digital™

Networks Marketing

DECNIS Architecture

An overview of the DECNIS multiprotocol bridge router architecture

Document Number: NIPG-007-01 December, 1993

Abstract

The DECNIS500 and 600 are scalable, high performance multi-protocol bridge routers. A distributed processing and buffer management architecture with hardware support, was used to achieve a forwarding performance of over 80,000 pps for IP routing, IPX routing and bridging, and over 35,000 pps for OSI routing, whilst at the same time conforming to the router correctness criteria, necessary for the safe operation of a backbone network.

This paper discusses the correctness issues central to the design of all backbone bridge/routers. It presents the processing considerations that led to the design of the distributed forwarding system used in the DECNIS 500/600 products. The paper then identifies the key hardware and software design features, and finally provides a summary of the achieved forwarding performance. More details of the operation and performance of the DECNIS bridge/routers are available in other Networks Marketing documents.

The information in this document is based on a paper written by Dr. Stewart Bryant, a Senior Consulting Engineer in the Networks Interconnect Product Group of Networks Engineering, who was responsible for much of the design of the DECNIS product.

Introduction

The DEC Network Integration Server 500 and 600 (DECNIS 500/600) products are high performance communications servers integrating multi-protocol routing, bridging, and gateway functions.

The basic system is a master-slave design, with a management processor controlling the system and supporting the forwarding operations and intelligent network interface cards providing the packet filtering and forwarding functions. The Futurebus+ based backplane provides the basic control and data paths between the major system elements, including network interface cards for Ethernet, FDDI, 2Mb/s WAN, and high fan-out 128Kb/s WAN connectivity. The DECNIS can connect to T3/E3 links via the FDDI interface and a suitable DSU/CSU. Higher speed synchronous line cards are under development for the DECNIS and will be available in the near future.

A distributed forwarding system is used by the network interface cards for Internet Protocol (IP), Novell IPX, DECnet PhaseIV, and DECnet/OSI, with other protocols (currently AppleTalk) forwarded by the management processor. The management processor controls the system, including loading and dumping of the linecards, administering the routing and bridging databases, generating routing and bridging control traffic, and network management. X.25 functionality, both for routing data and as an X.25 gateway, is also provided by the management processor.

Performance measurements on a system configured with 14 Ethernets have demonstrated a forwarding performance of over 80,000 packets per second as a router or a bridge. The performance is linearly scalable with the number of lines.

Correctness Criteria

The purpose of routing protocols is to create a virtual network of optimum paths between the nodes on the network from the physical paths present. All common routing protocols are timer based, relying on regular status reports from neighbouring nodes to confirm the currency of the network topology. The virtual network exists in a state of dynamic equilibrium, and remains stable only so long as the routers are able to reliably exchange and process routing messages.

Should a router consistently fail to receive or process routing control traffic, (for example, because it is being presented with an excessive forwarding load that it cannot seperate from the control traffic) the network will enter a state of instability. This instability will generate more routing messages, and greater

forwarding load (as end systems re-generate data lost in the miss-converged network). This in turn will act as positive feedback on the incorrectly designed bridge/routers, causing the instability to remain. In a backbone network a malfunctioning node overloading an under-performing router can cause a global communications breakdown.

Any bridge-router runs three major processes that compete with each other: the forwarding process, the routing process (topology management) and network management. As both processor speed and the sophistication of network applications rises, forwarding throughput and latency criteria become critical to application performance. In particular packet loss is only recovered by timer expiry in the common transport protocols, and consequently it is loss free throughput performance, not peak packet performance that determines the processing rate of the applications. Due to the inherent frailty of the network demonstrated above, the designer cannot simply hold the routing process hostage to the instantaneous forwarding requirements. Provision must therefore be made for the routing processes to execute to completion in a timely manner. Finally in times of crisis, monitoring and control of the network by the network manager may be the only means of recovering the situation. This requires provision for the safe transmission, reception and execution of network management messages.

We can therefore set down a set of universal bridge-router correctness criteria that are necessary conditions for the safe operation of a backbone network. Even under conditions of gross overload a bridge/router must:

1. Extract and process all routing control and spanning tree messages (at the expense of data messages if necessary).

2. Always have the resources to transmit, receive and process routing control and spanning-tree messages.

3. Ensure that topological convergence occurs in a timely fashion.

4. Ensure responsiveness to network management commands.

5. Maintain the integrity of the forwarded data, particularly for bridged traffic.

6. Provide fairness between competing network protocols allowing each to make continued and adequate progress.

Designing to these criteria requires a sophisticated hardware and/or software design capable of striking the correct balance between the demands imposed by these constraints.

To address the requirements of a high-performance multi-protocol bridge/router with the technology currently available, we split the functional requirements into two sets: those best handled in a distributed fashion and those best handled centrally.

The data-link and forwarding functions represent the highest processing load and operate in sufficiently local context that they can be distributed to a processor associated with a line or a group of lines. The processing requirements associated with these functions scale linearly with both line speed and number of lines attached to the system. Some aspects of these per-line functions, such as link initialization and processing of exception packets, require information that is available only centrally or need a sophisticated processing environment. However, these may be decoupled from the critical processing path and moved to the central processing function.

In contrast to the lower-level functions, the management of the system and the calculation of the forwarding database is best handled as a centralized function, since these processes operate in the context of the bridge/router as a whole. The processor workload is proportional to the size of the network and not the speed of the links. Network protocols are designed to reduce the amount of this type of processing, both to minimize control traffic bandwidth and to permit the construction of relatively simple low-performance routers in some parts of the network. These processing considerations led us to design the DECNIS 500/600 as a master-slave system, with central management processor (MPC) and a set of per-line forwarding processors.

The per-line processors communicate on a peer-to-peer basis, forwarding the normal packets that comprise the majority of the network traffic. The MPC behaves, in essence, like a normal mono-processing bridge/router, however its involvement in forwarding is limited to unusual types of packet. Having split the functionality between the peer-to-peer forwarding processors and the management processor, we designed a buffer and control system to efficiently couple these processors together.

The DECNIS 500/600 products use a central packet forwarding buffer pool (PRAM) of 256-byte buffers, shared among the linecards. Ownership of buffers is passed from one linecard to another by a swap which exchanges a full buffer for an empty one. This algorithm improved both the fairness of buffer allocation

and the performance of the buffer ownership transfer mechanism. Fractional buffers much smaller than the maximum packet sizes were used, even though this makes the system more complicated. The consequential economy of memory, however, made this an attractive proposition.

Analysis of the forwarding function indicated that to achieve the levels of performance we required, we would need hardware assistance in parsing and looking up network addresses. Considerations of economy of hardware cost, board area, and bus bandwidth led us to design an single address recognition engine (ARE) shared among all linecards. This address parser was fabricated as an ASIC. It will parse in excess of 240,000 MAC destination address, source address and protocol identifier triplets per second. Alternatively it will parse in excess of 80,000 OSI routing addresses per second.

Of the commonly used network address formats, the OSI NSAPs defined by ISO 8348/AD2 have the most complex parsing rules. These addresses contains several components, with a complex padding structure and abbreviation rules. The ARE therefore has sufficient performance to support a DECNIS 600 server fully populated with linecards that support each link with a bandwidth of up to 2 X 10 megabits per second. Above this speed, for example with a FDDI interface, a local address cache, refreshed from this shared ARE is required.

Distributed Forwarding

In understanding the distributed forwarding process used on the DECNIS 500/600, it is convenient to first consider the forwarding of routing packets, and then to extend this description to the processing of other packet types.

In the routing forwarding process, as shown in above, the incoming packets are made up of three components: the data-link header, the routing header, and

the packet body. The receive process (RXP) terminates the data-link layer, stripping the data-link header from the packet. The routing header is parsed and copied into PRAM unmodified. Any required changes are made when the packet is subsequently transmitted. The information needed for this is placed in a data structure called a packet descriptor, which is written into space left at the front of the first packet buffer. The packet body is copied into the packet buffer, continuing in other packet buffers if required.

The destination network address is copied to the ARE, which is also given instructions on which address type needs to be parsed. The RXP is now free to start processing another incoming packet. When the address lookup process has completed, the RXP is able to read from the ARE the forwarding parameters needed to complete the processing of the packet. These parameters contain information about the output port and channel to use, the destination data-link address for the next hop, and any translation information. The RXP combines this information with some information saved from parsing the packet to build the packet descriptor in PRAM.

The RXP builds a set of ring vectors for the packet, one for each buffer used. Each ring vector contains a pointer to the PRAM buffer used, plus some additional information used to decide on which queue the buffer should be stored and to determine its relative importance to the system. During congestion, this information is used by the linecards to discard the least important packets first.

These ring vectors are then exchanged with the transmit process (TXP) on the output linecard, which queues them ready for transmission. Before the TXP starts to process a packet for transmission, it reads the descriptor from the first PRAM buffer. From the information in the descriptor, the TXP is able to build the data-link header, determine the routing header translation requirements, and locate a number of fields in the header (such as the OSI segmentation and quality of service fields) without having to reparse the header. The TXP builds the data-link header, reads the routing header from PRAM, makes the appropriate modifications, and then completes the packet by reading the packet body from PRAM.

Since the transmit packet construction follows the packet transmission order byte for byte, implementations can be built without further intermediate transmission buffering.

The linecard processors were closely coupled to the line I/O controllers, to minimize the system latency, and hence maximize network performance. In this

design these processors only need sufficient transmit and receive buffering to cover the local latency requirements. In one instance, a linecard has significantly less than a full packet buffer. This small buffering requirement makes available a number of different implementation styles.

If the RXP discovers a faulty packet, a packet with an option that requires system context to process, or a packet that is addressed to this system (including certain multicast packets), it queues that packet to the management processor in exactly the same way that it would have queued a packet for transmission by a TXP. The management processor (MPC) contains a full-function mono-processor router that is able to handle these exception cases. The MPC sends packets by swapping them to the appropriate linecard TXP in exactly the same format as a linecard receivers uses.

The bridge forwarding process operates in a fashion similar to the routing forwarding process, except that the data-link header is preserved from input port to output port, and only the data-link header is parsed.

To comply with the correctness criteria, it is necessary to ensure that the receiver has sufficient processing power to examine all received packets to select the routing packets from the data-packets under conditions of system overload. In the case of the high fan-out, 128Kb/s WAN card, we can caluculate that the overload condition cannot occur. On the Ethernet linecards, we ensure correct operation by providing a software mechanism that is able to detect the onset of the congestion state, and change from a packet forwarding mode to a screening mode for the duration of the crisis.

As no processor capable of screening at the peak packet rate of 450,000 packets per second necessary for correct opeartion on FDDI LANs was commercially available, we therefore use a patented hardware mechanism that is able to recognise the various packet types based on their MAC PID, and to sort them into separate buffer streams prior to passing them to the FDDI forwarding processor in priority order.

In the line transmitters we provide a reserved buffer pool for the routing and network management packets, and give these transmission priority. This ensures that the control traffic is provided with the necessary resources to be transmitted, that it is transmitted in a timely fashion. This is essential to comply with the correctness criteria.

Distributed Buffer System

A schematic description of the DECNIS 500/600 buffer system is shown in the diagram below.

The RXPs have only sufficient buffering to cope with the latencies that must be sustained in their various stages of packet processing. All long-term storage of packets takes place while the packet is owned by the TXP. When an RXP has finished processing a packet, it swaps the PRAM buffers containing the packet for the same number of empty buffers owned by the TXP that is to transmit the packet. Only if the TXP is able to replace these buffers with empty buffers does the transfer of ownership take place. If the swap cannot complete due to lack of free buffers, the RXP re-uses these buffers for another packet. In this way, no transmitter is able to accumulate buffers and thereby prevent a receiver from receiving packets intended for other output ports.

The design of an efficient buffer transfer scheme is an important aspect of a high-performance multi-processor router. We solved this problem by using a set of single writer/single reader rings, with one ring associated with each pair-wise exchange of buffers that can take place in the system. Thus each TXP has associated with it one ring for each of the RXPs in the system (including its own), plus one for the management processor.

When an RXP has a buffer to swap, it reads the next transfer location in its ring corresponding to the destination TXP. If it finds a free buffer, it replaces it with the one to be sent, keeping the old one as a replacement. The transferred

information consists of a pointer to the buffer, its ownership status, and some information to indicate the type of information in the buffer. This structure is known as a ring vector. A single-bit semaphore is used to indicate transfer of ownership of a ring vector.

The buffer transfer scheme schematic shown above in illustrates how this works. Each transmit port (TXa or TXb) has a ring dedicated to each of the receivers in the system (RXa and RXb). RXa swaps ring vectors to the "a" rings on TXa and TXb, and RXb swaps ring vectors to the "b" rings on TXa and TXb. During buffer transfer, the TXP runs a scavenge process, scanning all its rings for new buffers, queuing these buffers in the transmit queues (TXQ) specified by the ring vector, and replacing the entries in the ring from the local free list. The buffer type information enables the transmit linecard to quickly determine the importance of the buffer. Thus if the linecard runs short of buffers due to congestion, it is able to discard less important packets in preference to those packets required to preserve the stability of the network.

Through judicious optimization of the ring vector encodings, we were able to condense this ring swap transaction into a single longword read followed by a single longword write for each buffer swap, for all unicast traffic. For multicast traffic, a second longword was required. To reduce the amount of bus traffic and the processor time associated with the scavenge process, the random-access memory (RAM) that holds the rings is physically located on the transmit linecard. Hardware is used to watch the rings for activity and report this to the TXP.

Analysis of the traffic patterns indicated that considerable economies in PRAM could be made if we fragmented long packets over a number of buffers. We achieved a satisfactory compromise between the processing overhead associated with buffer management and memory efficiency through the use of 256-byte buffers. With this buffer size, a large fraction of the packets are contained within a single buffer. When a linecard is driven into output congestion, it is no longer certain that a complete set of packet buffers will be swapped. We therefore had to introduce a simple protocol to ensure that a packet was queued for transmission only if it had been fully transferred to the transmitting linecard.

To cope with dissimilar swap and scavenge process speeds, we had to stage the transfer of buffers. Thus, the TXPs collect a complete set of buffers from an RXP before queuing the packet for transmission; this process is called "binning". In this way, a partial transfer due to congestion or a slow receiver does not block the progress of other ports in the system.

Bridging needs a mechanism to support the distribution of flooded and multicast packets to multiple output ports. In some distributed systems, this function is handled by replicating the packet via a copy process. In other systems, the packet is handled by a central multicast service. The use of a central multicaster gives rise to synchronization issues when a destination address moves from the unknown to the learned state. Replication by the linecards is not practical in this architecture since the linecards do not hold a local copy of the buffer after it has been copied to PRAM. We therefore use a system in which multicast buffers are loaned to all the transmit linecards. A "scoreboard" of outstanding loans is used to record the state of each multicast buffer. When a buffer is returned from all its borrowers, it is added to the multicast free queue and made available for re-use. The loan process and the return process are similar to the normal swap and scavenge process, but the ring vector is extended slightly to include the information needed for rapid de-referencing.

Centralized Resources

Three central resources are used in the DECNIS 500/600 products: MPC, PRAM, and ARE. Centralizing these resources reduced both the cost and the complexity of the system. There are two ways of building a distributed processing router. In one method, the router consists of a federation of full-function routers, each a separate network node. An alternative method is to employ a partially centralized design in which only one processor is the router in the traditional sense.

The central processor is the focus for network management, calculating the forwarding table and being a central repository for the context of the router, and the peripheral processors undertake the majority of the forwarding work. An analysis of the cost and complexity both from a system and a network perspective led us to choose the latter approach.

Thus the MPC provides all the software functionality necessary to bind the collection of forwarding agents located on the linecards together to form a router. To the rest of the network, the system appears indistinguishable from a traditionally designed router. The processing capability and memory requirements of the MPC are those associated with a typical medium-performance multi-protocol bridge/router.

Management Processor Software

The DECNIS 500/600 MPC software structure is as follows:

It consists of a full-function bridge/router and X.25 gateway, together with the software necessary to adapt it to the DECNIS 500/600 environment. The control code module, which includes the routing, bridging, network management, and X.25 modules is an extended version of Digital's WANrouter 500 software. These extensions were necessary to provide configuration information and forwarding table updates to the DECNIS 500/600 environment module. This module hides the distributed forwarding functionality from the control module. In this way, the control module is provided with an identical environment on both the MicroServer and DECNIS 500/600 platforms.

The major component of the DECNIS 500/600 environment module contains the data-link initialization code, the code to control the linecards, and the code to transform the forwarding table updates into the data structures used by the ARE. A second component of the environment module contains the swap and scavenge functions necessary to communicate with the linecards. Because of the real-time constraints associated with swap and scavenge, this function is split between the management processor on the MPC and an assist processor.

Because the control code module was designed as a full-function router, we are able to introduce new functionality to the platform in stages. If a new protocol type is to be included, it can be initially executed in the management processor with the linecards providing a framing or data-link service. At a later point, the

forwarding components can be moved to the linecards to provide enhanced performance.

Packet Buffering and ARE

We had a choice of three locations for the PRAM: distributed among the receiving linecards, distributed among the transmitting linecards, or shared centrally between these linecards. Locating the buffering at the receiver would have meant providing the maximum required transmitter buffering for each transmitter at every receiver. Locating the long-term packet buffering at the transmitters would have meant staging the processing of the packets by storing them at the receiver until the transmit port was determined and then transferring them to the appropriate transmitting linecard. This would have increased the system latency, the receiver complexity, and its workload. An analysis of the bus traffic indicated that for a router of this class there would be adequate bus bandwidth to support the use of a centrally located, single shared packet buffer memory. With this approach, however, every packet crosses the bus twice, rather than once as in the other approaches. Nevertheless, we chose to base the system around a single packet memory, and win the consequential economies in both linecard cost and board area.

An analysis of the processing power needed to parse and look up a network address led us to conclude that the linecards would need some form of assistance if the processing power associated with each line was to be constrained to a reasonably cost-effective level. This assistance is provided by the ARE.

Some advanced development work on the design of hardware search engines showed that it was possible to design a single address parser powerful enough to be shared among all the linecards. This search engine was adaptable enough to parse the complex structure of an OSI NSAP, with its two right-justified padded fields and its longest match semantics. In addition, the engine was able to cope with the other routing protocol address formats and the learning requirements of bridging. By centralizing the forwarding database, we also avoided the processing and bus overhead associated with maintaining several distributed forwarding databases and reduced the cost and board area requirements of the linecards.

The bus bandwidth and lookup rate needed to support multiple fiber distributed data interface (FDDI) linecards would place an excessive burden on the system. For FDDI, therefore, we equip the central lookup engine with a linecard-resident address cache.

Performance

Performance measurements have been made on the DECNIS 500/600 products for IP, IPX, DECnet Phase IV, DECnet/OSI, and bridged traffic. For a detailed description of the measurement methodology, and a comparison between the performance of the DECNIS 500/600 and the other competing bridge/routers the reader should refer to the independent test results published by Scott Bradner (7). A summary of these results is available in the "*DECNIS Performance*", Networks Marketing Document Number NIPG-001-02.

The performance measured by both Scott Bradner and are shown below. Table 1 shows the forwarding throughput for 6 Ethernet to 6 Ethernet for minimum sized packets.

These measurements show the maximum forwarding performance with no packet loss. The use of a no-loss figure for comparison between different designs is important, because this represents the maximum throughput usable by a network application. If the applications attempt to run at more than the loss free rate, the packet loss will cause the transport protocols to back off to the loss free operating point.

The Ethernet to Ethernet figures indicate the near linear scalability of performance with number of lines. Ethernet forwarding performance of this magnitude are well in excess of those required to operate on any practical LAN. The correctness software will ensure the reception of any routing packets for a significant period even after these rates are exceeded.

Measurements taken also indicated that the unidirectional and bi-directional forwarding performance were substantially the same, which is not the case for all router designs. This is of more than academic significance. Poorly designed

Ethernet sub-systems, that do not provide adequate transmit processing power under conditions of receive overload suffer from a condition known as "live-lock". In this condition the receiver uses up all the processing cycles, preventing the transmitter from attempting the transmission which would force a collision on the Ethernet, thereby restoring fair operation.

The FDDI forwarding performance is shown in the following diagram. These are also taken at the zero-loss operating point, and indicate industry leading performance results.

References

1. IEEE Standard 896.1-1991 Futurebus+ Logical Layer Specification.

2. IEEE Standard 896.2-1991 Futurebus+ Physical Layer and Profile Specifications.

3. A Digitallink DL3000 DSU/CSU.

4. S Bryant and D Brash, "The DECNIS 500/600 Multi-protocol Bridge Router and Gateway," To be published in Digital Technical Journal.

5. G. Cobb and E. Gerberg, "Digital's Advantage Networks Routing Software Design," To be published in Digital Technical Journal

6. E. Fredkin, "TRIE Memory," Communications of the ACM, vol. 3 (1960): 490-499.

7. Knuth, "Sorting and Searching -- the Art of Computer Programming," vol. 3

8. S Bradner, "Testing the Devices" Proceedings Fall Interop 1993. Available on the Internet by anonymous FTP from hsdndev.harvard.edu in /pub/ndtl

Digital believes the information in this publication is accurate as of its publication date. Such information is subject to change without notice. Digital is not responsible for any errors.

The following are trademarks of Digital Equipment Corporation: Digital, DECnet, and DECNIS.

IPX is a trademark of Novell Corp.

AppleTalk is a trademark of Apple Corp.