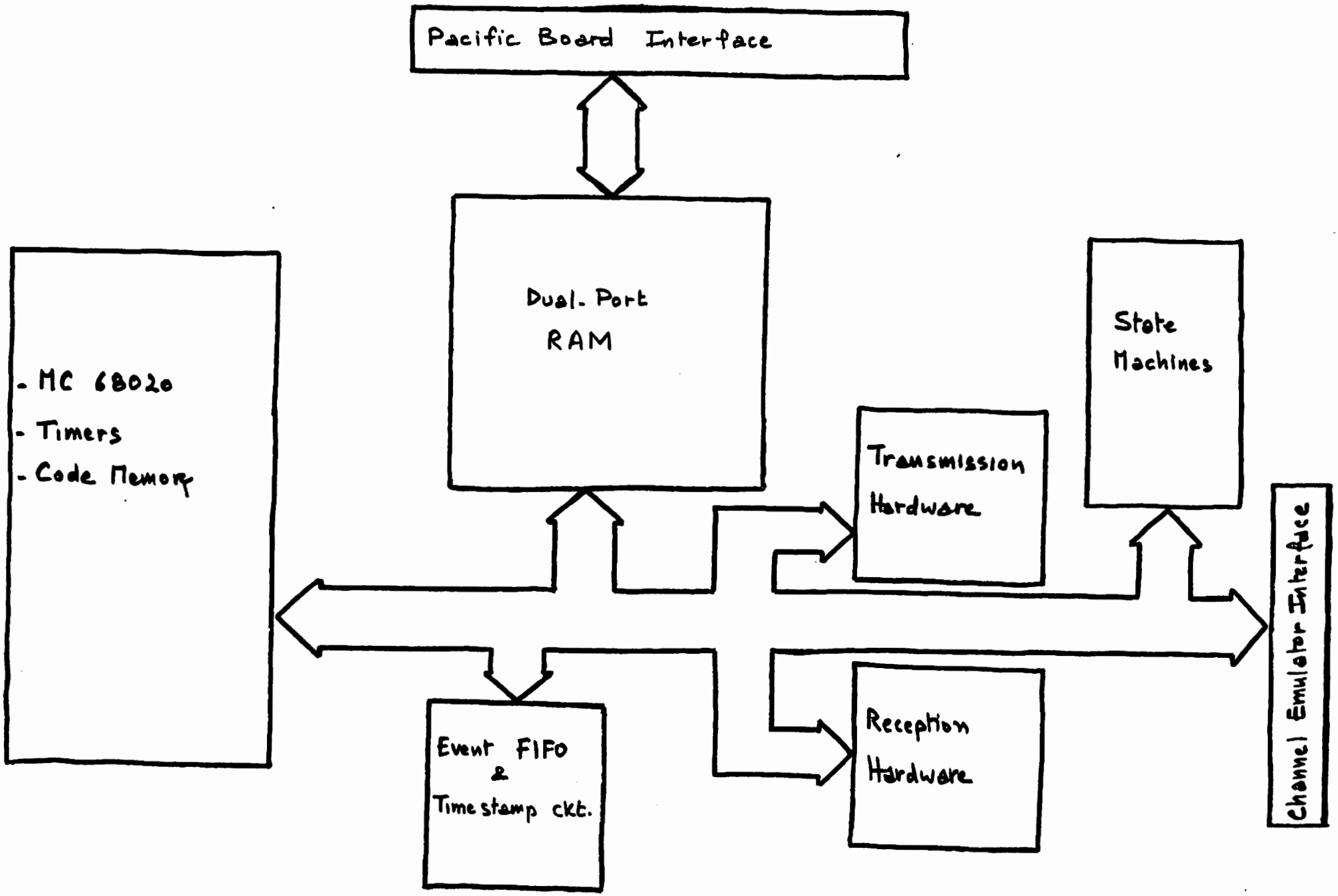


Figure 2. General Overview of the PW Controller Architecture

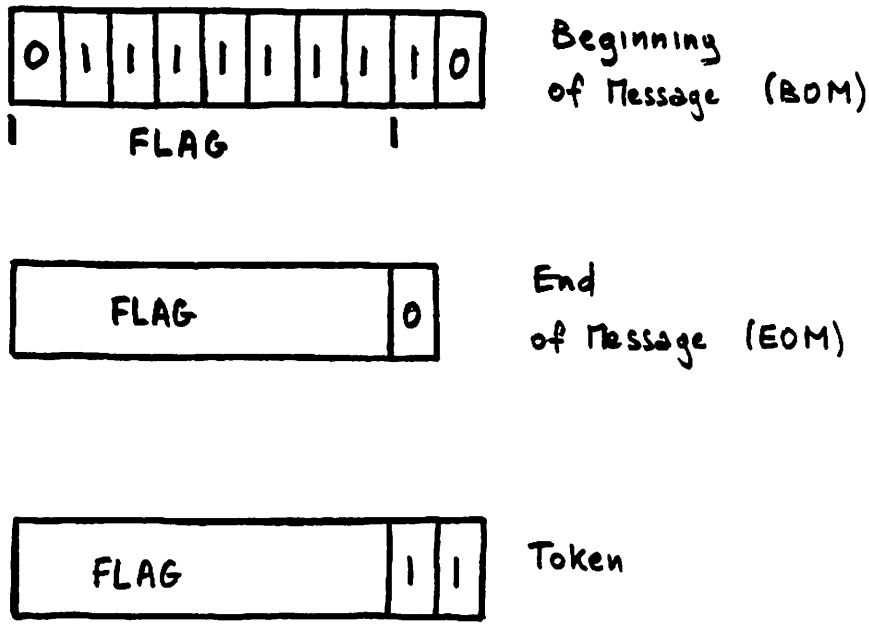


Figure 3. Pronet Control Characters