NASA Contractor Report 195470

Alternative Packet Switch Architectures for a 30/20 GHz FDMA/TDMA Geostationary Communication Satellite Network

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

June 1995

Prepared for
Lewis Research Center
Under Contract NAS3–25934

National Aeronautics and
Space Administration

ALTERNATIVE PACKET SWITCH ARCHITECTURES FOR A 30/20 GHz FDMA/TDMA GEOSTATIONARY COMMUNICATION SATELLITE NETWORK
Draft Final Report

Roy Stehle and Richard G. Ogier
SRI International

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 was to look 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 five 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 [4]. 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 TDMA high-power transmitting amplifier at each ground station.

The satellite will employ eight uplink beams and eight downlink beams to cover CONUS. A block diagram of the satellite system is shown in Figure 1, which is a copy of Figure 1 from [4]. Each of the uplink beams will be served by a multi-channel demodulator/demultiplexor (MCDD) which will provide packet synchronization and decoding for 1024 channels each having a data rate of 64 kbp. 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 8 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 is complicated by the necessity that a multicast capability must 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 it 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

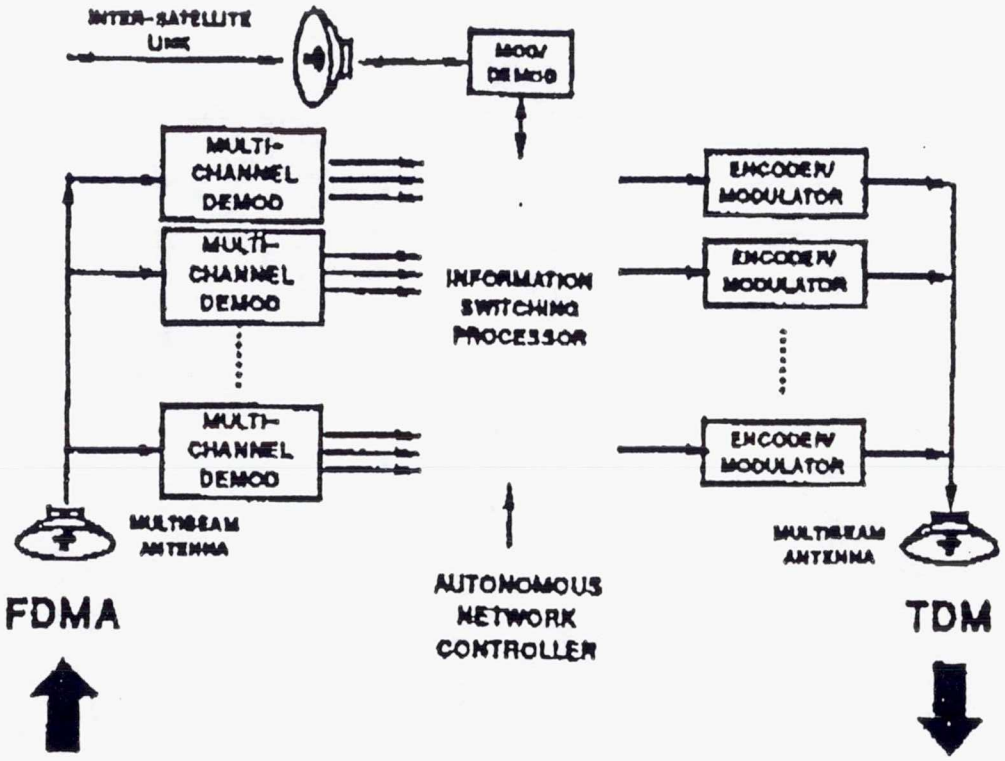

Figure 1 FDMA/TDM Network

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 Memory configuration shares one large memory buffer among all of the output devices (i.e., downlink modulators). We will describe each of these implementations, variations thereto, and the advantages and disadvantages of each 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 given in Figure 2. In this figure, the eight uplink beams and their associated multichannel demultiplexer/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; the width offers a form of space division multiplexing to allow for proportionally slower speed circuits for handling the interface. The width of 16 bits yields a word duration of 30 ns for our example, which is easily within the current state of the art. A bus width of 16 bits also matches the projected packet destination address space.

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 start 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 down link data stream or from a secondary timing source, such as a Global Positioning Satellite (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 MWords/sec) 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 less timing problems might result if a word from each of the channels associated with a single beam are 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.

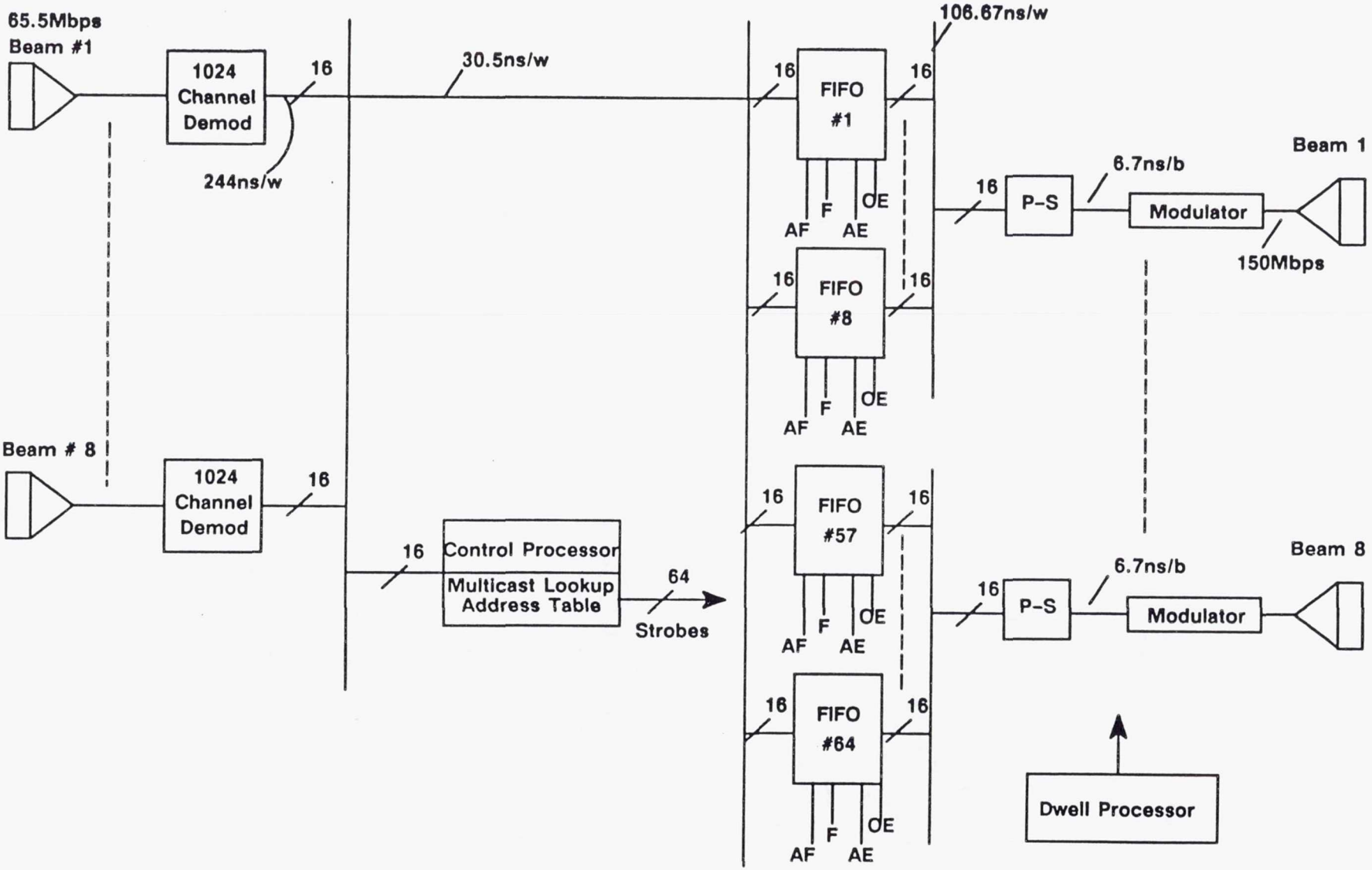

Figure 2 Shared Bus Implementation

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 an address into a routing table. In this manner, as few or as many of the addresses may 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 provides a pictorial of 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 which will receive the packet. These bits are used to generate latch strobes into the 64 individual FIFO memories assigned to each dwell. This is the implementation of a shared bus.

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 to generate the appropriate strobes to the FIFO memories until a packet is transferred. 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 in this case would not generate any strobes, in this case. Additional schemes are possible allowing variable packet lengths, with an increased processing requirement to have counters for each channel, rather than a single one for the entire block of 8192 channels.

The approach described requires that the header's destination address and the following words of each packet form a recognizable pattern to the ground stations, as 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. If such a distinction cannot be established, then full packets would need to be assembled for each channel in the MCDD before full packets could be transferred on the shared bus, significantly increasing the memory requirements for the MCDD.

It may be possible to overcome this problem, as explained in the next subsection. In any case, the shared memory design of Section 4 does not have this problem, since the words of each packet can be stored noncontiguously in the shared memory, and then reassembled into a full packet just before transmission. Thus, in the shared memory design, full packets need not be buffered at the MCDD.

3.3 Packet Division Scheme

In the packet division scheme, packets are divided into a number, say 8, of 'minipackets' (which can be larger than a word) such that only the first minipacket of a packet contains a header. Each minipacket is bussed to the appropriate downlink FIFO as soon as it arrives (thus the MCDD only needs to store two minipackets per channel). When the first minipacket of a packet is transmitted in a downlink slot (corresponding to the appropriate dwell), the corresponding slot in the next 7 frames is reserved for the remaining 7 minipackets of the same packet. The length of the downlink frame is chosen so that it is no longer than the time required to send one minipacket on the uplink (so that minipackets are retransmitted on downlink as fast as they are received on the uplink).

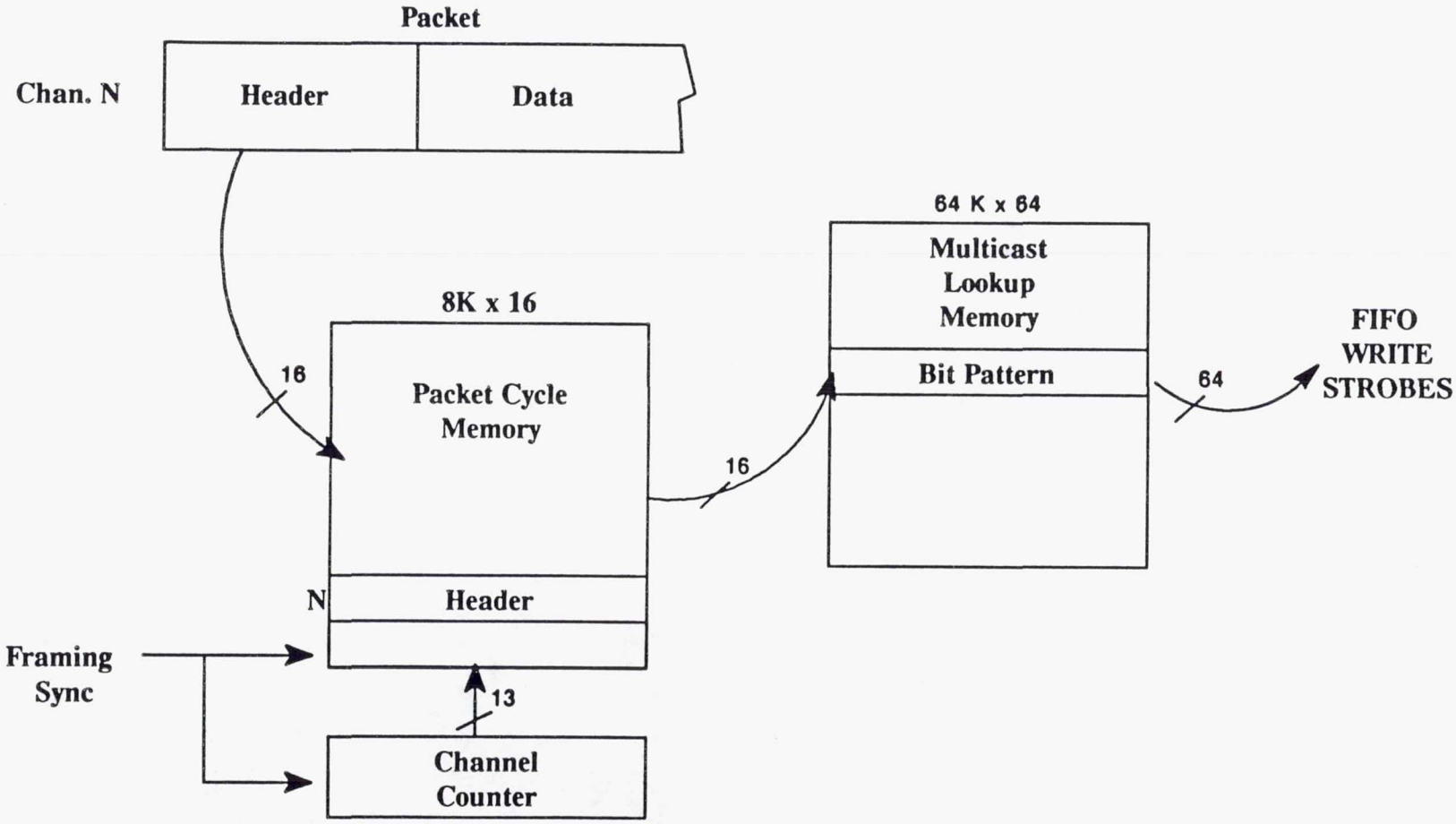

Figure 3 Multicast Address Lookup Memory

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 the frame to be very small, because of the dwell switching time. This can be avoided by making the packets (and thus minipackets) 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 of future study. The idea can be applied to both the shared bus and shared memory architectures.

3.4 Dwell-assigned FIFO Memories

The FIFO memories provide the temporary storage of packets as the downlink dwell order and 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. The discussion of contention and the influence of memory size will be discussed in Section 7.

The aggregate data rate of the uplink data determines the speed of the FIFO memory. Eight beams of 65.5 Mbps data divided into 16-bit words requires the transfer of a word every 30.5 ns. Currently available FIFOs, such as the Cypress Semiconductor CY7C474, offer this transfer rate in a 32K word by 9 bit configuration. The 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 correction. At a 150 Mbps downlink data rate, the capacity of one dwell FIFO storage represents 3.50 ms of traffic.

Referring to Figure 2, it can be seen that the eight dwell FIFOs 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 FIFOs associated with a single beam can be combined into a lookup table. The output of this table would control dwell time, on the present dwell, and selection of 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 might be 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 required. The assumption that beam steering will require a significant amount of time (i.e., ¿ 1 microsecond) relative to a bit time, coupled with resynchronization bits, dictates that the dwell time be somewhat to 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.

3.5 Conclusions

The advantage of the Shared Bus architecture just described is in 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, as each dwell has been assigned a FIFO. 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 dynamically reconfiguring 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 roundtrip 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, it is required to 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 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.

Just as the uplink demodulators and demultiplexors provided two word buffering for synchronization and processing, similar buffering is used for each of the downlink beams and their associated

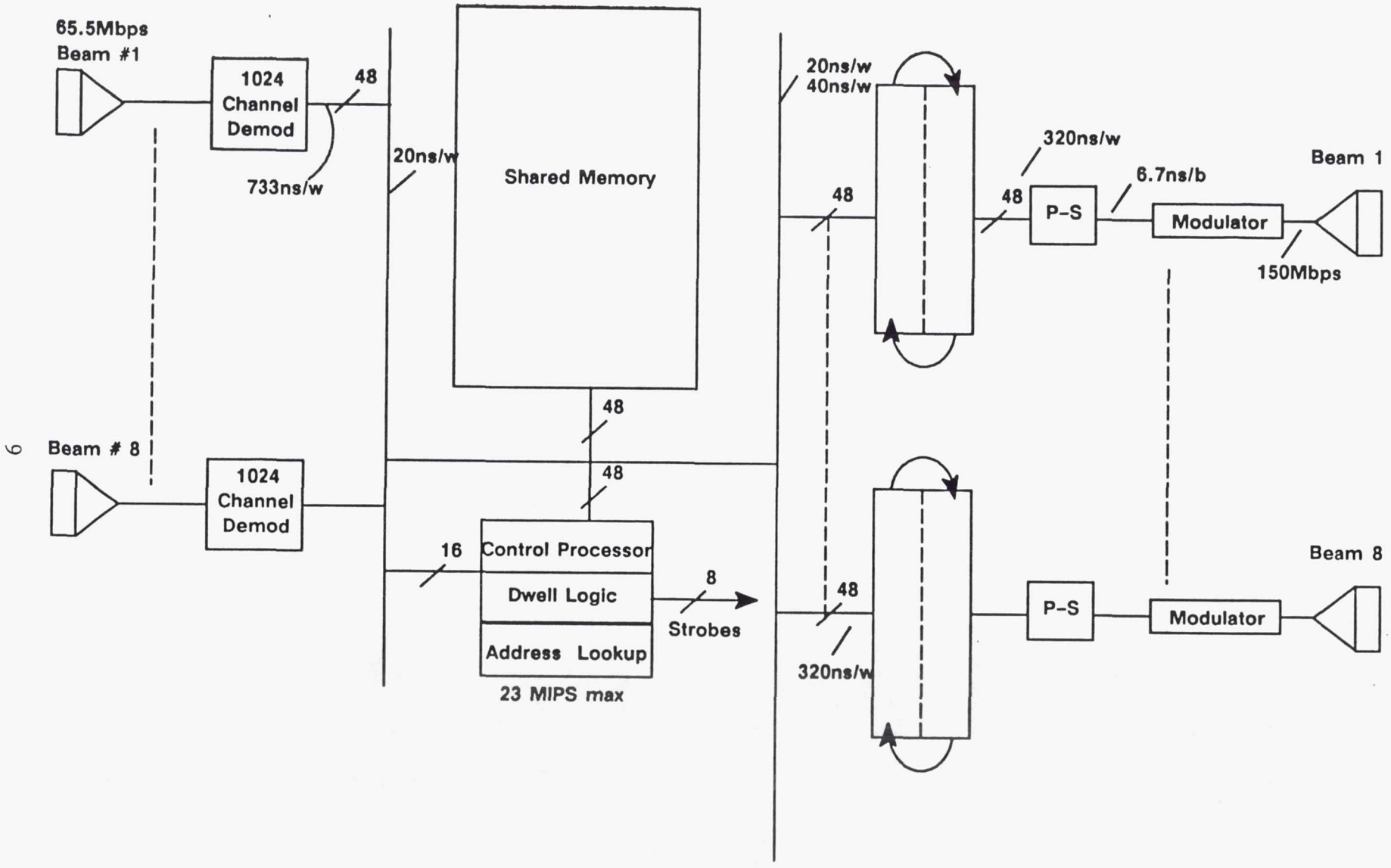

Figure 4 - Shared Memory Implementation

modulators. The 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 roundtrip propagation time relative to a 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 channel. 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.

With the storage for the first word of each channel's packet, the address field is examined to determine its destination(s). A reprogrammable look-up address table produces a 64-bit output word with bits set appropriate to the dwell(s) which 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, as was done in the case of the Shared Bus' FIFO memory. The Dwell Service Memory will be configured in the fashion 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 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. The processing for dwell management is a much more complex requirement. It may require that a dwell management processor be dedicated to each beam, as in the case of the Shared Bus implementation, due to the complex logic required.

By scanning the Dwell Management Memory or by examination of counters representing the number of packets destined for each dwell, the individual or composite Dwell Management Processor will decide 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

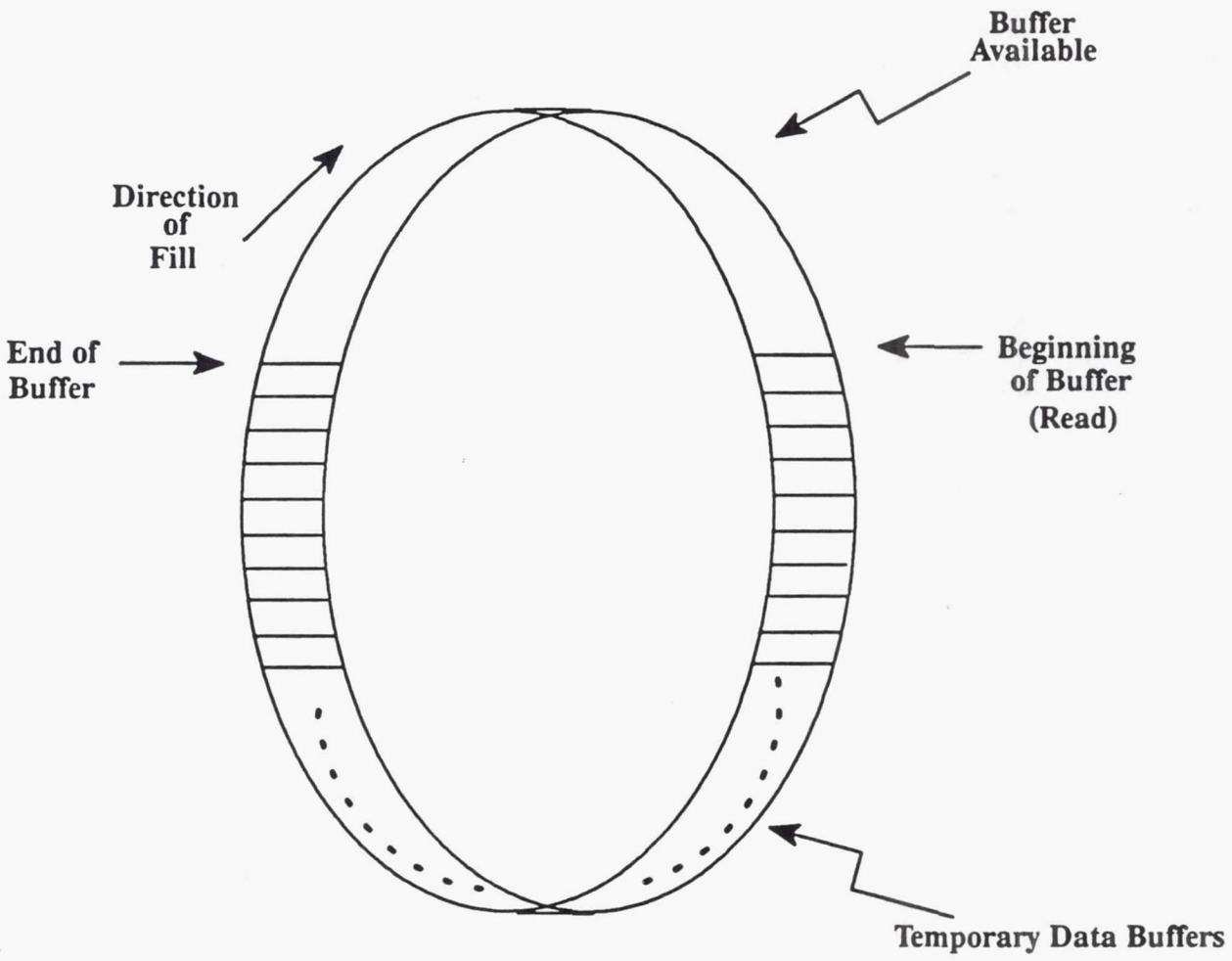

Figure 5 - Circular Buffer Storage

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 multi-channel 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 in 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) depends 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 was retained. The relocated packet could be stored in a contiguous memory block if an address coding bit was 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 an 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 a distinct advantage that the memory is shared by all processes. Its relatively tight packing of that memory in a continuous fashion increases the efficiency of usage. Only one copy of a packet is required for multicasting, offering increased efficiency. A large block of memory is also fairly efficient 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 three-fold 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 storage location. This comes at only a minor cost for a more complex storage address generator formed from a channel address counter, and offset memory, and a word counter. It has an additional advantage of being able to isolate arbitrary blocks of memory from usage in case of a fault. Large blocks of contiguous memory are not required. The disadvantage lies in the need for a potentially more sophisticated processor to find the buffers which are "empty" and able to accept new packets. This could use a simple algorithm of taking the next free buffer from the last one used, leading to the inclusion of the word "quasi" in the description of this approach.

4.5 Dual-Port Video RAM Approach

It would be desirable to use the random write, sequential read feature of a dual-ported 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 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 for using this type of memory in this switching application is the comparatively 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 be easily handled by this sequential access output port. While this is similar to constructing a software equivalent of the hardware FIFO, proposed in the Shared Bus implementation, the advantage lies in the sharing of the total memory based on traffic statistics. It would seem to require a high-speed processor to preload these buffers and/or it would require banks of memory dedicated per a beam, at least. This then becomes equal

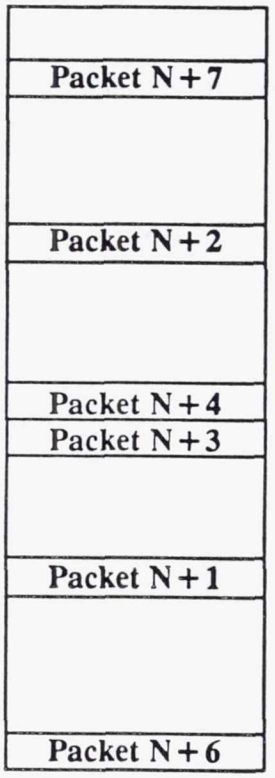

Figure 6 - Quasi -Arbitrary Buffer Storage

to or more difficult to implement than the (hardware) FIFO memory. This approach is mentioned to show some consideration for an architecture that does not seem feasible with current circuit designs but that might be implemented with new technology.

4.6 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 there are technical reasons why a direct connection parallel bus would not be 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 is 524 Mbps. Serial to parallel and parallel to serial converters have been implemented 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 with easier interconnection and reduced interference.

5 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 FIFOS, 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 to a custom circuit implementation lies in reliability.

Reliability is expected to be easily achieved in the Shared Bus implementation from the limited size of the data bus, reduced 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 FIFOs of the Shared Bus system must all be as large as the worst case traffic might dictate.

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 memory required and power consumption for each implementation is included in Table 1. We assume, as is implied by the analysis of Section 7, 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 ([1]); 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

	Shared Bus	Shared Memory
Total Memory Capacity	32 Mbits	16 Mbits
Power dissipation		
Memory	115W	10W
Bus Drivers	3W	16W
Memory Management Processors	35W	100W
Misc. Circuits	20W	20W
Total	173W	146W
Est. size ratio	1	1.4
Est. MTBF ratio	1	0.5

Table 1: Estimated Memory and Power Requirements

figure of 10 mW/Mb/MHz assuming a 50 MHz cycle time; a 1Mb ECL static RAM with 10ns cycle time has been used as a reference ([2]). 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 physical size and system availability estimates given in Table 1 represent engineering estimates, together with these for power consumption, should receive further study before decisions can be made on a design based on these parameters.

6 Accommodations for Circuit Switched Connections

6.1 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 packets length. This would add significant overhead to the channel, which is desirable.

Each of the packet switching schemes relied on information in the first word of the packet header to provide 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 Memory for those channels designated to comprise circuit switched data. In this manner, no bits are lost in the data stream to encode the routing in- formation.

Having determined the routing for the circuit switched data, including the possibility for multicast broadcast information, it is possible to pass the data to the downlink as if it were packets. A problem arrives in the time parsability of this data. If 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 memorys 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. For the case of the Shared Bus, an additional set of parallel FIFOs would be used to hold the circuit switched data. Output enables would select the appropriate FIFO based on a dwell reservation scheme.

6.2 Dwell Structure

For a switch handling 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 10ms; 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 1.25ms.

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 determined 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 these queues from overflowing. The dwell durations will vary according to the traffic statistics.

Referring to Figure 7, we see 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. A word indicates the number of packets may be set equal to zero to better allocate time to busy dwells. Figure 8 is an example of this process. Control of the channel setup process can provide for the limiting of circuit switched traffic if packet data are backing up and threatening the capacity of the on-board memory.

The figures portray packets and circuit switched data that are contiguous blocks. The structures for network switched in this report are capable of handling packets on an interleaved basis on the downlink as well as contiguous units.

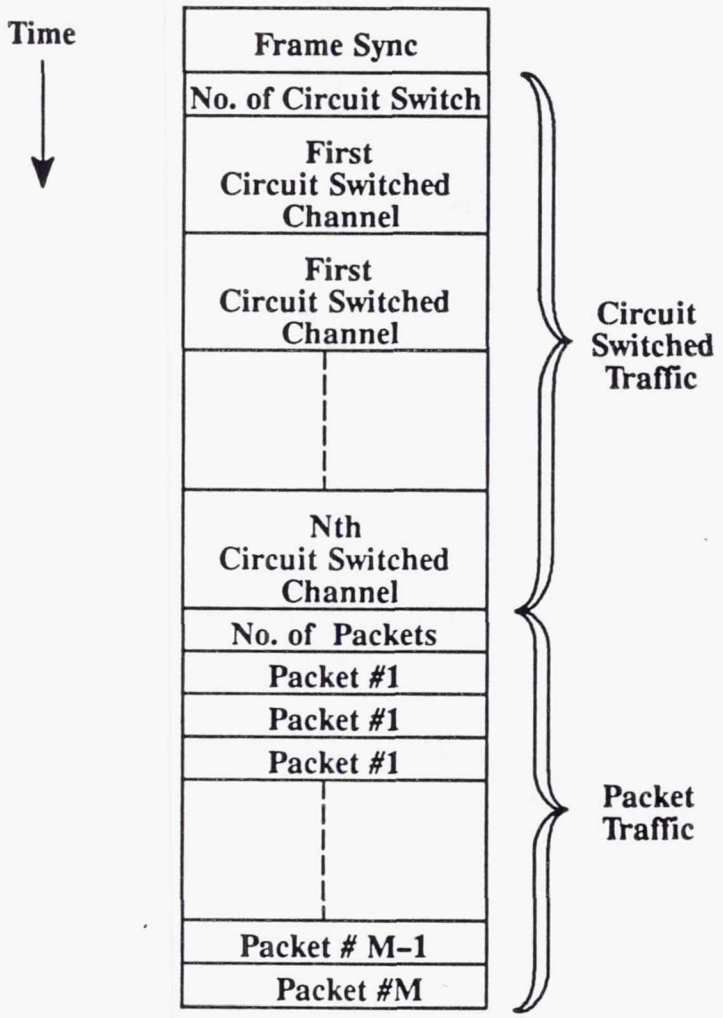

Figure 7 - Dwell Frame Structure

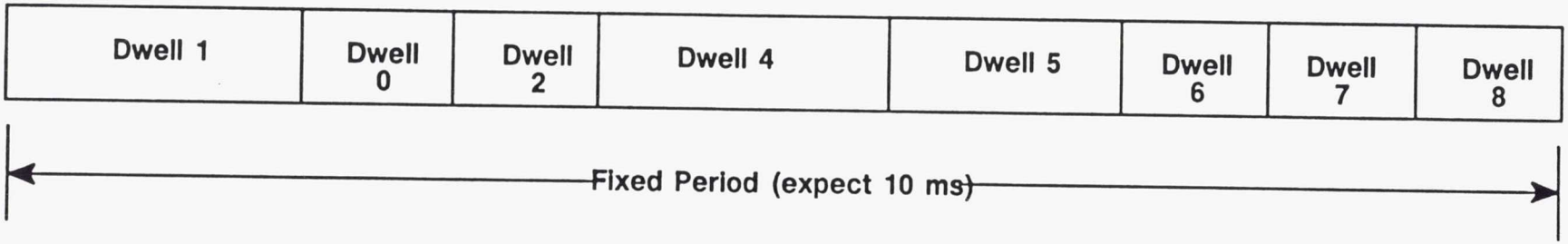

Figure 8 – Example Super Frame

7 Analysis

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

7.1 Packet Loss Probability vs. Buffer Size

Our analysis of probability loss makes use of the formulas of Hluchyj and Karol [3]. 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 [3] would assume that all dwells are served continuously at 18.75 Mbps rather than 1/8 of the time at 150Mbps.

It is easy to see that having to wait for a particular dwell to be served adds at most 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. Thus, to determine an upper bound for the buffer size required to achieve a given probability, we can compute the size according to [3], and then add D packets per output, i.e., $64D$ packets. In other words, we add enough memory for each downlink beam to hold a frame (8 dwells) of packets. However, we emphasize that in our shared memory design, all memory locations are shared among all dwells. This is in contrast to the design of [4], in which each location of the burst transmit buffer is dedicated to a particular dwell.

We consider three switch designs: output queuing (OQ) (i.e., shared bus), shared buffering per beam (SBPB), and completely shared buffering (CSB). The shared bus and completely shared buffering designs were presented earlier. SBPB is similar to CSB except that SBPB uses a separate shared memory for each downlink beam. In each of the designs there are 8 inputs and 64 outputs.

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.

Figure 9 gives the packet loss probability as a function of buffer size (in packets) for the three switch designs, assuming $p = .9$ and $D = 256$ (implying 2048 packets per frame). 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 2*8*1024 = 16384 if we assume that as many as two packets must be stored for each uplink channel. However, the designs presented earlier allow a much smaller buffer at the MCDD by requiring only a fraction of each packet to be stored there.

We define the multicast factor to be the average number of destinations to which each packet (including non-multicast 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 used is independent of the multicast factor, but assumes, for the CSM design, that every copy of each packet is stored in a separate location. For the case in which the CSM stores only one copy of each

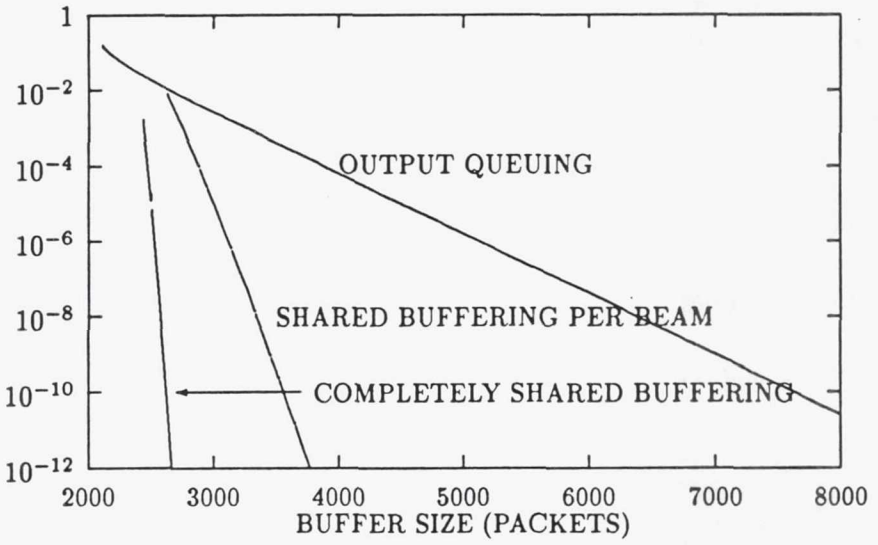

Figure 9: Packet loss probability vs. buffer size for three switch designs.

Load	Frame size	OQ	SBPB	CSB
.9	256	7040	3456	2596
.8	256	4544	2696	2287
.9	2048	21376	17796	16932
.8	2048	18880	17032	16623

Table 2: 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). This does not include storage at the MCDD.

Load	SBPB	CSB
.9	624	176
.8	312	81

Table 3: 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.

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 probability of packet loss 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.

The designs presented earlier allow the possibility 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 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 [3]. The OQ design is omitted because it is more difficult to analyze, but we expect it would also achieve a dramatic savings in memory with dynamic dwell control.

7.2 Computing the Optimal Packet and Dwell Lengths

It is clear that choosing a small dwell length, and thus a small frame length, results in a significant reduction in 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 a function 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 $D = 256$ and $D = 2048$. In general, the buffer requirement can be expressed as $B + 64D$, where B is independent of the dwell length. In this subsection, B will also include the size of the buffer at the MCDD (16384 packets).

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 effective data throughput, not including the packet header and the dwell switch times, is thus

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

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

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

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

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

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

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

which has the solution

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

Table 4 gives the optimal values for D and L, for M ranging from 1 to 10 Megabytes, assuming that $p = .9$, that B is chosen to achieve a packet loss probability of 10^{-9} using output queuing, that $H = 65$, and that $S = 1500$ (corresponding to a dwell switch time of 10 microseconds).

The table shows that, if the memory size is at least 24Mbits, then the optimal dwell size is nearly constant, and the optimal packet length increases nearly linearly with the memory size.

8 Conclusions

We have presented two feasible packet switch designs for the problem considered: the Shared Bus and the Shared Memory designs. For each, we have presented alternative methods for improving their performance, and have discussed the tradeoffs between these alternatives. We conclude that the Shared Memory design requires significantly less memory than the Shared Bus design, but requires only slightly less power because of its increased processing requirement.

Some of the ideas we have presented are only preliminary, and require future developement before they can be implemented. One of these ideas is the dynamic control of dwell lengths based on the instantaneous queue sizes, discussed in Sections 3.4 and 6.2. Another of these ideas is the packet division scheme presented in Section 3.3.

Memory size	Dwell length	Packet length	Data throughput
8Mb	75.469894	305.272736	0.665067
16Mb	81.676025	601.429932	0.778946
24Mb	83.724060	897.721802	0.818500
32Mb	84.744621	1194.045166	0.838576
40Mb	85.355904	1490.380859	0.850717
48Mb	85.762993	1786.722534	0.858851
56Mb	86.053558	2083.067627	0.864681
64Mb	86.271378	2379.415039	0.869064
72Mb	86.440720	2675.763672	0.872479
80Mb	86.576157	2972.113281	0.875215

Table 4: Optimal packet length (in bits) and dwell length (in packets) for Output Queuing for various limits on memory size, assuming that the dwell switch time is 10 microseconds and that 90% of the available downlink capacity (not including dwell switch time) is used.

References

[1] Leilani R. Tamura et al. A 4-ns bicmos translation-lookaside buffer. *IEEE Journal of Solid-State Circuits*, pages 1093–1101, October 1990.

[2] Masahide Takada et al. A 5-ns 1-mb ecl bicmos sram. *IEEE Journal of Solid-State Circuits*, pages 1057–1062, October 1990.

[3] M.G. Hluchyj and M.J. Karol. Queueing in high-performance packet switching. *IEEE Journal on Selected Areas in Communication*, pages 1587–1597, 1988.

[4] 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.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1995	Final Contractor Report

4. TITLE AND SUBTITLE

Alternative Packet Switch Architectures for a 30/20 GHz FDMA/TDMA Geostationary Communication Satellite Network

5. FUNDING NUMBERS

WU–506–72–21
C–NAS3–25934

6. AUTHOR(S)

Roy Stehle and Richard G. Ogier

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

8. PERFORMING ORGANIZATION REPORT NUMBER

E–9656

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR–195470

11. SUPPLEMENTARY NOTES

Project manager, Heechul Kim, Space Electronics Division, NASA Lewis Research Center, organization code 5600, (216) 433–8698.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 17

This publication is available from the NASA Center for Aerospace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

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 was to look 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 five years. Tradeoffs in memory requirement, power requirement, and architecture have been considered as a part of our study.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Information switching processor; Contention analysis	26
	16. PRICE CODE: A03

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