Title

Design implementation and measurement of a collision avoidance multiple broadcast tree network

Authors

Huang, Hung Khei
Suda, Tatsuya

Publication Date

1992

Peer reviewed

Design, Implementation and Measurement of a Collision Avoidance Multiple Broadcast Tree Network [1]

Technical Report 92-37

Hung Khei Huang, Tatsuya Suda
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717
Phone: (714) 856-5474
FAX: (714) 856-4056
e-mail: huang@ics.uci.edu, suda@ics.uci.edu

[1]This material is based upon work supported by the National Science Foundation under Grant No. NCR-8907909. This research is also in part supported by University of California MICRO program and Omron.

Abstract

Packet collisions and their resolution create a performance bottleneck in random access LANs. Collision avoidance switches are a hardware solution to this problem [1, 2]. Collision avoidance switches allow the implementation of random access protocols without the penalty of collisions among packets.

In this paper, we describe a design and implementation of a local area network architecture based on collision avoidance, called the Collision Avoidance Multiple Broadcast (CAMB) tree network. Our implementation follows the protocol layering architecture of the IEEE 802 local area networks, and includes CAMB tree switches, station/network interface boards, and support of transport protocols. We also present the performance measurements of our experimental CAMB tree network.

1 Introduction

Advantages of random access protocols such as ALOHA and CSMA/CD include simplicity and ease of implementation. Random access protocols also exhibit small transmission delays under light traffic conditions. However, they have a performance bottleneck under heavy traffic conditions. Under such conditions, a large number of collisions occur, resulting in low channel utilization. Channel capacity is wasted in the transmission of collided packets [3].

The authors have proposed and investigated a new network architecture that solves this performance bottleneck based on a collision avoidance scheme [2, 4, 5, 6]. This architecture is called the collision avoidance tree network. It uses collision avoidance switches. These switches allow implementation of random access protocols without the penalty of collisions among packets. Their key feature is that there is no collision. When two or more packets contend for the right to use the switch, one successful packet transmission is guaranteed. No channel time is wasted in the transmission of collided packets. The main disadvantage of a traditional random access network is eliminated.

Collision avoidance can be implemented with very little circuitry. Functional designs of collision avoidance hardware are proposed in [1, 4, 6, 7, 8]. Various station and switch protocols for collision avoidance networks are discussed in [2, 4, 5, 9].

An experimental broadcast star network, a simplest form of tree network, is discussed in [10, 11]. In [12, 13, 14], synchronous operation of a broadcast star is considered, and performance is analyzed. An exact analysis is developed for the network with an infinite number of stations, and an approximate analysis is developed for the network with a finite number of stations. Papers [15, 16] assume a broadcast star that operates under asynchronous mode, where transmissions of a packet are not confined to the beginning of slots. In [15, 16], a broadcast star is modeled as a polling system, and an approximate analysis is developed.

Paper [2] extends the idea of collision avoidance from a broadcast star to a tree network. This new architecture is called the Collision Avoidance Multiple Broadcast (CAMB) tree network. In a CAMB tree network, collision avoidance switches are organized into a tree topology. The switches are the internal nodes, and the stations are the leaves of the tree. A packet reaches its destination by climbing the tree and being broadcast by the switch that is the root of the minimal subtree containing both the source and the destination stations. Due to the segmented nature of the network, concurrent transmissions are possible in the CAMB tree network.

Station and switch protocols for the CAMB tree are studied in [4, 5]. A switch design based on TTL devices is given in [8]. A switch design based on photonic devices and simulated performance study for this network are given in [4, 6]. A performance study on the CAMB tree network through

theoretical analysis and simulations is given in [17, 18]. Results show that the CAMB tree shows a better performance in comparison with a broadcast star.

In this paper, we present a design and implementation for the CAMB tree network. In section 2, a description of the CAMB tree network is presented. A design and implementation of the CAMB tree network is given in section 3. In section 4, performance measurements done on the experimental CAMB tree network designed are presented. In section 5, concluding remarks are given.

2 The CAMB Tree Network

The CAMB tree network consists of collision avoidance switches organized in a tree topology. Each switch is a node in the tree. The stations are the leaves of the tree. Each transmission line consists of uplinks and downlinks. A station (or a switch) uses the uplink to send packets and uses the downlink to receive packets. A station sends a packet as soon as one is available. A packet reaches its destination by climbing the tree and being broadcast by its "proper ancestor", the root switch of the minimal subtree containing both the source and the destination stations (Fig. 1). Note that each switch on the minimal subtree is responsible for transmitting a packet to its parent switch and for broadcasting the packet to its children switches.

CAMB Collision Avoidance Switch Protocol

A CAMB switch is connected to children switches (or stations) by uplinks and downlinks. It is also connected to a parent switch by an uplink and a downlink. A switch can receive packets either from its children's uplinks or from its parent's downlink. When a switch receives packets from its children's uplinks, the switch executes the following protocol:

1. The switch selects one of the incoming packets at random and checks the header of the selected packet to see if it is the proper ancestor of the packet.
2. IF the switch is not the proper ancestor, the packet is transmitted to the parent uplink,
3. ELSE (if the switch is the proper ancestor)
   (a) IF there is a packet from the parent downlink being broadcast, the packet from a child is discarded (this is called "broadcast preemption"),
   (b) ELSE (if there is no packet from parent switch) the packet from a child is broadcast to the children's downlinks and to the parent uplink.

Note that while a selected packet is being transmitted, all the other children's uplinks are blocked.

When a switch receives a packet from its parent's downlink, the switch executes the following protocol:

1. IF the switch is idle, the packet from the parent is broadcast to the children's downlinks.
2. ELSE (if it is busy broadcasting a packet from a child) this packet from a child is truncated (this is refered to as an "abort") and the packet from the parent is broadcast to the children's downlinks.

In the switch protocol given above, packets received from the parent downlink are given priority over those received from the children's uplinks. If instead, the switch blocks the packets received from the parent downlink, then it would be possible for a source station to receive the broadcast of its packet without the broadcast being received by the destination station. This situation is incompatible with

the station protocol, in which the source station uses the receipt of its packet as an indication that the packet was also broadcast to the destination station.

Station Protocol

The CAMB station protocol is based upon the station monitoring its downlink for the broadcast of its packet. It is very simple and similar to random access protocols like ALOHA. The CAMB station protocol is as follows:

1. A station transmits a packet as soon as one becomes available (say at time t).
2. The station monitors its downlink for the broadcast of its packet.
3. IF the station does not see the start of its packet by time $t + R_{pa}$, where R_{pa} = round trip propagation delay time between the station and the proper ancestor, then it retransmits the packet immediately,
4. ELSE (the start of the packet is seen within this time)
   (a) IF it sees the broadcast of the packet truncated by the broadcast of another packet, then it retransmits the packet immediately,
   (b) ELSE the station sees the broadcast of the entire packet (i.e., the transmission is successful and the station can transmit a new packet).

Note that since the priority is given to parent's downlinks, packet broadcasts may be aborted at any point before completion. Therefore a station must see the broadcast of its entire packet before it can assume a successful transmission.

Concurrency of Transmissions

Concurrent transmissions are possible in the CAMB tree. The switch protocol is designed in such a way that, regardless of the level of a switch, a packet is always passed to the parent switch. Every time the climbing packet busies a switch above its proper ancestor, it prevents stations beneath that switch from using the switch as a broadcast point. In other words, every time a packet busies the switch, it creates partitions that are the children subtrees of that switch. Within each of these partitions, broadcasts may occur. Any attempt to transmit from within a partition to a destination outside the partition does not succeed. We note here that in [6], it is shown that the delay decreases significantly as the degree of concurrency increases.

Fig. 1 gives an example of a transmission in a CAMB tree network. Station 1 transmits a packet to station 4. Switch E, which is the proper ancestor of this packet, sends it to the parent switch G, and broadcasts it to its subtree (switch A and B). If the switch is not the proper ancestor, a packet is only passed to the parent switch. In Fig. 1 switch A passes the packet to its parent switch E only. Note that the packet sent by station 1 divides the network into two subtrees (X and Y). This allows a concurrent transmission within subtree Y.

3 Implementation of a CAMB Tree Network

Currently we have a completely operational prototype of the CAMB tree network. In this section, our design and implementation of the CAMB tree network is described.

In our implementation, we followed the protocol layering architectures of the IEEE 802 local area networks. Fig. 2 shows the layering architecture of the CAMB tree network and the components that implement each layer. The Physical layer consists of CAMB switches, transmission cables and transceivers located on the station/network [2] interface board. The MAC layer and a part of the LLC layer are implemented in the station/network interface board installed in the station. We have also provided the TCP/IP protocol (implemented in the workstation) on top of the IEEE 802 protocol stack.

3.1 Packet Format and Addressing Scheme

In the following, we describe the packet format and addressing scheme used in our implementation of the CAMB tree network.

Packet Format

The packet format used in the current implementation of the CAMB tree network has the following seven fields: Destination Address, Control Bit, Reserved, Source Address, Packet Size, Data and CRC (Fig. 4).

The Destination Address field (8 bits long) indicates the destination address. The next field, Control Bit, is used to indicate whether the packet has reached its proper ancestor. This bit is initially set to 1 by the sending station. It remains 1 until the packet reaches the proper ancestor. When this happens, the proper ancestor switch resets this bit to 0. The Reserved field (7 bits long) is reserved for future use. The Source Address field (8 bits long) indicates the source address. The Packet Size field (16 bits long) indicates the number of bits contained in the Data field. This information is used by the stations to determine whether a packet has been aborted. The Data field contains the user data, and it is variable in length. The CRC field is used to check the correctness of the received packet (header and data). The switch examines only the Destination Address and the Control Bit field. All the other fields are used by the receiving station.

Addressing Scheme

Our addressing scheme is shown in Fig. 5. Stations are ordered in ascending order from the left to the right of the tree. Addresses are assigned to the switches in the following way. Each switch has an eight bit address. However, at level i, only the $2 \times (i-1)$ left most bits are significant. The other bits are ignored. In our addressing scheme, an address is assigned to a switch in such a way that the first $2 \times (i-1)$ bits of the address of a switch on level i match with the first $2 \times (i-1)$ bits of the addresses of all of the stations in its subtree.

This numbering simplifies the determination of proper ancestor switches. A switch on level i checks the $2 \times (i-1)$ leftmost bits of the address field of a packet. If it is equal to the $2 \times (i-1)$ leftmost bits of its own address, then it is the proper ancestor of the packet. In this case, (as explained before), the packet is sent to both the parent switch and the children switches. Otherwise, the packet is only sent to the parent switch.

[2] The station used in this implementation is a LUNA workstation. The LUNA is a UNIX based workstation that uses the 32 bit CPU M68030 (20 MHz) and has a 68881 (20 MHz) floating point co-processor. The processing speed is 4 MIPS. The workstation supports communication protocols such as TCP/IP, NSF, and X.25.

We use an example to show how packets are transmitted on a CAMB tree using the above addressing scheme and packet format. Fig. 5 shows a transmission of a packet from station A (address 00000000) to station B (address 00000100). When station A sends a packet, its immediate parent switch, Z (address 000000xx), checks the destination address of the packet. As the six leftmost bits of the destination address in the packet header are different from the switch address, it sends the packet only to its parent switch (Y). Switch Y (address 0000xxxx) then checks the four leftmost bits of the destination address of the packet. This time, they match with the switch's address, so switch Y knows that it is the proper ancestor of the packet. The packet is thus broadcast to its children. The packet is also sent to the parent switch (X). Now, the switch X (address = 00xxxxxx) checks the two leftmost bits of the destination address field. They agree with the switch address. Therefore, the switch X thinks that it is the proper ancestor of the packet and becomes ready to broadcast the packet to its children. However, if the switch X actually broadcasts the packet, the packet is broadcast twice (once by switch Y and once by switch X). To avoid this inefficiency, the Control Bit is used (see packet format – Fig. 4). First, a switch checks the control bit in the packet header. If it is 1, the switch knows that the packet has already been broadcast by its proper ancestor, so it will transmit the packet only to the parent switch. Otherwise, it will check the address field to see if it is the proper ancestor of the packet. In the example shown above, the switch X first checks the control bit. As it is 0, the switch does not broadcast the packet.

3.2 CAMB Switch

The CAMB switch can be realized by the following three switch sections. See Fig. 3.

- The *Uplink Selector (US)* – is responsible for detecting an incoming packet from the children's uplinks and selecting one in case of simultaneous packet arrivals.
- The *Address Recognizer (AR)* – checks the header of the selected packet to see if the switch is the proper ancestor. If it is, the AR sends the packet to the DS and to the parent switch, otherwise the AR sends the packet only to the parent switch.
- The *Downlink Selector (DS)* – receives packets from the AR and from the parent's downlink from the parent switch, and broadcasts to all the children switches. Priority is given to a packet from the parent downlink. In other words, in case of simultaneous arrivals of a packet from the AR and a packet from the parent downlink, the packet from the parent downlink is selected. If the DS is broadcasting a packet received from the AR upon the arrival of a packet from the parent downlink, the transmission of the packet from the AR is aborted, and the packet from the parent is broadcast. If the DS is broadcasting a packet received from the parent upon the arrival of a packet from the AR, the packet from the AR is simply discarded.

3.2.1 TTL Switch Design

We present a switch design which supports four children switches (Fig. 3). Thus, each switch can interface with four devices (stations or other switches). Note that this design can be easily extended to support more than four children per switch. Each transmission link (either up- or down-link) consists of a control path and a data path. The control path signals the beginning and end of each packet. This line is high whenever there is a valid packet transmission on the data line. Otherwise it is low, signaling the switch that there is no packet being transmitted. The data path carries the packet sent to the switch.

We first present an implementation based on TTL devices and then give a VLSI implementation. We use the packet format and the addressing scheme discussed above. The design of each of the three sections (US, AR, DS) of the switch is explained below.

Uplink Selector (US)

Fig. 6 shows a possible design of the US section of the CAMB switch. The US has been broken down into four units.

- The *Synchronizer* – aligns an incoming packet with the switch's internal clock. Note that if several switches in the network share the same clock signal, this unit is not required between interconnected switch nodes. Removal of the unit will speed up packet transmission.
- The *Start Recognizer* – detects the start of a new packet by checking the control line. This unit is also responsible for blocking all other uplinks while a selected packet is being transmitted.
- The *Priority Resolver* – selects one packet at random when more than one packet from children's uplinks arrive simultaneously. Note that the Start Recognizer and the Priority Resolver only handle the control path, namely, they do not have connections to the data path. The Priority Resolver selects the packet based on the signal from the control lines and then signals the Multiplexer as to which packet has been selected. The *New'* line from the AR signals the end of a packet transmission, reseting the Priority Resolver.
- The *Multiplexer* – allows only the selected packet to go through to the Address Recognizer.

In summary, the synchronizer receives packets from uplinks; the start recognizer detects the beginning of the packets, activating the priority resolver, which will randomly select one of the packets based on the control lines; the selected packet is allowed to go through the multiplexer.

Address Recognizer (AR)

Fig. 7 shows the schematic of the AR section of the switch. It consists of 5 units.

- The *Shift Register* – allows the Comparator to check the destination address of a packet. The destination address of the packet is contained in the Shift Register for one clock cycle. During that cycle, the Comparator can check the the destination address of that packet. An incoming packet goes through the shift register and is always sent to the parent switch.
- The *Comparator* – generates a signal (*Cmp*) to indicate whether the destination address of the incoming packet matches with the switch address. It indicates whether the switch is a proper ancestor of the packet. The switch address is configured in a block of 8 SPST switches.
- The *Routing Logic* – generates a signal (*Broadcast*) to indicate if the incoming packet will be sent to the Downlink Selector to be broadcast or not. This signal is generated based on three signals received by the routing logic: *Cmp*, *Control Bit*, *Parent Int'*. When *Cmp* is low (indicating the switch is a proper ancestor of the packet), and the Control Bit of a packet is high (indicating the packet has not been broadcast by its proper ancestor yet), and *Parent Int'* is high (indicating there is no transmission from the parent switch), then a high *Broadcast* signal will be generated in order to broadcast the incoming packet. Otherwise, the *Broadcast* signal will be low. *Ddn* and *Cdn* lines send the data and control lines, respectively, from the US to the DS.

- The *Reset Control Bit Logic* – resets the Control Bit (see packet format – Fig. 4) in the incoming packet whenever the switch recognizes that it is the proper ancestor of the packet. As explained earlier in this section, this will prevent broadcast of a packet by a switch at a higher level.

- The *End Recognizer* – which detects either (1) the end of a packet going through the shift register (by checking if the control line from the US is low) or (2) a transmission abort (by checking if the *Parent Int'* line is low, which indicates a transmission from the parent). When one of these events happens, the End Recognizer sends a reset signal (*New'*) to the Uplink Selector. Upon the reset, the US waits for new incoming packets. Whenever there is a transmission from the parent, the *Parent Int'* line is low.

Downlink Selector (DS)

The DS section is shown in Fig. 8. It is divided into two units.

- The *Selector* – selects a packet from the parent downlink or from the Address Recognizer and broadcasts it to all the children switches. As described previously, priority is given to the parent's downlink over the children's uplink. If a packet from the parent downlink arrives during transmission of a packet from the AR, the *Selector* will abort the packet transmission from the AR. In addition, any packet from the AR will be blocked during the entire period of transmission from the parent switch.

- The *Flip-flop* – is used to set the control lines to children switches low in case of an abort. If the control line goes low, it indicates the end of a transmission. This can be the end of a packet transmission or a truncated packet (abort). To detect an abort, a station needs to check the packet size indicated in the packet header against the actual size of the received packet.

A full schematic of the TTL tree switch based on the above design is given in [19].

We also implemented a switch in VLSI. The resulting layout is 1497.5 microns (width) by 1549.5 microns (height). In our performance tests, a switch is driven with the clock speed of 10MHz, which is the normal Ethernet channel speed. The result obtained shows correct functionality with a maximum input to output delay of 32.78 ns.

3.3 Station/Network Interface Board

The interface board is designed to connect a station to the CAMB tree network. As described in the beginning of section 3, the interface board provides the functions corresponding to part of the physical layer, MAC layer and a part of LLC layer (Fig.2). It implements the station protocol described in section 2.

Interface Board Design

The interface board functions can be realized by the following four sections (Fig. 9).

- The *Station Interface Section (SIS)* – is responsible for the transfer of data and control information between the station and the interface board.
- The *Microprocessor Control Section (MCS)* – manages data movement, data buffering, and header processing for packets transmission and reception.

- The *Transmission Section (TS)* – accepts a packet from the MCS, generates CRC for the packet, and transmits the packet to the switches.
- The *Reception Section (RS)* – receives incoming packets, performs CRC error checking, and buffers the incoming packets for transfer to the MCS.

The Station Interface Section (SIS)

Fig. 10 shows the schematic of the SIS section of the interface board. The SIS links directly with the station. SIS has been broken into the following units.

- The *Bus Transceiver* – provides an asynchronous two-way communication between (1) the address and data busses within SIS and (2) the address and data busses within the station.
- The *Interface Buffer (IB)* – stores packets to be transmitted.
- The *Interface Status Register (ISR)* – stores a control word from the station, which indicates "new packet in Interface Buffer" or "Station Buffer free".
- The *Station Buffer (SB)* – stores a packet to be passed to the station.
- The *Station Status Register (SSR)* – stores a control word from the interface board, which indicates "new packet in Station Buffer" or "Interface Buffer free".

The Microprocessor Control Section (MCS)

The block diagram of the MCS is given in Fig. 11. The MCS controls the other three sections of the interface board (SIS, TS, RS). The MCS consists of the following units:

- The *Central Processor Unit (CPU)* – implements the control logic that drives all the units in the interface board. It executes a small firmware code contained in the Read Only Memory (ROM). The CPU performs packet processing (creates and appends a header to outgoing packets, processes the header of incoming packets), moves packets among Interface and Station Buffers (in SIS), RAM (in MCS) and FIFOs (in TS and RS), and controls transmission and reception of packets. The CPU is a 16-bit microprocessor, operating at 10 MHz. Internally it has the following modules.
  - Two DMA (direct memory access) modules, one used to transfer data from Random Access Memory (RAM) to FIFO (in TS), in case of transmission of a packet. The other is used to transfer data from FIFO (in RS) to RAM, in case of reception of a packet.
  - A clock generator module, used to generate a clock signal to the CPU and the other units.
  - A programmable timer module, used as a retransmission timer. When the timer counter reaches zero, an internal interrupt is sent to the CPU, and a retransmission takes place.
- The *Read Only Memory (ROM)* – contains the code that is executed by the CPU.
- The *Random Access Memory (RAM)* – provides the internal transmission buffer and the internal reception buffer. It is also used to store information necessary for the operation of the CPU such as variables and interrupt vectors.

The Transmission Section (TS)

The functions of the Transmission Section are provided by the following three units (Fig. 12):

- The *Transmission FIFO* – is a First-In First-Out memory used to store the packet that is currently being transmitted. The MCS loads the packet into the FIFO and starts the transmission. If the broadcast of the transmitted packet is not received before the timeout, the MCS triggers the retransmission of the packet in the FIFO. Note that the FIFO is responsible for the conversion of the packet from parallel to serial format.
- The *CRC Generator* – is used to generate the CRC code on the fly for the currently transmitted packet.
- The *Transmission Transceiver* – is used to interface the board with the transmission cable (of a tree network).

The Reception Section (RS)

Fig. 13 shows the block diagram of a design of the Reception Section. It consists of the following four units:

- The *Reception FIFO* – is a First-In First-Out memory used to store incoming packets from the network (i.e., a CAMB switch). When the first bit of a packet is received, the CRC Checker starts calculating the CRC, and the incoming packet is stored in the reception FIFO. As soon as the first byte of the packet is stored in the reception FIFO, the DMA module (in the MCS) starts transfering the data from the FIFO to the MCS internal reception buffer. When the DMA reads the last byte of an incoming packet, the MCS is informed that a new packet has just arrived. Note that the FIFO is also responsible for the conversion of the packet received from serial to parallel format.
- The *Interrupt Logic* – starts DMA transfer when a packet arrives in the FIFO and informs the MCS the a packet has been received.
- The *CRC Checker* – is used to check the CRC code of the packet being received on the fly.
- The *Reception Transceiver* – is used to interface the cable with the board.

A complete circuit schematic of the Station/Network interface board is given in [19].

3.3.1 The Interface Software

In this section, the firmware (contained in the ROM, located in the MCS) that the CPU executes is described (Fig. 11).

The firmware consists of two concurrent processes: a packet transmission process, and a packet reception process. These processes are both interrupt driven. In other words, an appropriate process is triggered either when (1) a new packet to be transmitted is left in the Interface Buffer, or (2) when the RS signals that a packet is being received.

The Transmission Process

The transmission process consists of two steps: the transfer of the packet from the station to MCS and the transfer of the packet from the MCS to the CAMB switch. First we describe how the packet is transferred from the station to the MCS.

The transmission process starts when the station writes a packet to the IB (in SIS) and a control word to the ISR (in SIS). This interrupts the microprocessor within the MCS in the interface board to start the following transmission algorithm.

1. The microprocessor reads the control word at the ISR (in SIS).

2. The control word indicates that there is a packet in the IB (in SIS). This causes the microprocessor to transfer the packet to either (1) the TS if the TS is not busy, or (2) the MCS internal transmission buffer (in RAM) otherwise.
3. The microprocessor writes a control word in the SSR (in SIS), indicating that the IB is available to receive another packet.

At this point, a packet has now moved from the station and is stored in the internal transmission buffer in the MCS. In the following we describe how the packet is transmitted to the network from the MCS.

If the TS is not transmitting a packet, the microprocessor resets the TS and programs its DMA (in the MCS) to transfer the packet from the MCS internal transmission buffer to the TS's FIFO. When the transfer is complete, the microprocessor starts the retransmission timer and signals the FIFO to start the transmission. If a broadcast of the packet is not received by the RS when the time-out occurs, the microprocessor resets the timer and retransmits the packet. This process continues until a broadcast of the packet is received by the RS (successful transmission).

When a broadcast of the transmitted packet is received, the reception process will set a flag to stop the transmission process. The transmission process resets the TS and looks in the transmission buffer for a new packet to be transmitted.

The Reception Process

The reception process starts when a packet (from a CAMB switch) arrives at RS on the interface board. Upon the arrival of a packet at RS, the first byte of the incoming packet is moved into the reception FIFO. This triggers the DMA module in MCS to start the byte-by-byte transfer of the packet from the reception FIFO to the internal reception buffer in MCS.

When the last byte of the packet is transferred from the reception FIFO to the MCS internal reception buffer, an interrupt is generated to signal the reception process to check the last bit of the received packet (indicates if CRC check was correct or not). If the bit is equal to one (CRC correct) and the number of bytes received matches that of the Packet Size field, the packet has been received correctly. Otherwise, an error has been detected (CRC error or packet aborted).

In the case where an error has been detected, the erroneous packet is discarded, and if this packet had been sent by the station, a flag is set to inform the transmission process to retransmit the packet.

Suppose the packet has no error (CRC and packet size correct). After the first three bytes of the packet (the destination address, a reserved byte and the source address) are moved to the internal reception buffer, the reception process examines them and determines which of the following three cases are true.

1. The destination address matches that of the receiving station. In this case, the packet will be kept in the internal reception buffer (in RAM) for transfer to the station. When the Station Buffer in the SIS is empty, the packet is transferred from the internal reception buffer to the Station Buffer. When the station finishes reading the packet from the Station Buffer, it writes a control word in the Interface Status Register, interrupting the interface board. This interrupt signals the reception process to search for more packets in the internal reception buffer (in MCS) to be transferred to the station.
2. The source address matches that of the receiving station. In this case, the reception process notifies the transmission process of the successful transmission of the packet.

3. Neither the source nor the destination address of the packet matches that of the station. In this case, the incoming packet is discarded from the internal reception buffer.

3.4 Network Protocol

In the previous sections, we described the design and implementation of the Physical layer and Data link layer (MAC and LLC layers) of the CAMB tree network. In this section, we describe transport protocol support for the CAMB tree network.

In our CAMB tree network implementation, we used Unix-based LUNA [3] workstations as network stations. LUNA workstations support TCP/IP and UDP/IP. To support these transport protocols on the top of our CAMB tree network layering architecture, we implemented an interface between IP and the interface board.

The most efficient way to support IP on our CAMB tree network is to design and implement a device driver for this purpose. However, this will make the implementation hardware dependent and not easily portable to other machines. Therefore, our approach is to provide an interface that is hardware independent by using the LUNA workstation's general device driver as a base. A similar design approach was used at the University of California at Irvine to connect tty lines into the ARPA Internet [20].

In our implementation (Fig. 14), TCP passes a packet to IP, and IP then passes the packet to the pseudo-device called the Raw Packet Interface (RPI). The packet is then handed to the Tree Network Daemon process through a raw socket interface. Finally, the packet is handed to the Station Device Driver that sends it to the Tree Network interface board. (The reception procedure is the reverse of the transmission procedure.)

4 Performance Measurements

4.1 Configuration

In this section, we present performance measurements of the CAMB Tree Network derived through experiments. Our experimental network consists of two configurations: (1) a single CAMB switch supporting two stations and a traffic generator, and (2) three CAMB switches organized in a two level-tree structure supporting one station and two traffic generators. The traffic generator is used to create (artificial) traffic on the network so that various statistics can be collected under different network traffic loads.

In the single switch configuration, the performance measures of interest include the transmission delay (TD) and various processing times involved in sending and receiving packets. The TD is the time from the beginning of transmission of a packet to the time of a successful reception of the packet. TD consists of the Transmission Time (TT) and the Reception Time (RT). TT includes the following times associated with the transmission of a packet.

- Transmission Processing Time (TPT) – includes the time required to process a packet header and to initialize a timer.
- Application-to-Transmission-FIFO data move time (ATF) – It includes the time to move data from the application to the transmission FIFO.

The Reception Time (RT) includes the following times associated with the reception of a packet.

[3] LUNA is a trademark of Omron.

- Reception Processing Time (RPT) – includes the time required to process a packet header and to manage buffer.
- FIFO-to-Reception-Buffer data move time (FRB) – includes the time to move data from the FIFO to the reception buffer.
- Reception-Buffer-to-Application data move time (RBA) – includes the time to move data from the reception buffer to the application.

In the two level tree configuration, the performance measurements include the transmission delay under various source/destination (S/D) traffic distributions to study the effect of various degrees of traffic locality in the tree network.

In the following numerical examples, we measure delays under various traffic loads on the network. In our experiments, we vary the traffic load by changing the rate at which packets are generated by a traffic generator. The traffic load ρ is defined as:

$$\rho = \frac{t_{cb}}{t_{cb} + t_{ci}},$$

where t_{cb} is the time period when the channel is busy transmitting a packet, and t_{ci} is the time period when the channel is idle. In all the figures, the delays are measured in milliseconds, and the packet size is given in bits.

4.2 Experimental Results

Fig. 15 shows the average of the Transmission Delay ($E[TD]$) as a function of the traffic load (ρ). The packet size used is 800 bits. It is observed that the average Transmission Delay remains very small until the traffic load reaches 0.93. After this point, the average Transmission Delay grows rapidly. This figure shows that the performance degradation due to conflicts at a switch among different packets is insignificant for most traffic loads. The average Transmission Delay remains very small for a wide range of traffic load ($0 \geq \rho \geq 0.93$).

In Figs. 16 through 18, we will show measurement results of various delay elements and examine how the Transmission Delay and various processing times are affected when the packet length changes. In these figures, there is no interfering traffic on the network (i.e., zero traffic from the traffic generator).

Fig. 16 shows the average Transmission Delay $E[TD]$, the average Reception Time ($E[RT]$) and the average Transmission Time ($E[TT]$) as a function of the packet size when the traffic from the traffic generator is zero. Note that $TD = RT + TT$. As the packet size increases, the average delays increase linearly. This is because the time required to move data (which is in proportion to the packet size) contributes to the delays most significantly. This is verified in Figs. 17 and 18.

Fig. 17 divides the average Transmission Time ($E[TT]$) into the average Transmission Processing Time ($E[TPT]$) and the average Application-to-Transmission-FIFO data move time ($E[ATF]$). Fig. 18 divides the average Reception Time ($E[RT]$) into the average Reception Processing Time ($E[RPT]$), the average FIFO-to-Reception-Buffer data move time ($E[FRB]$), and the average Reception-Buffer-to-Applcation data move time ($E[RBA]$).

From Figs. 17 and 18, we see that the average Transmission Processing Time ($E[TPT]$) and the average Reception Processing Time ($E[RPT]$) remain constant as the packet size increases. However, the data move times ($E[ATF]$, $E[FRB]$ and $E[RBA]$) increase as the packet size increases. Furthermore,

when the packet size is large, the times required for packet processing ($E[TPT]$ and $E[RPT]$) is very small compared to the time required for the data move. Since the data move is the predominant overhead, the transmission delay increases as the packet size increases.

Fig. 19 shows the average transmission delay of the the two-level CAMB tree under various S/D traffic distributions. The horizontal axes gives the traffic load ρ provided by each traffic generator.

The experimental results show the effect of locality of transmission on performance in the two-level tree. When all packets are destined to stations located in the other subtree (0% locality), and hence, all packets must be broadcast by the root switch, the average delay in a two-level tree is the same as that in a broadcast star. (This case gives the upper bound on the delay in a two-level tree network because no concurrent transmissions are possible.) On the other hand, if all communications are local within each subtree (100% locality), the tree behaves like two independent broadcast star networks. (This case gives the lower bound on the delay in a two-level tree network.) In between these two extremes, varying the degree of locality of communication yields different performance for the tree network. To show the effect of intermediate degrees of locality, we measured the transmission delay of a two-level tree network with two S/D distribution: 20% and 80% locality (With a 20% O/D matrix, 20% of the traffic goes to a station within the local subtree and only 80% of the traffic passes through the root switch, whereas with the 80% locality O/D matrix, 80% of the traffic goes to a station within the local subtree and only 20% of the traffic passes through the root switch). From Fig. 19 we see that a tree network with a higher degree of locality yields smaller delay. This reflects a decrease in the preemption and abortion of packets at the lower level tree switches, and thus an increase in the concurrency of successful transmissions in the network.

5 Concluding Remarks

In this paper, a design and implementation of the CAMB Tree Network is described. CAMB switches, the Interface Board and station protocols provide the physical layer, data link layer and higher layer protocols (TCP/IP) for the Tree Network. Performance measurements on a one level and a two level configuration of the Tree Network were also described. The results show transmission delay remains small for a wide traffic range. Our experiments show that the largest overhead is in moving the incoming data from FIFO to the reception buffer (FRB). We also showed that the the two level tree gives a better performance than the one level tree (broadcast star), specially when the percentage of traffic locality is high. The next step of this research is (1) to build a simulation model of the CAMB tree Network, (2) to show that the results obtained from the model correspond to the ones obtained from the experimental network, and (3) to use the simulation model to study the behavior of a larger size tree network.

References

[1] A. Albanese, "Star Network with Collision-Avoidance Circuits," *The Bell System Technical Journal*, vol. 62, pp. 631–638, Mar. 1983.

[2] T. Suda, Y. Yemini, and M. Schwartz, "Tree Network with Collision Avoidance Switches," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pp. 105–113, IEEE, 1984.

[3] J. Kurose, M. Schwartz, and Y. Yemini, "Multiple-Access Protocols and Time-Constrained Communication," *ACM Computing Surveys*, vol. 16, Mar. 1984.

[4] T. Suda, S. Morris, and T. Nguyen, "Tree LANs with Collision Avoidance: Protocol, Switch Architecture and Simulated Performance," in *Proc. of the ACM SIGCOMM Symposium*, pp. 155–164, ACM, 1988.

[5] T. Suda and S. Morris, "Tree LANs with Collision Avoidance: Station and Switch Protocol," in *Computer Networks and ISDN Systems 17*, pp. 101–110, North-Holland, 1989.

[6] S. Morris, T. Suda, and T. Nguyen, "Tree LANs with Collision Avoidance: Photonic Switch Design and Simulated Performance," in *Computer Networks and ISDN Systems 17*, North-Holland, 1989.

[7] F. Closs and R. P. Lee, "A Multi-Star Broadcast Network for Local-Area Communications," in *Local Networks for Computer Communications*, pp. 61–80, IFIP, 1981.

[8] H. K. Huang, T. Suda, and Y. Noguchi, "LAN with Collision Avoidance: Switch Implementation and Simulation Study," in *Proceedings of the* 15^{th} *Conference on Local Computer Networks*, pp. 84–92, IEEE, 1990.

[9] Y. Yemini, "Tinkernet: or, Is There Life Between LANs and PBXs?," in *Conference Record of the International Conference on Communications (ICC)*, pp. 1501–1505, IEEE, 1983.

[10] E. S. Lee and P. I. P. Boulton, "The Principles and Performance of Hubnet: A 50 Mbit/s Glass Fiber Local Area Network," *IEEE Journal on Selected Areas in Communications*, vol. 1, pp. 711–720, Nov. 1983.

[11] E. S. Lee, P. I. P. Boulton, and B. W. Thomson, "Hubnet Performance Measurement," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1025–1032, July 1988.

[12] T. Suda and K. Goto, "Performance Study of a Tree LAN with Collision Avoidance," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pp. 59–68, IEEE, 1989.

[13] K. Goto and T. Suda, "Performance Analysis of a Broadcast Star Local Area Network with Collision Avoidance: Part 1, Infinite Station Population Model," Tech. Rep. 91-10, University of California, Irvine, Feb 1991.

[14] K. Goto and T. Suda, "Performance Analysis of a Broadcast Star Local Area Network with Collision Avoidance: Part 2, Infinite Station Population Model," Tech. Rep. 91-11, University of California, Irvine, Feb 1991.

[15] A. E. Kamal, "A Performance Model for a Star Network," in *Proceedings of the Conference on Global Communications (GLOBECOM)*, IEEE, 1986.

[16] A. E. Kamal, "Star Local Area Networks: A Performance Study," *IEEE Trans. on Computers*, vol. c-36, pp. 483–499, Apr. 1987.

[17] S. Marano and A. Volpentesta, "Performance Evaluation of Alberonet by Simulation and Theoretical Analyses," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pp. 1137–1147, IEEE, 1987.

[18] V. Ielapi, S. Marano, and A. Volpentesta, "A Simulation Study for a Tree Local Area Network with Concurrent Transmissions," in *Local Communications Systems: LAN and PBX*, pp. 437–450, IFIP, 1987.

[19] H. K. Huang and T. Suda, "Design and Implementation of a Collision Avoidance Multiple Broadcast Tree Network," Tech. Rep. 91-51, University of California, Irvine, 1991.

[20] M. T. Rose, "Low Tech Connections into the ARPA Internet: The Raw Packet Split-Gateway," Tech. Rep. 216, University of California, Irvine, Feb 1984.

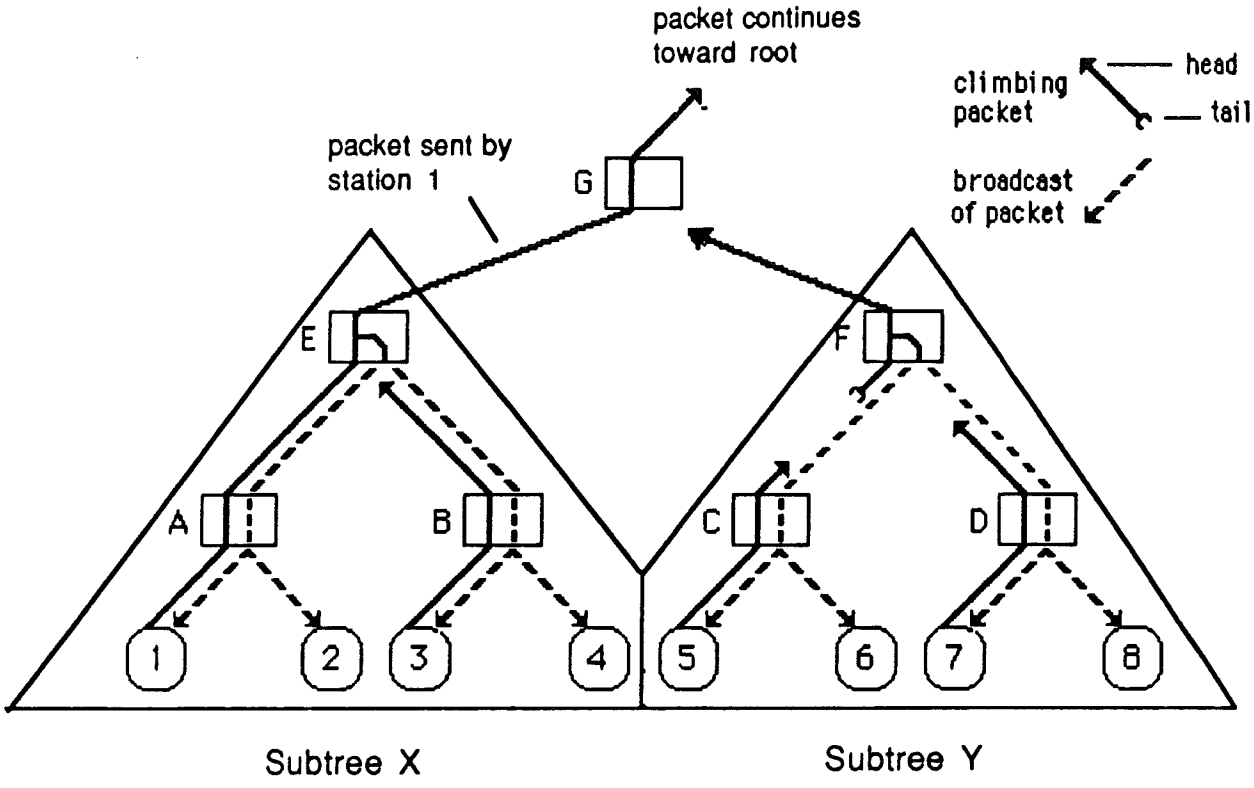


Figure 1: Trasmission in a CAMB Tree

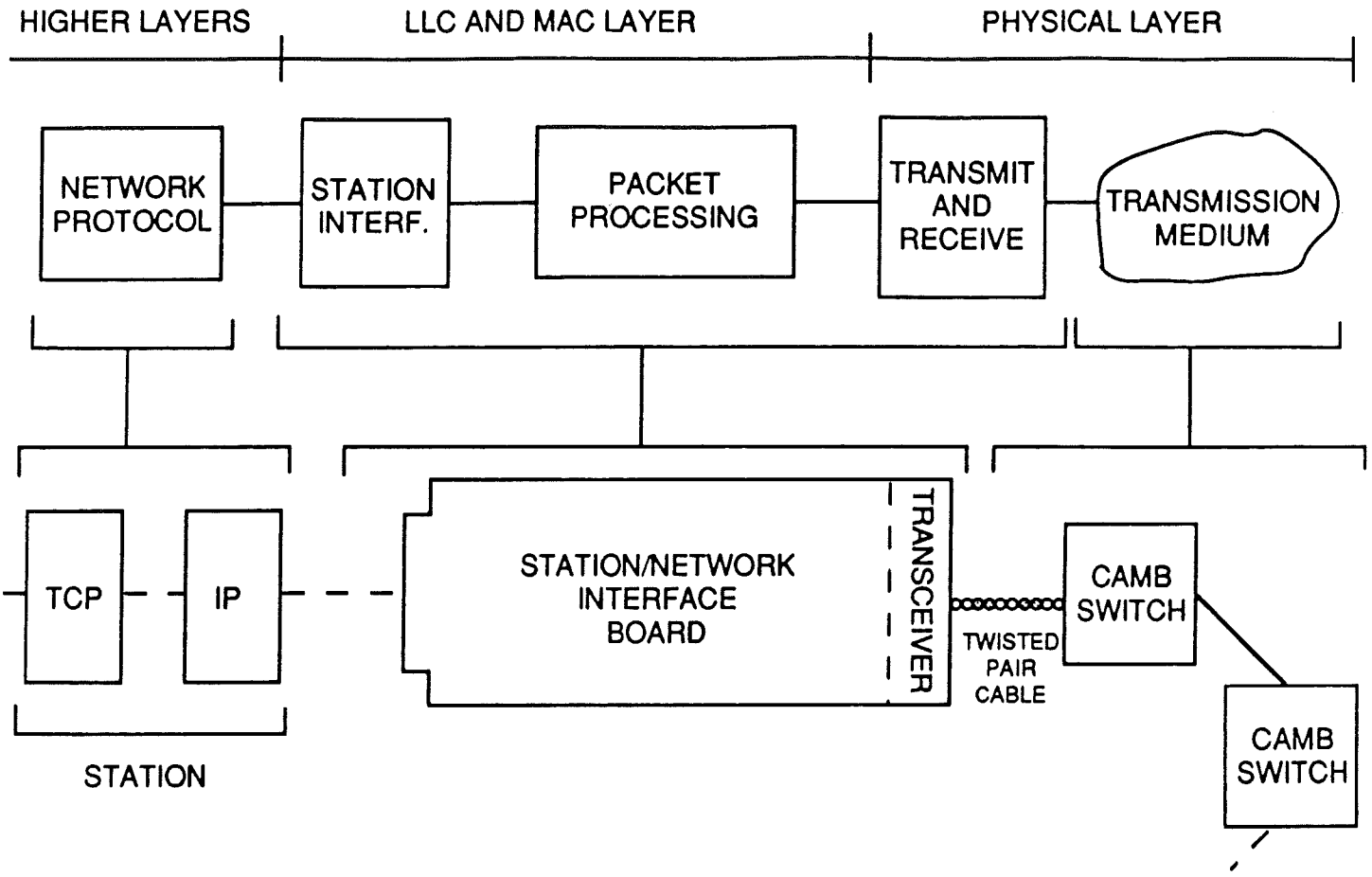


Figure 2: Layering Architecture of the CAMB Tree

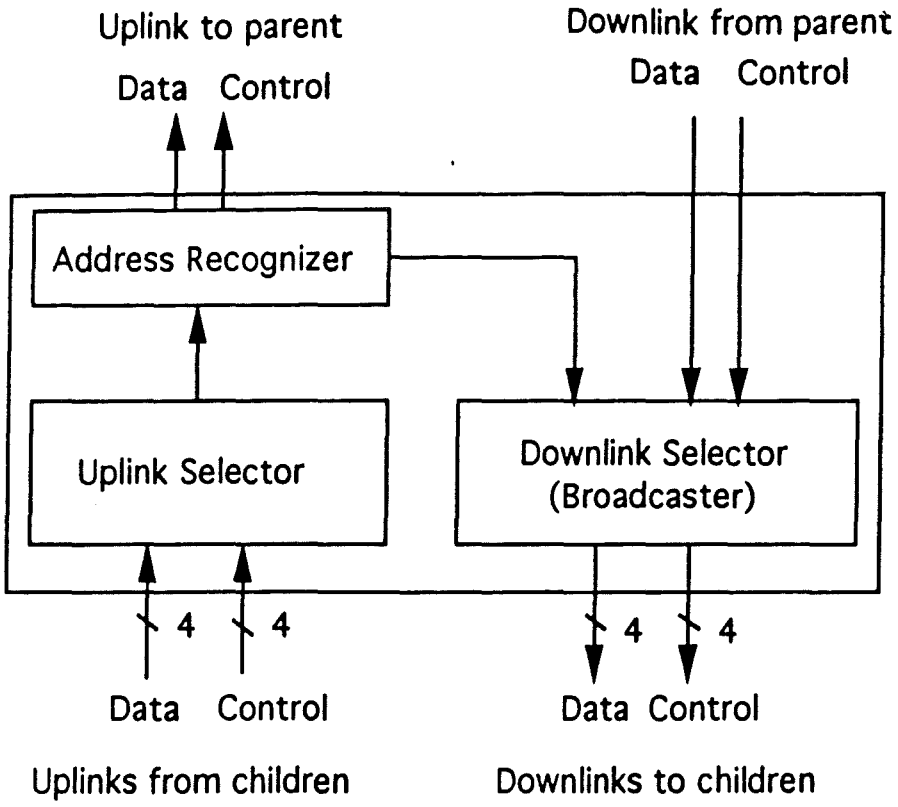


Figure 3: CAMB Switch

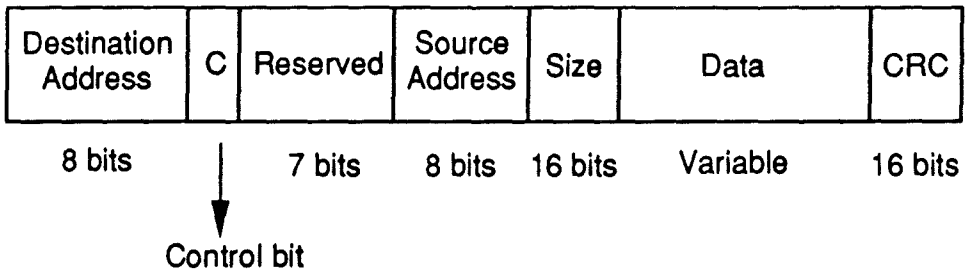


Figure 4: Packet Format

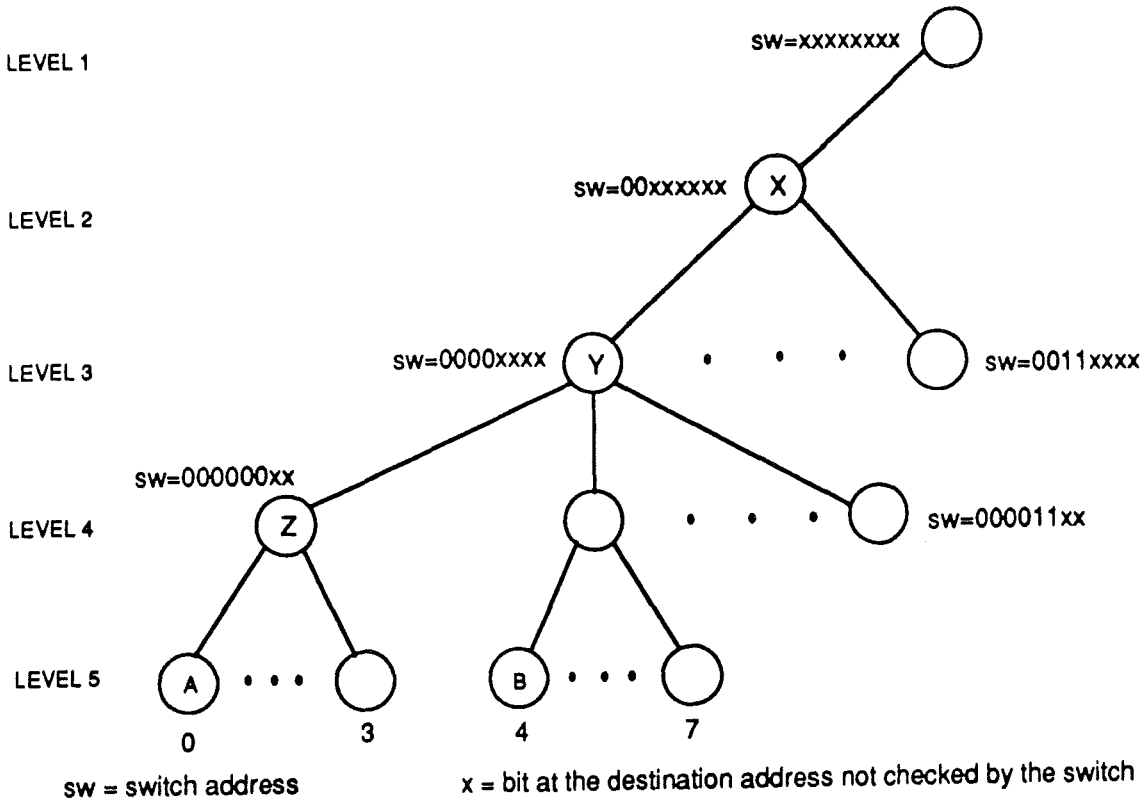


Figure 5: Addressing Scheme

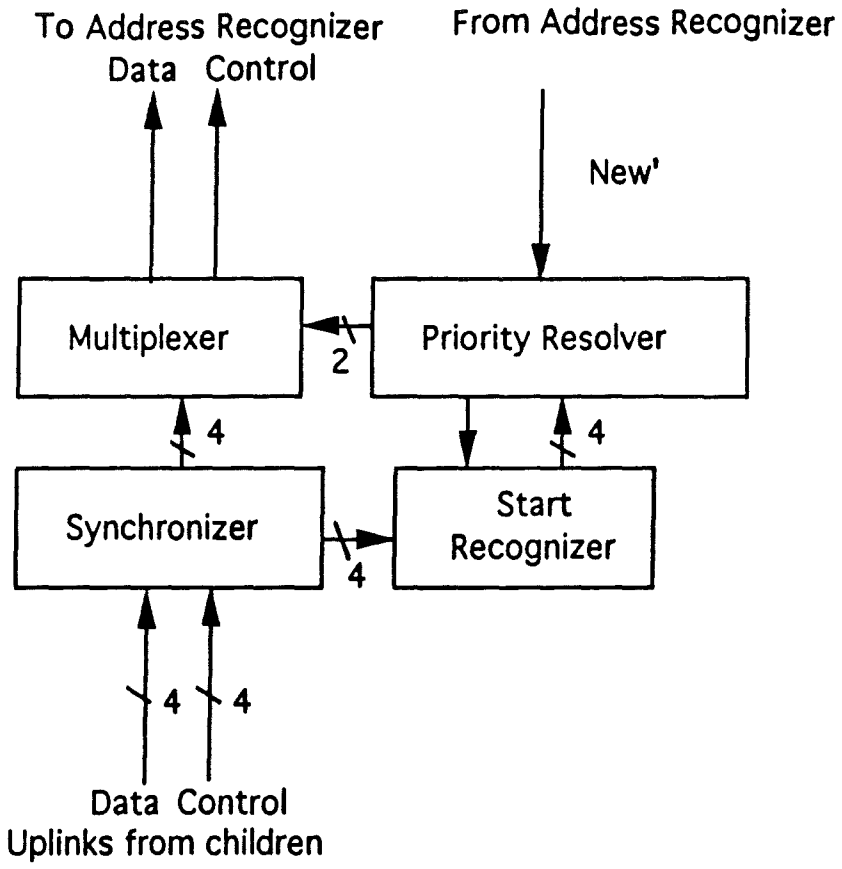


Figure 6: Uplink Selector Block Diagram

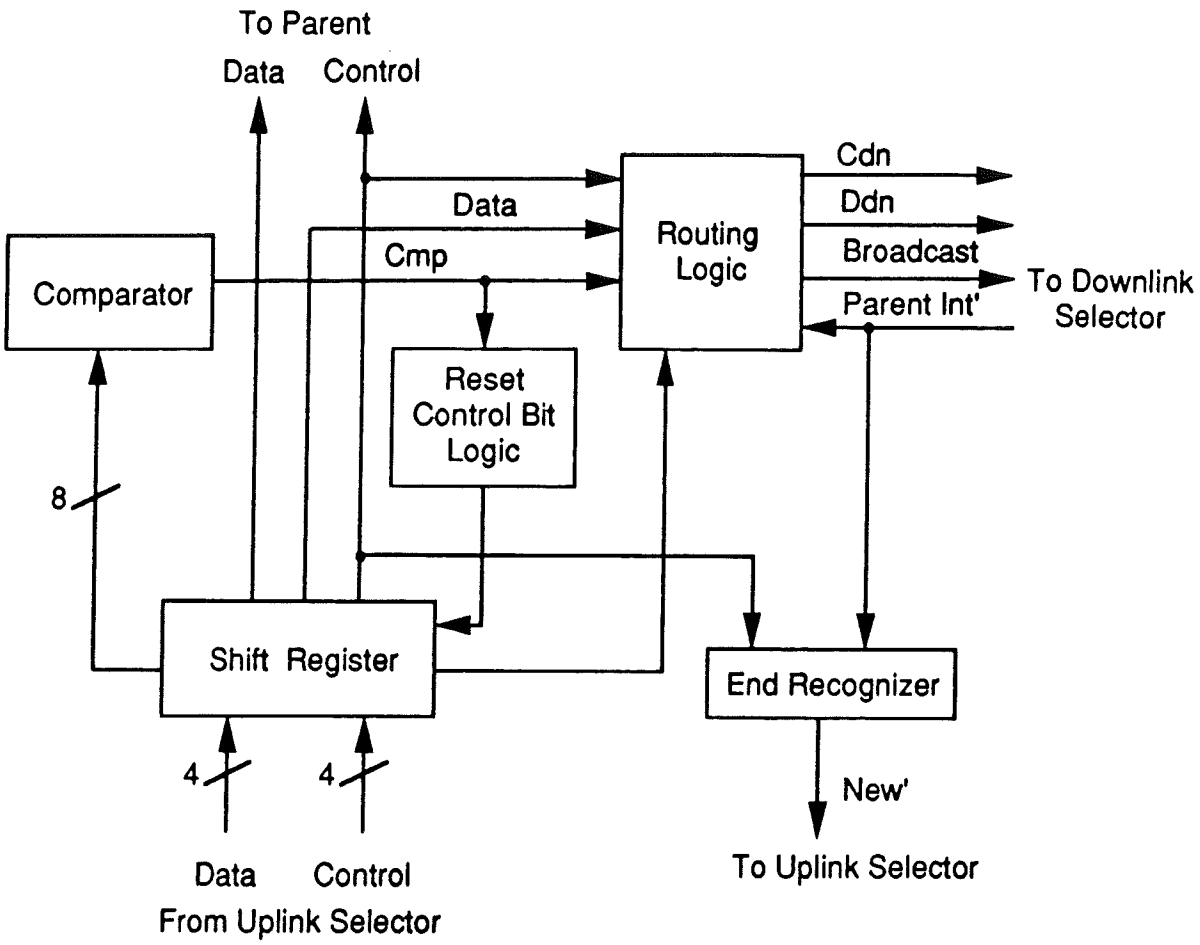


Figure 7: Address Recognizer Block Diagram

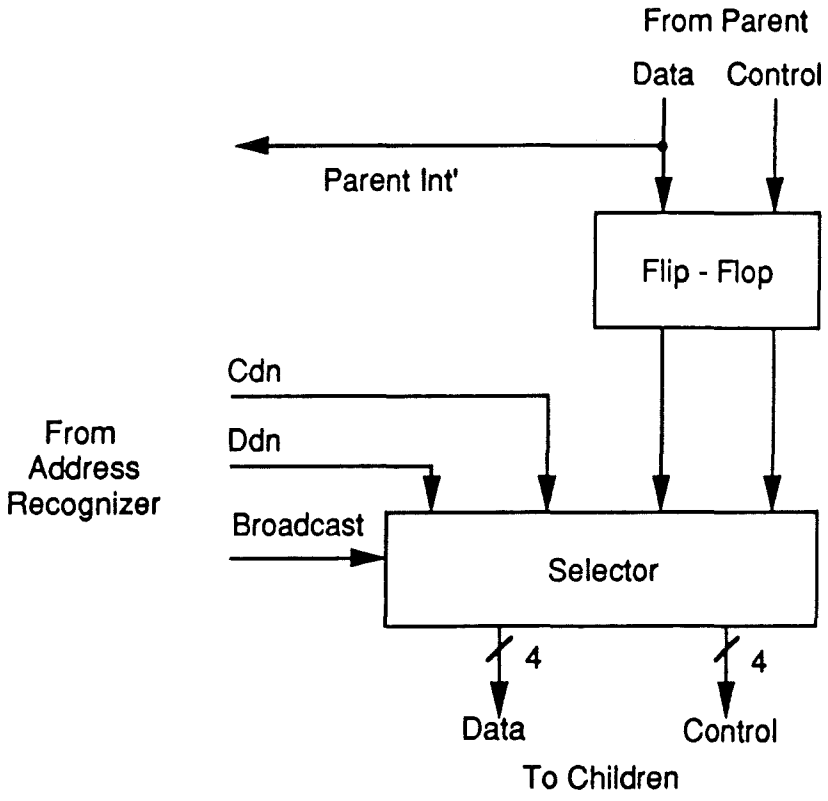


Figure 8: Downlink Selector Block Diagram

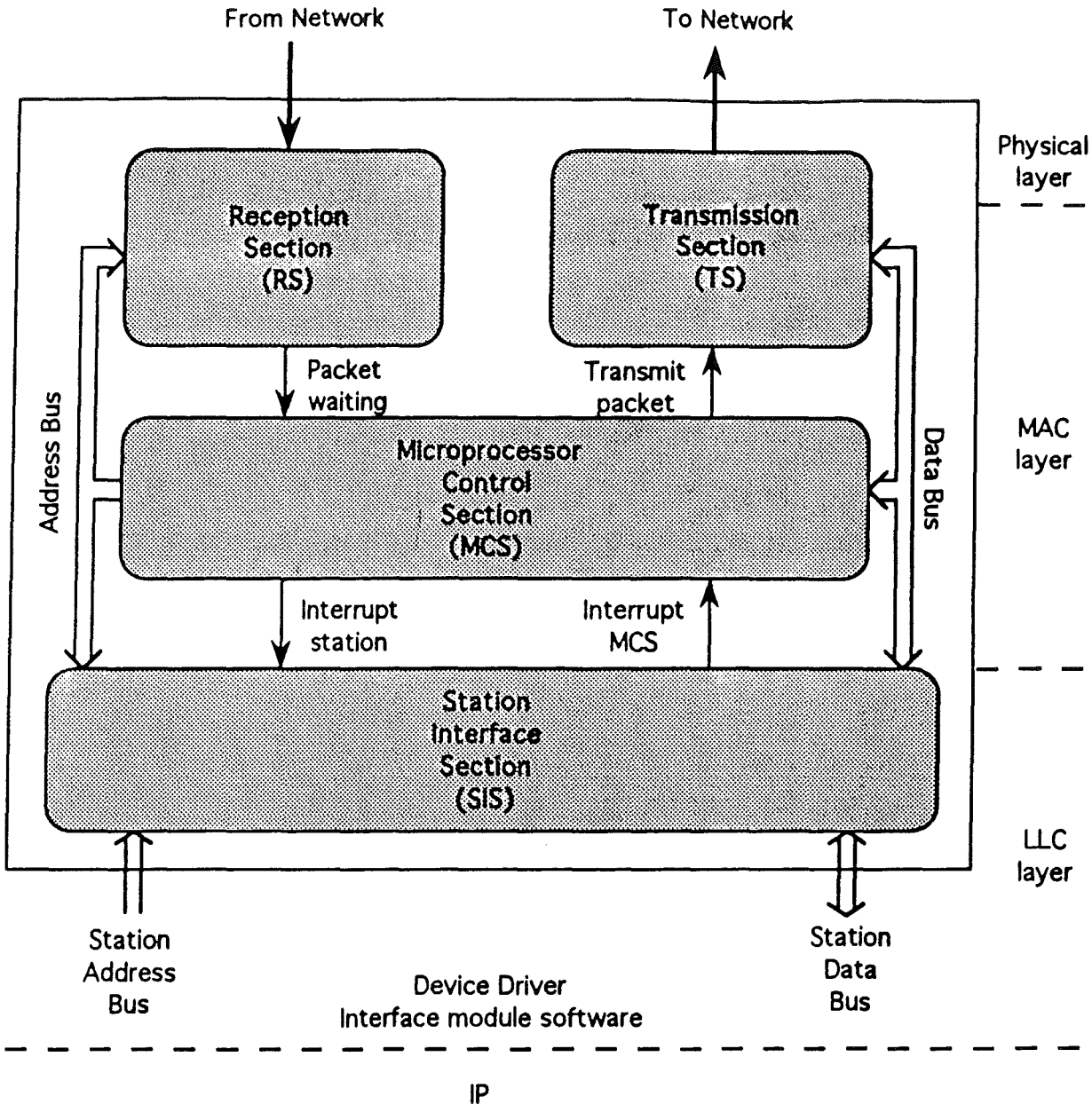


Figure 9: Station/Network Interface Board

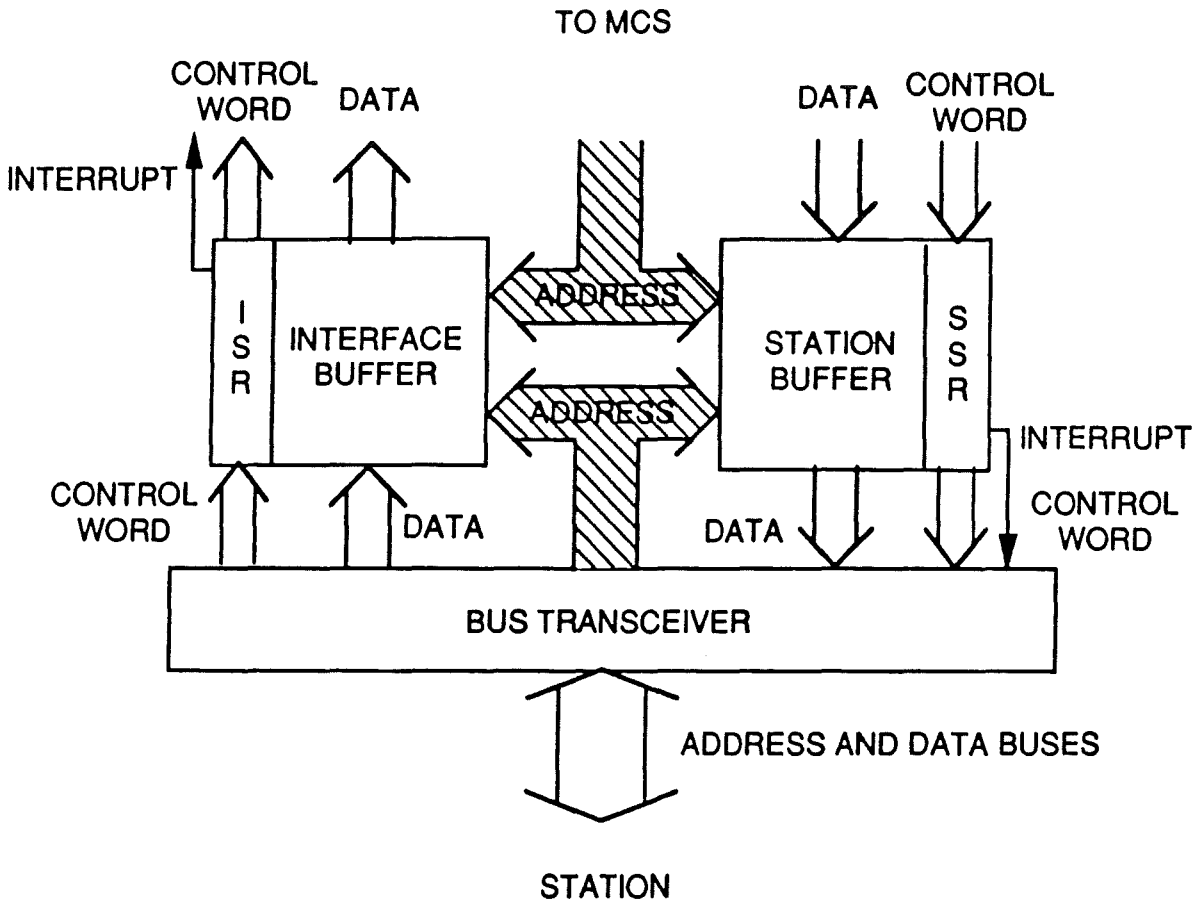


Figure 10: Station Interface Section Block Diagram

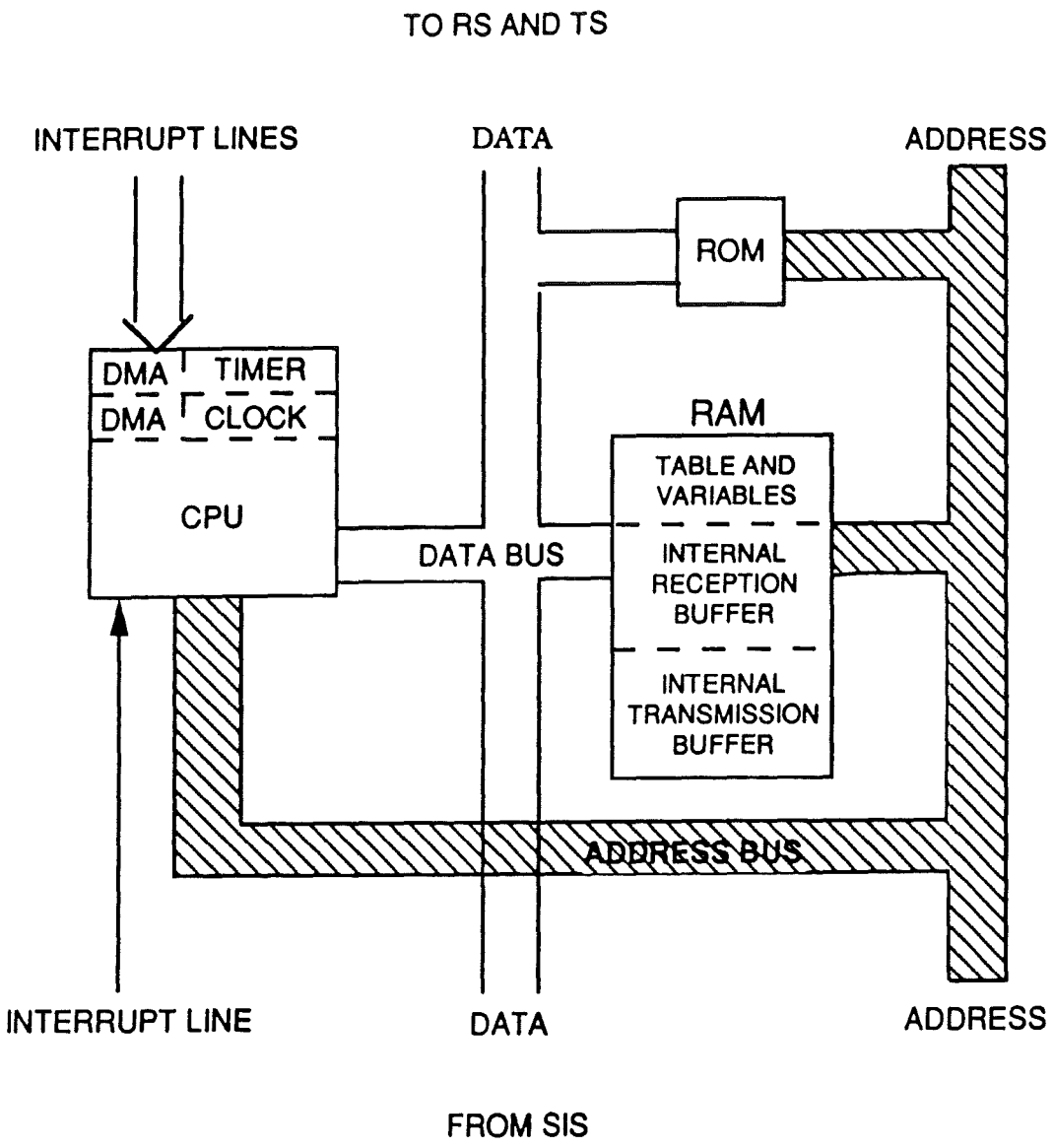


Figure 11: Microprocessor Control Section Block Diagram

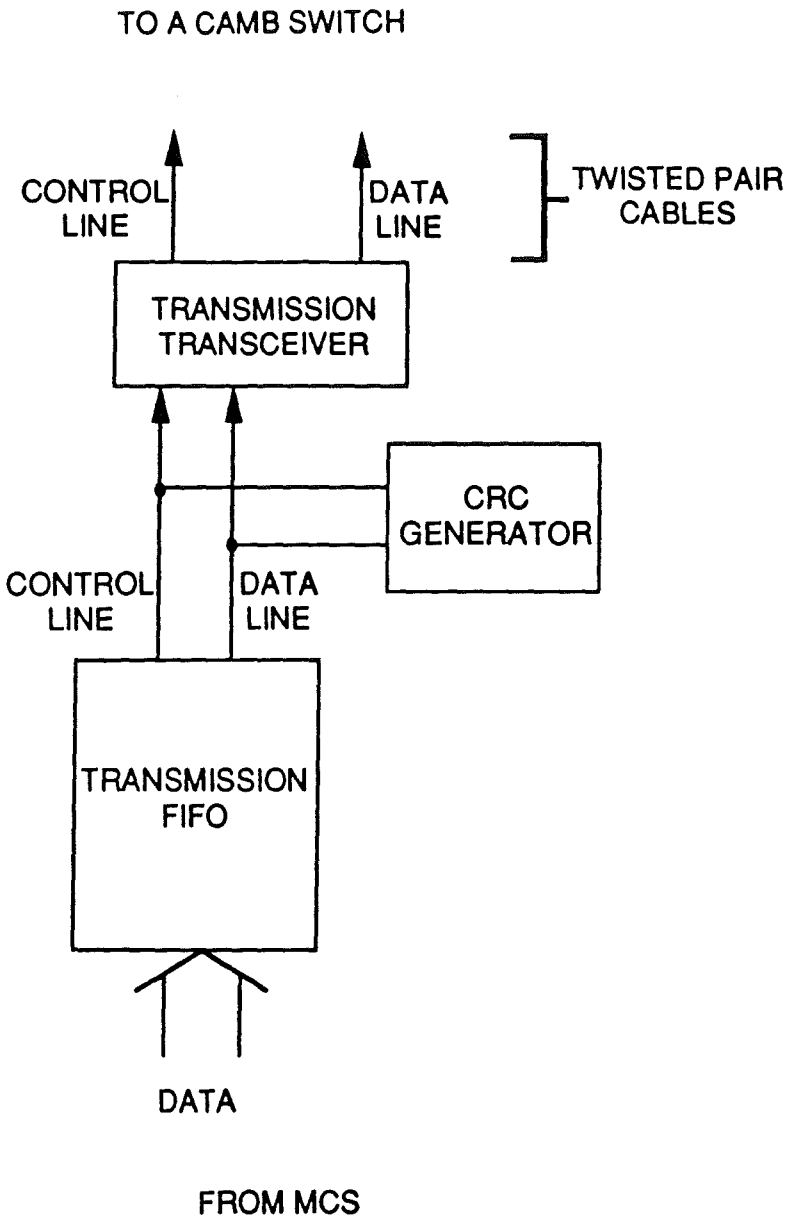


Figure 12: Transmission Section Block Diagram

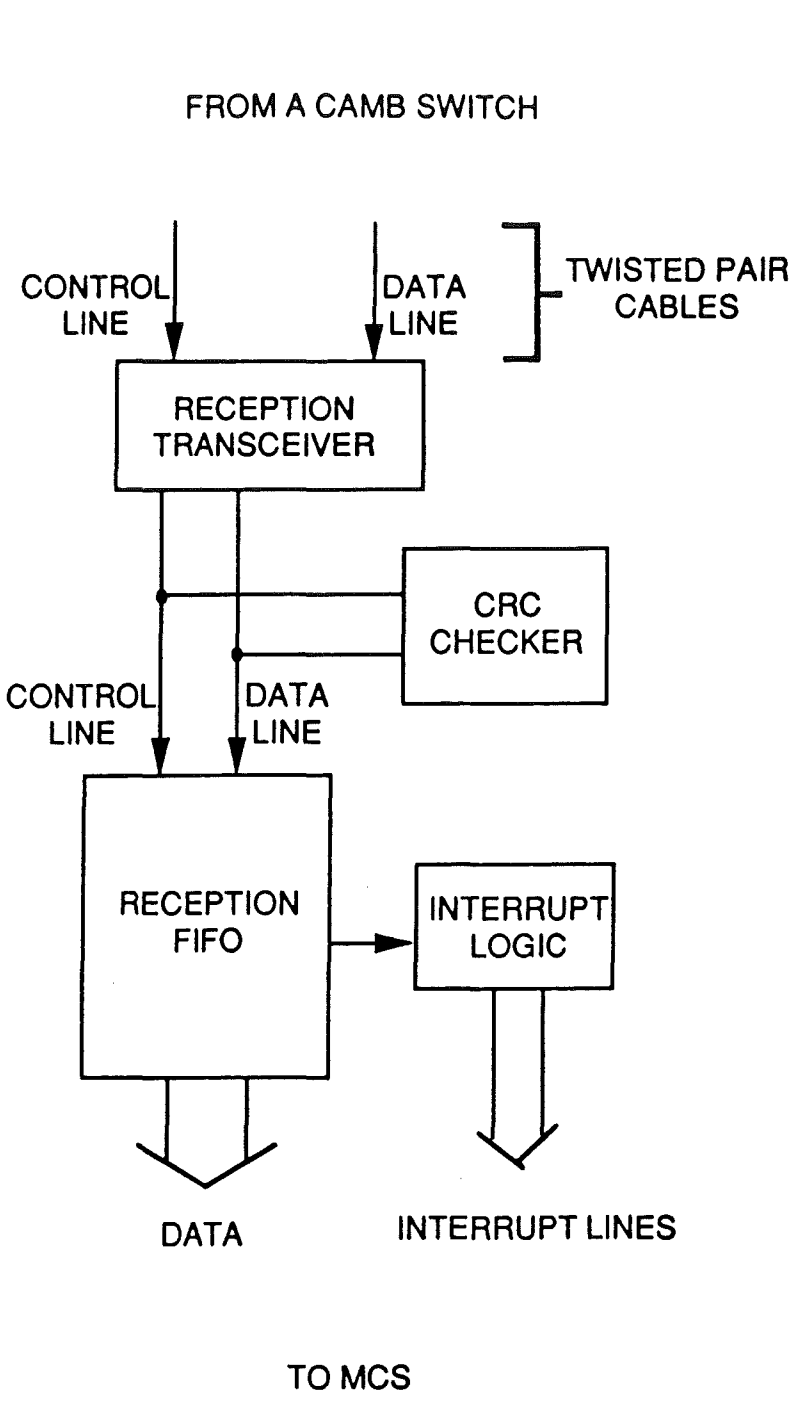


Figure 13: Reception Section Block Diagram

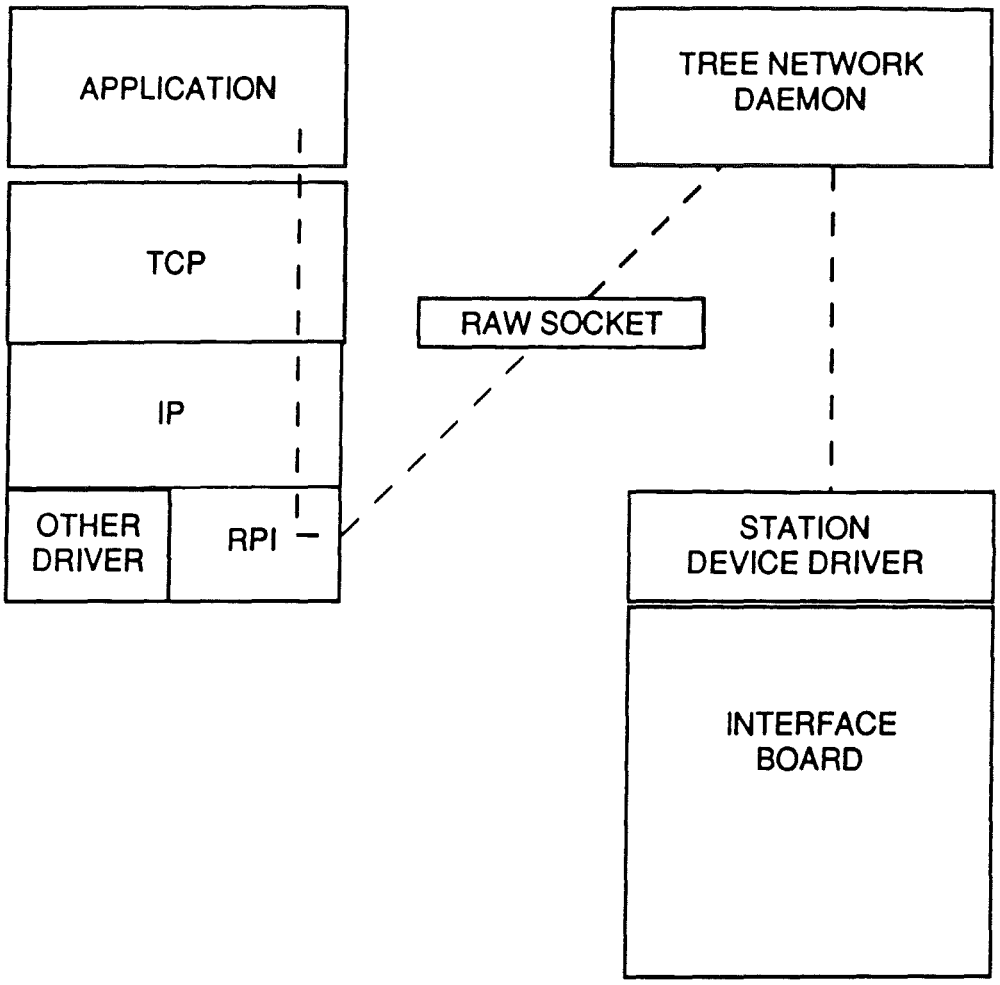


Figure 14: Network Protocol Structure

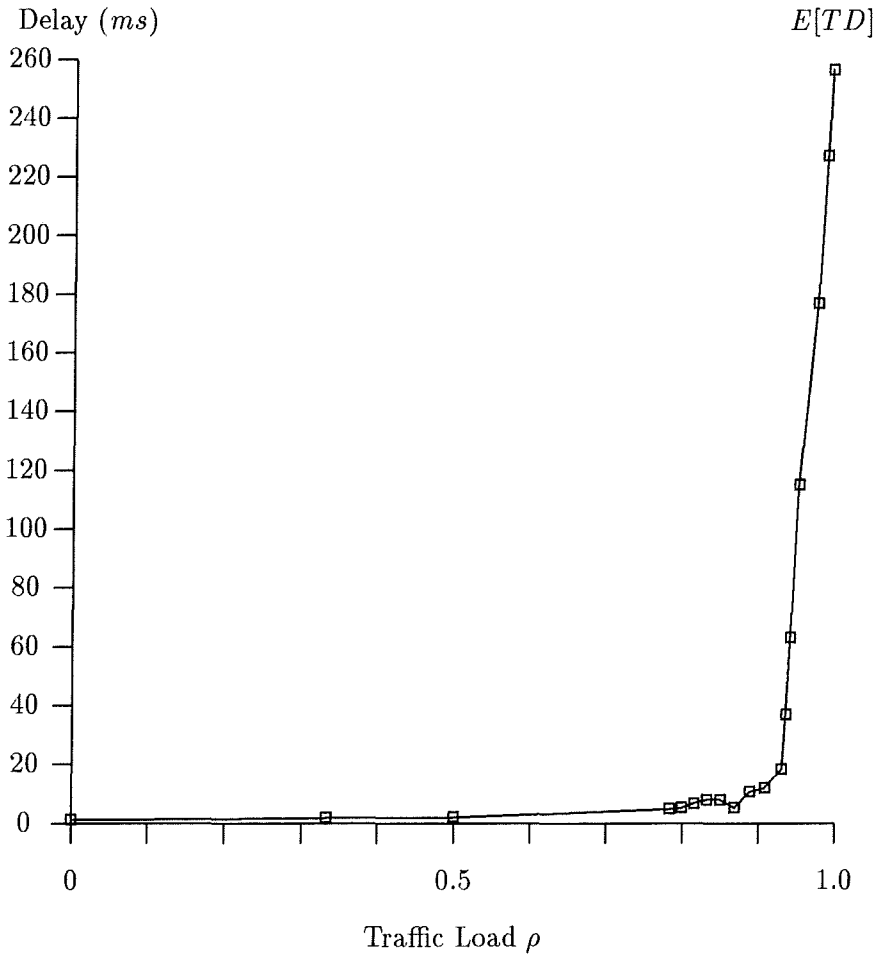
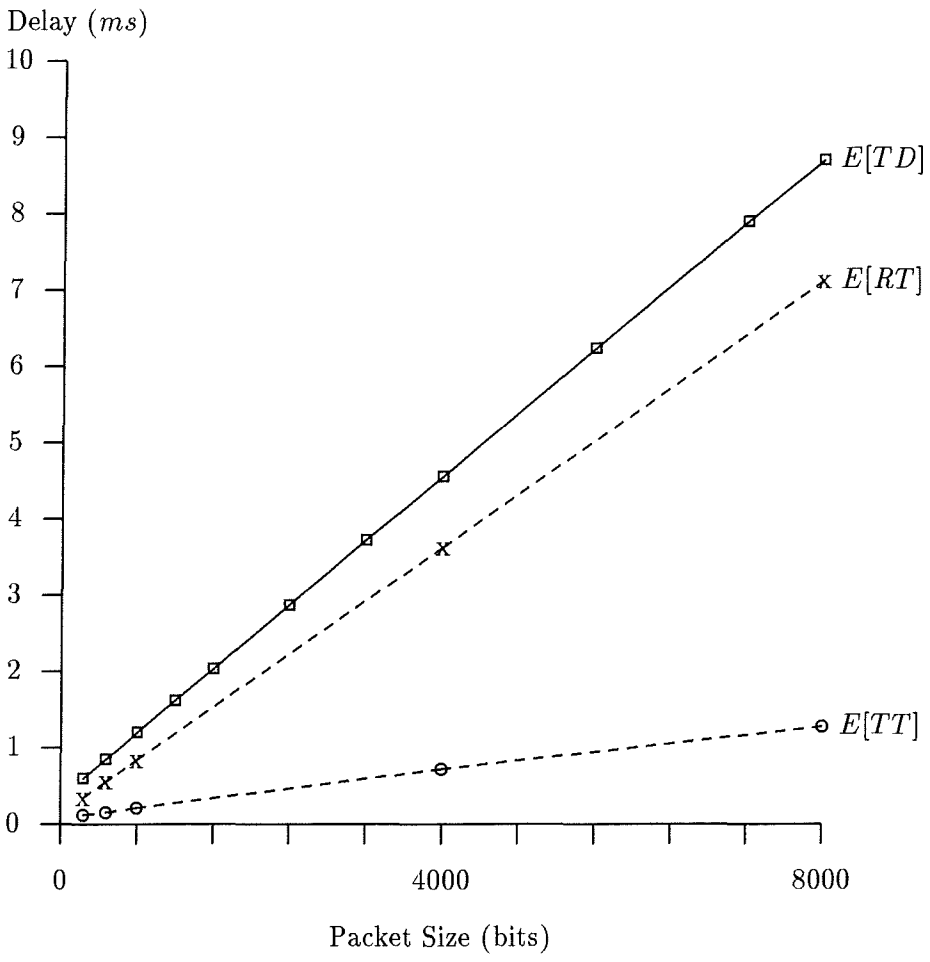


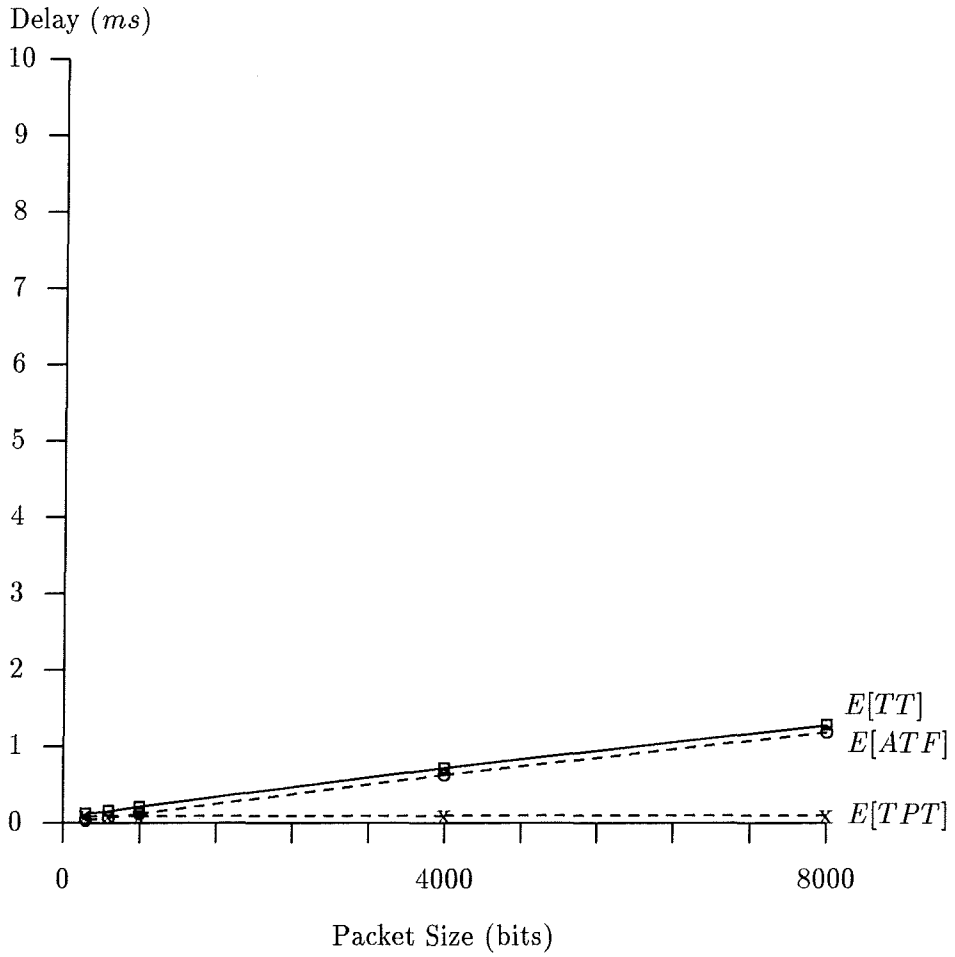
Figure 15: Transmission Delay vs. Traffic Load



E[TD] = E[TT] + E[RT], where

E[TD] – total transmission delay,
E[TT] – transmission time,
E[RT] – reception time.

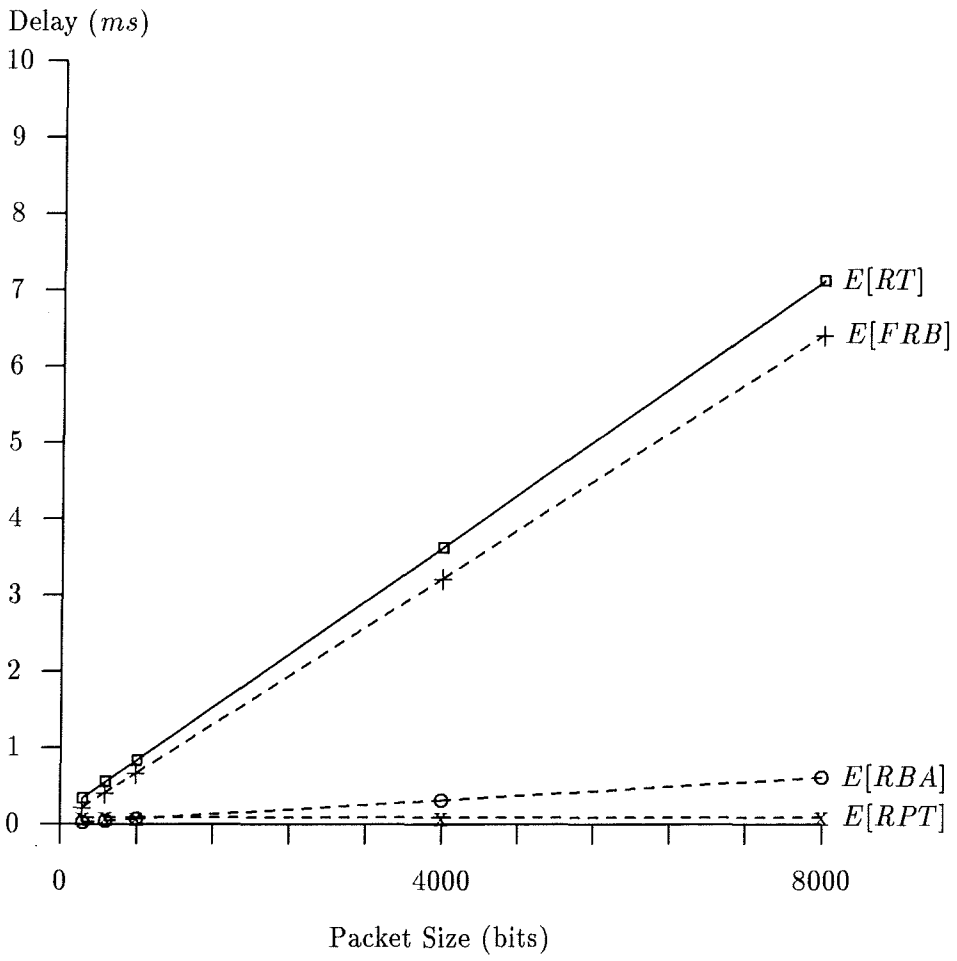
Figure 16: Transmission Delay vs. Packet Size



E[TT] = E[TPT] + E[ATF], where

E[TT] – transmission time,
E[TPT] – transmission processing time,
E[ATF] – application-to-transmission-FIFO data move time.

Figure 17: Transmission Time vs. Packet Size



E[RT] = E[RPT] + E[FRB] + E[RBA], where

E[RT]	– reception time,
E[RPT]	– reception processing time,
E[FRB]	– FIFO-to-reception-buffer data move time,
E[RBA]	– reception-buffer-to-application data move time.

Figure 18: Reception Time vs. Packet Size

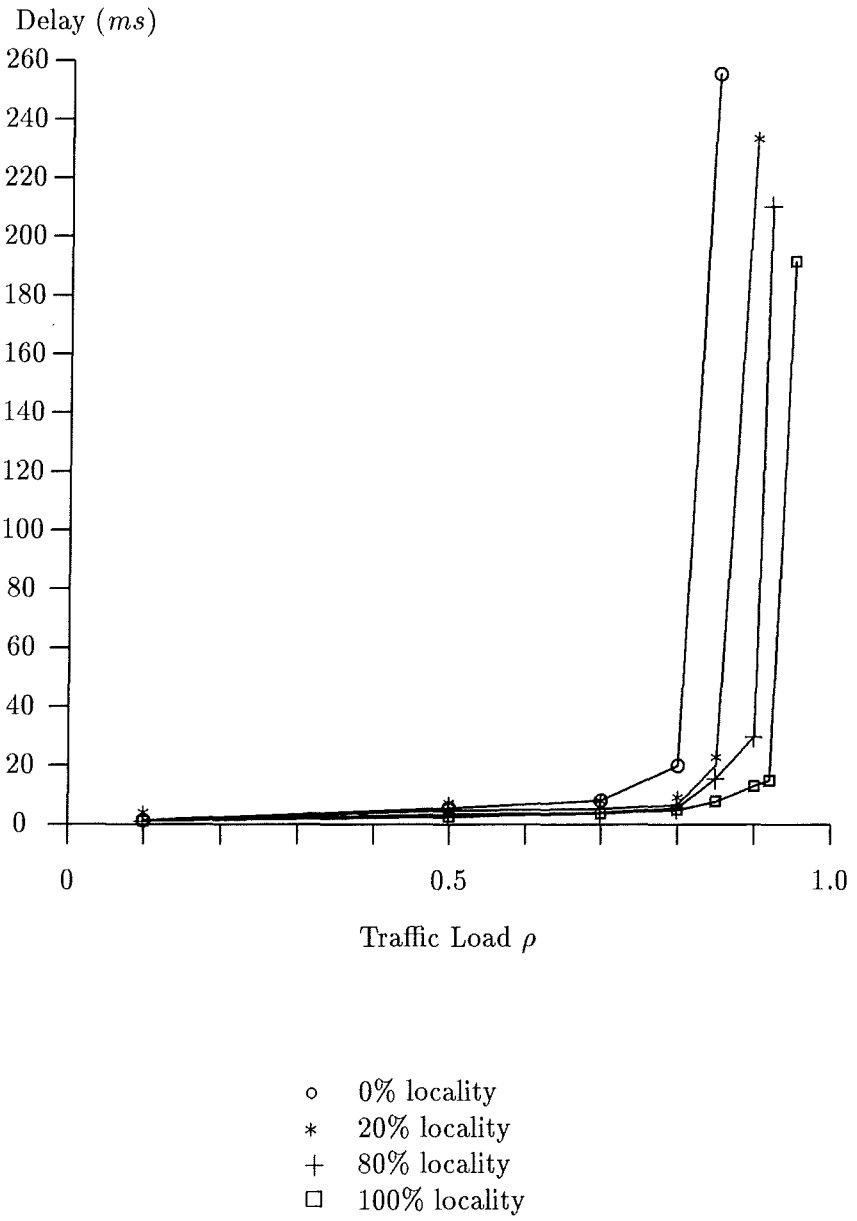


Figure 19: Transmission Delay in a 2-Level CAMB Tree