8/25/93
E7564

NASA Contractor Report 190758

Technical Support for Digital Systems Technology Development Task Order 1 ISP Contention Analysis and Control

Roy H. Stehle and Richard G. Ogier
SRI International
Menlo Park, California

January 1993

Prepared for
Lewis Research Center
Under Contract NAS3-25934

NASA
National Aeronautics and
Space Administration

Technical Support for Digital Systems Technology Development

Task Order 1

ISP Contention Analysis and Control

Roy H. Stehle and Richard G. Ogier
SRI International
Menlo Park, California 94025-3493

Abstract

This study has investigated alternatives for realizing a packet-based network switch for use on an FDMA/TDM geostationary communication satellite. Each of the eight downlink beams supports eight directed dwells. The design needed to accommodate multicast packets with very low probability of loss due to contention.

Three switch architectures were designed and analyzed. An output-queued, shared bus system yielded a functionally simple system, utilizing a first-in, first-out (FIFO) memory per downlink dwell, but at the expense of a large total memory requirement. A shared memory architecture offered the most efficiency in memory requirements, requiring about half the memory of the shared bus design. The processing requirement for the shared-memory system adds system complexity that may offset the benefits of the smaller memory. An alternative design using a shared memory buffer per downlink beam decreases circuit complexity through a distributed design, and requires at most 1000 packets of memory more than the completely shared memory design.

Modifications to the basic packet switch designs have been proposed to accommodate circuit-switched traffic, which must be served on a periodic basis with minimal delay. Methods for dynamically controlling the downlink dwell lengths have been developed and analyzed. These methods adapt quickly to changing traffic demands, and do not add significant complexity or cost to the satellite and ground station designs. Methods for reducing the memory requirement by not requiring the satellite to store full packets have also been proposed and analyzed. In addition, optimal packet and dwell lengths have been computed as functions of memory size for the three switch architectures.

1 Objectives

This study has investigated alternatives for realizing a packet-based network switch for deployment on a communication satellite. The emphasis was on the avoidance of contention problems that can occur due to the simultaneous arrival of an excessive number of packets destined for the same downlink dwell. The study looked ahead beyond the current Advanced Communications Technology Satellite (ACTS) capability to the next generation of satellites. The study has not been limited by currently available technology, but has used university and commercial research efforts as a basis for designs that can be reliably constructed and launched within the next 5 years. Tradeoffs in memory requirement, power requirement, and architecture have been considered as a part of our study.

2 Design Considerations

The proposed communications system, including both space and ground segments, is similar to that described by Ivancic and Shalkhauser [1]. To allow for lower cost ground terminals, it has been proposed that a hybrid modulation scheme be employed. The system will employ the traditional time division multiplexed (TDM) method of transmitting data to the ground. High power amplifiers can be used effectively on the satellite, where a high transmission duty cycle exists. To lessen the cost of the ground station, the system will use frequency division multiple access (FDMA) on the uplink to the satellite, eliminating the need for a costly time division multiple access (TDMA) high-power transmitting amplifier at each ground station.

The satellite will employ eight uplink beams and eight downlink beams to cover the continental United States (CONUS). A block diagram of the satellite system is shown in Figure 1, which is a copy of Figure 1 from Reference [1]. Each of the uplink beams will be served by a multichannel demodulator/demultiplexor (MCDD), which will provide packet synchronization and decoding for 1024 channels each having a data rate of 64 kbps. Groups of 32 channels may be combined to provide circuit-switched service at an aggregate data rate of 2048 kbps. The combined data rate on each of the eight uplink beams will be 65,536 kbps. Each of the eight downlinks will be served by an encoder and TDM burst modulator operating at 150 Mbps. Each beam will have a beam steering network that will provide eight dwell locations; there will be no overlap in dwell centers for any of the total 64 dwell possibilities.

The network switch that controls the routing of data between the input demodulators and output modulators is the subject of this report. The design of the switch is complicated by the necessity that a multicast capability be supported; uplink packets may be addressed to ground stations in as few as one beam dwell or in as many as all 64 beam dwells (i.e., full broadcast capability). This capability is supported by the higher data rate of the downlink beams (150 Mbps vs. 65.5 Mbps) but can be limited by the aggregate traffic to any one beam, the physical dwell switching time for a beam, and the percentage of packets requiring multicast. The temporary storage needed to guarantee packet delivery and avoid contention can be realized in several architectures, which will be discussed in later sections. Also to be covered are the considerations for the storage capacity required as a function of the percentage of multicast packets and their final destination distributions.

Two classical architectures have been developed for constructing a packet switch for this application: the Shared Bus and the Shared Memory architectures. In the Shared Bus implementation, each packet appears on a common output bus that is read (i.e., shared) by multiple destinations; in our case, each destination is a buffer associated with each of the 64 beam dwells. The Shared Mem-

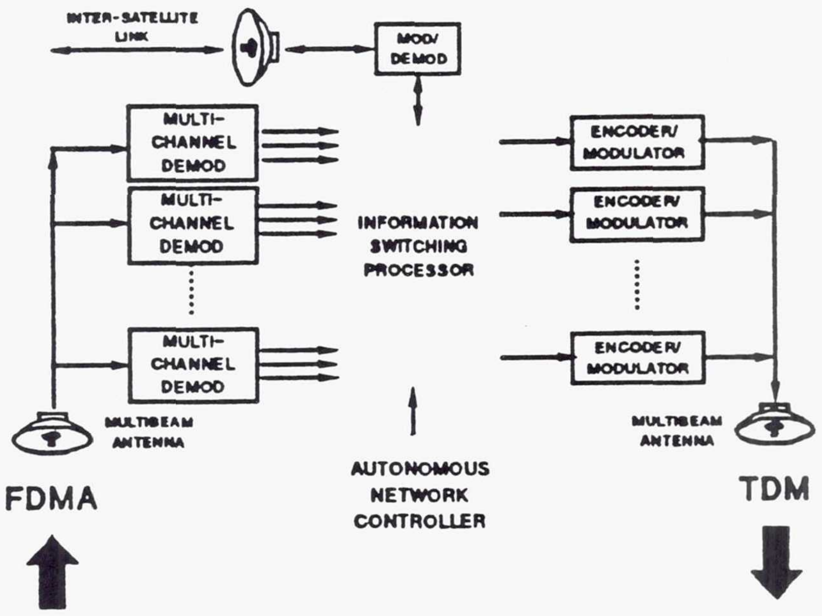

Figure 1: FDMA/TDM Network

ory configuration shares one large memory buffer among all of the output devices (i.e., downlink modulators). A variation of the Shared Memory architecture is to have a separate shared memory for each downlink beam. In this architecture, packets are transferred from the MCDD to the eight shared memories via a shared bus.

We will describe these implementations and variations of them, and will discuss their advantages and disadvantages in the following sections, beginning with the conceptually simpler Shared Bus approach.

3 Shared Bus Implementation

3.1 Design Overview

One possible implementation of a Shared Bus switch is shown in Figure 2. In this figure, the eight uplink beams and their associated multichannel demultiplexor/demodulators are shown on the left-hand side of the figure. The eight downlink beams with their time division multiplex burst modulators are indicated on the right-hand side of the figure. A data bus, the shared bus, interconnects the eight MCDDs, 64 first-in, first-out (FIFO) memories, and a central control processor. Each of the FIFO memories is assigned to an individual dwell. In our example, the data bus is 16 bits in width; this width offers a form of space division multiplexing to allow for proportionally slower speed circuits to handle the interface. The width of 16 bits yields a word duration of 30.5 ns for our example, which is well within the limits of the current state of the art. A bus width of 16 bits also matches the projected packet destination address space.

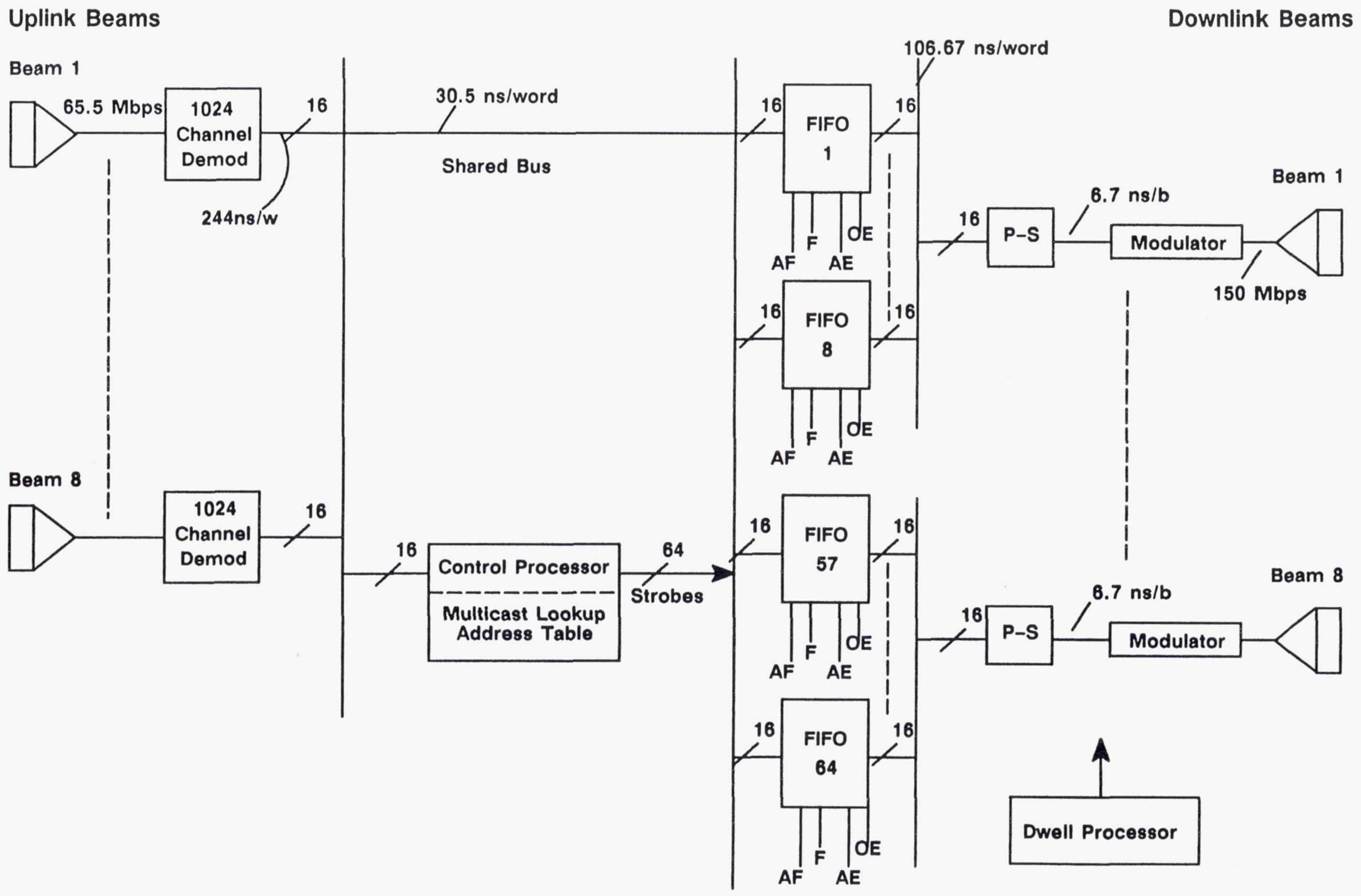

Figure 2: Shared Bus Implementation

The control processor will generate synchronization signals that will be decoded to provide output enable signals to each of the MCDDs. Each channel will be given a dedicated time slot in a frame providing access to all 8192 channels. It should be possible to time synchronize the tart of packets so that their headers will arrive at the satellite at the same time. This can be achieved by having the ground station derive synchronization from framing flags on the downlink data stream or from a secondary timing source, such as a Global Positioning System (GPS) signal. Since the satellite is stationary, adjustments to path length differentials, due to geographic deployment, can be calculated and used to adjust the synchronization.

3.2 Data Bus and Control Processor

The timing on the data bus is fast enough (33 million words per second) to allow the data from all 8192 channels to be transferred in the assembly time for a 16-bit word from an individual channel. Therefore, the MCDD requires only two words of storage for each channel: one word for the data to be multiplexed onto the bus and one word for the incoming data assembly. This results in considerable memory savings compared to storing entire packets (which could be 1000 bits long) at the MCDD. The order in which the channels are sequenced onto the bus is not restricted, since the timing of the header from the ground station can be appropriately programmed to prevent buffer overflows. It is suggested that fewer timing problems might occur if a word from each of the channels associated with a single beam is read before words are read from the next uplink beam. This would also be consistent with grouping of data from 32 channels assigned to carry DS-1 circuit-switched data.

If header synchronization is achievable, as discussed in the previous paragraph, then the timing logic in the control process can be simplified. The first 16 bits of the packet's header is the destination address. To allow for multicasting, this address is actually structured to be a pointer into a routing table. In this manner, as few or as many of the addresses as necessary can be devoted to multicasting; it is not necessary to devote one-half, one-fourth, or other preset fraction of the address space to multicasting.

A multicast address lookup table is constructed from high-speed read-write memory. Figure 3 shows the processes associated with this table. The table is 64K words in depth; a word is 64 bits in width. Each word represents a possible address with each bit of the word assigned to the dwell that will receive the packet. These bits are used to generate latch strobes into the 64 individual FIFO memories assigned to each dwell.

Through an appropriate control channel, the mapping of header (lookup) address to output strobe bit pattern can be changed in orbit to dynamically accommodate new multicast requirements, as the user database changes during the lifetime of the spacecraft.

A second high-speed read/write memory is used to store the header address for each of the 8192 channels. This memory is cycled through sequentially to transfer succeeding words from each channel in order to generate the appropriate strobes to the FIFO memories until a packet is transferred. High speed memories suitable for this address translation task have been reported in the literature [2]. This simplified clocking scheme requires that all packets be of the same length and that the headers be synchronized. There is a potential for saving of FIFO memory in the case of idle channels in the uplink data stream: the header fill address would not generate any strobes for such channels. Additional schemes are possible allowing variable packet lengths, with an increased processing requirement to have counters for each channel, rather than a single counter for the entire block of 8192 channels.

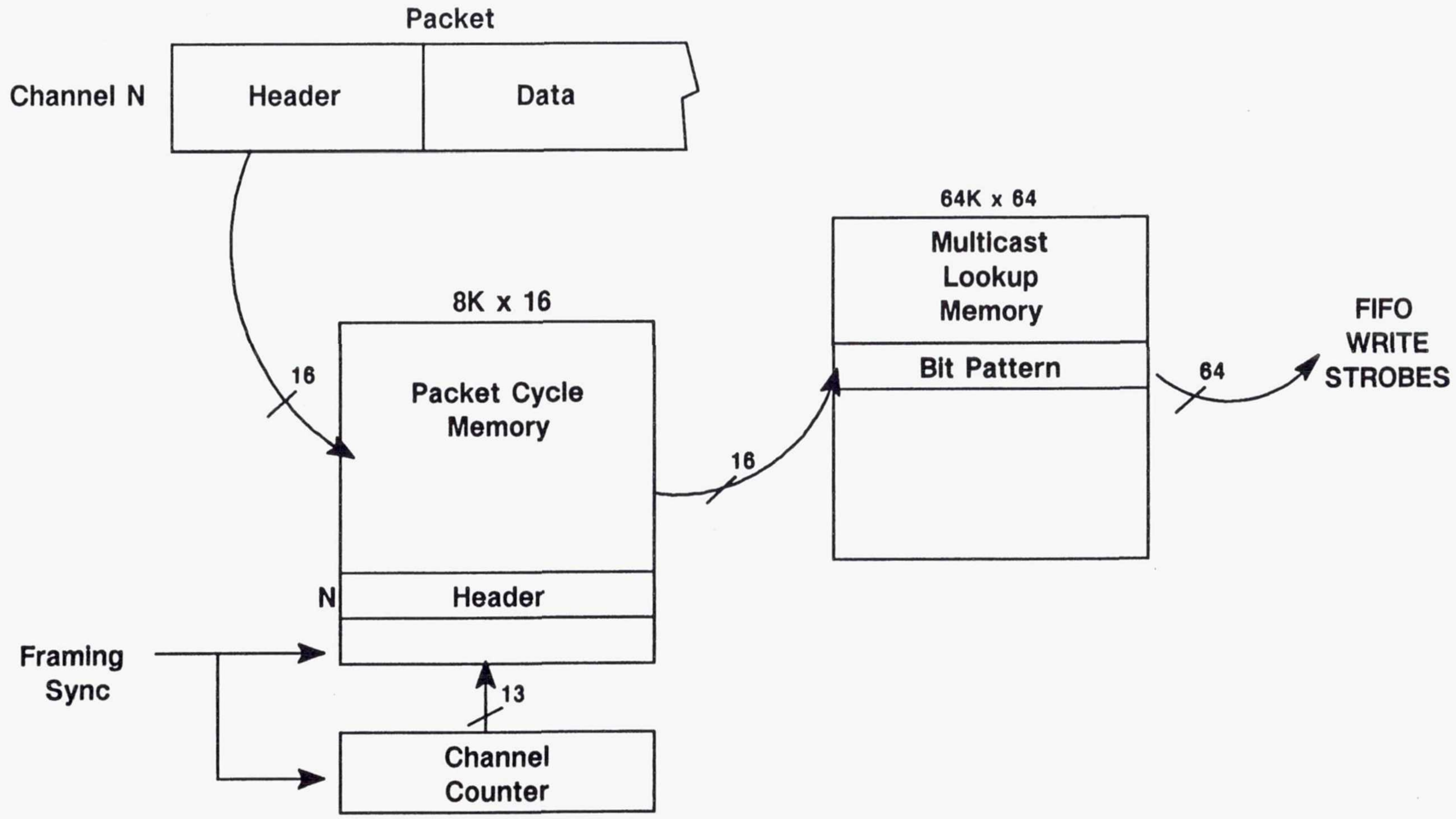

Figure 3: Multicast Address Lookup Memory

The approach we have described requires that the ground stations be able to distinguish between the first word of a packet (containing the destination address) and subsequent words, since the packets assembled for each dwell from multiple uplink channels will be interleaved in the FIFO buffer. Without this constraint, the address of another packet could be interpreted as control or data information of the first packet. This distinction can be made by letting the first bit of each 16-bit word indicate whether the word is the first word of a packet. Since labeling the first bit in each 16-bit word decreases data throughtput by 6.25%, we can instead group words into larger "subpackets," each containing only one labeled bit.

If the ability to make such a distinction is not established, then full packets would need to be assembled for each channel in the MCDD before they could be transferred on the shared bus, significantly increasing the memory requirements for the MCDD. An alternate method for reducing the memory requirement by dividing packets into subpackets is discussed in Section 5.2.

3.3 Dwell-Assigned FIFO Memories

The FIFO memories provide temporary storage for packets while the downlink dwell timing is adaptively controlled based on message traffic statistics. The amount of memory required in each FIFO is dependent on the projected message traffic, including such parameters as percent multicast packets and destination dwell. Contention and the influence of memory size will be discussed in Section 8.

The aggregate data rate of the uplink data determines the speed of the FIFO memory. Since there are eight beams of data rate 65.5 Mbps, a 16-bit word must be transferred every 30.5 ns. Currently available FIFO memories, such as the Cypress Semiconductor CY7C474[1], offer this transfer rate in a configuration of 32K words by 9 bits. Input (write) and output (read) operations can occur asynchronously; each can occur up to the full transfer rate of the chip. Two chips would be required to store the 16-bit word on the bus, with the extra bits allocatable to parity or other error-checking and correcting codes. At a 150 Mbps downlink data rate, the capacity of one dwell FIFO storage represents 3.50 ms of traffic.

Figure 2 shows that the eight dwell FIFO memories associated with a downlink beam share a common output data bus. The data are clocked from the appropriate FIFO memory under the control of a Dwell Processor. A parallel-to-serial converter follows the FIFO output bus if a bit serial data stream is required by the TDM downlink modulator. The Dwell Processor adaptively controls the length and order of dwell selection, based on packet traffic stored in the FIFO memories. The FIFO chips selected for illustration are equipped with three output status pins: Empty/Full, Programmable Almost Full/Empty, and Half Full. These outputs can be decoded to determine one of six states of the memory: Empty, Almost Empty, Less than Half Full, Greater than Half Full, Almost Full, and Full. The "almost" thresholds are programmable, allowing for in-flight adjustment of dwell logic if traffic statistics change after launch.

The logic states from the eight FIFO memories associated with a single beam can be combined into a lookup table. The output of this table would control the length of the present dwell, and select the next dwell. The dwells could be selected in random order, but a quasi-sequential order is expected to service circuit switched data channels. If a dwell FIFO is empty (or nearly so), it could be passed over in favor of a dwell FIFO that is nearly full.

Simulations of projected traffic can determine if the Dwell Processor may be configured from something as simple as a 24-bit input combinational logic element. Up-down counters can provide more quantization levels on the capacity of the FIFOs if more precise knowledge of capacity is

[1]All product names mentioned in this document are the trade marks of their respective owners.

required. The assumption that beam steering will require a significant amount of time (i.e., greater than 1 microsecond) relative to a bit time, coupled with resynchronization bits, dictates that the dwell time be set to be significantly larger than the switching time. Adequate time is provided for comparisons of counter values (i.e., buffer capacity) to decide the next dwell selection without demanding ultrafast computational capacity. This simplicity of design compensates for the fact that a Dwell Processor is required for each beam.

3.4 Conclusions

The advantage of the Shared Bus architecture just described is its simplicity of processing. The routing decisions are handled by lookup tables or combinatorial logic. A disadvantage of this architecture is the lack of a sharing of memory storage; each dwell needs the maximum computed capacity. This does not appear to be a serious drawback, as the FIFO memory is a highly integrated element offering good total power and space tradeoffs when compared to random read-write memory and a much more complicated control processor. A secondary benefit comes from the saving of buffer memory for idle channel packets. The moderate word width and minimized interconnection requirements offer advantages in reliability.

Reliability is important, because each dwell has been assigned a FIFO, and a memory failure could disable one service area on the ground. With additional programming logic on board the satellite, a FIFO failure could be accommodated in a quasi-static manner by dynamic reconfiguration of the lookup tables to associate a low-traffic FIFO to serve the dwell of the failed FIFO. The dynamic switching would take place on a cycle that was a fraction of the round trip delay (e.g., 50 ms). The period needs to be short enough so that circuit-switched data does not appear to be significantly delayed. Commands on the downlink channel could direct ground stations on the timing of the switchover, so that traffic to the temporarily reassigned dwell might be held back until it is reconfigured. Such a scheme will degrade traffic flow by more than one-eighth, because of timing margins, but it does retain data flow to an otherwise lost target area (dwell).

4 Shared Memory Implementation

4.1 Design Overview

A block diagram of a Shared Memory implementation is given in Figure 4. The uplink portion, on the left-hand side of the figure, is identical to that previously discussed for the Shared Bus implementation, with the exception of the use of a wider data bus. Aside from double-word synchronization buffers on the uplink and downlink channels, the memory is contained in one large memory block shared by the uplink and downlink hardware. A sophisticated control processor manages the memory space for maximum utilization.

Since input and output functions for all uplink channels and downlink dwells share the memory buffer, the shared memory must process data at the aggregate of both the uplink and downlink data rates. To scale the data rates to more easily handled speeds, the data bus is scaled in width. A width of 48 bits has been selected for our discussions; smaller or greater widths are possible. An assumption of memory cycle times of 20 ns yields a capacity of 2.4 Gbps, offering a comfortable margin for housekeeping functions over the aggregate of the 0.524 Gbps uplink and 1.200 Gbps downlink.

The Control Processor manages the placement of packets into the shared memory. It determines the distribution of packets, including multicast packets, and controls the assembly order of dwell

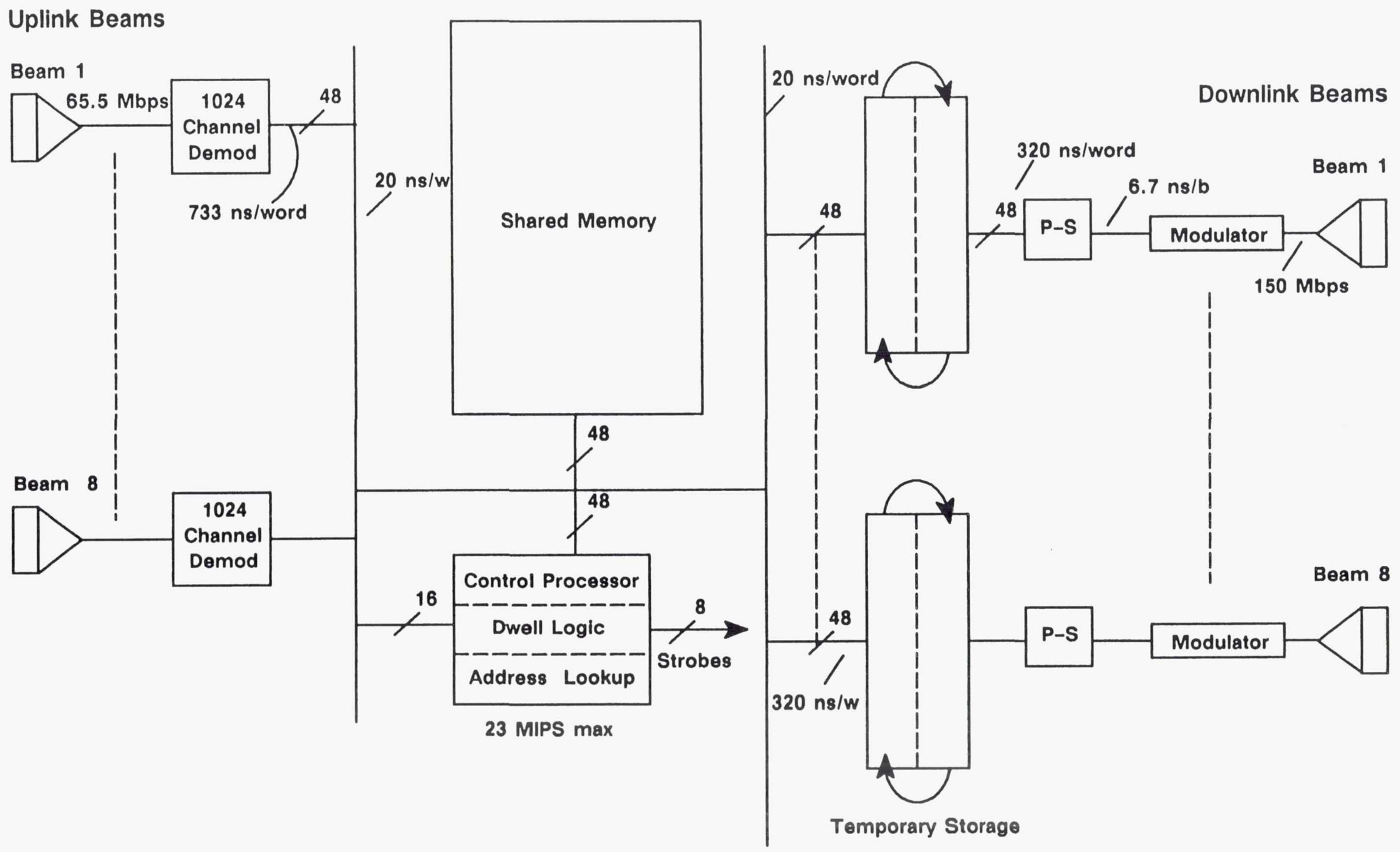

Figure 4: Shared Memory Implementation

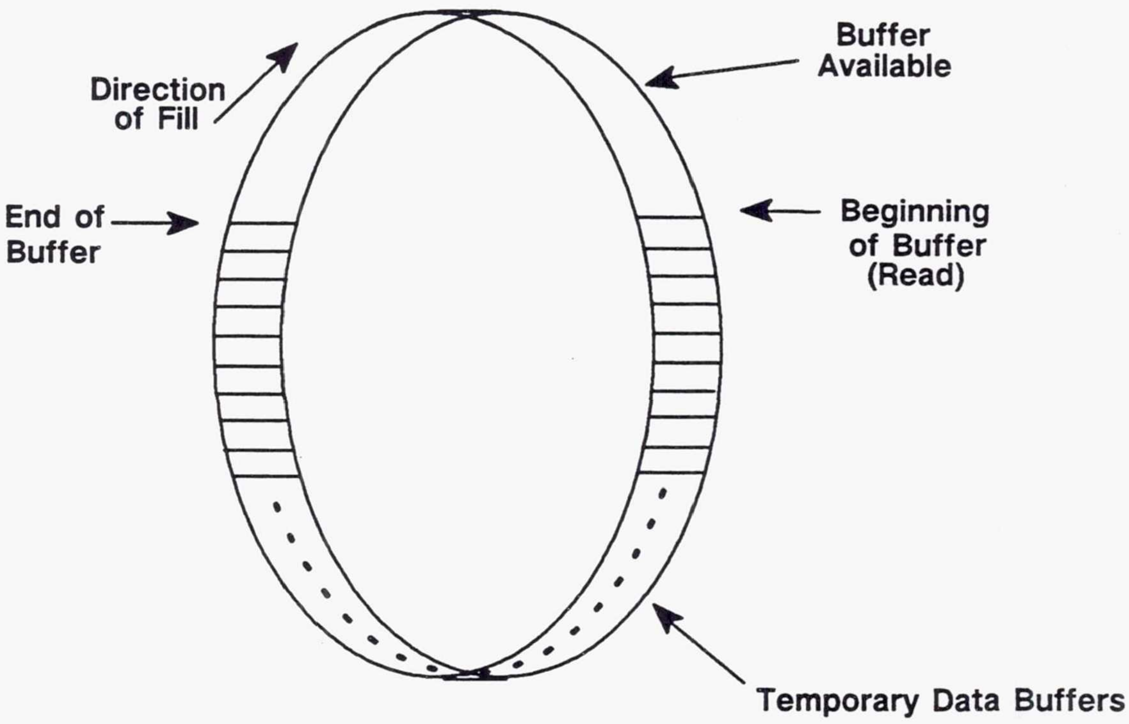

Figure 5: Circular Buffer Storage

downlink data streams. An address mapping into dwell destinations is accomplished in a lookup table similar to that used in the Shared Bus implementation. An additional memory contains main memory address pointers to the packets assigned to each dwell. Address counters and address offset registers are prominent in the data transfer design, to accommodate the high throughput required. A multiprocessor implementation will quite likely be needed to handle the routing decisions and dwell frame assembly.

Like the two-word buffering provided by the uplink demodulators and demultiplexors for synchronization and processing, similar buffering is used for each of the downlink beams and their associated modulators. The downlink buffers convert from the parallel mode of the data bus to the serial data stream required by the modulator.

4.2 Circular Buffer Design

In this implementation, the shared memory is configured as a large circular buffer. Incoming data is written in ascending, consecutive address order; successive words from an individual uplink channel will be stored 8192 memory addresses apart. When the memory address range allocated for the circular buffer overflows, the address is "wrapped" back to the start. Only one copy of a packet is stored in memory, even in the case of a multicast packet. Figure 5 presents a graphic representation of a circular buffer.

In the extreme case of no multicast packets, it would only be necessary to provide a few words of buffering for each channel's packet before passing it on to the downlink; full packet storage would not be required with properly synchronized uplink and downlink timing. In the case of multicast packets, the traffic statistics are not easily controllable, because of the large value of the round trip propagation time relative to packet duration. Because packets destined for the same downlink dwell may arrive simultaneously on several different channels, it is necessary to provide for the storage of multiple packets for each downlink dwell. It is not required, however, that complete packets be assembled before transmissions begin on the downlink, unless interleaved packets cannot be handled by the ground station. Interleaved packets have been assumed for the Shared Bus implementation, and should be assumed in this case.

After the storage of the first word of each channel's packet, the address field of the packet is examined to determine its destination(s). A reprogrammable look-up address table produces a 64-bit output word with bits set appropriately to the dwell(s) that will send the packet. The bit pattern can be used as strobes for special-purpose memory or as flags for a special-purpose processor that controls writes into a Dwell Service Memory. The word written into the Dwell Service Memory is the main memory's address of the first word of the channel's packet, not the word itself, as was done in the case of the Shared Bus' FIFO memory. The Dwell Service Memory will be configured in the structure of 64 FIFOs. Software or hardware up-down counters can also be driven by the pattern of look-up table output bits, to facilitate the logic of dwell management. The 64-bit output from the look-up table is stored in a memory block with an address range correlated to the circular buffer modulo the length of a packet. This Packet Distribution Memory will serve as input to the housekeeping and buffer management process.

4.3 Dwell Management Processor

The data flow and memory management functions described thus far are quite straightforward and can be implemented in high-speed logic or may be handled by a high-speed processor. Because the processing for dwell management is much more complex, a dwell management processor dedicated to each beam may be required, as in the case of the Shared Bus implementation.

By scanning the Dwell Management Memory or by examining counters representing the number of packets destined for each dwell, the individual or composite Dwell Management Processor decides the next dwell to be serviced (per beam). Once a dwell has been decided upon, the Dwell Service Memory is searched to find the main-memory starting addresses for the packet destined for the dwell; this is done by searching for the bit or flag corresponding to that dwell that is set in the Dwell Distribution Memory. Counters are set with main memory starting addresses for the packets destined for the selected dwell. Time sharing of the main memory bus allows the packet contents to be transferred to the buffering registers associated with each beam. Timing must be critically controlled to maintain the flow of data to each TDMA burst modulator.

Since it is possible for a multichannel packet transmission for a given dwell to be interrupted to service a higher priority dwell, the order of the channel packets must not be disturbed until they have been transmitted in their entirety. Adding a new packet to a process already underway is likely to cause design difficulties for the ground receiver's demultiplexor.

Once a packet has been transmitted for its destination dwell, the corresponding bit in the Dwell Distribution Memory is reset to signal this condition. An all-zero word in the Dwell Distribution Memory is an indication to the Memory Management Processor that that area of memory may be reused. In the ideal case, the "highest" writable address in the circular buffer will always remain a constant distance from the last address written.

In the case of multicast packets, it is possible that a packet for a low-traffic dwell might reside in memory for a comparatively long time. Without proper memory management, the packet would be at risk of being overwritten. A housekeeping process (or processor) would sense this impending condition and take avoidance procedures. One method would be to relocate the packet to the current end of the circular buffer. This might be done at the epoch when all 8192 uplink channel packets have been received, or it might occur when all eight downlink beams were being redirected. It is important that dwell management memory pointers be updated in a consistent fashion, to prevent corruption of a packet.

The packet relocation process has a problem of memory conservation. Since the counters used for retrieval of the packet for transmission on the downlink(s) depend on addresses that are incremented by 8192 (i.e., the number of channels), a large block of memory would be wasted if this storage algorithm is retained. The relocated packet could be stored in a contiguous memory block, if an address coding bit is associated with the new packet starting address to indicate address increments of unity, rather than 8192.

An alternate approach to packet relocation would be to use the space of an idle channel. It can be expected that not all uplink channels will be occupied continuously. Even in high traffic periods, such as those likely to create straggling packets, there is a reasonable likelihood that an unused channel of single-packet duration would exist in the recent history of the active portion of the circular memory. Alternatively, the highest-addressed, already delivered packet buffer could serve as a destination for the packet to be relocated.

Straggler packets may also be avoided by assigning a higher priority to the queue for the undelivered dwells for that packet, to encourage earlier delivery of that packet. This is algorithmically more complicated than the storage relocation process and would have to be simulated for optimization.

The circular buffer implementation has the distinct advantage that the memory is shared by all processes. Its relatively tight, continuous packing of that memory increases the efficiency of usage. Only one copy of a packet is required for multicasting, which increases efficiency. A large block of memory is also fairly efficient in its use of power and space, compared to an equivalent amount of memory distributed amongst a modular hardware implementation. Reliability can be gained by allowing the Memory Control Processor to redefine circular buffer size to avoid a faulty page of memory. The disadvantage is that the circular buffer requires a fairly large continuous block of memory to store an adequate number of packets for distribution.

The disadvantages of this architecture relate to device speed and logic complexity. Even with a threefold increase in memory bus data word width (48 bits vs. 16 bits), the projected memory cycle time for the shared memory is two-thirds of that projected for the Shared Bus FIFOs (20 ns vs. 30 ns). The efficient use of memory will reduce costs and power, but the processing logic needed to manage the memory allocation will add significantly on both counts. The additional drivers for the wider data bus will also increase power and space demands.

4.4 Quasi-arbitrary Buffer Allocation

An alternate scheme for allocating memory uses an arbitrary assignment of individual packet buffers. The operation is similar to the circular buffer approach except for the manner in which packets are stored. Blocks of memory equal to a packet are assigned on a quasi-arbitrary basis. An auxiliary offset memory stores the starting address for each channel's incoming packet. The channels continue to be sampled in a sequential manner and word by word; the packet data for each channel is stored in consecutive memory. An adder forms the physical address from the channel number designator and the pointer stored in the offset memory. The mapping into a Dwell Distribution Memory is similar to that of the Circular Buffer approach, including the flag bits per dwell. An example of how packets might be stored with this scheme is given in Figure 6.

The ability to assign incoming packets to arbitrarily distributed buffers has the advantage that buffer overflow housekeeping is not required: delinquent packets can remain in place in their initial

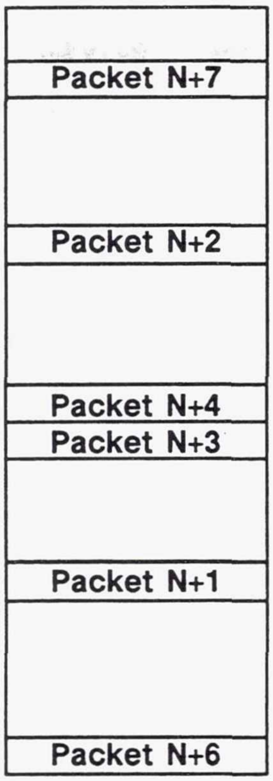

Figure 6: Quasi-arbitrary Buffer Storage

storage location. This storage method incurs only the minor cost of a more complex storage address generator formed from a channel address counter, offset memory, and a word counter. It offers the additional advantage of being able to isolate arbitrarily sized blocks of memory from usage in case of a fault; large blocks of contiguous memory are not required. The disadvantage of this method lies in the need for a potentially more sophisticated processor to find the buffers that are "empty" and able to accept new packets. This method could use a simple algorithm in which the next free buffer is taken from the last one used — the reason for including the prefix "quasi" in the name of this approach.

4.5 Shared Memory per Beam Variation

The fully shared memory implementation has great benefit in its efficiency of memory usage. However, it suffers from hardware implementation issues (e.g., wide data bus, fast processors) that can impact its overall reliability. A variation to the fully shared memory approach is the Shared Memory per Beam design. This is shown schematically in Figure 7.

As will be shown in Section 8, the memory efficiency of the shared memory per beam implementation is only slightly degraded from that of the fully shared memory design. The bus speeds are only slightly higher than that of the shared bus design while utilizing a similarly small (e.g., 16-bit) bus word width. The processing tasks associated with shared memory are distributed among the eight downlink beam control processors. Moderately fast control processors are still required to maintain a constant stream of data to the downlink modulators, but the requirements are more easily met with standard technology.

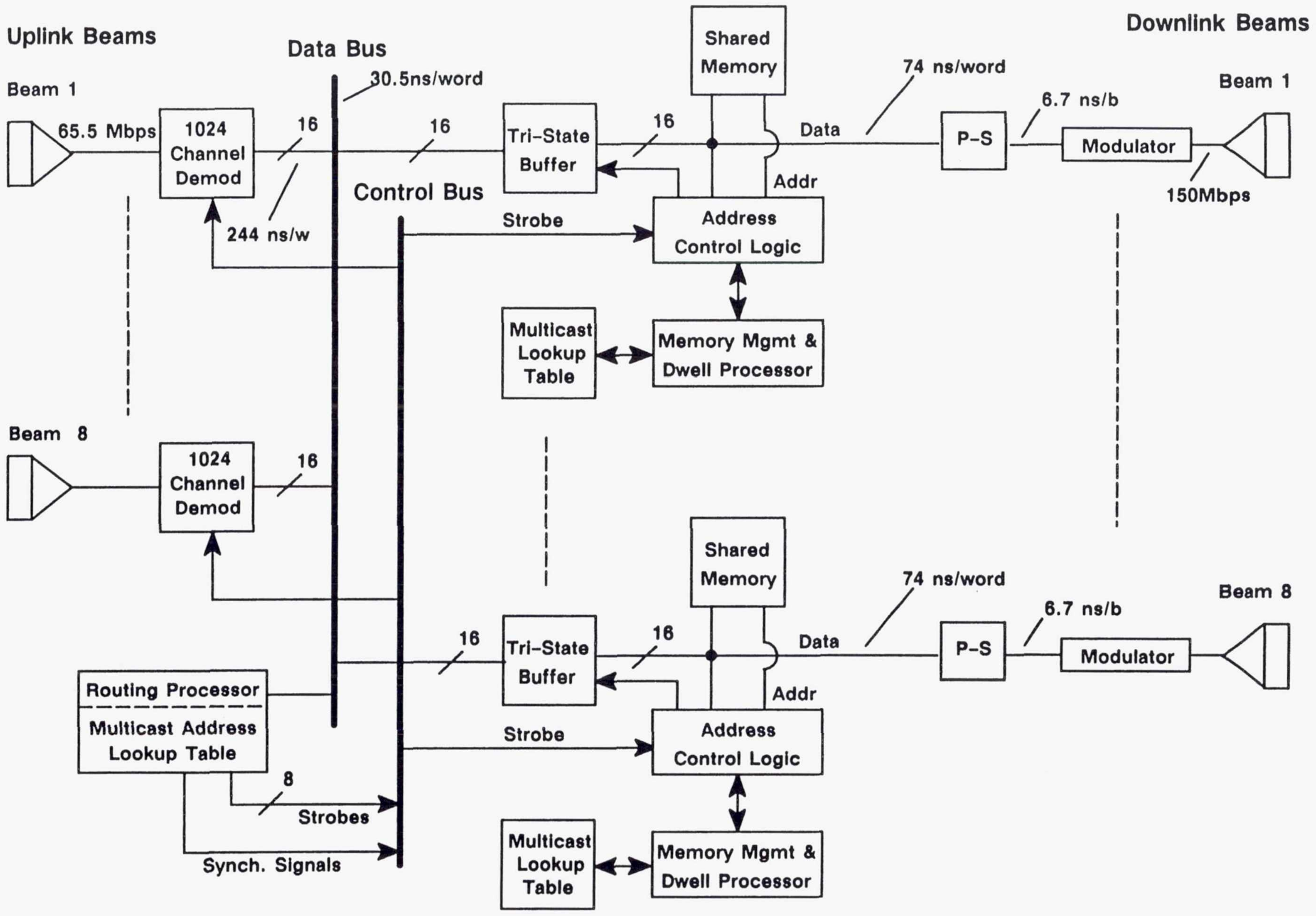

Figure 7: Shared Memory Per Beam Implementation

Referring to Figure 7, the data from the uplink MCDDs are time multiplexed onto the 16-bit Common Data Bus in a manner similar to that of the Shared Bus approach. The MCDDs have sufficient intelligence to identify the packet headers; at the least, the MCDDs must present the data on "word" boundaries. The intelligence for header identification might be aggregated into the Routing Processor, as an alternative. The Routing Processor is responsible for determining the downlink beam destination of the uplink packet; it determines this information from the header's address field in a process similar to that of the Shared Bus design. In this case, it is only necessary to generate up to eight strobes that would correspond to the downlink beams serving the packet's destination. Packet synchronization is not required on the uplink if the Routing Processor maintains counters and pointer registers for the data of each uplink packet. Circuit-switched data are accommodated as in the case of the Shared Bus by using order wire lookup destinations rather than packet header address fields.

The time multiplexing of the uplink data onto the Common Data Bus is under the control of the Routing Processor. The processor can provide timing or synchronization signals to the logic associated with each downlink beam. This timing information is used by the Address Control Logic block of each downlink processor to reserve an internal bus timing slot to write the uplink data into the shared memory (assuming that the selection strobes have been provided by the Routing Processor). The internal data bus for each downlink beam needs to have a transfer speed at least equal to the speed of the Common Data Bus (i.e., 30.5 ns/word in this example) because it may need to accept data from consecutive timeslots. On the average, however, it will see a datarate that is only slightly higher (i.e., 74 ns/word) than the average required by the downlink beam.

Each downlink beam will have a Memory Management and Dwell Processor that will handle the storage of incoming packets and the formation of dwell data structures. The multicast routing information will need to be accessible from each of the downlink processors, so that each may route the packets and circuit switched data to the correct dwells. Timing and reliability considerations may allow the processors to share a common lookup table. Note that this multicast routing information needs to be encoded in the lookup table in the Routing Processor, although only beam, and not dwell, information need be stored or accessed there.

The parallel-to-serial processing is similar to each of the other implementations. The serial bit stream is fed to the TDM downlink modulator. This parallel-to-serial conversion may be accomplished internally to the modulator and need not be a separate process.

4.6 Dual-Port Video RAM Approach

It would be desirable to use the random write, sequential read feature of a dual-port video RAM to improve the data handling and bus timing requirements for the shared memory approach; in a sense, this approach is a hybrid of the Shared Memory and Shared Bus architectures.

A video RAM offers random access to any word in the memory. Its operation on the random access port is similar to any other RAM for read and write operations. The video RAM possesses a second port that can be programmed to sequentially read from blocks of the random access memory. An internal register and counter provide the function of buffer addressing for output that the other shared-memory devices have time multiplexed on the address bus. It is usually intended that the output will continuously be refreshed with the contents of a block of memory.

The drawback of using this type of memory in this switching application is the long time that it takes to reprogram the starting address, compared to the read cycle time. Therefore, it is not possible, with current commercial devices, to reprogram the output to sequentially access multiple buffers; a complete dwell buffer would need to be assembled for continuous readout.

If an efficient method could be devised for writing dwell buffers from the incoming data, then the high speed output to each dwell modulator could easily be handled by this sequential access output port. While this method would be similar to constructing a software equivalent of the hardware FIFO, as proposed in the Shared Bus implementation, the advantage lies in the sharing of the total memory based on traffic statistics. A high-speed processor would probably be required to preload these buffers, and/or banks of memory dedicated per beam, at least. This approach then becomes equally or more difficult to implement than the (hardware) FIFO memory. Although this architecture may not be feasible with current circuit designs, we consider it because it may become feasible with new technology.

5 Further Design Alternatives

5.1 Fiber-Optic Data Bus

The implementations proposed have used the parallel hardware data bus for illustration. This is a valid implementation and has been used in narrow and wide bus schemes in many systems. If further studies indicate that a direct-connection parallel bus is not suitable, then it is possible to achieve the same function using optical means. Some cases that might justify an optical approach might involve spacially separated MCDDs, a long data bus that requires heavy current drivers to achieve high cycle times, or a bus that might be susceptible to radiated or conducted interference.

The aggregate data rate for the uplink beams is 524 Mbps. Serial-to-parallel and parallel-to-serial converters have been implemented that are capable of transferring data at this rate. Vitesse Semiconductor offers the G-TAXI V760 series implementation of the Advanced Micro Devices TAXI chip set. The GaAs version offers data conversion and transfer at data rates to 1.25 Gbps. The system is modular and can easily accommodate parallel bus widths to 40 bits. The power tradeoff in the optical conversion may be readily offset by easier interconnection and reduced interference.

5.2 Packet Division Scheme

In the packet division scheme, packets are divided into a number, say 8, of "subpackets" such that only the first subpacket of a packet contains a header. Each subpacket is bussed to the appropriate downlink FIFO as soon as it arrives (thus the MCDD only needs to store two subpackets per channel). When the first subpacket of a packet is transmitted in a downlink slot (corresponding to the appropriate dwell), the corresponding slot in the next seven frames is reserved for the remaining seven subpackets of the same packet. The length of the downlink frame is chosen so that it is equal to the time required to send one subpacket on the uplink (so that subpackets are retransmitted on the downlink as fast as they are received on the uplink).

This scheme should result in significant memory savings, since entire packets would not have to be stored in the satellite. However, it is not desirable to choose a very small frame, because of the dwell switching time. A small frame can be avoided by making the packets (and thus subpackets) longer, which requires more memory but also increases the effective data throughput by increasing the ratio of the packet length to the header length.

Exactly how to implement this scheme is a topic for future study. The scheme can be applied to both the shared bus and shared memory architectures.

6 Comparisons

Two distinctly different approaches to the implementation of a packet switch have been offered. The Shared Bus approach favors a system that could be implemented with special function circuits such as FIFO memories, dwell duration and selection logic, and data routing lookup functions. While these functions could be implemented using microprocessor elements, such as a CPU and memory, the main advantage of a custom circuit implementation is its reliability.

Reliability is expected to be easily achieved in the Shared Bus implementation because of the limited size of the data bus, reduction of the pin count of the custom circuit elements, and reuse of a limited number of specialized circuits.

The Shared Memory approach gains some reliability because the memory can be shared and reconfigured to accommodate some hardware faults. It would not suffer the restrictions of a FIFO memory dedicated to a specific dwell. The Shared Memory approach offers more efficiency in the amount of memory needed to handle the worst case traffic.

The high data rate that must be handled by the packet switch places extreme demands on the high speed processor that must make storage, routing, and dwell decisions in the case of a Shared Memory system. What might be saved in power and space due to increased memory efficiency might easily be offset by the multiprocessor control processor dictated by the demands associated with the high data rate.

A rough estimate of the memory and power requirements for the Shared Bus and Shared Memory implementations is included in Table 1. We assume, as is implied by the analysis of Section 8, that the Shared Memory design requires about half as much memory as the Shared Bus design. The power consumption for the FIFO memories is taken from the data sheet for the densest part currently available from Cypress Semiconductor; other parts may be available that are equally suitable for the function. The power estimate for the static RAM used by the Shared Memory example is based upon a figure of 10 mW/Mb/MHz, assuming a 50 MHz cycle time; a 1 Mb ECL static RAM with a 10 ns cycle time has been used as a reference [3]. Although the FIFO memory consumes considerably more power than does the random-access RAM of the shared memory, the additional circuit elements, particularly the control processors, balance out the power requirements. The mean time between failures (MTBF) is estimated to be twice as large for the Shared Bus design as for the Shared Memory design. The numbers given in Table 1 are only rough estimates, and should receive further study before decisions can be made on a design based on these parameters.

7 Accommodations for Circuit-Switched Connections

7.1 Design Considerations

The designs presented thus far have been based on the switching of multicast packets. Time-dependent circuit-switched data traffic is expected to be carried by the switch. These circuits represent a data rate of 7 Mbps each and are expected to comprise sets of 32 adjacent uplink channels, each at the basic data rate of 64 kbps. It is possible to treat these circuits in a packet fashion, assigning headers with routing addresses for each group of bits amounting to a packet's length. This would add significant overhead to the channel, which is undesirable.

Each of the packet switching schemes relies on information in the first word of the packet header to obtain routing information through a lookup table (see Figure 3). With order wire control information passed through a supervisory channel to the control processor, it is possible to cause the address fed to the Multicast Lookup Table to be derived from an Order Wire Lookup

Table 1: Estimated Memory and Power Requirements

	Shared Bus	Shared Memory
Total Memory Capacity	16 Mbits	8 Mbits
Power dissipation		
Memory	60W	5W
Bus Drivers	3W	16W
Memory Management Processors	35W	100W
Misc. Circuits	20W	20W
Total	118W	141W
Estimated size ratio	1	1.25
Estimated MTBF ratio	1	0.5

Memory for those channels designated to comprise circuit-switched data. In this manner, no bits of the data stream need to be used for encoding the routing information.

After determination of the routing for the circuit switched data, including the possibility of multicast broadcast information, it is possible to pass the data to the downlink as if it were packets. A problem arises in the time perishability of this data. If the data is handled as packets, the ground stations would be required to have buffers large enough to store the data for a time equivalent to the maximum expected delay. The first bits of circuit-switched data would not be released by the ground station until this time delay had occurred, to ensure that there would be no breaks in the data stream caused by delayed packets.

To avoid this additional delay and ground station cost, the circuit-switched data needs to be handled in a special manner. In the Shared Memory implementation, special circuit-switched data buffers and/or assignment memories could be established, to ensure high priority to the buffer's selection for timely transmission on the appropriate dwell. This might be an application for video RAM to handle the circuit-switched data stream. In the case of the Shared Bus, an additional set of parallel FIFOs would be used to hold the circuit-switched data. Output enable signals would select the appropriate FIFO based on a dwell reservation scheme.

7.2 Dwell Structure

For a switch that handles only packet data, it is possible to derive an algorithm for sensing dwells in a manner that serves the largest queues first. The selection of a given dwell can be truly random, based on the traffic statistics. The requirement for the synchronousness of circuit-switched data calls for an approach that ensures uniform service to this data.

To keep time synchronization, each circuit-switched channel needs to be served at the average of the uplink rate. With a two-dwell-length buffer at the ground station, variations in the time a dwell is served can be accommodated.

If we assume a fixed-length frame for each of the eight downlink beams, and if we assume a linear progression through the eight dwells of a beam, we can describe a method of handling both packet- and circuit-switched data. As an example, we might set the duration of a frame to be 10

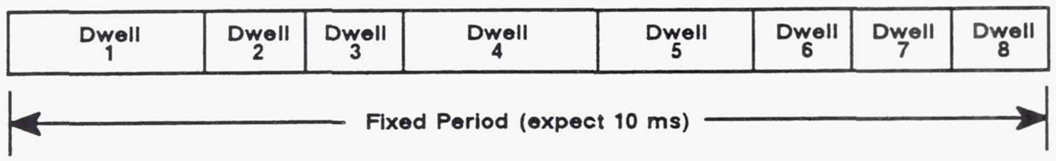

Figure 8: Example Frame

ms; this is the uplink duration for a 600-bit packet on a 64 kbps channel. The average dwell time, including beam steering switching time, is 1.25 ms.

Before the end of the current frame, a Dwell Management processor examines the queue length for each of the eight dwells. The number of circuit-switched channels is also known. The processor determines the amount of time occupied by the circuit-switched traffic and subtracts this amount from the total time available for data; this leaves the amount of time available for packet data traffic. The largest queues will be allocated the largest portion of the available time, to prevent those queues from overflowing. The dwell durations will vary according to the traffic statistics.

Figure 8 gives an example of a frame consisting of eight dwells of various lengths. Figure 9 shows a possible structure for a single dwell. After the dwell synchronization header, a word, if necessary, indicates the number of circuit-switched channels; the data for these channels follows, satisfying the requirement for a constant flow of this data. Packets for the current dwell are then transmitted. Control of the channel setup process can provide for the limiting of circuit-switched traffic if packets are backing up and threatening the capacity of the on-board memory.

Figure 9 portrays packets and circuit-switched data that are contiguous blocks. The architectures presented in this report are also capable of handling packets on an interleaved basis on the downlink. This is illustrated in Figure 10, where the first bit of each packet word indicates whether the word contains the packet header, as discussed in Section 3.2.

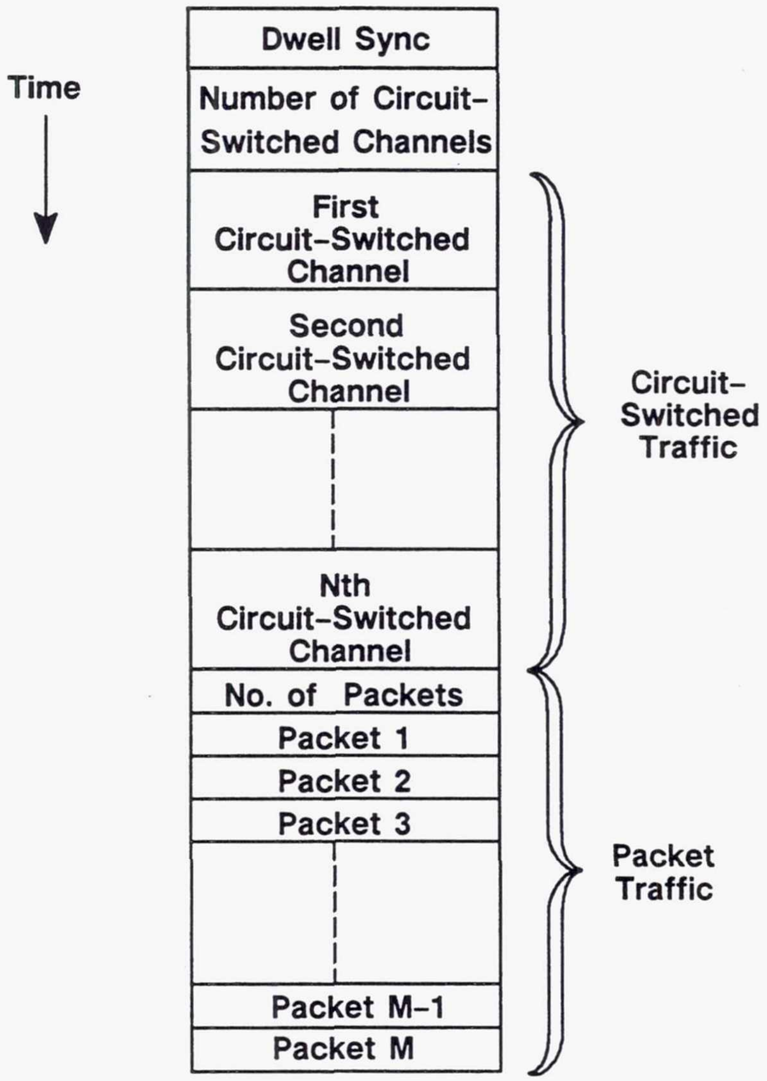
Time
Dwell Sync
Number of Circuit-Switched Channels
First Circuit-Switched Channel
Second Circuit-Switched Channel
Nth Circuit-Switched Channel
No. of Packets
Packet 1
Packet 2
Packet 3
Packet M-1
Packet M
Circuit-Switched Traffic
Packet Traffic

Figure 9: Dwell Structure

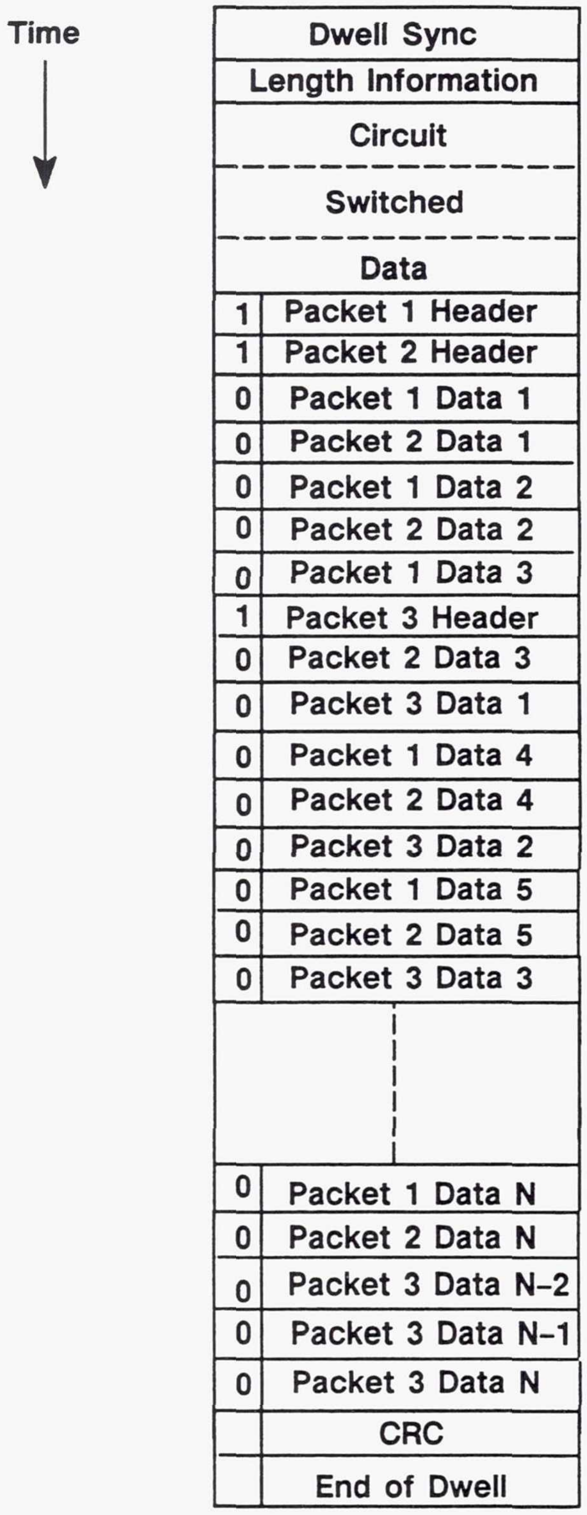

Figure 10: Example of a Dwell with Circuit Switched and Interleaved Asynchronous Packet Data

8 Analysis

In this section, we analyze the relationship between memory size and the probability of packet loss for the switch designs presented earlier. In particular, we determine for each design the amount of memory required to achieve the desired packet loss probability of 10^{-9}. We also compute the optimal packet length and dwell length as functions of the packet header length, the memory size, and the dwell switch time.

8.1 Packet Loss Probability vs. Buffer Size

We consider the three basic switch architectures presented earlier: output queuing (OQ) (i.e., shared bus), shared buffering per beam (SBPB), and completely shared buffering (CSB). In each of the designs there are 8 inputs and 64 outputs. We note that in our shared memory designs, all memory locations are shared among all dwells. This is in contrast to the design of Reference [1], in which each location of the burst transmit buffer is dedicated to a particular dwell.

For the traffic model, we assumed that during each time slot (the time required to transmit one packet on a downlink beam), each of the 64 outputs receives a packet from each of the 8 inputs with probability p/64. Thus, each output receives an average of p/8 packets per slot. Since there are 8 outputs per downlink beam, p is the fraction of downlink slots that are used. Since each input is the aggregate of 1024 independent streams, we can assume that input traffic is not bursty.

8.1.1 Analysis for Constant Dwell Lengths

Our analysis of probability loss makes use of the formulas of Hluchyj and Karol [4]. Although their paper assumes an NxN switch, their analysis is easily extended to an MxN switch. One important difference between their model and ours is that they assume each output is served continuously. In our model, an output corresponds to one of 64 dwells (8 per beam), and at most one dwell per beam can be served at a time. Thus, packets queued for a particular dwell may have to wait for the other 7 dwells to be served before being served. The model of Reference [4] would assume that all dwells are served continuously at 18.75 Mbps rather than 1/8 of the time at 150 Mbps.

It is easy to see that having to wait for a particular dwell to be served adds at most $(7/8)D$ packets per output to the buffer requirement (while achieving the same loss probability), where D is the number of packets in a single dwell. This is because up to $7D$ time slots may have passed since a given dwell has been served, during which $(7/8)D$ packets would have been served in the model of Reference [4]. Thus, to determine an upper bound for the buffer size required to achieve a given probability, we can compute the size according to [4], and then add $(7/8)D$ packets per output, i.e., $56D$ packets. Equivalently, this gives an upper bound on the packet loss probability corresponding to a given buffer size.

For the shared buffering models CSB and SBPB, the number of additional packets per output that must be buffered is half of that for OQ, i.e., $(7/16)D$ packets. This is because, on the average, only $(7/2)D$ time slots have passed since a given dwell has been served, during which $(7/16)D$ packets would have been served in the model of Reference [4]. Since memory is shared among dwells, the total additional buffering required for each beam is 8 times this average, or $(7/2)D$ packets. For all 8 beams we therefore add $28D$ packets.

Figure 11 gives the computed upper bounds for packet loss probability as a function of buffer size (in packets) for the three switch designs, assuming $p = 0.9$ and $D = 32$ (implying 256 packets per frame). The dashed curves show the packet loss probabilities for OQ and SBPB obtained from

Table 2: Buffer Size Requirements

Load	Dwell length	OQ	SBPB	CSB
.9	32	6784	2304	1444
.8	32	4288	1544	1135
.9	256	19328	8580	7716
.8	256	16832	7816	7407

Storage requirement in packets required to achieve a packet loss probability of 10^{-9} for Output Queuing (OQ), Shared Buffering Per Beam (SBPB), and Completely Shared Buffering (CSB). MCDD storage is not included.

simulations, using the above traffic model. As predicted, the upper bounds are greater than the simulated probabilities. For a given packet loss probability, the corresponding computed buffer size is a good approximation to that obtained from simulations.

Table 2 shows the buffer size required to achieve a packet loss probability of 10^{-9} for the three switch designs, for different values of p and D. The buffer sizes in the figure and table do not include the buffering required at the MCDD, which would be 8*1024 = 8192 if we assume that a single packet must be stored for each uplink channel. Table 2 shows that, for $p = .9$, CSB requires 21% as much memory as OQ if $D = 32$, and 40% as much memory as OQ if $D = 256$, not including MCDD storage. However, if we include MCDD storage, CSB requires 64% as much memory as OQ if $D = 32$, and 58% as much memory as OQ if $D = 256$. In the limit as D increases to infinity, CSB will require about half as much memory as OQ (due to the terms $28D$ and $56D$ discussed above). In all cases, SBPB requires at most 1000 packets of memory more than CSB.

We define the multicast factor to be the average number of destinations to which each packet (including nonmulticast packets) is addressed. If the uplinks and downlinks were fully utilized, the multicast factor would be 150/65.5 = 2.3. However, the factor can be higher if the uplinks are underutilized, and lower if the downlinks are underutilized. The traffic model we use is independent of the multicast factor, but assumes, for the CSB design, that every copy of each packet is stored in a separate location. For the case in which the CSB stores only one copy of each packet, the storage requirement can be obtained by dividing the appropriate size in Table 2 by the multicast factor.

Table 2 assumes that all dwells of each downlink beam are of equal length, and that the traffic load is the same for each dwell. This assumption of uniform demands is actually the worst case. For example, if all traffic on each downlink beam used the same dwell (at 80% or 90% of the downlink beam capacity), then, assuming that the dwell lengths can be adapted according to the near real time traffic demands, the storage requirement would be lower than that shown in Table 2. This is because the full capacity of each downlink beam is continuously dedicated to the single utilized dwell.

8.2 Analysis of Packet Division Schemes

In the packet division schemes discussed in Sections 3.2 and 5.2, each packet is divided into K subpackets, where only the first subpacket contains a header. The above analysis can be applied to these schemes simply by letting a subpacket replace the role of a packet. Therefore, in this case,

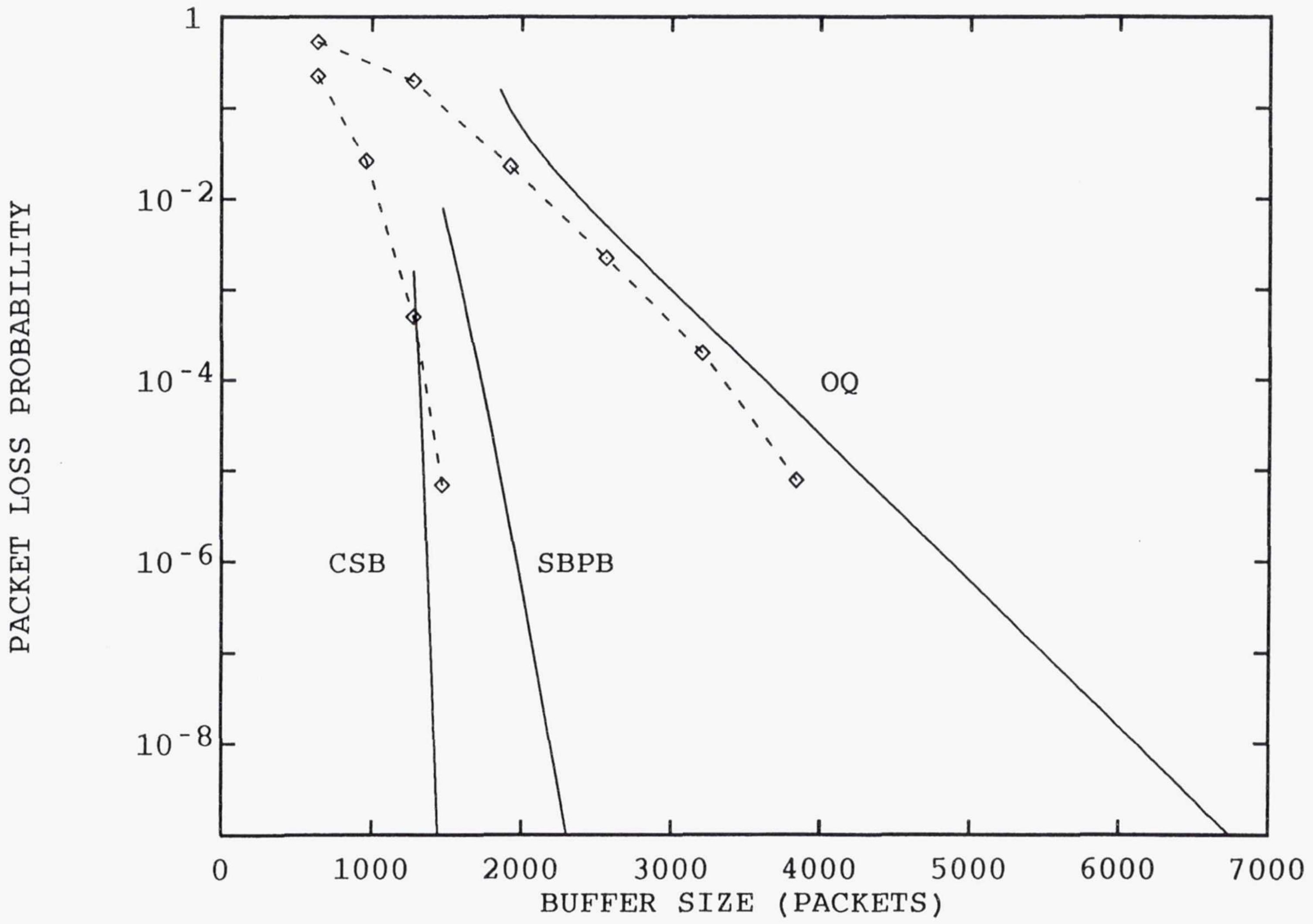

Figure 11: Upper Bound for packet loss probability vs. buffer size for three switch designs. The dashed curves show simulation results for OQ and SBPB (MCDD storage is not included).

Table 3: Theoretical Storage Requirement

Load	SBPB	CSB
.9	624	176
.8	312	81

Theoretical storage requirement in packets required to achieve a packet loss probability of 10^{-9} for SBPB and CSB, assuming arbitrarily fast dwell control and negligible dwell switch time. MCDD storage is not included.

Table 2 gives the storage requirement in subpackets, and D represents the number of subpackets per dwell.

As an example, we compare the storage requirements for two schemes: Scheme 1 transmits full packets, and Scheme 2 divides each packet into 8 subpackets. In Scheme 1, $D = 32$; we therefore let $D = 256$ in Scheme 2, so that both schemes have the same dwell duration. Assuming $p = 0.9$, and adding the MCDD storage of 8192 packets (subpackets for Scheme 2) to the storage requirement of Table 2, we see that Scheme 1 requires 9636 packets of storage for CSB and 14976 for OQ. Similarly, Scheme 2 requires 15908 subpackets of storage for CSB and 27520 for OQ. Since a subpacket is 1/8 of a packet, Scheme 2 requires the equivalent of 1989 packets of storage for CSB and 3440 for OQ. Thus, the packet division scheme requires only 21% and 23% as much memory as Scheme 1 for CSB and OQ, respectively. Further reductions in the memory requirement can be achieved by choosing D to be less than 256 for the packet division scheme, at the expense of a decrease in throughput due to the larger fraction of time required for dwell switching.

8.3 Analysis for Dynamic Dwell Control

The designs presented in this report allow the option of dynamic dwell control, in which the dwell lengths can be adjusted on a frame-by-frame basis according to the current number of packets queued for each dwell. Table 3 shows the dramatic savings in memory that this method can achieve when combined with memory sharing. This analysis assumes arbitrarily fast dwell control and negligible dwell switch time, and thus represents only a theoretical limit.

Since shared memory is used, the analysis is independent of the protocol used, as long as the protocol is "work conserving," i.e., has the property that each downlink beam transmits packets as long as there is a packet waiting for any dwell of the beam. One possible protocol is as follows. The beam first selects the dwell for which the most packets are buffered. It then continues to serve that dwell for a preassigned length of time D, or until all packets for that dwell have been transmitted, whichever comes first. This step is then repeated. Although this scheme may require the beam to switch very rapidly from dwell to dwell when few packets are buffered, thus wasting bandwidth during the dwell switch time, it does not have this problem in the more critical situation in which the number of buffered packets is large (since the dwell length is D in this case).

CSB with dynamic dwell control is equivalent to shared buffering with 8 outputs, and SBPB with dynamic dwell control is equivalent to output queuing with 8 outputs (one output per downlink beam) and with 64 inputs (since as many as 8 packets addressed to the same downlink beam can arrive in a time slot from each of the 8 uplink beams). Thus, we can use the analysis of Reference [4]. The OQ design is omitted because it is more difficult to analyze.

8.4 Simulation of Dynamic Dwell Control

The dynamic dwell control protocol described above would be difficult to combine with circuit switching, since it does not use a fixed-length frame. We simulated a dynamic dwell control protocol that uses a fixed-length frame and can be combined with circuit switching as discussed in Section 7.2. In this protocol, the 8 dwell lengths for each frame are assigned at the beginning of the frame, based on the current number of packets queued for each dwell. The dwell lengths for each beam are chosen so that the length of each dwell j ($j = 1, ..., 8$) is proportional to the number of packets queued for dwell j plus $j/8$ times the expected number of packets for dwell j arriving during a frame. This expected number of packets can be estimated by maintaining a time average. Of course, the dwell lengths for each beam are normalized so that they sum to to frame length. Thus, the length of each dwell is roughly proportional to the estimated number of packets that will be queued for that dwell the next time it is served.

This protocol is not work conserving, since it is possible that there are no packets to transmit for the currently served dwell while there are packets waiting for another dwell of the same beam. However, since the length of each dwell is proportional to the estimated queue length of that dwell, the fraction of time that each beam is idle is minimized.

Figure 12 gives the simulation results for the OQ (shared bus) and SBPB designs, with and without dynamic dwell control. We used the same traffic model described in Section 8.1.1, with $p = 0.9$ and a frame length of 256 packets. Each data point represents a simulation of at least 10 million packets. Since obtaining probabilities less than 10^{-6} would require more than this many packets, the curves were extrapolated assuming constant slopes. This is a reasonable assumption, since the computed curves of Figure 11 have constant slopes for small probabilities.

For output queuing, using dynamic dwell control reduces the buffer size required to achieve a 10^{-9} packet loss probability by about 23%. For SBPB, the reduction in buffer size is only about 9%, but for a fixed buffer size the probability is reduced by a factor of about 35.

In our simulations, we assumed constant traffic demands. However, it is clear that dynamic dwell control will react quickly to changes in traffic demands. Since the demand for each dwell can change, some method is needed for adapting the dwell lengths. It is also possible to use "quasi-dynamic" dwell control, in which the dwell lengths are changed based on the short-term average demands. However, dynamic dwell control is preferable since it reacts more quickly.

8.5 Optimal Packet and Dwell Lengths

It is clear that choosing a small dwell length, and thus a small frame length, significantly reduces the memory requirement. However, because of the dwell switch time, a small dwell length also reduces the effective data throughput. Similarly, choosing a small packet size reduces the memory requirement, but because of the packet header, it also reduces the effective data throughput.

In this subsection, we compute the optimal packet length and dwell length as functions of the packet header length, the memory size, and the dwell switch time. Let D denote the number of packets per dwell. Table 2 gives the required buffer size for the two cases $D = 32$ and $D = 256$. In general, the buffer requirement can be expressed as $B + 56D$ for OQ and $B + 28D$ for CSB and SBPB, where B is independent of the dwell length.

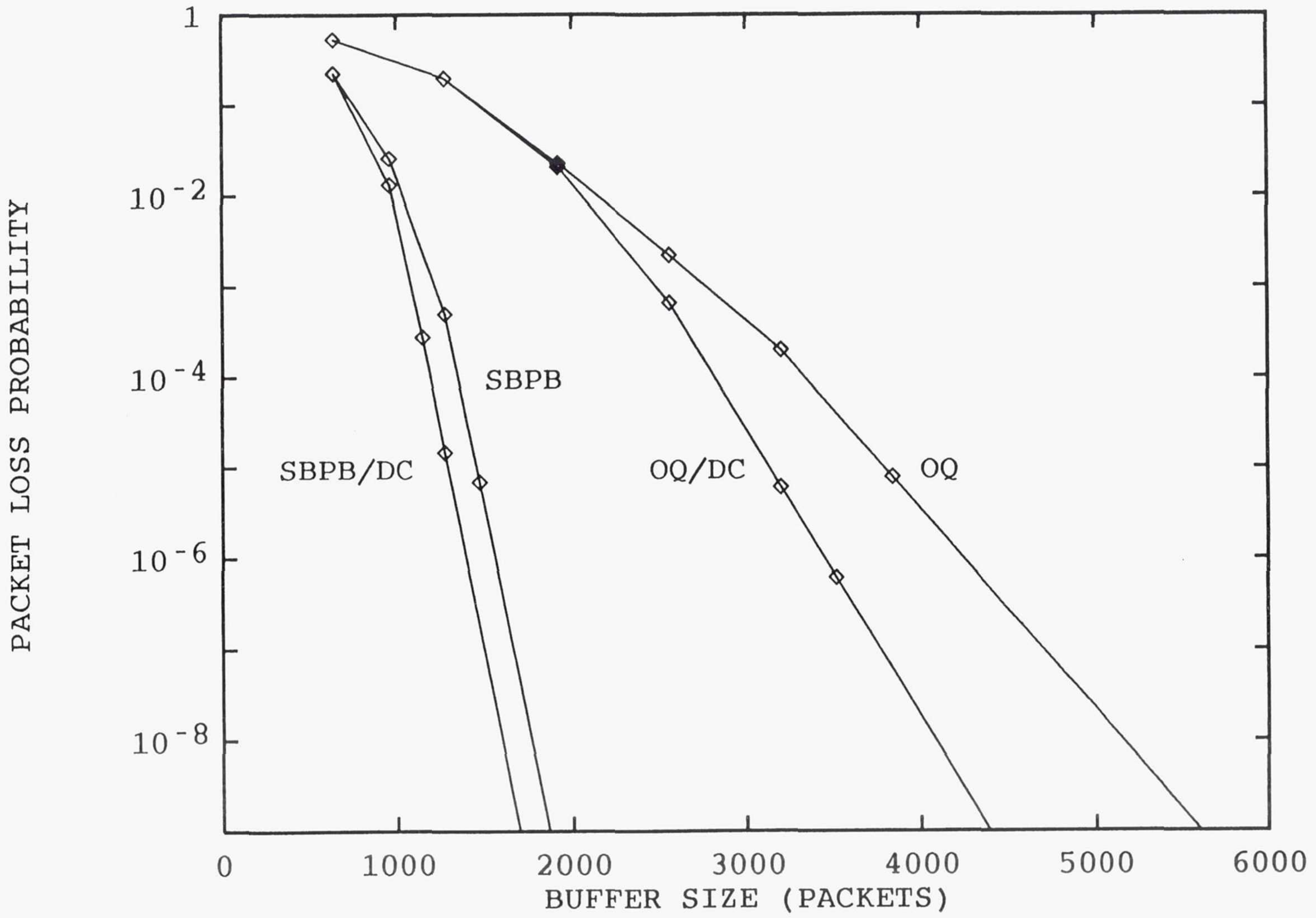

Figure 12: Simulated packet loss probability vs. buffer size for OQ and SBPB, with and without dynamic dwell control (MCDD storage is not included).

We let L denote the packet length in bits, H denote the number of bits in the packet header, S denote the dwell switch time in bits, and M denote the amount of available memory in bits. We let p denote the fraction of available downlink bandwidth (not including the dwell switch times) used for packet transmissions.

The downlink data throughput (i.e., the fraction of the downlink capacity that is used by data, not including headers), is thus

$$T = p\,\frac{L-H}{L}\,\frac{DL}{DL+S} \quad .$$

The total memory requirement in bits is $L(B+64D)$. Our objective is to choose D and L to maximize T subject to the constraint $L(B+64D) = M$. This constraint implies

$$L = \frac{M}{64D+B} \quad .$$

Substituting the equation for L into the equation for T, we obtain

$$T = .9\frac{-aD^2+bD}{cD+e} \quad ,$$

where $a = 64H$, $b = M - BH$, $c = M + 64S$ and $e = BS$. Setting the derivative of T with respect to D to zero, we obtain the quadratic equation

$$acD^2 + 2aeD - be = 0 \quad ,$$

which has the solution

$$D = \frac{-ae + \sqrt{a^2e^2 + abcd}}{ac} \quad .$$

For the packet division schemes in which each packet is divided into K subpackets, the above analysis can be applied by letting a subpacket replace the role of a packet. Thus, L is replaced by the length $L' = L/K$ of a subpacket, and D is replaced by the number $D' = K * D$ of subpackets per dwell. Since a header occurs only in 1 of every K subpackets, H is replaced by $H' = H/K$ (i.e., H' is the average header length per subpacket).

Tables 4–7 give the optimal values of D and L for M ranging from 2 to 16 megabits, for OQ and SBPB with and without packet division. We assume that $p = 0.9$, $H = 65$, and $S = 1500$ (corresponding to a dwell switch time of 10 microseconds); that B is chosen to achieve a packet loss probability of 10^{-9}; and that the MCDD must store 8192 packets or subpackets. For the packet division schemes, we assume that $K = 8$.

The tables shows that, when the downlink data throughput is above 80%, then the optimal dwell size is nearly constant, and the optimal packet length increases nearly linearly with the memory size. Note that the optimal packet lengths are much greater when packet division is used. For a memory size of 2 Mb, SPBP with packet division gives the best throughput, but OQ with packet division gives almost the same throughput.

9 Conclusions

We have presented three feasible packet switch designs for the problem considered: the shared bus, shared memory per beam, and completely shared memory designs. We have presented alternative methods for improving the performance of each architecture, and have discussed the tradeoffs

Table 4: Optimal Packet Length (in Bits) and Dwell Length (in Packets) for Output Queuing for Various Limits on Memory Size

Memory Size	Dwell Length	Packet Length	Data Throughput
2Mb	68.903931	132.333679	0.393245
4Mb	87.157555	256.009644	0.629195
6Mb	92.955582	380.065399	0.715692
8Mb	95.812744	504.198822	0.760365
10Mb	97.514809	628.361145	0.787620
12Mb	98.644669	752.537415	0.805977
14Mb	99.449409	876.721497	0.819181
16Mb	100.051735	1000.910278	0.829134

Table 5: Optimal Packet Length (in Bits) and Dwell Length (in Packets) for Output Queuing with Packet Division for Various Limits on Memory Size

Memory Size	Dwell Length	Packet Length	Data Throughput
2Mb	34.293221	766.809387	0.779259
4Mb	35.561707	1513.015137	0.837974
6Mb	35.989861	2259.277588	0.858274
8Mb	36.204952	3005.554199	0.868563
10Mb	36.334335	3751.836426	0.874782
12Mb	36.420727	4498.121582	0.878947
14Mb	36.482502	5244.408203	0.881931
16Mb	36.528870	5990.695801	0.884174

Table 6: Optimal Packet Length (in Bits) and Dwell Length (in Packets) for SBPB for Various Limits on Memory Size

Memory Size	Dwell Length	Packet Length	Data Throughput
2Mb	66.305321	174.572647	0.500090
4Mb	77.807587	339.598572	0.688645
6Mb	81.554939	504.900116	0.756575
8Mb	83.415344	670.262085	0.791486
10Mb	84.527657	835.647095	0.812735
12Mb	85.267639	1001.043274	0.827027
14Mb	85.795448	1166.445801	0.837298
16Mb	86.190903	1331.852173	0.845034

Table 7: Optimal Packet Length (in Bits) and Dwell Length (in Packets) for SBPB with Packet Division for Various Limits on Memory Size

Memory Size	Dwell Length	Packet Length	Data Throughput
2Mb	29.641747	985.236755	0.799556
4Mb	30.536077	1946.462402	0.848531
6Mb	30.838190	2907.724365	0.865405
8Mb	30.990009	3868.995361	0.873946
10Mb	31.081345	4830.270020	0.879106
12Mb	31.142338	5791.546387	0.882559
14Mb	31.185955	6752.823730	0.885033
16Mb	31.218697	7714.102051	0.886892

between them. We conclude that the completely shared memory design requires significantly less memory than the shared bus design, but requires only slightly less power because of its increased processing requirement, and is less reliable because of its increased complexity. A good compromise is achieved with the shared-memory-per-beam design, which requires at most 1000 packets of memory more than the completely shared memory design, while achieving reduced complexity and increased reliability.

A dramatic reduction in memory requirement can be achieved by using a packet division scheme, which allows part of a packet to be transmitted on the downlink before the entire packet has arrived on the uplink. If this scheme is used with shared memory per beam, then a downlink data throughput (excluding packet headers) of 80% can be achieved with only 2 Mb of memory (including MCDD storage). If the same scheme is used with the shared bus design, a downlink data throughput of 78% can be achieved with the same amount of memory.

We have also described and simulated a method for dynamically controlling the dwell lengths based on the instantaneous queue sizes, and have presented a design that combines this method with circuit switching. This method adapts quickly to changing traffic demands, and does not add significant complexity or cost to the satellite and ground station designs. Dynamic dwell control is therefore recommended, regardless of the choice of the switch design.

Some of the ideas we have presented are only preliminary, and require future work before they can be implemented. For example, there are many ways to implement dynamic dwell control, and more research is needed to determine the optimal scheme. The same can be said for the packet division scheme discussed in Section 5.2. Also, more complete power, size, and reliability estimates are needed, and the impact of the proposed designs on ground station complexity needs to be studied.

References

[1] W.D. Ivancic and M.J. Shalkhauser. Destination Directed Packet Switch Architecture for a 30/20 GHz FDMA/TDMA Geostationary Communication Satellite Network. Technical report, NASA Lewis Research Center, 1991.

[2] Leilani R. Tamura et al. A 4-ns BiCMOS Translation-Lookaside Buffer. *IEEE Journal of Solid-State Circuits*, pages 1093–1101, October 1990.

[3] Masahide Takada et al. A 5-ns 1-Mb ECL BiCMOS SRAM. *IEEE Journal of Solid-State Circuits*, pages 1057–1062, October 1990.

[4] M.G. Hluchyj and M.J. Karol. Queueing in High-Performance Packet Switching. *IEEE Journal on Selected Areas in Communication*, pages 1587–1597, 1988.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE January 1993	3. REPORT TYPE AND DATES COVERED Final Contractor Report

4. TITLE AND SUBTITLE Technical Support for Digital Systems Technology Development Task Order 1 ISP Contention Analysis and Control	5. FUNDING NUMBERS WU–690–60–21 NAS3–25934
6. AUTHOR(S) Roy H. Stehle and Richard G. Ogier	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SRI International 333 Ravenswood Avenue Menlo Park, California 94025–3493	8. PERFORMING ORGANIZATION REPORT NUMBER E–7564
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135–3191	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR–190758

11. SUPPLEMENTARY NOTES

Project Manager, William D. Ivancic, Digital Systems Technology Branch, NASA Lewis Research Center, (216) 433–3494.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 17	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This study has investigated alternatives for realizing a packet-based network switch for use on an FDMA/TDM geostationary communication satellite. Each of the eight downlink beams supports eight directed dwells. The design needed to accommodate multicast packets with very low probability of loss due to contention. Three switch architectures were designed and analyzed. An output-queued, shared bus system yielded a functionally simple system, utilizing a first-in, first-out (FIFO) memory per downlink dwell, but at the expense of a large total memory requirement. A shared memory architecture offered the most efficiency in memory requirements, requiring about half the memory of the shared bus design. The processing requirement for the shared-memory system adds system complexity that may offset the benefits of the smaller memory. An alternative design using a shared memory buffer per downlink beam decreases circuit complexity through a distributed design, and requires at most 1000 packets of memory more than the completely shared memory design. Modifications to the basic packet switch designs have been proposed to accommodate circuit-switched traffic, which must be served on a periodic basis with minimal delay. Methods for dynamically controlling the downlink dwell lengths have been developed and analyzed. These methods adapt quickly to changing traffic demands, and do not add significant complexity or cost to the satellite and ground station designs. Methods for reducing the memory requirement by not requiring the satellite to store full packets have also been proposed and analyzed. In addition, optimal packet and dwell lengths have been computed as functions of memory size for the three switch architectures.

14. SUBJECT TERMS Contention control; FDMA; TDM; Packet-switch; Shared-bus	15. NUMBER OF PAGES 32
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT