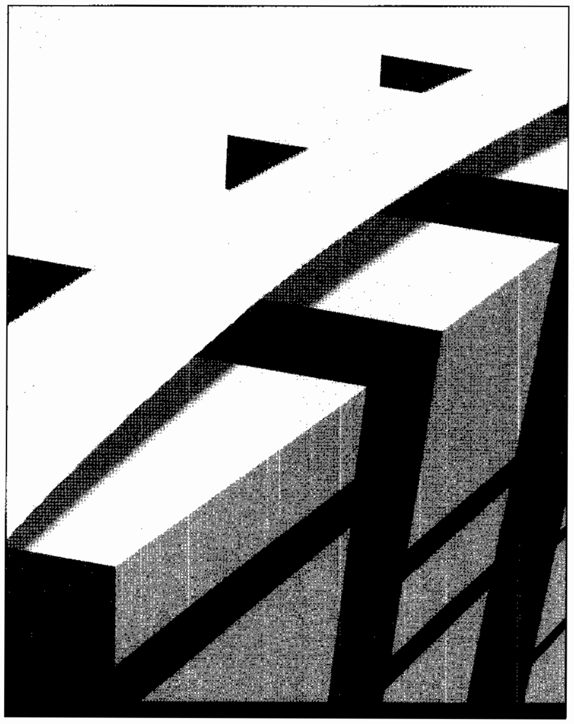
TANDEM SYSTEMS REVIEW



Himalaya IPB

Client/Server Availability

Call Applications Manager

Product Update

T A N D E M

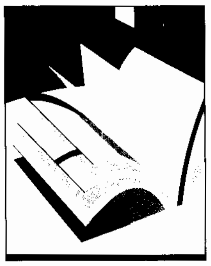
SYSTEMS REVIEW

VOLUME 10, NUMBER 2

APRIL 1994

The Tandem Systems Review *publishes technical information about Tandem software releases and products. Its purpose is to help programmers, analysts, and other IS professionals to plan for, install, use, and tune Tandem systems.*

TANDEM



Editor's Note

The first two articles in this issue of *Tandem Systems Review* reflect topics of high priority for business applications: high performance, fault tolerance, and high availability. "The NonStop Himalaya K10000 Interprocessor Bus" describes enhancements made to the high-speed interprocessor bus subsystem of the newest and most powerful Tandem system, the NonStop Himalaya K10000 server. The enhancements support the K10000 RISC processor's increased performance capabilities. The latter part of the article presents a detailed technical discussion of the K10000 IPB architecture.

Next, "Client/Server Availability" is a study of availability in today's complex client/server environment. This two-part article begins by presenting a predictive model for evaluating client/server availability and applies it to a representative client/server environment. The second part of the article discusses a number of ways to increase client/server availability.

The third article, "Automating Call Centers With CAM," describes the Tandem Call Applications Manager, software that links applications on Tandem NonStop systems with telephone switches to provide computer-telephone integration. The article discusses basic functions and benefits of CAM, how CAM works in an automated call center, and architectural considerations for developers who are building call center applications with CAM.

Finally, we encourage you to fill out and return the Reader Survey found at the end of this issue. We are very interested in responding to your technical information needs.

–AL

EDITOR
Anne Lewis

ASSOCIATE EDITORS
David Gordon, Steven Kahn, Mark Peters

PRODUCTION MANAGER
Anne Lewis

ILLUSTRATION AND LAYOUT
Donna Caldwell

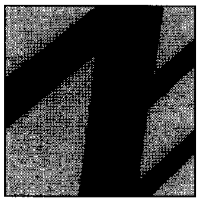
COVER ART: Brian Jeung, Steve Sanchez

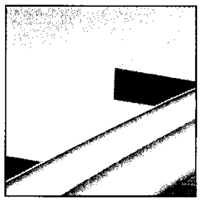
SUBSCRIPTIONS: Elaine Vaza-Kaczynski

ADVISORY BOARD
Mark Anderton, Richard Carr, Jim Collins, Moore Ewing, Terrye Kocher, Randy Mattran, Mike Noonan

Tandem Systems Review is published quarterly by Tandem Computers Incorporated. All correspondence and subscriptions should be addressed to *Tandem Systems Review*, 10400 Ridgeview Court, Loc 208-65, Cupertino, CA 95014.

Subscriptions: $75.00 per year; single copies are $20.00. Detailed subscription information is provided on the subscription order form at the end of this book.





Product Update

Integrity Systems

Integrity FT CM-1475 and CO-1475 Systems

February 1994

The new CM-1475 and CO-1475 systems, based on the MIPS® R4400™, 150MHz processor, are high-end Integrity FT systems for fault-tolerant computing. These systems provide more than 30 percent improvement in price/performance over comparable configurations of the older CM-1450 and CO-1450 systems.

The base configurations of both 1475 models include a MIPS R4400, 150MHz processor, 64 megabytes of local memory, and 16 megabytes of global memory. Maximum local memory on the 1475 systems is 128 megabytes and maximum total memory (local plus global) is 192 megabytes. These base configurations do not include disks, tapes, or SCSI device controllers. The new packaging scheme allows users flexibility in choosing media and system options.

The CO-1475 model incorporates specialized features to support telecommunications central office applications. These features include compliance with the stringent safety, fire resistance, earthquake resistance, temperature, power, and grounding standards required in telco central offices.

The CM-1475 and CO-1475 models use the same system cabinets and mass storage cabinets as previous CM and CO systems. The system cabinet can store up to 9 disk or tape devices. In addition, these systems can support two mass storage cabinets with up to 21 devices per cabinet.

Integrity FT CM-1455 and CO-1455 Systems

February 1994

The new CM-1455 and CO-1455 systems, based on the MIPS R4000®, 75MHz processor, are midrange Integrity FT systems for fault-tolerant computing. These systems provide improved price/performance over comparable configurations of the older CM-1450 and CO-1450 systems.

The base configurations of both 1455 models include a MIPS R4000, 75MHz processor, 64 megabytes of local memory, and 16 megabytes of global memory. Maximum local memory on the 1455 systems is 128 megabytes and maximum total memory (local plus global) is 192 megabytes. These base configurations do not include disks, tapes, or SCSI device controllers. The new packaging scheme allows users flexibility in choosing media and system options.

The CO-1455 model incorporates specialized features to support telecommunications central office applications. These features include compliance with the stringent safety, fire resistance, earthquake resistance, temperature, power, and grounding standards required in telco central offices.

The CM-1455 and CO-1455 models use the same system cabinets and mass storage cabinets as previous CM and CO systems. The system cabinet can store up to 9 disk or tape devices. In addition, these systems can support two mass storage cabinets with up to 21 devices per cabinet.

The Product Update department provides brief descriptions of new products announced by Tandem. For more information on any of these products, please consult your local Tandem representative.

Integrity FT CM-1300E System

February 1994

The new CM-1300E system, based on the MIPS R3000®, 25MHz processor, is an entry-level Integrity FT system. It offers users the benefits of the Integrity FT architecture and the flexibility for a cost-effective future upgrade to a CM-1455 or CM-1475 system.

The base configuration of the CM-1300E system includes a MIPS R3000, 25MHz processor, 16 or 32 megabytes of local memory, and 16 or 32 megabytes of global memory. Maximum total memory (local plus global) is 64 megabytes. The base configuration does not include disks, tapes, or SCSI device controllers. This packaging scheme allows users flexibility in choosing media and system options.

Integrity NR/4401 Server

January 1994

The Integrity NR/4401 is a new low-end network resource server that offers a cost-effective upgrade path from the existing NR/4001 server. The NR/4401 provides a 28-percent price/performance improvement and a 56-percent absolute performance improvement over the NR/4001.

The new server uses a MIPS R4400 RISC processor running at 150MHz. It has a memory capacity of 384 megabytes and an internal disk capacity of 3 gigabytes. Other standard features are the same as for the NR/4001 server.

Client/Server Computing Products

Tandem Workflow Image System 3.1

November 1993

Tandem Workflow Image System 3.1 is an enhanced version of the previously announced Tandem Workflow Management product. It incorporates all of the basic features of the earlier version and includes a number of enhancements such as ViewStar version 3.1 software, an expanded compatible-hardware base, and the availability of related Tandem Education courses and Professional Services.

Version 3.1 of ViewStar software offers client/server tools and capabilities that improve the reliability and performance of complex, distributed, high-volume business applications. It also makes the development and deployment of workflow applications faster and easier. The compatible-hardware base of Tandem Workflow Image System 3.1 has been extended to include a number of powerful database servers, such as the Tandem NonStop Himalaya servers and Tandem Integrity NR servers, as well as additional workstations and peripheral devices.

The Tandem training courses and Professional Services related to this product are aimed at helping users plan, design, and implement appropriate, cost-effective document management and workflow automation systems using the ViewStar software and Tandem products.

Windows NT

January 1994

Tandem now offers Windows NT on its PSX and NDX platforms. Windows NT is a full 32-bit, preemptive multitasking operating system. Its fast, 32-bit drivers directly manipulate disk hardware, which results in better system performance and more consistent throughput. This performance gain is especially apparent in client/server or other environments where heavy client processing loads are the norm. Windows NT also protects users' current software investments by providing backward compatibility for applications running on MS-DOS and Windows operating systems.

Windows NT Advanced Server

January 1994

Windows NT Advanced Server is a networking version of Windows NT for users implementing client/server applications such as database servers, messaging servers, or communications gateways on different networks. This networking version builds on the Windows NT operating system, adding centralized network management and security functions, as well as connectivity to remote clients. The Windows NT Advanced Server is a complete file and print server, and excels as a platform for building the server portion of client/server applications.

Communications and Networking Products

NonStop Access for Networking

December 1993

NonStop Access for Networking is a collection of network components that provides the automatic switching of Ethernet paths from a primary LAN to a secondary LAN in 10Base-T networks. The components are a 10Base-T version of the 3615 Ethernet controller and a Gemini dual-path network interface card (NIC).

To eliminate a single point of failure for the LAN attachment to Tandem servers, a minimum of two 3615 controllers are required. Tandem LAN Access Method (TLAM) software manages and controls both 3615 controllers. The Gemini NIC provides dual paths from each LAN-attached PC to the dual networking hubs. The Gemini NDIS software driver senses both paths and automatically switches from the active path to the backup path in the event of a failure.

These components work in conjunction with existing Ungermann-Bass 10Base-T local area networking products. To take advantage of NonStop Access for Networking, two Ungermann-Bass networking hubs are redundantly internetworked to provide primary and backup 10Base-T Ethernet paths to the 3615 controllers. Standard Ungermann-Bass dual Ethernet backbone LANs provide backup paths between the hubs.

Access/Stax Networking Hub

October 1993

Access/Stax Networking Hub is a stackable, segmentable networking hub that provides for a port density greater than that of any other stackable solution on the market today. Up to five Access/Stax hubs can be stacked to accommodate departmental growth to a maximum of 120 ports. In addition, the new 10Base-T hub supports up to 3 10Mbps Ethernet segments in a single 24-port hub, or up to 15 segments in a stack of five hubs. These hubs can be either unmanaged or managed through consoles supporting Simple Network Management Protocol (SNMP) or Hub Management Interface (HMI).

New Token-Ring Adapter

October 1993

The new token-ring adapter now available from Tandem provides full 4Mbps or 16Mbps support for IBM and non-IBM token-ring networks. The new adapter is fully software configurable for ease of use and trouble-free installation. It features a complete software diagnostic package and is designed to automatically select either unshielded twisted pair (UTP) or shielded twisted pair (STP) wiring to provide full flexibility in network configuration.

Workstation and Terminal Products

Indigo^2 XL Workstation

January 1994

The Indigo^2™ XL is a high-end workstation with new desktop packaging, faster and more fully-featured graphics, and a 30-percent price/performance improvement over the Indigo® R4000 workstations. Two Indigo^2 XL workstations are available: one based on the MIPS R4000 RISC processor running at 100MHz and one based on the R4400 RISC processor running at 150MHz. Both workstations feature 24-bit color and high resolution 19-inch monitors and offer a graphics level that is optimized for advanced program development, commercial client/server applications using the X-Windows system, and the display of networks, traffic patterns, or other scenarios that require monitoring.

PSX SP P/60 Desktop Computer

December 1993

The new PSX SP P/60 desktop computer, housed in a low-profile chassis, combines a Pentium processor with 16 kilobytes integrated memory cache and 256 kilobytes of second-level cache to provide a great increase in processing speed. The PSX SP P/60 uses an integrated PCI (Peripheral Component Interconnect) bus, the latest standard in local-bus technology. The computer's two PCI slots run peripheral devices at accelerated speeds. In addition, with the video subsystem on the PCI local bus, there's a direct express route between CPU and graphics, resulting in faster video performance.

The PSX SP P/60 provides five expansion slots (two PCI local bus, three ISA), and four drive bays (three external, one internal). The two PCI slots are backward-compatible with ISA, making it possible for users to continue to take advantage of the wide variety of existing ISA-compatible products.

Enhanced PSX EP-Series Desktop Computers

October 1993

Tandem now offers the next generation of PSX EP-series desktop computers. The new computers provide all of the standard EP-series features such as memory expansion to 64 megabytes, 512-kilobyte graphics memory (expandable to 1 megabyte), and four 16-bit ISA bus expansion slots. In addition, the new EP-series offers these enhancements: accelerated 16-bit local bus graphics, three external drive bays, and upgradability to an Intel Pentium OverDrive processor.

PSX LP-Series Desktop Computers

October 1993

The new PSX LP-series computers offer high performance and cost effectiveness as well as a number of sophisticated security features. These computers are housed in a low-profile desktop chassis and are Tandem's first "green PC" offering, exceeding the guidelines established by the EPA's Energy Star power management program.

The PSX LP-series computers feature 32-bit local bus graphics, 512 kilobytes of standard graphics memory, and Pentium OverDrive and cache upgradability. They also provide such security features as protected FlashBIOS, asset management, write protection, dual passwords, and AST Walk-n-Lock.

NDX ST P/60 Workstation

October 1993

The NDX ST P/60 workstation, based on the AST CUPID architecture, is a new computer designed to provide the power needed for advanced networking. The ST P/60 uses the same tower chassis as the current NDX ST systems and can be used with the existing NDX ST memory and peripherals. With the 64-bit data paths of its Pentium 60MHz processor, the ST P/60 eliminates the memory bottleneck typical of 32-bit systems.

The ST P/60 workstation incorporates eight EISA expansion slots, five drive bays (three external and two internal), up to 128 megabytes of RAM (expandable on the system board), and 256 kilobytes of second-level cache memory (standard). The tower chassis provides slide-out, cable-free SCSI backplanes that make it easy to add enough hard-drive storage for even the largest network. To protect system components and data, the ST P/60 offers a number of security features such as AST Walk-n-Lock, protected FlashBIOS, write protection, and asset management.

New NDX ST Servers

December 1993

Tandem now offers two new models of NDX ST servers. Driven by Intel 486DX or 486DX2 processors, the new high-performance NDX file servers excel where power and reliability are critical. Whereas most present systems provide a Pentium upgrade path only through a P24T socket for a future overdrive processor, the new NDX ST servers support both the future overdrive chip and a real Pentium processor card with up to 512 kilobytes of second-level cache. The Pentium card slot on the system board can also accept a cache upgrade.

Both new NDX models incorporate 11 drive bays and eight EISA slots. The 8 slide-out, cable-free SCSI drives make it easy to add extra hard drives quickly, or to perform a quick hot-swap with little or no downtime should a hard drive fail. BIOS upgrades are made simple with AST FlashBIOS. The new servers also feature up to 128 megabytes of RAM, second-level cache memory upgradable to 512 kilobytes (256 kilobytes is standard on the 486DX2 model), and local-bus graphics with 1 megabyte of video memory (expandable to 2 megabytes). Hardware and software security is provided through a chassis lock, floppy and I/O write protection, asset management, dual-level passwords, and AST Walk-n-Lock.

MS-DOS 6.2

January 1994

All new shipments of Tandem personal computers now include MS-DOS 6.2. This latest release of MS-DOS makes it safer and easier to double one's disk space through the DoubleSpace compression feature of MS-DOS 6.0. Additional performance enhancements include faster CD-ROM access through an improved SmartDrive and new data-protection technology.

Programming Languages

C++ Programming Language

December 1993

Tandem's C++ programming language environment is available for systems using the Tandem NonStop Kernel operating system, version D20 or later. The Tandem C++ development environment has four components: C++ Translator based upon USL Cfront; ISO C compiler; Inspect debugger; and Tools.h++ foundation class library by Rogue Wave Software.

Cfront is the de facto reference implementation of C++. The C++ Translator converts C++ source code to C source code, which is then compiled by the C compiler. The C compiler is a prerequisite for C++ programming for Tandem NonStop systems. Tandem's Inspect symbolic debugger features direct debugging of C++ source code. Inspect is supplied on all Tandem NonStop systems. Tools.h++ is an industry-standard C++ data structure library that provides fundamental structures for C++ programming.

The NonStop Himalaya K10000 Interprocessor Bus

The Tandem™ NonStop™ Himalaya™ K10000 server delivers high performance, data integrity, and fault tolerance for large, business-critical applications. Tandem's newest and most powerful computer system, the K10000 server is designed to support a mixture of online transaction processing (OLTP), decision support, and batch applications.

The K10000 servers have much in common with the Tandem NonStop systems that preceded them. However, to achieve the highest levels of performance, K10000 servers differ from previous NonStop systems in several significant ways (Kong, 1994). In addition to enhancing the processor itself, designers made several significant changes affecting the interprocessor buses (IPBs), the high-speed buses used by the multiple processors in a NonStop system to communicate with one another. Specifically, designers improved the performance of the IPBs to support the increased capabilities of the K10000 processor.

This article introduces the changes made in the Himalaya K10000 IPB subsystem. It begins by describing the purpose and functions of these enhancements. It then briefly discusses K10000 performance results relative to previous NonStop systems, which indicate that the IPB enhancements achieved their performance goals. The remainder of the article describes the architecture of the K10000 IPB, focusing on the recent changes; these sections are intended for readers who want a more complete technical understanding of the IPB.

NonStop System Architecture

Tandem NonStop systems are multiprocessor systems (sometimes called multicomputer systems). To provide fault tolerance, the processors in a NonStop system do not share memory, as many other multiprocessor systems do. Instead, each processor is a completely separate, fully functional computer with its own main memory, caches, I/O channels, and other components. The processors communicate with one another over a pair of high-performance buses, collectively called the Dynabus™, which allows the processors to cooperate and coordinate their work so they can function as a single-server system. Figure 1 illustrates the NonStop system architecture.

Figure 1

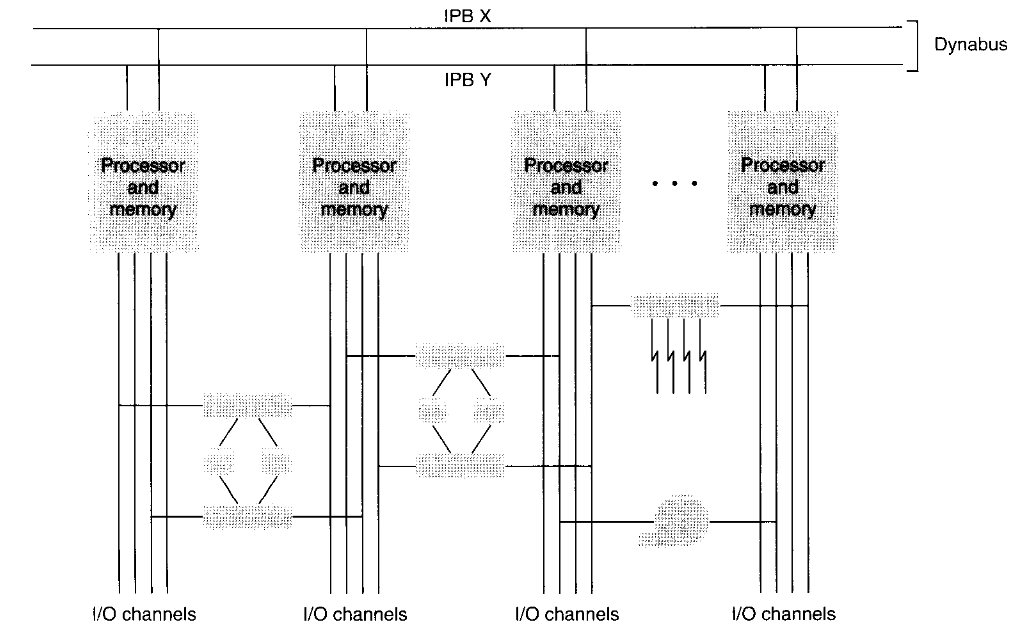


Figure 1.
NonStop system architecture.

As shown in Figure 1, the IPBs connecting the multiple processors play a central role in a NonStop system. The buses must be highly reliable, because all communication between processors depends on them. They must also be fault tolerant; therefore, the Dynabus comprises two IPBs. During normal operation, consistent with the Tandem philosophy of avoiding unused standby equipment, the software uses both IPBs for interprocessor communication. However, if a failure occurs, the software automatically routes all traffic over the remaining bus.

The IPBs must also provide both high bandwidth (aggregate transfer rate) and low latency (elapsed time for a transfer) for interprocessor communications. Because the Dynabus is one of the few shared resources in a NonStop system, it could become a bottleneck to system performance and scalability if it were not carefully designed for high performance.

Over the years, these performance considerations have led to several incremental changes in the design of the IPBs and the processor's interface to them. In the NonStop VLX™ systems (introduced in 1986), the bus protocol was altered and the bus clock speed increased relative to the NonStop TXP™ and earlier systems. The NonStop VLX and NonStop Cyclone™ systems increased the sizes of the packet queues that serve as buffers between the processors and the buses. In this tradition, the IPB subsystem was further improved to support the increased performance of the K10000 processor.

Supporting the Performance of the K10000 Processor

At the core of a K10000 server is the powerful MIPS® R4400™ (RISC) microprocessor, running at an internal clock rate of 150 MHz. The microprocessor contains 32 kilobytes of on-chip caches and is further supported by a 4-megabyte off-chip cache. Along with other hardware support, a K10000 processor can deliver up to twice the performance of a NonStop Cyclone processor.

The designers of the K10000 server had to determine whether the existing IPB design (the one used on the NonStop Cyclone and NonStop Cyclone/R™ processors) could support the increased performance of the K10000 processor without becoming a bottleneck. To answer this question, they performed several modeling and simulation studies. The designers developed a detailed model of a K10000 system and stressed it with a variety of workloads. With these models, they could predict various properties of the IPB subsystem as well as the overall throughput of the system under various conditions.

The conclusions from the study were as follows:

- The existing IPB design would be a system-performance bottleneck with processors of the anticipated performance.
- The performance bottleneck could be alleviated by changing the processor's interface to the IPB; the IPB itself did not have to be changed.
- The Dynabus+™ architecture of the NonStop Cyclone systems would provide the highest performance, with four processors per section being the optimal configuration.[1]

[1]The Dynabus+ architecture uses a dual fiber-optic ring to connect up to four sections of up to four processors per section. This architecture provided better performance than the flat, single-section architecture of the NonStop Cyclone/R systems.

Changes in the K10000 IPB

Guided by the study, the designers modified the processor's interface to the IPB subsystem. The K10000 servers introduce a new level of performance in the IPBs, while retaining much of the design and many system components from previous systems. The K10000 IPB subsystem includes the following enhancements:

- Direct memory access (DMA) mechanisms that transfer packets between main memory and the inbound and outbound interface packet queues.
- Asynchronous (nonwaited) sending mechanisms.
- *Send chaining* to link together multiple send requests into a single composite operation.
- Multiple send channels per bus.

Introduction to DMA

The concept of DMA is simple: an agent other than the processor itself moves data between memory and, in this case, the IPB. Previous Tandem NonStop systems used DMA agents for I/O operations, but not for the IPB. The processor's microcode or millicode[2] performed the data transfer between memory and the IPB, thus taking processor cycles away from the execution of processes and the NonStop Kernel.

For processors based on RISC technology (such as the K10000), the cost of moving data by the processor is even more significant. RISC designs rely on high internal clock rates and large caches, so references to main memory and to external interfaces such as the Dynabus are relatively more costly. DMA engines for both sending and receiving data on the IPBs relieve the K10000 processor of these burdens, freeing up more execution cycles for other work.

[2]Designers used the name *millicode*, which retains some of the flavor of *microcode*, to indicate that parts of the millicode software perform the same functions on RISC processors that the microcode did on previous Tandem NonStop CISC processors. The primary difference is that the microcode was programmed in a special-purpose language and operated below the TNS instruction-set level, whereas RISC millicode is a set of MIPS R3000® or R4400 instructions executing on the hardware platform, just as instructions in the NonStop Kernel do.

Asynchronous Sending

The concept of asynchronous sending is often misunderstood; indeed, it is often confused with that of DMA transfers. The basic concept has to do with waiting. In previous NonStop systems, when an IPB send operation was performed, the processor had to wait for the send operation to be completed before it continued executing. The processor could not perform other work until the entire message had been split into packets and moved from memory into the Out Queue (OUTQ).

Asynchronous transfer is not the same as direct memory access. In fact, NonStop systems have always had asynchronous receiving mechanisms, even without DMA. These systems were interrupt-driven. When a packet arrived in the In Queue (INQ), the processor was interrupted, and millicode (or microcode) moved the data from the INQ into memory. However, between packets, the processor resumed executing instructions; it did not wait for the next packet.

The NonStop Himalaya K10000 processors use asynchronous sending as well as DMA transfer engines for both sending and receiving data.

Send Chaining

Since the D00 version of the NonStop Kernel, a message on the NonStop system has consisted of two parts: a control part that describes the message and a data part that contains the actual data of the message. When a message is sent from one processor to another, the message system prefixes a setup section to the message. This section helps the receiving processor prepare to receive the rest of the message. (For some messages going to remote processors, the message system prefixes an additional section.) Thus, a typical message transmission over the IPB requires sending three separate, usually noncontiguous parts.

In previous NonStop systems, each part would require a separate SEND instruction. To reduce the number of interrupts and other required overhead, the NonStop K10000 implementation allows all parts of a message to be bundled together and described by a single chain of commands for the millicode and hardware to perform.

Multiple Send Channels per Bus

Often, the NonStop Kernel executing in one processor has messages that could be sent concurrently to several other processors (on the same system or remote systems). Furthermore, these messages frequently have different urgencies or other delivery requirements.

Sometimes, in previous NonStop systems, an *urgency inversion* condition could exist. Assume, for example, that a low-urgency but long message was first started for one destination, and that a high-urgency and shorter message had to be sent to a different destination before the first send operation was finished. Even if it used DMA and asynchronous sending, the NonStop Kernel would not be able to start the high-urgency send operation until the long send was completed.

The K10000 processors use asynchronous sending and DMA send and receive engines.

To alleviate this potential problem, the K10000 server implements multiple send channels on each bus, which allow multiple send operations to be outstanding to different destinations simultaneously. In the preceding example, the long send operation no longer blocks the short, time-critical send operation, since the latter can proceed concurrently on a different send channel.

Figure 2

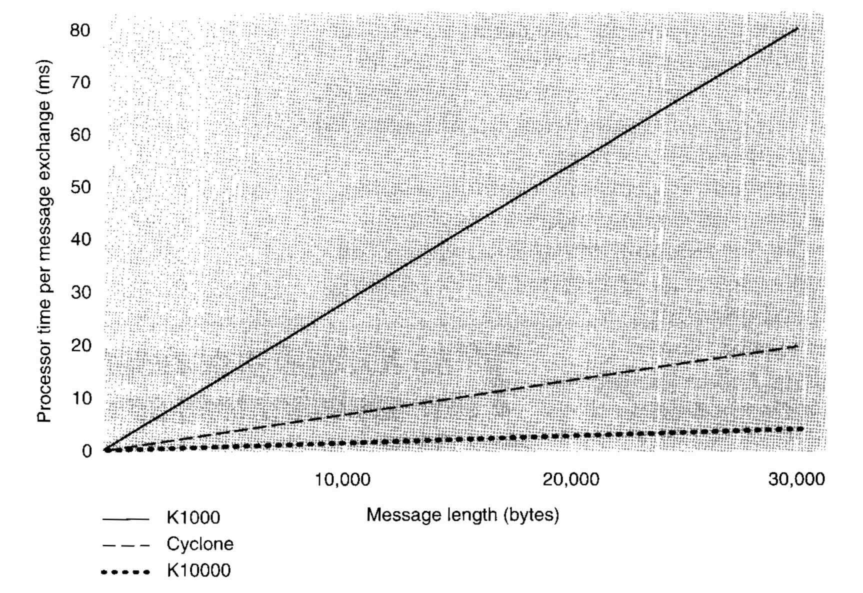


Figure 2.

Processor costs for interprocessor communication (sending and receiving same-sized messages) on the K10000, Cyclone, and K1000 processors.

Performance Measurements

Performance measurements show that the changes in the Himalaya K10000 IPB subsystem, summarized above, were sufficient to support the performance capabilities of the K10000 processor. The graphs in Figures 2 and 3 illustrate these performance results.

Figure 2 shows the processor time required to send and receive messages of various size on three processors: a Himalaya K1000, a NonStop Cyclone, and a K10000. Of the three, only the K10000 processor has DMA IPB and asynchronous sending capabilities. The cost of transferring data between processors on the K10000 is extremely low. Moreover, it is nearly constant for all message lengths. The Cyclone time, while quite low, shows that many processor cycles are consumed in transferring long messages. The K1000 time shows that this effect is relatively worse on a RISC processor that has neither DMA nor asynchronous sending capability.

Figure 3 shows the throughput for the same three processor types, again as a function of message size. For long messages, the K10000 begins to approach the theoretical point-to-point throughput limit of the IPB, slightly over eight megabytes per second. (This limit applies only to a single processor-to-processor transfer; the maximum throughput for multiple senders is higher.)

The message system performance shown in these figures is measured using LINKER, a development tool that measures the time (both processor and elapsed) needed for a round-trip message between a linker process and a listener process in different processors. Both processes use the privileged (link-level) interface to the message system. Therefore, the reported times include all of the operating system overhead (except for the file system) as well as any hardware transport delays that a message-based application might typically see. The size of the linker's request and the listener's reply can be varied independently. However, in the tests reported here, the message sent and the reply returned were of equal length, for each message size measured.

Architecture of the K10000 IPB Interface

The IPB features introduced in the preceding sections are implemented within a three-layer hierarchy: a hardware layer, a millicode layer, and a software layer. The hardware provides the DMA transfer engines and supporting hardware. The millicode provides a layer of services that serve mainly to isolate the software from certain implementation details of the hardware, providing a more abstract, machine-independent interface to the software. The software layer is the lower layer of the message system, a module within the NonStop Kernel operating system.

The remainder of this article describes how these three layers provide the IPB interface for the K10000 processor. For information about the IPB in general and another perspective on the K10000 IPB implementation, refer to the *NonStop Himalaya K10000 Server Description Manual* (1993).

Hardware Layer

The hardware is the lowest of the three layers implementing interprocessor communication. In addition to providing data integrity, single-fault tolerance, and the other standard Tandem features, the hardware was designed to meet three major objectives:

- Minimize the processor time costs associated with messaging.
- Maximize the efficiency of IPB bus use.
- Eliminate certain sources of failures known from previous designs.

Figure 3

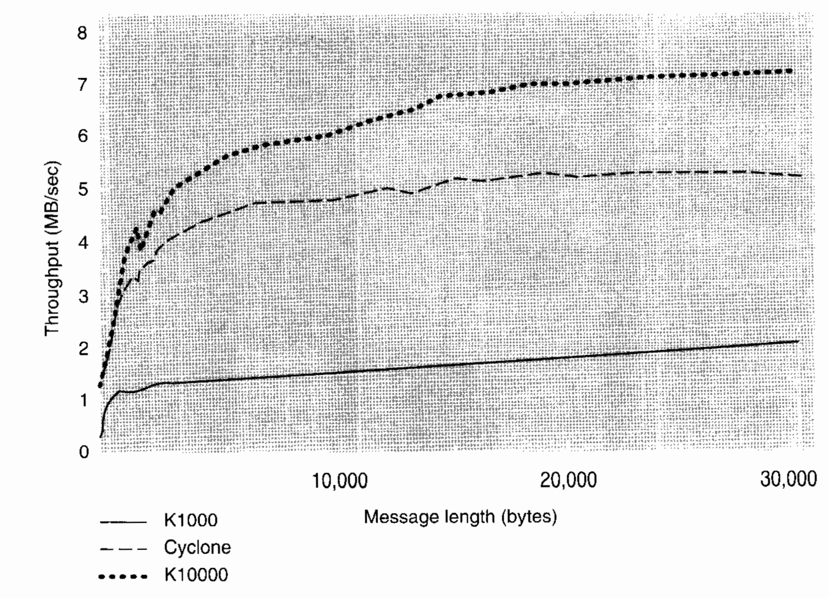


Figure 3.
Throughput comparison for interprocessor communication (sending and receiving same-sized messages) on the K10000, Cyclone, and K1000 processors.

Another goal was to lower the design time and costs by using some of the same components (outside of the processor) used on other NonStop systems. For example, the K10000 uses the same FOX™ and Dynabus+ logic boards that are used in NonStop Cyclone systems. Furthermore, the various bus controllers, terminators, and FOX and Dynabus+ adaptors are functionally (although not physically) identical to their Cyclone counterparts.

Figure 4

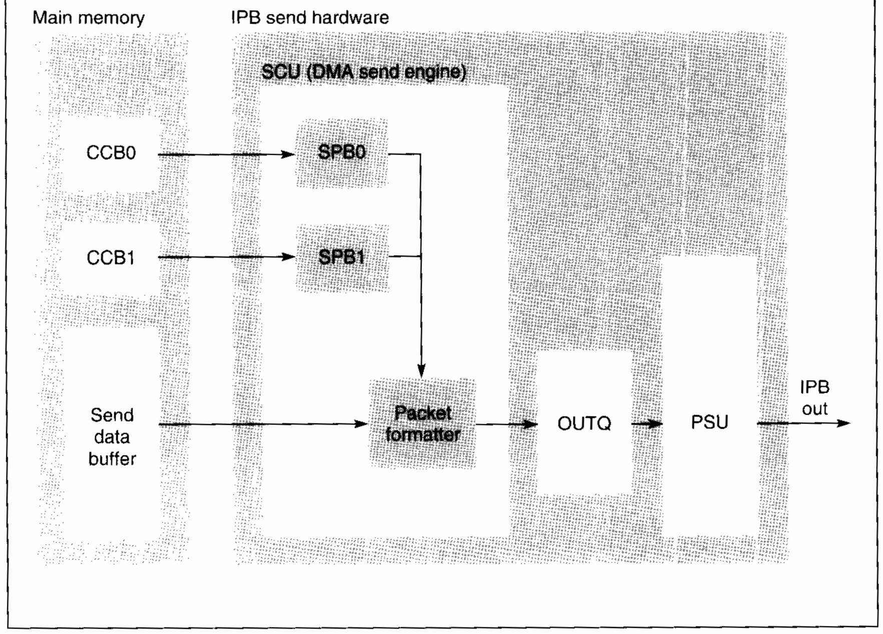


Figure 4.
IPB send engine functional block diagram.

Send Hardware

To meet the design objectives for send operations, the send hardware incorporates several key features. The hardware uses a command-chaining DMA engine controlled by the processor through a *mailbox* located in main memory. The DMA engine performs send-data fetching and packet formatting. A multichannel send structure is implemented; the hardware has two independent DMA send channels per bus that multiplex on a packet basis into the send stream. The send channels are completely interruptible and restartable at packet boundaries. Finally, the send hardware includes two traffic-shaping features, *poll limiting* and *bandwidth limiting*. Poll limiting prevents a sender from wasting bus bandwidth when the receiver is busy. Bandwidth limiting minimizes the likelihood of a fast sender overrunning a slow receiver.

Figure 4 shows the functional organization of the K10000's send hardware for a single bus. The processor can perform a sequence of one or more sends on a single channel by describing those sends in a mailbox data structure in memory, called a circular command buffer (CCB), which can contain up to 16 send segment commands. There is one mailbox per send channel. (Although a mechanism exists for a mailbox to chain multiple CCBs, the millicode layer currently uses only one CCB per channel.) After the CCB is set up, the processor starts the DMA send hardware for that channel.

The DMA send hardware, called the send control unit (SCU), fetches a single command from the CCB into its local storage (shown as SPB0 or SPB1 in Figure 4) and then executes it. The single command can describe a send segment of up to 16 kilobytes of contiguous physical memory (though the current millicode limits segments to a single page frame of 4 kilobytes). The send command is executed by handling as many packets as needed. For each packet, the SCU fetches data from memory, formats it into an IPB packet, and places that packet into the OUTQ. Once the packet has been placed in the OUTQ without encountering any errors, the SCU updates its local working pointers so the next packet can be fetched. It then informs the other send hardware unit, the packet send unit (PSU), to send the packet already in the OUTQ.

The PSU operates in the IPB bus clock regime and follows the protocol set by the IPB bus controller. The PSU asks to send a packet when the bus controller permits it and sends the packet when the bus controller orders it to do so. After the packet is sent, the PSU informs the SCU whether it was sent successfully or given a NAK handshake by the receiver. (A NAK is a flow-control feature of the bus controller's protocol that causes the packet to be retransmitted.) The PSU does not retry NAKed packets on its own; the decision to retry is made by the SCU, allowing effective use of the traffic-shaping and multiple-channel features.

When all of the data in a single segment has been fetched, formatted into packets, and sent by the PSU, the SCU fetches the next command in the CCB mailbox. The new command is executed just as the first one was. This process continues until the end of the chain of commands is reached. At the end of the chain, if so requested by the last command, the SCU posts a send-completion notification interrupt to the processor. In addition, if the send should terminate abnormally, such as with an uncorrectable memory error (UCME) detected in the data buffer, the SCU posts an abnormal-completion interrupt to the processor.

The SCU and PSU form a two-stage pipeline so that they can operate concurrently, fetching and formatting one packet while the previous packet is being sent. This pipeline is completely emptied at intercommand boundaries within a command chain. Therefore, the full performance benefit of this pipelined organization is not realized unless send segments are at least three or four packets long. With shorter messages, a single channel typically cannot send at the maximum bus rate much of the time.

Once the CCB is set up and the DMA engine is started, the processor is not involved in send activities (except for performing some minor monitoring functions described later in the Millicode Layer section of this article). With the hardware structure as shown in Figure 4, the processor can set up and start a DMA transfer independently on each of the two send channels. Whenever both channels are actively processing through their command chains, the PSU can be kept busy at the maximum bus rate even if the two streams each consist of short segments. Thus, the dual send channels increase efficiency in addition to avoiding the time-critical blockages described above.

Though the structure of the send hardware makes maximum use of common hardware between send channels, some unique hardware cost exists for each channel. Since many channel-use schemes could be optimal at the system level, it was hard to judge the value of the hardware expenditure for additional channels. It was decided that the best answer was to limit hardware costs to two channels (the minimum needed for concurrency, robustness, and maximum performance) and allow the millicode to emulate as many channels as desired. To support switching at a granularity smaller than a CCB, each channel is efficiently interruptible and restartable at packet boundaries.

Figure 5

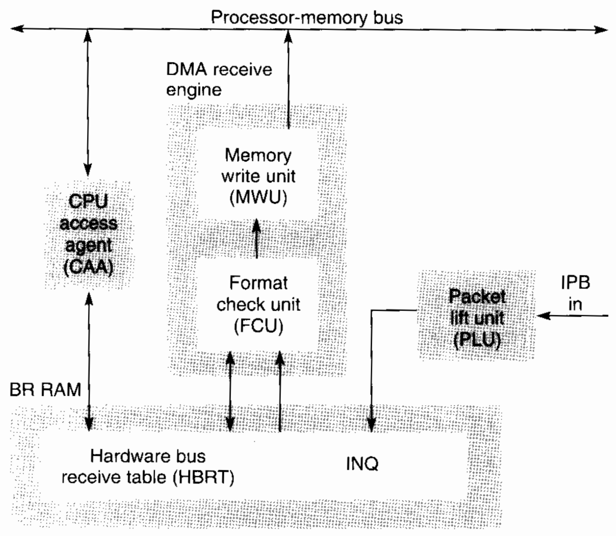


Figure 5.
IPB receive engine functional block diagram.

It had been observed on the NonStop Cyclone and previous processors that a noticeable portion of packet transfers on the IPB were NAKed. Two major sources of NAKs were discovered: receiver INQ overruns due to long MUTEX (mutual exclusion) intervals and FOX INQ overruns due to small INQ size and slower fiber transmission rates. The two traffic-shaping features, poll limiting and bandwidth limiting, cannot speed up the receivers, but they do permit other senders to use those packet-transmit windows that would have been NAKed, so that, at the system level, IPB throughput is higher.

Receive Hardware

Figure 5 shows the functional organization of the IPB receive hardware for a single bus on the K10000. Receive operations underwent fewer conceptual changes on the K10000 than did send operations. The basic change was the use of DMA for receive data. This change required moving the packet format-checking and data-extraction processes into hardware, which in previous systems had been handled by the microcode (or millicode). Moving those processes strongly suggested creating a hardware copy of the bus receive table (BRT), the architectural table that describes receive buffers. It was decided, therefore, that millicode would encapsulate the BRT and use the IPB hardware as its primary storage. Some hardware support had to be added to provide the processor with atomic access to the BRT. (See the CPU access agent in Figure 5.) DMA for receive operations (hardware format-checking) permits both buses to receive data into memory simultaneously, whereas in previous systems the processor could perform format checking on received packets from only one INQ at a time. Further, the K10000 pipelines operations so that packet reception and packet checking can execute concurrently.

The hardware performs high-bandwidth packet checking by using a dedicated and specialized programmable microcontroller called the format check unit (FCU). The working logic in the FCU is designed specifically to support IPB receive functions. It includes a route-word checker, a sequence-number checker, a checksum checker, and other checking functions. The FCU is controlled by an extensible microcontroller with writeable control store. Since packet format checking is complex, and it is this process that interfaces with the millicode layer, designers considered the flexibility of the FCU to be important. The FCU design achieves flexibility without sacrificing performance. The main microcode loop that handles most packets was optimized to process packets about 50 percent faster than the maximum rate at which they can arrive into the INQ.

After the FCU strips out the formatting information from each packet, the memory write unit (MWU) transfers the packet data into memory. While the MWU is performing the DMA for a packet, the FCU can proceed to handle the next packet concurrently, so that a high receive streaming rate can be maintained.

Meeting the performance goals of the FCU requires quick access to INQ and BRT data. This, in turn, requires that addresses in the BRT be physical instead of virtual. (There is no time to translate them.) Therefore, the hardware bus receive table (HBRT) is made an extension of the BRT structure. It includes space for the physical addresses of all pages touched by the receive buffer. The millicode sets up this information when the transfer is enabled.

Receive interrupts in a K10000 normally occur at most once per datagram (a sequence of consecutively numbered packets), rather than once per packet. Although far fewer receive interrupts are handled than on previous processors, care was taken to minimize the processor costs of these interrupts. The IPB hardware optimizes interrupt encodings so that the most frequent receive interrupts require the least expensive processor activity.

In the K10000, the format-checking and packet-arrival processes are given full concurrent access to the INQ. The packet lift unit (PLU) can place packets in the INQ independently of and simultaneously with the FCU's INQ activity. The INQ has high enough bandwidth to support multiple FCU accesses while picking up packets at the maximum IPB rate. The INQ size is also programmable up to 256 packets, so that overrun conditions can be made less likely. (Millicode currently uses a 16-packet INQ.)

Finally, the K10000 includes some additional checks on the proper functioning of the IPB protocol. These checks allow for earlier detection of and smoother recovery from errors that occur on the bus, including, for example, the loss of bus clocks due to the failure of or removal of a bus controller or terminator.

Figure 6

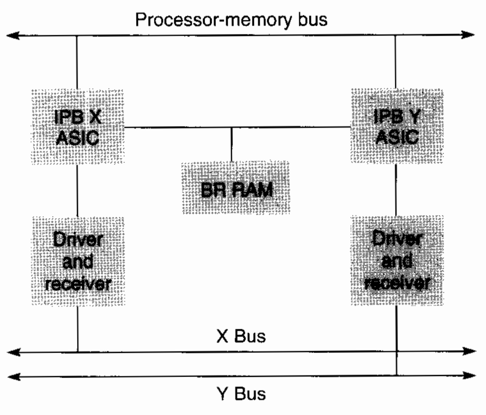


Figure 6.
IPB subsystem hardware block diagram.

Physical Implementation

Figure 6 shows the physical implementation of the K10000 IPB subsystem. The subsystem contains two application-specific integrated circuit (ASIC) chips, one for each bus. These share a small, fast memory comprising static RAM chips. This memory, called the bus receive RAM (BR RAM), holds both X and Y INQs, the BRT, and the FCU microcode. Finally, discrete driver and receiver chips provide the electrical interface to the IPB itself.

Millicode Layer

The Himalaya K10000 IPB millicode is a layer of software between the IPB hardware and the NonStop Kernel message system. The design of the millicode layer had three primary objectives:

- Provide an effective method for the message system to use the advanced features provided by the K10000 IPB hardware for sending and receiving data over the IPBs.
- Hide as many nonessential implementation details of the K10000 processor hardware and IPB hardware as possible.
- Achieve the preceding objectives with a minimum of overhead.

Some essential features exposed to the message system are the existence of multiple send channels per bus, the asynchronous sending capability, and the encapsulation of the BRT. The interface between the message system and the millicode was designed so that it could be used for a wide variety of future IPB designs that have the same general characteristics.

Examples of nonessential implementation details hidden from the NonStop Kernel are the cache-implementation and cache-coherency protocols, the hardware-register formats, and the addressing and accessing protocols of these registers.

The interface to the NonStop Kernel comprises a set of privileged millicode routines that the kernel can call (for example, to initiate or queue a send operation) and an interrupt interface (which is a minor extension of the existing BUSRECEIVE interrupt interface on previous NonStop systems).

Millicode Send Operations

The millicode enables the software to take advantage of asynchronous sending by providing a queue-driven interface to the send hardware. A new data structure, the send information block (SIB), carries all the information about a send operation (or set of related sends) across this interface.

The fundamental operation of starting a send was previously performed by the execution of a SEND instruction. Instead, the message system now constructs a SIB describing the send to be performed and calls a millicode routine, Enqueue_Send, to request the initiation of the send operation. This millicode routine never waits for that send (or any earlier send) to finish. Instead, if the send can't be initiated immediately, the SIB is placed in one of several queues for later processing. In either case, whether the send is initiated immediately or not, the Enqueue_Send routine returns control to the caller without waiting.

The millicode provides an efficient routine, Send_Status, that the NonStop Kernel can use to check on the progress of a send operation. For cases in which the kernel must override the usual K10000 practice of not waiting for a send to complete, the millicode provides the Wait_on_Send routine to do so. To cancel an in-progress send, the kernel can call the Cancel_Send routine. In all of these cases, the SIB is a parameter to the routine.

As described previously, the K10000 IPB hardware can chain send commands (and send buffers) together for greater efficiency. A limited form of this feature is exposed to the NonStop Kernel by allowing information about multiple sends to be grouped together in a single SIB. This allows the message system to group together frequently related sends such as the *setup* packets, the user's *request control*, and *request data* packets. Significant efficiency is gained by grouping these together in a single SIB. The millicode keeps these operations together as it links the SIB into queues, converts the SIB information into hardware commands, and initiates and tracks the send operation's progress. In most cases, the millicode uses the hardware send-chaining feature to initiate all the send areas described in a SIB with a single operation.

As mentioned earlier, the K10000 server provides multiple send channels per bus. The ideal number of send channels is at least four (for a system with FOX or TorusNet™ connections), because this number allows the splitting of local and remote sends on separate channels and also allows for a time-critical send channel separate from the normal channels. However, gate limitations within the IPB ASICs limited the number of separate send channels to two per bus. Thus, designers decided to provide the illusion of having four separate send channels per bus to the message system. The millicode accomplishes this by maintaining a dynamic mapping between the four channels seen by the software and the two channels provided by the hardware.

The IPB millicode maintains a queue of SIBs for each of the four send channels (per bus) that it provides. At any given time, for each bus, two queues are *mapped* to the two hardware-provided send channels, while the other two queues are *pending*. (That is, the two corresponding emulated channels are not mapped.) Whenever a send (SIB) is completed or a new send (SIB) is initiated, the millicode reevaluates the mapping to achieve the maximum sending concurrency, while maintaining the priority order of the queues.

When a SIB is placed into an empty queue that is mapped to a send channel, or when a SIB reaches the head of a mapped queue because earlier SIBs have been completed, the millicode initiates sending the data described by that SIB by converting the information in the SIB into hardware commands placed in the channel's CCB. At this time, cache flushing of the send areas is performed (because the R4400 processor's write-back cache may contain copies of data newer than those in memory), and the millicode handles packets that cross page boundaries. (Designers determined that it would be more expedient and cost-effective to have the millicode copy page-crossing packets to separate memory areas than to have the hardware send channels deal with page crossings in midpacket.)

Care was taken to minimize the number of IPB interrupts that the software must handle. One method has been described: the grouping of related send operations together into a single SIB at the NonStop Kernel-millicode boundary and the grouping of send commands together in a single CCB at the millicode-hardware boundary. In addition, the software takes advantage of the fact that in most circumstances, it does not need explicit notification of send-completion events.

Under the normal, relatively light IPB loads that are typical of OLTP applications, the send channels are idle when each send is started. In these cases, no interrupt is generated when a send is completed. Instead, the channel's status is checked when each new send is initiated; only at that time does the millicode notice the completion status of the previous send operation.

The millicode enables the software to take advantage of asynchronous sending.

However, when traffic is heavy, queues may temporarily form. In these cases, it is preferable to incur the expense of handling an interrupt to get the next send started at the earliest opportunity, rather than use some time-based status polling. The hardware optionally generates send-completion interrupts, a facility the millicode uses only when its queues are not empty. Similarly, the millicode optionally generates send-completion interrupts, a facility the message system uses only when its queues are not empty.

The millicode periodically monitors the progress of a send operation according to a parameter supplied in the SIB. If a send is not making progress within the time prescribed, it is cancelled and a send-timeout interrupt is generated. (The millicode then initiates the next queued send, if any.)

Millicode Receive Operations

As discussed earlier, the BRT has changed significantly in the K10000 server. In previous NonStop systems, the BRT was a simple data structure in main memory, initialized by the NonStop Kernel and updated by the microcode (or millicode) on each packet reception. However, with the advent of a DMA receive engine, the overhead and coordination problems of keeping the BRT in main memory were unacceptable. Instead, designers decided to encapsulate the BRT, allowing only controlled access to update it, so that the hardware could keep its own copy of the BRT information in fast RAMs and registers.

The millicode performs this encapsulation by providing a set of routines to the message system to access and update the BRT, while hiding the details of the hardware BRT and its accessing protocols.

To coordinate the sharing of the BRT between the software and hardware, each entry in the BRT (corresponding to a single source processor) is considered to be controlled either by the hardware or by the software. This distinction is actually implemented by an enable bit within the BRT entry. When the enable bit is on, the entry is controlled by the hardware, and the DMA engine is free to use the entry to transfer packets to memory from the corresponding source processor, updating the BRT information appropriately. When the enable bit is off, the entry is controlled by the software, and the DMA engine must not transfer packets to memory from the corresponding source processor, nor may it alter the entry. The message system uses five millicode routines (Disable_BRT, Enable_BRT, Read_BRT, Read_and_Disable_BRT, and Write_and_Enable_BRT) to read, write, and transfer control of the BRT entries.

In addition, the millicode provides two interface routines to allow the message system to turn the DMA receive engines on and off. Recall that this function was previously performed by the interrupt masks; when IPB interrupts were disallowed, transfer from the INQs to memory was also disallowed. However, more concurrency is achieved if these transfers can take place even while the processor is under MUTEX for some reason unrelated to the IPB operation. Therefore, this function is now separately controlled by the two millicode routines, Set_Receive_Enable and Reset_Receive_Enable. The X and Y buses can be controlled separately. In addition, when the hardware completes an IPB reception, the receiving DMA engine always disables itself until the software has had a chance to process the completion interrupt.

Message System Layer

The message system is the component of the NonStop Kernel that provides messaging services to the rest of the kernel, to system programs, and, through the NonStop Kernel file system, to applications programs (Chandra, 1985). It provides a request-reply message interface to processes running on the same processor or different processors in a Tandem NonStop system connected by the Dynabus. It also provides the same message protocols to processes running on remote systems interconnected by the FOX or TorusNet fiber-optic networks or by Tandem's Expand™ networking software, which supports a number of physical interconnect media.

The message system is the highest layer of the IPB subsystem described in this article. It communicates with the IPB hardware (the lowest layer) through routines supplied by the millicode (the middle layer). Of the three layers discussed here, the message system has undergone the least amount of change. Its highest sublayer, interfacing to its clients, has not changed at all, while its lowest sublayer, interfacing to the hardware, was entirely rewritten to use the new millicode routines.

Interface Changes

Because the message system's high-level interfaces to its clients are entirely unchanged, client software does not need to be modified, in spite of the significant changes in the IPB subsystem. The main clients of the message system are the NonStop Kernel file system, TMF™ (Transaction Monitoring Facility), the Input Output Process Request Manager (IOPRM), privileged I/O and communications processes, and privileged user applications.

A few NonStop system processes use the IPB to communicate with processors that are "down" (not running the NonStop Kernel) and cannot use the request-reply protocol provided by the message system. Such processes include the RELOAD utility, which loads the operating system image to down processors and brings them up; the RCVDUMP utility, which receives an image of the memory from a halted processor; the Tandem Maintenance and Diagnostics System (TMDS); and the IPBMON process, the system process that controls and monitors FOX and TorusNet. Before the K10000 IPB changes, these processes used the SEND NonStop processor instruction to send data on the bus. They also directly manipulated the BRT before receiving data on the bus. With the K10000 IPB changes, the message system has encapsulated the IPB for its low-level clients through two new processor-independent routines, XMIT_PROC_ and UPDATE_BRT_. The XMIT_PROC_ routine sends data on the bus and the UPDATE_BRT_ routine sets up a BRT entry before receiving data on the bus.

Major Features of the K10000 Message System

The message system uses most of the new facilities provided by the K10000 IPB hardware and millicode. The main features of the K10000 message system that differ from those on earlier processors are described in the following paragraphs.

Asynchronous Sending. Sending a message does not block the processor from doing other work. The message is queued to a send engine that asynchronously sends the message.

On earlier processors, calls to the MSG_LINK_ and MSG_REPLY_ routines waited (with some exceptions) until the entire message was placed in the IPB hardware OUTQ. For messages larger than the OUTQ size, this implied that the processor idled without executing useful instructions until all of the message (except the last OUTQ worth) was sent out on the IPB. When a processor idles, waiting for something to happen, it is said to be *busywaiting*. On the K10000, the MSG_LINK_ and MSG_REPLY_ calls queue the request to the millicode and return immediately without waiting. Waited calls such as MSG_READCONTROL_ and MSG_READDATA_, which require data transfer from the requester processor, do not behave differently on the K10000 processor than they do on previous processors.

Client software does not need to be modified, in spite of the significant changes in the IPBs.

Asynchronous sending results in better processor utilization. For an IPB-bound application, the improvement is dramatic. Figure 2 shows the differences in processor utilization between the Himalaya K10000, Himalaya K1000, and NonStop Cyclone processors.

Multiple Buffers per Send Operation. As discussed earlier, a typical message sent over the IPB has three noncontiguous parts: the setup, control, and data areas. Since the SEND instruction on earlier processors could send only one buffer at a time, multiple SEND instructions were required to send a message over the IPB. With the synchronous IPB interface on those processors, the multiple sends did not increase the message overhead appreciably, but with the asynchronous IPB interface on the K10000, multiple sends would require extra send-completion interrupts. The interrupt overhead is eliminated by describing the separate send buffers in a single SIB and using a single millicode Enqueue_Send operation.

Multiple Send Channels. For each bus, the IPB millicode provides the message system with four send channels with different send priorities. The two channels with highest priority are used to send waited messages generated by the NonStop Kernel (such as the *IamAlive* messages exchanged by all NonStop processors every 1.2 seconds). The two channels with lowest priority are used to send user messages.

Encapsulated BRT. The BRT is not shared by the IPB millicode (or microcode) and the NonStop Kernel. It resides in the IPB hardware and is encapsulated for the message system by the millicode. The message system uses millicode routines to read and update the BRT.

Separate Mechanism for Disabling Buses. Separate mechanisms are used to disable bus receive software interrupts and transfers from the IPB INQ into memory. On earlier processors, a single interrupt mask bit served both purposes. On the K10000, millicode routines are called to stop and restart IPB transfers into memory. This enhancement means that bus receptions are not stopped when interrupt handlers (except the BUSRECEIVE interrupt handler) are running and when interrupts are disabled.

DMA Sending and Receiving. Both send and receive engines use DMA to access the data buffers. The DMA engines use physical addresses to access memory, while the message system clients work with virtual addresses. The message system uses a memory manager routine to translate the virtual addresses to page-frame numbers before invoking the DMA engines.

Send Implementation

The two primary changes in the message system send implementation are the use of asynchronous sending and the use of multiple send channels.

Use of Send Channels. As stated earlier, the K10000 IPB millicode provides four send channels per bus, two for local transmissions and two for remote transmissions. The channels are assigned different send priorities, with lower channel numbers having higher priority.

The message system uses channel 0 exclusively for sending unacknowledged datagrams to other processors on the same NonStop system. These datagrams include time-critical unsequenced IPB packets (with the special sequence number of -1) such as the *IamAlive* packets, regroup and poison packets (sent when a processor is slow to respond or has been declared down), and datagrams sent using the XMIT_PROC_ interface by system utilities such as RELOAD and RCVDUMP. All sends on channel 0 are waited. The processor busywaits until the transmission is completed by calling the Wait_on_Send millicode routine. (The processor can service interrupts while busywaiting, but it cannot run processes.)

The message system uses channel 1 exclusively for sending unacknowledged datagrams to other processors over the FOX network or TorusNet. These include time-critical unsequenced handshake packets exchanged by IPBMON processes when establishing or modifying remote connections, periodic idle packets exchanged to preserve these connections, and packets sent to the FOX or TorusNet controller board, the local bus unit (LBU). These send operations also cause the processor to busywait until the transmission is completed.

Channels 2 and 3, the low-priority channels, are used to send acknowledged messages (all user messages and most NonStop Kernel messages) and their acknowledgements (which are sent as unsequenced packets). The message system uses channel 2 to send to other processors on the same system, and channel 3 to send to remote processors on the network. These messages are queued to the millicode and do not cause busywaiting.

Limiting Millicode Queue Lengths. The message system supports a transmission window of four outstanding unacknowledged transmissions to each destination processor. On a FOX or TorusNet system, if there were no other limiting factor, the millicode send queues could contain up to 892 such outstanding messages (60 to local processors and 832 to remote processors). Although the millicode does not restrict the size of the message queues for the send channels, there are practical reasons to limit the queue lengths.

A potential problem with allowing send queues to contain up to four SIBs to each destination is that the time to receive acknowledgements for the outstanding transmissions could get very long. The message system uses a timeout mechanism for these acknowledgements to detect lack of progress of a message. The time allowed for the timeout must be significantly greater than the elapsed time to get an acknowledgement back under heavy IPB message traffic. Allowing the queues to hold hundreds of SIBs would require such a long timeout that it would delay the timely detection of real problems.

Long millicode queues would also consume large amounts of memory. Each queue element is a SIB, a fairly large data structure.

To avoid both problems, the K10000 message system implementation limits the length of a nowaited channel queue to one sequenced message and one unsequenced acknowledgment packet. On each processor, one sequenced and one unsequenced SIB are preallocated for each channel on each bus. These SIBs are used for sending messages to all destinations using that channel.

Sending a Message. When a message is to be sent to another processor, the message system picks the current bus (X or Y) and a channel (based on whether it is a local or remote message). Then it checks to see if the SIB for the selected bus and channel is free by calling the Send_Status millicode routine. If the SIB is free, the message system copies the send information into it. The SIB contains multiple send areas to describe the multiple buffers that constitute a message going over the IPB.

The message system then calls the Enqueue_Send millicode routine to initiate the send operation. A single millicode operation is sufficient to send the multiple buffers forming a message. The millicode places the SIB in a queue for the channel. When the channel becomes free, the millicode commands the send channel to begin sending the data associated with the SIB at the head of the queue.

If the Send_Status routine indicates that the SIB is in use (either being transmitted or awaiting transmission), the message system requests the millicode to invoke a special interrupt when the transmissions associated with the SIB are completed. The message is then placed at the tail of a queue of messages waiting for the SIB to become free. When the SIB-completed software interrupt is invoked (through the BUSRECEIVE interrupt vector), the interrupt handler removes the message at the head of the queue of messages waiting for the SIB and arranges to have the message sent.

The millicode provides the message system with four send channels per bus.

Receive Implementation

There are two primary changes in the message system receive implementation. First, the BRT resides in the IPB hardware and is encapsulated by the millicode using a procedural interface. A new message system routine, UPDATE_BRT_, was created for the NonStop Kernel to update the BRT. This routine invokes the millicode BRT procedures.

Second, in normal operation, bus reception (transferring data from the IPB INQ into memory) is disabled only while processing the completion of a message reception. In previous processors, IPB data transfer was also disabled when BUSRECEIVE software interrupts were disabled.

Receiving a Message. As indicated above, on the K10000 server, the BRT resides in the IPB hardware, and the message system uses millicode routines to access it. To read a BRT entry, the Read_and_Disable_BRT millicode routine is called if the entry is enabled and the Read_BRT routine if it is already disabled. To update a BRT entry, the message system disables the BRT entry (unless it is already disabled or the buses are both disabled) and updates it by calling the Write_and_Enable_BRT millicode routine.

When all packets of a sequenced transmission are received from a source processor on a particular bus, the IPB hardware turns off receptions on that bus and disables the BRT entry for the source processor. The hardware then posts a millicode interrupt. The millicode interrupt handler sets up the interrupt parameters and invokes the BUSRECEIVE software interrupt handler. BUSRECEIVE processes the received message and sets up for the next message to be received from the same source processor. It then updates and reenables the BRT entry by calling the Write_and_Enable_BRT millicode routine and reenables receptions on the bus by calling the Set_Receive_Enable millicode routine.

When an unsequenced packet is received on a bus, no BRT entry is used and hence no BRT entry is disabled. The hardware turns off receptions on the bus and interrupts the millicode, which in turn invokes the BUSRECEIVE interrupt handler. BUSRECEIVE processes the packet received, but does not update the BRT. It then reenables receptions on the bus by calling the Set_Receive_Enable millicode routine.

BUSRECEIVE can also be invoked in the case of receive errors (such as checksum errors or sequence errors), bus errors (such as bus-controller parity errors), and some send errors (such as send timeouts). Some of these errors do not cause BRT entries to be disabled, but others do. Through special bits in the interrupt parameter words, the millicode informs BUSRECEIVE if the BRT entry or bus has been disabled. In these error cases, BUSRECEIVE enables the BRT entry or the bus only if necessary.

Thus, bus reception is turned off by the hardware when message reception is completed on a bus and reenabled at the end of BUSRECEIVE processing for that reception. Bus reception and BRT entries are not disabled by the MUTEX facility or by entry into software interrupt handlers, as was done on previous processors. The only time reception on a bus is turned off by the NonStop Kernel is when it brings down a bus because too many receive errors (such as checksum errors) occurred. To do this, it calls the Reset_Receive_Enable millicode routine. When this happens, the error condition should be corrected. An operator can then reenable receptions by using the TACL™ (Tandem Advanced Command Language) XBUSUP or YBUSUP command.

BRT Accesses Outside of BUSRECEIVE. When a system is coldloaded, all BRT entries are initialized to expect a setup message with a packet sequence number of zero. The BRT is also accessed by Tandem system processes such as RCVDUMP and IPBMON to set up receptions from down processors (processors not running the NonStop Kernel). The RCVDUMP process uses the BRT to set up memory-dump reception from a halted processor and the IPBMON process uses it to receive messages from the FOX or TorusNet controller board, which is addressed as a special processor on the network.

To simplify updating a BRT entry, the message system now provides the UPDATE_BRT_ routine, which takes as parameters the BRT entry values to be updated. The routine works on all hardware platforms. On the K10000, it disables the hardware BRT entry and then calls the Write_and_Enable_BRT millicode routine. UPDATE_BRT_ is called by the processor initialization code to initialize the BRT when the processor is coldloaded or reloaded. It is also called by the RCVDUMP and IPBMON processes.

Error Packet Reception Areas. When the DMA engine receives an unsequenced IPB packet or a packet with an error, the packet is written into a special memory buffer called the error packet reception area. On the K10000, when one bus is disabled due to an error packet reception, the other bus is still enabled and can receive packets and detect errors. Therefore, a separate error packet area is provided for each bus. (In previous processors, a single error packet reception area was sufficient because reception on both buses was disabled when the error packet area was filled by either bus.)

Conclusion

The NonStop Himalaya K10000 server provides higher performance than any previous Tandem system while maintaining compatibility with all existing Tandem user applications. Its Dynabus subsystem was designed to provide high performance so that the K10000 processor can deliver its full potential, not only in the OLTP applications Tandem systems have traditionally supported, but also in batch and decision support applications, and in combinations of these applications. System performance measurements have verified that these design goals have been met.

References

Chandra, M. 1985. The GUARDIAN Message System and How to Design for It. *Tandem Systems Review*. Vol. 1, No. 1. Tandem Computers Incorporated. Part no. 83934.

Kong, C. 1994. A Hardware Overview of the NonStop Himalaya K10000 Server. *Tandem Systems Review*. Vol. 10, No. 1. Tandem Computers Incorporated. Part no. 104396.

NonStop Himalaya K10000 Server Description Manual. 1993. Tandem Computers Incorporated. Part no. 84559.

Acknowledgments

The authors wish to thank all of those who helped in the design and implementation of the Himalaya K10000 IPB subsystem. Specifically, thanks to Srinivasa Murthy and Dave Kinkade for their wise comments and insights into the message system, and for acting as sounding boards for the K10000 message system design. Thanks also to Yi-Fan Hsu for his tireless help in making the hardware real, to Dale Southgate for providing excellent assistance in the millicode area, and to Andrew Samuels for testing and inspecting above and beyond the call of duty. The authors would also like to thank the reviewers for their helpful comments on earlier drafts of this article.

Steve Hamilton was the technical leader for development of the Himalaya K10000 IPB subsystem hardware. Including his 4 years at Tandem, he has had 18 years of experience doing board-level and ASIC design, custom IC design, and CAD software development. He holds two bachelors degrees, in mathematics and electrical engineering, from Duke University.

Robert Jardine designed the millicode layer of the Himalaya K10000 IPB subsystem. He joined Tandem in 1984 to work on the NonStop Cyclone processor design and microcode. His experience prior to joining Tandem included 12 years of design and implementation of compilers, operating systems, and processor microcode. He holds a degree in mathematics from Harvey Mudd College.

Karoor Krishnakumar designed the Himalaya K10000 message system. During his five years at Tandem, he has been an operating system designer supporting the message system and FOX. Before joining Tandem, he spent four years as a CAD software designer. He has a bachelors degree in electrical engineering from the Indian Institute of Technology and a masters degree in computer science from the University of Southern California.

Client/Server Availability

Wherever computers are used, loss of user time through the failure of a hardware or software component has always been a significant concern. Ten years ago, when the most common configuration was a set of terminals connected to a mainframe, computer systems and their interconnections were usually fairly simple, and analyzing the causes and costs of user downtime was relatively straightforward.

The current world of client/server computing is far more complicated. In a client/server environment, individual workstations on a local area network (LAN) may be connected to multiple servers and possibly a mainframe. A single network may involve several types of transmission media, such as Ethernet and fiber-optic cable, and require multiple bridges, routers, and other network devices. Successful execution of a single database query often requires that five or more pieces of equipment and the software that runs on them all be operating properly. In such an environment, it can be difficult to analyze all possible types of failures and their costs in user downtime. This, in turn, makes it difficult to arrive at conclusions about probable weak points in a client/server environment and the most effective ways to reduce downtime.

This article discusses client/server availability and ways of reducing user outage times.[1] User outage time is the amount of time that individual users in a client/server environment would be expected to lose due to outage causes of all types, whether at the level of a single client, a server, or an entire network. The article consists of two main parts. Part 1, "Client/Server Availability Model," presents a predictive model for evaluating client/server availability and applies the model to a representative client/server environment. It provides outage statistics covering many types of failures that can affect hardware and software components. A significant finding is that without fault tolerance, servers account for more user outage minutes than any other single factor. Part 2, "Improving Availability in a Client/Server Environment," beginning on page 32, discusses ways of increasing client/server availability through operational improvements, greater fault tolerance, and other means.

[1]For a longer version of this article, with a list of references on the topic of availability and sources for outage statistics, see *NonStop Availability in a Client/Server Environment* (Wood, 1994).

Part 1: Client/Server Availability Model

Given the hardware of ten years ago, a complex client/server environment would probably be down as often as it was up, and an analysis of availability would primarily be concerned with hardware outages. However, hardware reliability has improved dramatically in the last decade, and the analysis of availability must give equal consideration to many other causes of outages, such as software bugs, operator error, and power failures. Planned outages must also be included in an analysis, since scheduled maintenance periods are being eliminated as the need for continuous availability of business-critical applications increases.

Defining and Measuring Availability

Availability, classically defined as

$$\text{Availability} = \frac{\text{system uptime}}{\text{system uptime} + \text{system downtime}}$$

is a standard system performance measure. Availability is usually expressed as a probability or percentage, for example, 99.9 percent availability, but this metric can be difficult to use or envision. Unavailability, or outage time, defined as *1 – availability* and expressed as annual downtime, is more useful. For example, a design change that increases availability from 99.9 percent to 99.99 percent may not appear very significant. In contrast, expressing the change as a decrease in downtime from 500 minutes per year to 50 minutes per year does appear significant and better conveys the true impact of the design change on business productivity. Table 1 shows the relationship between percentages and annual outage minutes.

In a distributed client/server environment, it is not very meaningful to define availability in terms of system failure and to measure it in terms of system uptime and system downtime. If a single PC fails, the user of the PC cannot access the application, but all other users can continue operating without any loss of performance (unless the PC has failed in a mode that causes an outage for other users.) If 1 out of a 1000 users cannot access the system, is the system down? If not, what about 10 out of a 1000 or 100 out of 1000?

Table 1.
Relationship between percentages and annual outage minutes.

Availability	90%	99%	99.9%	99.99%	99.999%
Annual outage minutes*	50,000	5,000	500	50	5

*For purposes of illustration, maximum availability is treated as 500,000 minutes, rather than the full 525,600 minutes of a calendar year.

To accurately reflect the impact and scope of an outage, it is important to measure client/server availability from a user point of view rather than a system point of view. Such a measure needs to include both the duration of an outage and the number of users affected. Accordingly, the predictive availability model presented in this article is concerned with user availability, defined as

$$\text{User availability} = \frac{\text{user uptime}}{\text{user uptime} + \text{user downtime}}$$

where *user uptime* is time during which users can effectively use their applications. If an average of 1 user out of a 1000 cannot effectively use needed applications, that is equivalent to 99.9 percent user availability or, in terms of Table 1, an average of 500 annual outage minutes per user. For the remainder of this article, user availability is expressed in terms of annual user outage minutes. Measuring client/server availability as user availability and user outage time avoids making an artificial decision about the number of down users that is equivalent to a system outage.

Table 2.
Definitions of outage categories.

Outage category	Definition
Physical	Physical faults or failures in the hardware. Any type of hardware component failure belongs in this category.
Design	Bugs in design and design failures in hardware and software. An example is an application change that introduces unexpected problems.
Operations	Errors caused by operations personnel and users, due to accidents, inexperience, defective procedures, or malice.
Environmental	Power or cooling system failures, failure of external network connections (for example, a leased-line outage), natural disaster (earthquake, flood), accidents, terrorism.
Reconfiguration	Any planned outage. Examples include downtime required for planned maintenance, such as upgrading hardware or software, and configuration changes, such as adding a new disk or restructuring a database.

Client/Server Outage Data

An effort was made to collect as much client/server outage data as possible. In the process, it was surprising to find that very few companies keep complete client/server outage data. Most companies approached did not keep outage data of any type. A few companies had data limited to network outages, and a few had data on server outages. None kept data on individual client outages. The best source of data was a company with a large networking application and about 15,000 users. The company keeps detailed records on all outages that affect four or more users and provided us with their entire set of outage records for 1992.

All the data collected was categorized according to five types of outage: physical, design, operations, environmental, and reconfiguration. Table 2 lists the outage categories and their definitions. Table 3 gives examples of outages in each category.

The wide variety of outages in Table 3 reflects the complexity of the client/server environment. Note the potentially disastrous effect some of the outages could have on a business, especially the building transformer that caused 1027 users to be down for 575 minutes, resulting in a total of 590,525 user outage minutes.

One of the most interesting findings to emerge from the outage data is the difference between the actual causes of client/server failure and users' perceptions of those causes. When people think of failures, they tend to think of physical hardware failures, but according to the outage data, physical failures cause only 10 to 20 percent of unplanned outages. Design, operations, and environmental failures are all as common as, or more common than, physical failures. Planned outages (reconfiguration) are also more common than physical failures. Another interesting point is that client/server users tend to blame all outages on the LAN. However, the data shows outages of non-fault-tolerant servers to be the primary source of user outage minutes (see the discussion under "Fault-Tolerant Servers" later in the article).

Outage Causes

The complexity of a client/server environment introduces a wide range of new failure mechanisms. To establish a list of outage causes and obtain outage times for the availability model, outage data was augmented with literature surveys (for example, Caginalp, 1992; Caldwell, 1992; Saal, 1990), articles quoting downtimes (for example, Bowen, 1992), university studies (for example Feather, 1992; Lee et al., 1993), quotes on hardware reliability from vendors, and internal outage data. Table 4 shows a number of the outage causes derived from these sources, listed according to outage category. Some specialized terms for outage causes that may be unfamiliar to the reader are defined in Table 5.

Table 3.
Outage examples classified by type of outage.

Outage category	Cause of outage and action taken	Outage minutes	Number of users down	Total user outage minutes
Physical	Clients lost access to the application due to a broadcast storm* on the LAN caused by a bad supervisor card in the hub. The card was replaced to restore service (a spare was available on site).	9	287	2,583
Physical	Sixty workstations did not have access to the application due to a router problem. Operations traced the problem to a loose cable on the download server. The cable was tightened and the routers reloaded, and the clients rebooted their workstations.	143	60	8,580
Design	Fifteen workstations were unable to access the application. The problem was traced to a hung file server. Operations performed a stop and start of the file-server process to restore service to the client.	13	15	195
Design	Clients could not access the application. Operations stopped and restarted the name-server process to allow access.	5	9	45
Design	Clients could not access the application due to a hub problem. Operations reset the supervisory card on the 8th floor and the Ethernet cards on the 3rd and 4th floors.	90	58	5,220
Design	Clients lost access to the application. Operations traced the problem to an out-of-sync condition between the file-server and line-handler processes that occurred following a failure of the host application. Both processes were stopped and restarted to resynchronize.	24	30	720
Design	Clients experienced degraded service following an application maintenance release. The release was backed out to restore service.	31	205	6,355
Operations	After power was restored, the Ethernet card to a file server did not properly restore because an operator had configured it incorrectly.	360	25	9,000
Operations	Clients lost access to the application because a maintenance technician on site inadvertently placed equipment in a test mode. The technician normalized the equipment and users softbooted their workstations to restore service.	5	25	125
Environmental	Construction group accidentally set off fire-extinguishing (Halon) system in computer room.	420	25	10,500
Environmental	Clients lost access to the application due to a failed remote router link. The failure was traced to a cross-connect problem in the link caused by a power failure. The router was reset and service restored.	486	40	19,440
Environmental	Clients could not access the application due to a bad fuse in the building transformer. Due to the location of the transformer, it took several hours to replace the fuse and restore power.	575	1,027	590,525
Reconfiguration	Server network software was upgraded to include Appletalk.	120	50	6,000

* *Broadcast storm* is defined in Table 5.

Table 4.
Examples of client/server outage causes.

Physical	Design (cont.)	Operations	Environmental	Reconfiguration
CPU, LAN card, disk, etc., fail	Database corruption	Cable bumped accidentally	Commercial power fails	Upgrade system
Babbling node*	Disk access error	Wrong cable pulled	Natural disaster	Add disk
	LAN protocol error	Incorrect network address entered	Heating, ventilation, or air conditioning fails	System move
Design	Access denied	Data not backed up	Circuit breaker tripped	New release
PC with lock on database fails	Router algorithm conflict*	Stopped wrong process	Power failure recovery error	Bug fix
Firmware error	Packet errors (runts*, jabbers*)	Table or log erroneously deleted	Virus	Workload balancing
Self-test failure	Timeouts	Erroneously stopped application		
Operating-system crash	Application freeze			
Broadcast storm*	Network paging*			
Server hung				

* Term is defined in Table 5.

Table 5.
Definitions of specialized terms for describing outages.

Term	Definition
Babbling node	The transmission of random, meaningless packets onto the network; often caused by a failed LAN card.
Broadcast storm	A broadcast is a special message or packet that all network hosts must receive and process. A broadcast storm is a condition in which excessive broadcasting occurs, potentially disabling the entire network. Broadcast storms are usually due to software errors.
Jabbers	Packets that are larger than the maximum length allowed by the network protocol (for example, 1518 bytes for Ethernet).
Network paging	Occurs when a workstation runs a job too large for its memory and has to page over the network, causing very heavy network traffic.
Router algorithm conflict	Routers use some form of shortest-path algorithm. If router A thinks that the shortest path to router C is through router B and router B thinks that the shortest path to router C is through router A, packets for router C will be sent back and forth between routers A and B. This type of problem usually occurs because of a breakdown in the routers' shortest-path updating strategy, due to software design errors.
Runts	Packets that are smaller than the minimum length allowed by the network protocol (for example, 60 bytes for Ethernet).

Most of the outage causes in Table 4 apply to any type of computing environment, but some are *propagating* outages unique to client/server computing. In a propagating outage, a hardware or software failure affects more than the users or equipment dependent on the failed item. For example, a router failure should only cause downtime for users dependent on the servers and hubs connected to the router (see Figure 1 for an illustration of multiple routers with associated hubs, servers, and users). However, if a router failure causes a router algorithm conflict (defined in Table 5), this will result in very heavy traffic and poor response times throughout the LAN and cause all users to think the LAN is down. In general, propagating outages can affect many users and result in very large amounts of user outage minutes. In Table 5, broadcast storm, babbling node, network paging, and router algorithm conflict are all examples of propagating outages.

Defining a Representative Client/Server Environment

In demonstrating a predictive model of availability, it is useful to define a representative client/server environment. Such an environment is not meant to be an optimal configuration or a model for designing other environments. Its purpose is to provide a basis for illustrating the effects that different outages would have on users and to make it easier to interpret numerical values derived from the availability model.

Typical Features of a Client/Server Environment. Client/server environments are very diverse. They range from a small departmental LAN with a single server to E-mail networks connecting tens of thousands of users to hundreds of servers. Even business-critical applications encompassing networks of ATMs have begun to adopt client/server architectures. Although it is difficult to define a standard client/server environment, it is possible to identify characteristic features, such as those described in the following paragraphs.

In the typical client/server network, clients use local servers connected by a LAN. Clients are usually physically near their primary server, although this is not a requirement. A communications server on the LAN provides wide area network (WAN) access.

There is no standard client/server configuration, but there are standard client/server components. Servers are generally powerful workstations, although they can also be minicomputers or mainframes. Servers often perform a specific type of service, specializing, for example, as file servers, database servers, and print servers. Clients are most often PCs or other workstations, although there are certainly other types of clients, such as ATM machines, handheld devices, and smart phones. Typical networking components include transceivers, bridges, routers, hubs, and gateways.

Figure 1

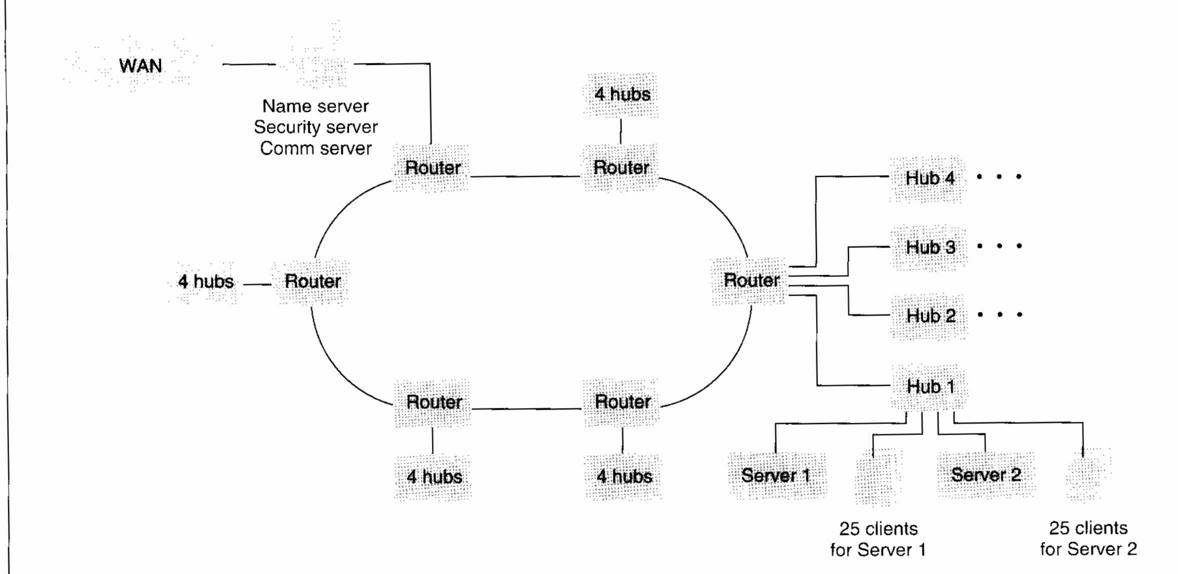


Figure 1.
A representative database-server architecture containing 1,000 clients, 20 hubs, 6 routers, 40 database servers, and 1 network management server.

In a typical LAN with database servers, there are 10 to 50 clients for each database server. For present purposes, 25 clients per server is used as a reasonable average, based on the client/server environments studied and the judgements of experienced client/server operations managers. Equipment in the LAN is layered hierarchically, with subnets joined to form larger networks.

A Representative Client/Server Environment. A representative environment for use with a predictive model of client/server availability should be similar to real-world configurations in which availability is a primary concern. Since availability is of particular importance for a business-critical database application, the representative environment used in this article is a LAN architecture based on database servers. It is illustrated in Figure 1. The LAN incorporates the features described in the preceding section. Servers on the LAN are generic, non-fault-tolerant servers. Most conclusions reached on the basis of the representative LAN apply equally well to other architectures, such as the typical campus LAN on which file and print servers predominate.

In Figure 1, clients and their primary servers are connected to a hub. There are 25 clients for each primary server and two servers to a hub, for a total of 50 clients per hub. Four hubs are attached to a router, making a total of 200 clients per router. Six routers are connected in a ring, such as a fiber distributed data interface (FDDI) ring. Five of the routers support hubs with servers and clients, resulting in a total of 1,000 clients. The sixth router functions as a gateway for communications outside the LAN, and also provides a server for network management. Although network management activities such as name services and security services are often spread throughout a network, the design in Figure 1 is sufficient for purposes of interpreting the availability model of the next section.

Table 6.
Calculation of annual user outage minutes.

Outage cause	Annual outage minutes per item	Total number in c/s environment	Total annual outage minutes	Number of clients affected	Annual user outage minutes
Hub client LAN-card failure	4	100*	400	10	4,000
Hub server LAN-card failure	4	40	160	50	8,000
Hub LAN-card failure causing babbling-node outages	1	140**	140	200	28,000
Server-application software failure	280	40	11,200	50	560,000

* Based on 1,000 clients and 10 clients per LAN card
** Sum of client and server LAN cards

Figure 2

$$\begin{array}{c}\text{Annual}\\ \text{user outage}\\ \text{minutes}\end{array} = \left\{\begin{array}{c}\text{Average annual outage}\\ \text{minutes due to a specific}\\ \text{failure type and element}\end{array}\right\} \times \left\{\begin{array}{c}\text{Number of such}\\ \text{elements in the}\\ \text{network}\end{array}\right\} \times \left\{\begin{array}{c}\text{Number of users}\\ \text{affected by}\\ \text{each failure}\end{array}\right\}$$

Figure 2.
Formula for calculating annual user outage minutes.

Table 7.
Summary outage statistics for the representative client/server environment.

Total annual user outage minutes from all causes (1,000 clients)	12,031,800
Annual outage minutes per client	12,032
User availability	97.7%
Percentage of user outage time due to unplanned outages	68.0%
Percentage of user outage time due to planned outages (reconfiguration)	32.0%

A Predictive Model of Client/Server Availability

The client/server availability model adopted here uses data on outage causes to calculate annual user outage minutes. The model is predictive. Given a specific network configuration and statistics for individual outage causes, the model predicts annual user outage minutes due to outages at each network component. The model is illustrated by applying the outage data from sources described earlier to the representative database-server architecture. The formula for calculating annual user outage minutes resulting from a given outage cause is given in Figure 2.

Table 6 shows the calculation of annual user outage minutes for four types of outages. In the table, statistics for annual outage minutes are derived from the outage sources cited earlier. Values in column 3, "Total number in c/s environment," are based on the representative client/server environment of Figure 1 and the component causing an outage. "Total annual outage minutes," column 4, is the product of columns 2 and 3. Column 5, "Number of clients affected," is the number of clients ultimately affected by an outage. "Annual user outage minutes," column 6, is the product of columns 4 and 5.

The full procedure for calculating annual user outage minutes for a given outage cause can be illustrated with reference to Table 6. For LAN cards, vendors quote a mean time between failures (MTBF) of 300,000 hours. To service the failure, user data shows a mean time to repair (MTTR) of three hours. Thus, on average, a LAN card causes one hour of outage for every 100,000 hours of operation.

Given 525,600 minutes in a year, the calculation of expected annual outage minutes for a LAN card is:

Annual outage minutes per LAN card
= 1/100,000 x 525,600 min/yr
= 5.3 min/yr

Thus, for an individual LAN card, there is an expected annual outage time of about five minutes.

Based on user data, about 20 percent of LAN-card failures are babbling-node (propagating) failures. Thus, approximately one minute of annual LAN-card outage time is for babbling-node failures (row 3 in Table 6) and four minutes are for nonpropagating failures (rows 1 and 2).

On average, each LAN card in a hub can support 10 clients. Since there are 1000 clients in the representative client/server environment, there are 100 client cards in hubs on the LAN. Total annual outage minutes (column 4 in Table 6) is 4 x 100, or 400 minutes. Finally, since 10 clients are affected by each nonpropagating client LAN-card failure, 400 x 10, or 4,000 annual user outage minutes are due to hub client LAN-card failures.

Results from the Model

Statistics in this and the following sections result from applying the predictive model to the representative client/server environment.[2] Unless otherwise specified, server data is for non-fault-tolerant servers. Table 7 provides summary statistics.

As Table 7 shows, the model predicts a total of 12,031,800 annual user outage minutes in the representative client/server environment. With 1,000 clients, there are approximately 12,000 annual outage minutes, or 200 hours of annual outage time, per client. This is equivalent to 97.7 percent user availability. This level of user availability is in accord with availability figures from other sources (see Wood, 1994). Table 7 shows that unplanned outages account for approximately 68 percent of total outage time and planned outages (reconfiguration) account for approximately 32 percent of total outage time. These figures are also similar to other availability findings.

Figure 3 shows the proportion of total outage time attributable to client, server, and network outages, and to operations or environmental outages not specific to a single type of equipment. An example of a nonspecific outage is a power failure (environmental outage) that causes all equipment in a building to shut down. The most significant finding is that server outages account for almost two-thirds of all user outage minutes. Another significant finding, in view of the common perception that the LAN is always to blame for outages, is that network outages only account for about 10 percent of total user outage minutes.

Figure 4 shows the percentage of annual user outage minutes attributable to each outage category defined in Table 2. Each category is subdivided into outage percentages for server, client, network, and all. In the figure, "All" refers to events such as power failures or building moves that affect all types of equipment.

Note that design outages account for 39 percent of user outage minutes, reconfiguration outages account for 32 percent, and physical outages account for only about 11 percent of all outage minutes. Thus, although users typically perceive outages as hardware failures, applying the predictive model to the representative client/server environment shows hardware outages to play a relatively minor role compared to other outage categories. This is the case even though the data does not include fault-tolerant servers or other fault-tolerant components. Reducing outage times by adding fault-tolerant components is described in Part 2.

[2]Wood (1994) shows the derivation of annual outage minutes for each of the outage causes listed in Table 4.

Figure 3

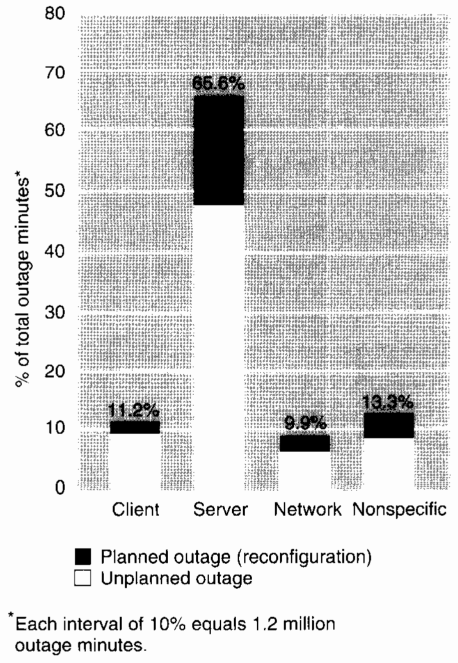


Figure 3.
Proportion of total outage time attributable to different types of equipment outages.

Figure 4

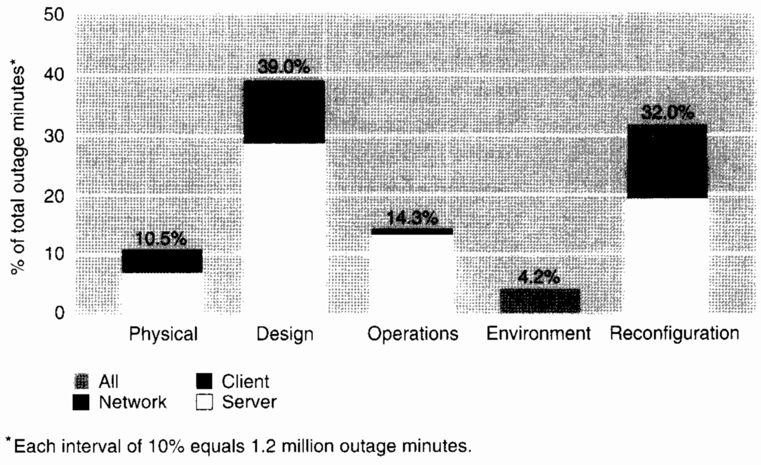


Figure 4.
User outage minutes by outage category.

Part 2: Improving Availability in a Client/Server Environment

Client/server availability can be improved by

- Decreasing the frequency of outages (increasing the time between outages).
- Decreasing the duration of outages (decreasing detection, repair, and recovery times).
- Decreasing the number of users affected by an outage.

It is not possible to simply "throw hardware" at the problem. For example, if the number of client LAN cards is doubled without adding fault tolerance, so that the number of users affected by a LAN card failure is cut in half, the expected number of LAN card failures doubles, because there are now twice as many cards. Therefore, the net effect of doubling the LAN cards is zero. There are, however, a number of other approaches to improving availability, as described in the following sections.

Improved availability provides a savings in user downtime, but may also involve a certain cost, such as the expense of providing an uninterruptible power supply. The predictive model and representative client/server environment can show how much a specific enhancement or new feature is likely to reduce user outage time. The overall cost savings from an outage reduction will vary from site to site. In evaluating methods for improving availability, sites need to carry out their own analyses of costs and benefits.

Operational Improvements

Operations errors such as improper installation of hardware or software, incorrectly entering network addresses, and accidentally or incorrectly halting system or user processes account for just over 14 percent of user outage minutes. Although users other than operations personnel frequently contribute to such outages, improving the effectiveness of operations staff would significantly reduce outage times.

As a useful start toward improving availability, whenever there is an outage, operations can attempt to identify its cause, time its duration, calculate user outage minutes, and record the information in a log. The logged information can then be used in setting priorities and planning ways of minimizing outage times.

Improved Training and Tools. In the course of researching client/server availability, there were many discussions with operations personnel. Operations staff members felt that with improved training and tools they could more quickly detect problems and their causes, make repairs, and carry out recovery procedures. Since there is no data on the subject, it is difficult to predict the number of user outage minutes that might be saved through improved training or tools. However, if better training and tools could reduce the average duration of outages by 10 percent, this would be equivalent to a 10 percent reduction in total outage times, a savings of over 1,200,000 annual user outage minutes.

An interesting and important finding from the model is that propagating failures account for over a third of user outage minutes. If an operations staff could be trained in ways to find and localize the effects of a fault, it might be able to significantly reduce user outage times due to propagating failures. For example, if operations personnel can quickly determine the source of a babbling node or broadcast storm, they can temporarily remove it from the network and prevent the outage from affecting large numbers of other nodes.[3]

[3]Tools that allow an operator to view network performance significantly improve the operator's ability to troubleshoot network problems. An example of such a tool is the Ungermann-Bass® NetDirector® product.

Testing to Ensure Successful Restarts After an Outage. Equipment recovery problems following an outage often increase the outage time. Approximately 2.5 percent (300,000 minutes) of user outage time in the representative client/server environment is directly attributable to equipment that does not reset properly following a power outage. Advance power-cycle testing of equipment to make sure it will restart after an outage can help to reduce outage durations. Restarting after recovery from other kinds of outages can also be a problem. One way to reduce such outages is to test systems in advance using simulated failure conditions. Often this can be done by testing a few components offline without affecting normal operations.

Improved Cable Layout and Problem Tracing. Cabling is a frequent source of problems in a client/server environment. About 1 percent of user outage minutes in the representative LAN are directly attributable to cable problems caused by operations personnel, such as accidently disconnecting power cords. However, other problems may be created by the difficulty of tracing cables through the spaghetti that often exists in LAN closets. In many cases, improving cable layout would make it possible for operators to trace problems more quickly and thus reduce recovery times. Improving cable layout may require expenditures for added hubs or other hardware and for additional personnel to trace and mark cables and possibly redo the existing cable layout.

Adding Fault-Tolerant Components

Adding fault-tolerant components to the representative LAN could reduce user outage times by as much as 70 percent. Maximum fault tolerance would require fault-tolerant servers, fault-tolerant clients, fault-tolerant client and server LAN connections, and a fault-tolerant LAN.

Fault-Tolerant Servers. The model showed non-fault-tolerant servers to be the primary source of client/server outages, accounting for almost two-thirds of lost user time in the representative client/server environment. Fault-tolerant servers can provide a major reduction in user outage times. Outage data for Tandem™ fault-tolerant servers has been gathered since 1992. Based on this data, if all servers in the representative client/server environment were fault-tolerant, server outage times would be reduced by a factor of six and total annual user outage times would be cut in half, dropping from over 12 million minutes to under 5.5 million minutes.

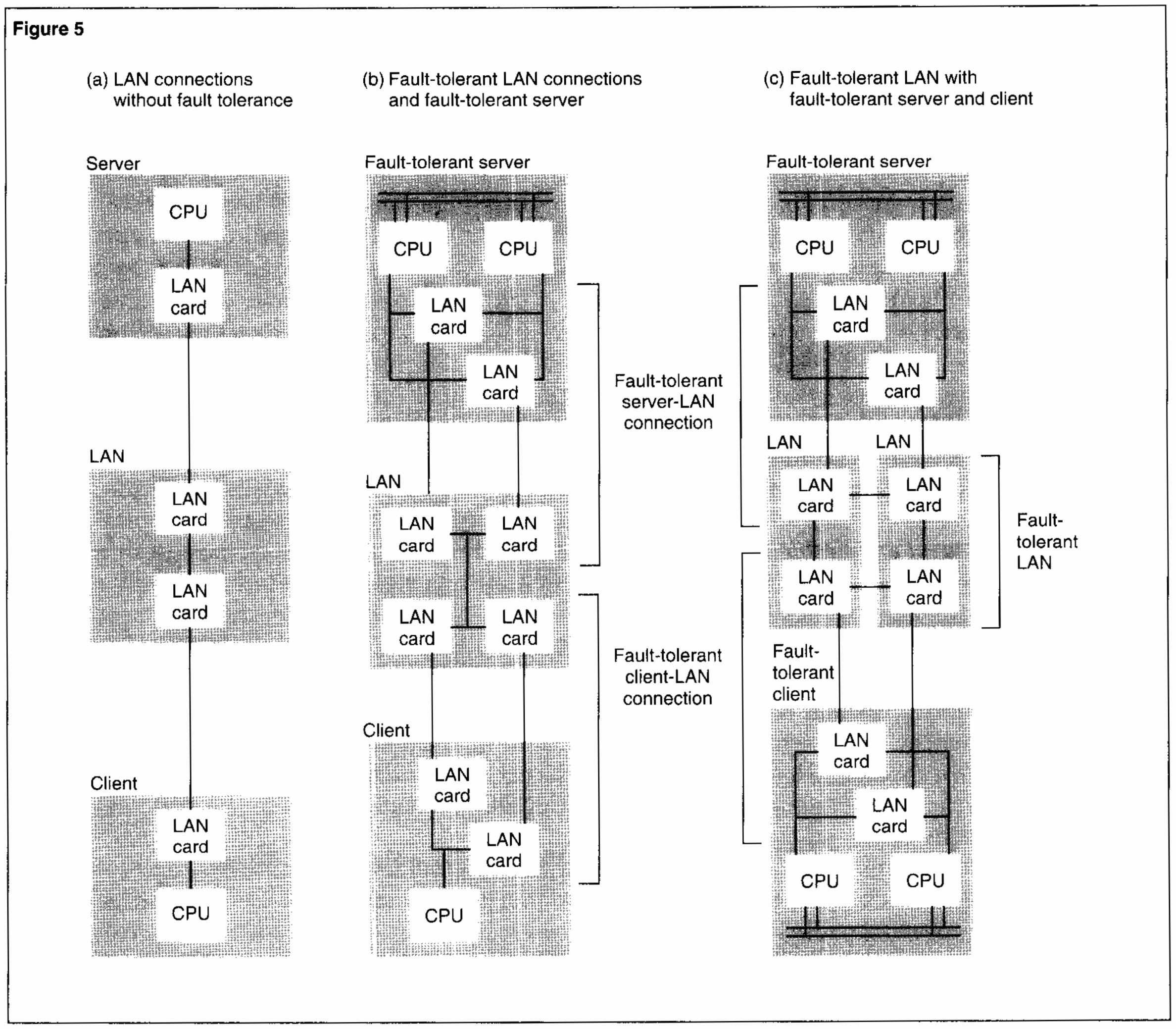
The improvement in user outage times provided by fault-tolerant servers can be understood by referring to Figure 4, which shows the proportion of server outages in each of the basic outage categories. Fault-tolerant servers almost entirely eliminate physical server outages because of the redundancy they provide. Their software fault tolerance significantly reduces server design outages, since most design bugs are transient and cause a failure in only a single process or processor. Operations outages are reduced because stopping a single process or processor, whether by mistake or for purposes of testing, does not cause a system outage. In addition, the ability to reconfigure Tandem servers online can considerably reduce reconfiguration outage times.

Fault-Tolerant LAN Connections. Fault-tolerant server-to-LAN and client-to-LAN connections can reduce user outage times by approximately 3 percent, or 362,000 minutes in the representative LAN.

A fault-tolerant server-to-LAN connection effectively eliminates user outage minutes due to LAN card failures in the server and in the hub or other server-to-network connection point. In the representative client/server environment, fault-tolerant server-to-LAN connections would result in a savings of 90,000 user outage minutes per year.

Figure 5.

Types of client/server LAN connections.



In a fault-tolerant client-to-LAN connection, dual-ported or redundant LAN cards in clients can take advantage of redundant paths through the network. This eliminates outage minutes due to LAN-card failures in the client and in the server-to-client connection. Fault-tolerant client-to-LAN connections can be an inexpensive alternative to using fault-tolerant PCs or client workstations. In the representative client/server environment, fault-tolerant client-to-LAN connections would result in a savings of 272,000 user outage minutes per year.[4] Figure 5a shows a client/server configuration with no fault-tolerant components. Figure 5b shows a configuration with fault-tolerant servers and fault-tolerant LAN connections.

[4]Recently announced Tandem and Ungermann-Bass NonStop™ Access for Networking products provide the hardware and software necessary for fault-tolerant server-to-LAN and client-to-LAN connections.

Fault-Tolerant LAN. The most effective way to reduce network outages is to make a LAN fault-tolerant. This will almost completely eliminate user outage times due to the failure of network equipment and network reconfigurations. In addition, it can greatly reduce outage times resulting from propagating failures such as broadcast storms and babbling nodes. Implementing a fault-tolerant LAN requires that there be (1) multiple, independent paths between all clients and servers and (2) software that can automatically detect the loss of a server-to-client connection and switch to an alternative path without user intervention.[5] If the representative LAN were fault tolerant, it would save 1,197,000 user outage minutes, almost 10 percent of total user outage minutes throughout the LAN. Figure 5c shows a fault-tolerant LAN configuration.

Fault-Tolerant Clients. Fault-tolerant clients can protect against most client hardware failures, including disk and power failures. Adding fault-tolerant clients to the representative client/server environment would save 388,000 user outage minutes.[6]

Currently, fault-tolerant clients are not designed to protect against software failures or to allow online reconfiguration. Because fault-tolerant clients are expensive, a reasonable alternative is to have extra clients available in a hot standby mode.

Reestablishing Interrupted Client Sessions

The most common client failure mode is a transient error in the client, network, or server that forces the user to reboot or restart an application. By itself, this generally takes at most a few minutes, but the user then has to reestablish the session and ascertain the status of any outstanding transactions. If the client lacks information about the interrupted session, reestablishing the session can take several minutes. Once the session is reestablished, the client must determine the status of transactions that were in process when the session was disrupted.

If both client and server maintain a transaction log and other information necessary for reestablishing a client session, downtime can be reduced significantly. Once the client reboots and logs in, the server can reestablish the session, provide the status of client transactions, and resume processing for the client. Such measures can reduce user outage times by 2.4 percent, or 292,000 minutes in the representative client/server environment.

Uninterruptable Power Supplies

Environmental outages are dominated by commercial power failures. The simplest way to protect against such outages is to provide uninterruptable power supplies (UPSs) and to improve site power filtering and conditioning where possible. Introducing UPSs into the representative client/server environment could reduce user outages by 389,000 minutes, a 3 percent reduction in total user outage minutes. A problem introduced by client/server computing is that site consolidation makes it easier to protect against power outages, but the nature of client/server environments tends toward site expansion.

Online Reconfiguration for System Moves

The model predicts about 777,000 user outage minutes for the representative client/server environment due to massive reconfigurations such as system moves. A fault-tolerant server, network, and client architecture can help reduce this downtime if it can be configured as two independent client/server systems. Conceptually, this allows half the client/server hardware to be moved to a new site while the other half continues to run the application. The hardware at the new site can then be configured to run required applications while the remaining equipment is moved. This could provide a 7 percent reduction in total user outage minutes.

[5]Recently announced Tandem and Ungermann-Bass NonStop Access for Networking products include the hardware and software necessary for a fault-tolerant LAN.

[6]Here and in Table 8, data for fault-tolerant clients is based on fault-tolerant PCs.

Table 8.
Methods for reducing user outage minutes.

Feature	Minutes saved (in 1000s)*	Percent of total outage time**	Comments
Fault-tolerant Tandem servers	6,542	54.4%	Based on Tandem data
Fault-tolerant server-LAN connections	90	0.7%	Reduces server-LAN card outages by 90%, babbling node by 50%. Reduces hub-card and router-card failures by 15%.
Fault-tolerant client-LAN connections	272	2.3%	Reduces client and network LAN-card failures by 50%, and babbling node outages by 50%.
Fault-tolerant LAN	1,197	9.9%	Reduces most network failure modes by 90-99%, router algorithm conflicts by 75%, broadcast errors by 50%, operations cable accidents by 90%, operations testing by 50%.
Fault-tolerant clients (PCs)	388	3.2%	Reduces client (PC) hardware outages by 95%.
Reestablish interrupted client session	292	2.4%	Reduces client OS and application outage times by 75% through faster recoveries.
Uninterruptable power supplies	389	3.2%	Reduces power outages by 90%.
Online reconfiguration for system moves	777	6.5%	Reduces user outage time due to system moves by 90%.

* Number of user outage minutes saved by the feature, based on a total of 12,032,000 user outage minutes for the representative LAN.
** Reduction in total user outage minutes, expressed as a percentage.

Summary of Methods for Improving Availability

With the exception of operational improvements, Table 8 lists each of the methods for improving client/server availability and its potential effect on user outage minutes. In each case, the savings in user outage time provided by a feature should be weighed against the possible cost of implementing the feature. Operational improvements are not included in the table because they are not associated with specific reductions in user outage times.

Conclusion

The most useful way to evaluate client/server availability is in terms of user outage times. Using outage data from a number of independent sources, a predictive model of client/server availability generated statistics on annual user outage minutes for a representative client/server environment. Outages were categorized as physical, design, operations, environmental, or reconfiguration. Results were presented by equipment type and outage category.

The major finding concerning equipment type was that non-fault-tolerant servers accounted for almost two-thirds of all user outage minutes in the representative client/server environment. Based on data for Tandem servers, it was found that fault-tolerant servers could reduce server outages by a factor of six and cut total user outage times in half. The findings for outage categories showed that design outages and planned (reconfiguration) outages accounted for the great majority of user outage minutes. Design outages were responsible for 39 percent of user outage minutes, reconfiguration outages were responsible for 32 percent of user outage minutes.

A number of approaches to increasing client/server availability were presented. These included operational improvements, the use of fault-tolerant servers and fault-tolerant LAN connections, uninterruptable power supplies, and the reestablishment of client sessions following a temporary break.

Acknowledgments
The author would like to thank Tim Chou for his encouragement and support in both the development of the client/server availability model and the documentation of results. The author also thanks the reviewers of this article for their many constructive comments.

References

Bowen, C. October 1992. Study by 3M Focuses on PC Data Loss. *Online Today.*

Caginalp, E. G. February 1992. The Lowdown on Downtime. *CRN Extra.*

Caldwell, B. June 1992. Program Lifts Grounded Users. *Information Week.*

Feather, F. 1992. *Fault Detection in an Ethernet Network via Anomaly Detectors.* Doctoral Thesis. Carnegie Mellon University.

Lee, I., et al. 1993. Measurement and Analysis of Operating System Fault-tolerance. *IEEE Transactions on Reliability.* Vol. 42, No. 2.

Saal, H. March 1990. LAN Downtime: Clear and Present Danger. *Data Communications.*

Wood, A. March 1994. *NonStop Availability in a Client/Server Environment.* Tandem Technical Report No. 94.1. Tandem Computers Incorporated. Part no. 106404. This report is available from Tandem Computers, Inc., Corporate Information Center, 10400 North Tantau Ave., MS 248-07, Cupertino, CA 95014-2599.

Alan Wood joined Tandem in May 1990. He is currently part of the Reliability Engineering group and works extensively on the NonStop Availability Initiative and the Client/Server Initiative. His specialty is mathematical modeling and cost-benefit analyses of computer system reliability and availability.

Automating Call Centers With CAM

Many call centers are turning to automation to improve productivity, profitability, and customer service. (A call center is any business center that handles the majority of its work over the phone. A good example is an airline reservation center.)

To automate a call center, however, one must have intimate knowledge of the telephone switch and the protocol it uses to communicate with the computer. Making the transition from the traditional call center to an automated call center can be costly and time-consuming.

The Tandem™ Call Applications Manager (CAM) software provides computer-telephone integration (CTI), linking applications on Tandem NonStop™ systems with telephone switches. CAM reduces the time it takes to integrate existing call center applications, or write new ones, through its simplified application program interface (API). This switch-independent API command set allows users to tailor their application's architecture to meet their specific needs. Users can focus on the application instead of worrying about CTI protocol and how to manage it.

This article discusses the role CAM plays in an automated call center. It includes two scenarios that describe how CAM interacts with an application to handle incoming calls. It then explains the basic call center functions provided by CAM and the features and benefits CAM offers to programmers developing call center applications. Finally, the article explores the architectural choices developers can make when building call center applications with CAM.

Benefits of Using CAM

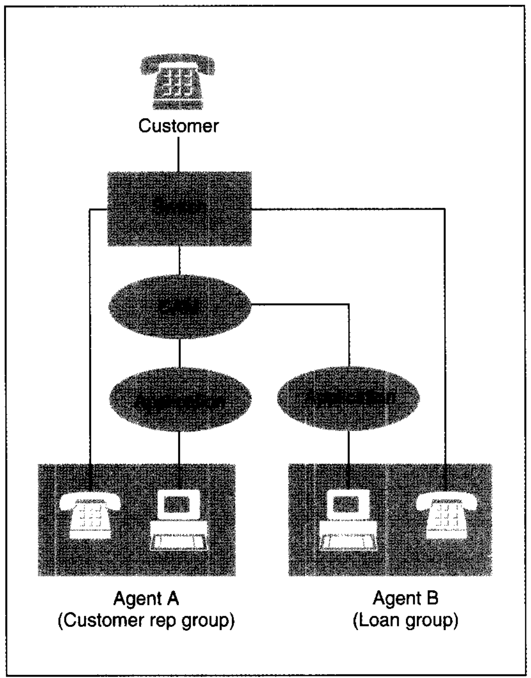
In a traditional call center, the telephone agent receives a call and then has to use the terminal keyboard to retrieve information related to the customer's call. This can take many seconds. CAM can automatically retrieve a caller's phone number from the telephone switch and pass it to the application. By using CAM, a call center application can deliver information about a customer to an agent's terminal screen at the same instant the agent receives the customer's phone call. If a customer requires special information or service, CAM can route the call, along with the accompanying account information, to a second agent.

CAM also supports integrated voice response (IVR) applications, which use recorded or automated voice response units (VRUs) to offer the caller a menu of services. An IVR application can use CAM to route calls from the recorded voice menu to a live agent. Moreover, an application using CAM can automatically make outbound calls for agents, which helps agents work productively when they are not receiving incoming calls.

Scenario 1: A Customer Requests Two Services From a Call Center

In this scenario, a bank's automated call center provides two different services for a customer. The customer owns a small business and uses this bank for all his business needs.

1. The customer phones the bank's call center from his business and gets connected to the first available agent in the general customer representative group (Agent A). The switch also notifies CAM of the call just sent to Agent A. This message contains, among other information, the customer's phone number, provided by automatic number identification (ANI).
2. CAM passes this information to the application, which uses the ANI to find the customer's account information and display it on Agent A's terminal screen at the same time that Agent A answers the phone.
3. The customer asks Agent A some questions about deposits he made last week. Agent A can help him right away because the customer's account information is already on the terminal screen.
4. The customer then asks about a new business loan advertised by the bank. Loans being a specialty item, Agent A uses the terminal keyboard to transfer the customer to the loan department.
5. The application sends a transfer-request message to CAM, together with a data buffer that contains the customer's account information and his request for information about the business loan.
6. CAM saves the transfer buffer and sends the transfer request to the switch.
7. The switch sends the call to an agent in the loan department (Agent B) and also notifies CAM of the call transfer.
8. CAM determines that the call offered to Agent B is a transfer call from Agent A. CAM attaches the original data buffer and passes it to the application used by Agent B.
9. The application displays the customer's account information and a history of the previous transactions on Agent B's terminal screen. Agent B can now speak to the customer knowing who he is, what has just transpired, and that he wants information about the new business loan.



Thus, CAM makes it possible for a call center to provide enhanced services to customers. CAM also helps reduce the cost of maintaining a call center. The two scenarios on the following pages show how an automated call center using CAM produces both tangible savings and other benefits.

Many call centers pay for incoming calls by offering 800-number services. By displaying customer information automatically, the call center application can save the agent many valuable seconds. With the account information already on the screen, the agent can dispense with basic questions and serve the customer promptly and efficiently.

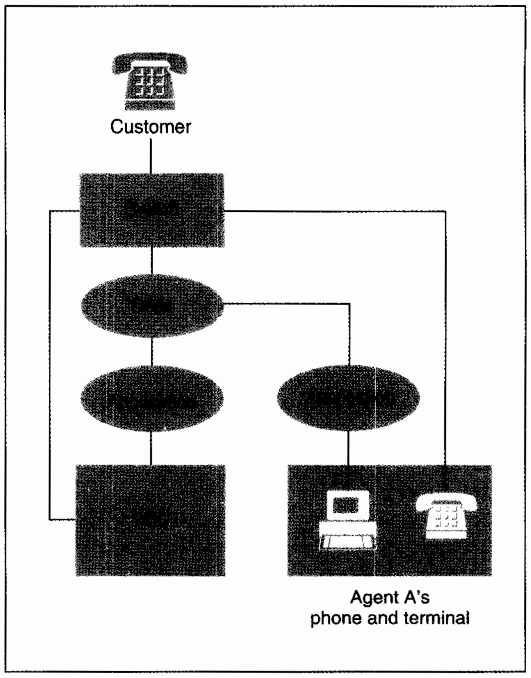
More time is saved when a call is transferred to another agent. The second agent immediately knows what conversation took place with the previous agent and what the customer wants. These time savings translate into dollar savings in the phone bill. Moreover, fewer agents can handle more phone calls, and the agents can provide more personable service when they have immediate access to customer information.

Scenario 2: An Automated Call Center Using a Voice Response Unit

In this scenario, a bank's automated call center uses a voice response unit (VRU) to take incoming calls. The customer calls in and has a dialog with a VRU. The VRU collects information from the customer and transfers the call to a live agent, who offers further help.

1. The customer phones the call center and gets connected to the VRU. The switch notifies CAM that a call has been sent to the VRU. CAM then passes the information to the application.
2. The customer and the VRU have a dialog. The customer obtains her account balance from the VRU after entering her account number and personal identification number (PIN).

   After completing this transaction, the customer asks to speak to a customer representative. She wants information about a home equity loan.
3. The application controlling the VRU sends a request to CAM to transfer the call to the loan department. The data buffer attached to the transfer request contains information about the customer's account and the transaction that just took place.
4. As in Scenario 1, CAM sends the transfer request to the switch, which transfers the call to Agent B in the loan department and notifies CAM. CAM sends the accompanying information to the application used by Agent B, which displays it on Agent B's terminal screen.



By the time the call is transferred, Agent B knows who the customer is and is ready to discuss the home equity loan with her.

Automated Call Centers Using CAM

An automated call center integrates the telephone (switch) and the terminal screen (host computer) through the CTI link. Automation allows the agent to manipulate telephone functions from the terminal keyboard as well as receive detailed call information passed from the switch.

CAM is an application software product that provides the link between the telephone switch receiving calls and the application that manipulates customer information related to those calls. In particular, CAM performs the following functions:

- Maintains the communication link to the switch.
- Facilitates the transfer of data or sessions screens to other agents.
- Routes messages to the appropriate application processes.
- Provides access to switch-specific features such as logging an agent in and out and making an agent's phone available or unavailable.
- Handles the application protocols of different switches.
- Converts messages from different switches to a common format.
- Supports interfaces to the Tandem NonStop Kernel operating system and the Tandem Pathway transaction processing environment.

The CTI protocol allows the switch to pass call information such as the calling and dialed numbers to CAM. CAM selects the appropriate application program for handling the call and passes on the information. The application obtains additional data required to handle the call and sends it to the agent's terminal screen. The application can use Pathway software to access, format, and deliver data to an agent's terminal screen. One can also use a client application on an agent's workstation to deliver screen data. Figure 1 shows a diagram of an automated call center using CAM.

Using Phone Numbers to Customize Call Processing

When a switch connects an incoming call, the call center application uses the information received from CAM to customize call processing. If the telephone network provides automatic number identification (ANI) or calling line identification (CLID), CAM relays the calling number to the application, so caller-specific data can be retrieved from the application database and custom scripts delivered to the agent.

CAM can also obtain the number dialed by the caller through the Dialed Number Information Service (DNIS), provided by the telephone network. CAM passes this number to the application, which can use it to identify the nature of the call and handle the call accordingly. Using the DNIS, the application not only delivers the call to the appropriate agent group, but also displays an appropriate menu on the agent's terminal screen.

The application can also use DNIS information to allow a single agent to handle multiple functions independently and efficiently. The application can deliver different functional screens to the agent's terminal, based on the DNIS number. As a result, the agent instantly knows which service the customer wants and greets the customer appropriately.

To help monitor call center activity, CAM identifies and tracks all calls. CAM informs the relevant application when each call is connected, transferred, disconnected, or abandoned. In addition, telemarketing functions can be enhanced with the make-call feature that can be invoked from the terminal keyboard. The host-enhanced call routing feature also allows the application to dynamically control to which agent the call should be offered.

Figure 1

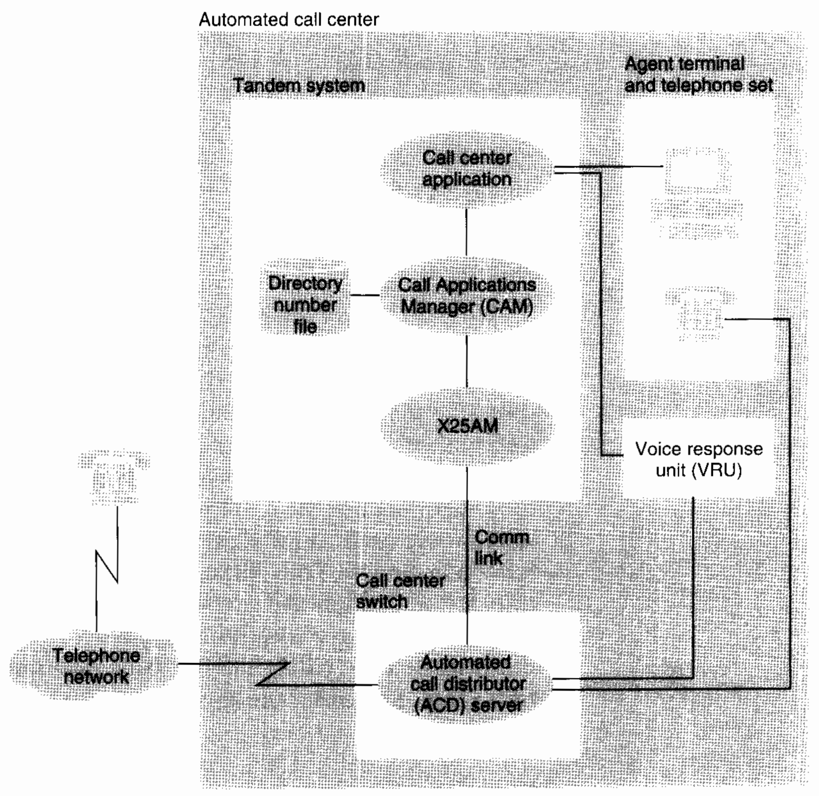

Automated call center
Tandem system
Agent terminal and telephone set
Call center application
Directory number file
Call Applications Manager (CAM)
X25AM
Voice response unit (VRU)
Comm link
Call center switch
Telephone network
Automated call distributor (ACD) server

Figure 1.
An automated call center using CAM.

IVR Applications

IVR applications can use CAM to perform all the same functions as applications that support live agents. CAM notifies IVR applications of call status events and allows the applications to control switch functions. Synchronized delivery of voice calls and call data over the CTI link offers IVR applications enhanced flexibility in call control. For example, an application can customize call handling for different callers or dialed numbers, allowing the VRU to respond to a call in a specific language, with a special script, or using customized processing logic.

Coordinated Voice and VRU Data Transfer

The application must coordinate voice and data transfer when a call is screened by a VRU before being transferred to an agent, or when a live agent must escalate a call to a supervisor or transfer it to another agent in a different department. For customer satisfaction and agent efficiency, a voice-VRU coordinated application should provide to the receiving agent the data already obtained from the caller (either by a VRU or by an agent).

When a call is transferred from a VRU to an agent, CAM coordinates with the IVR application and the switch to ensure that all data is routed to the application controlling the agent's terminal. If the call is transferred to a second or third location, CAM ensures that all collected data accompanies the call by directing it to the appropriate application delivering data to the receiving agents. Thus, the agent answering a transferred call has the necessary background information to go to work immediately. Agents don't waste time collecting the same information again, and caller problems or requests are resolved faster.

Host-Enhanced Routing

With CAM, a call center application on the Tandem system can help select the appropriate destination for incoming calls. CAM then passes routing commands to the switch.

Before an incoming call is connected by the switch, CAM provides information about the call to the application, which can determine where to route the call. This determination can be based on the characteristics of available agents and information provided with the call. As well as making call routing more efficient, this feature allows the application to customize routing for specific callers such as those with special language needs or those requiring priority handling.

Automated Outbound Dialing

Through CAM, a call center application can make outbound phone calls automatically. By using CAM, the application can ask the switch to place a call to an internal or external location. CAM gives the application access to other outbound functions such as speed dialing or preview dialing that enhance agent productivity and eliminate dialing errors. In addition, the application can track successful calls and save abandoned or incomplete calls so that agents can automatically call back the unreached customers during less busy hours. Agents can make calls by selecting items from a menu; they don't have to dial the numbers manually.

Automated Control of Agent Status

CAM offers agents a single point of access to automated call center functions. Agents can log into or out of the switch through the terminal keyboard. Without using the telephone set, they can automatically make themselves unavailable for call connection after they terminate a call. This simplifies agent activity and reduces errors, while providing time for activities such as wrapping up a call. Agents can thus be more effective; there is no risk of getting an incoming call during call wrap-up. Moreover, agents can choose to be made available automatically when the application determines that they are ready to take a new call.

Application Development Features

CAM offers several benefits for call center application developers. By maintaining the CTI link and handling session-management functions (such as logging into the switch to request the use of CTI services and registering agents), CAM simplifies the application and allows developers to concentrate on the call handling. CAM translates call information received over the CTI link into a simpler and more uniform format, presenting the formatted information to the application. Developers can also choose to request the call-related information in its raw format to get any information they wish.

In addition to supporting ease of development, the CAM API supports switch independence. By allowing a single application to communicate with multiple telephone switches, CAM can increase the flexibility of the application and further reduce development costs.

Application Design

Applications using CAM can be written as multiple programs. Developers can write each application function as a separate program that can tell CAM to deliver the appropriate messages to it. For example, Process A can handle enhanced routing functions, while Process B handles status events. One can even separate the general call handling by agent groups that handle different functions. Thus, a separate process would manage each agent group. (See Figure 2.)

CAM determines where to deliver specific messages by means of entries in the Directory Number (DN) file. (Developers must provide these entries.) CAM uses the DN file to determine where to send route-request messages, status events, and call information for every agent. Developers can choose whether to use a single program or multiple programs in their application design.

Message-Based Interface

To promote ease of development, the CAM API is a message-based interface with fixed-length messages and common fields in the same location for all messages. For example, every message begins with the message type, message length, and return-code fields. All switch-specific parameters are located at the end of the message. The API uses fixed-length messages to make writing Pathway applications simpler and more efficient. By having common fixed-length messages, the Pathway application can allocate a single buffer for its receive buffer instead of multiple buffers for the different-sized messages.

The API provides an optional user header and CAM identifier to reduce the need to change an existing application. The user header gives developers some flexibility in integrating CAM into an existing application. The user header allows the application to identify the CAM message type by using the preexisting methods, which look for the message type in a particular offset from the beginning of the message. Similarly, the optional CAM identifier gives developers an easy way to distinguish between CAM messages and messages from another program.

Figure 2

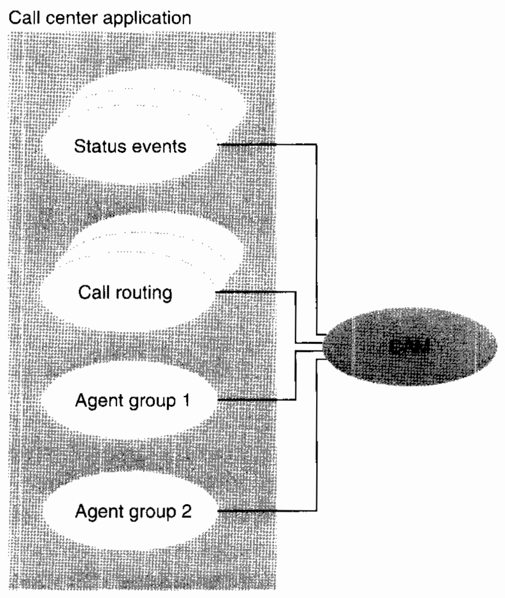


Figure 2.
A call center application consisting of multiple programs.

Table 1.
Switches supported by CAM.

Manufacturer	Switch	CTI link
Aspect	CallCenter ACD	ApplicationBridge
AT&T	Definity G3	CallVisor ASAI
Ericsson	MD110	ApplicationLink
Northern Telecom	Meridian 1	Meridian Link
Northern Telecom	DMS100	Compucall
Northern Telecom	SL100	SL100 Meridian SCAI
Rolm	9751	CallBridge CSTA
Siemens	Hicom 300	CallBridge ACL

Switch Independence

The CAM API is a switch-independent command set. CAM currently supports seven different switches, each of which behaves differently in many ways. CAM hides most of these differences from the application, which provides several advantages for writing an application. Some call centers have multiple telephone switches, which may come from different manufacturers. With the common interface provided by CAM, users can change to a different switch without having to change the application. Table 1 lists the switches supported by CAM.

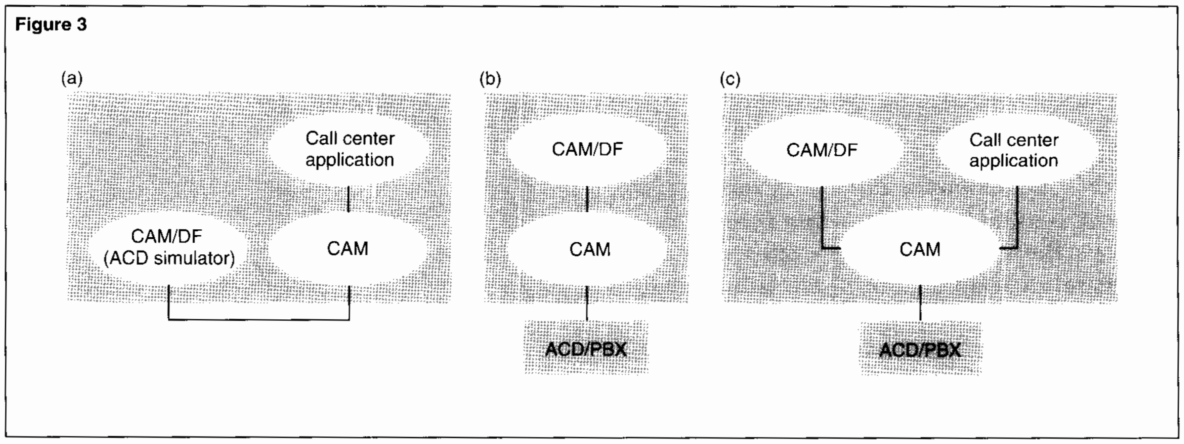


Figure 3.
Three uses of the CAM/DF feature. a) CAM/DF used as a switch simulator. b) CAM/DF used as an application simulator. c) CAM/DF supplementing some functions of the application.

The Tandem Alliance partners that write call center applications can benefit especially from the switch-independent command set, since they have to write applications that communicate with all the switches. Using CAM will reduce costs for the Alliance partners; they can pass on some of these savings to Tandem users in the form of simpler and quicker application development.

Development Tools

The CAM product comes with a development tool kit, the CAM Development Facility (CAM/DF), which helps speed up application development time. CAM/DF contains a switch simulator and an application simulator. As a switch simulator, CAM/DF gives programmers the flexibility of developing the application without having a switch. The switch simulator reduces the complexities that arise from dealing with the details of the real switch and isolates the effort of integrating CAM with the application.

Conversely, one can use the CAM/DF application simulator to integrate CAM with the switch before introducing the application. The application simulator can coexist with the call center application during its development phase. The application simulator can monitor and handle some functions, while the call center application performs other functions. As programmers continue to develop the application, they can gradually eliminate the dependence on the CAM/DF. Figure 3 illustrates the uses of the CAM/DF.

Call Center Architecture

With CAM, one can design an automated call center in many ways. Some call centers have a VRU that can take incoming calls, collect information such as account numbers, and provide for the transfer of calls to live agents. The application program logic can automatically transfer a call from the VRU to an agent; the application tells CAM to transfer the call. Furthermore, the application can pass any collected data to the destination agent by attaching the data to the transfer-request message, which the application sends to CAM to initiate the transfer.

Another option is to use a switch that has a built-in, integrated VRU instead of using a VRU provided by a third party. This architecture has the advantage of using a single communication interface to the switch. Architectures that have separate VRUs need two communication interfaces, one for the CTI link and a separate one for the VRU. Having a single interface can simplify the design of the application.

Regardless of the architecture and the switch type one chooses, one needs to make only minimal changes to an existing application, since the CAM API is virtually the same for all switches. The only code one must change is the processing of the switch-specific data fields in certain messages.

Furthermore, using the CAM routing feature, one can move into the application many call-routing functions traditionally made by the switch. The advantage is that the application can dynamically control call routing.

Assume, for example, that a home-shopping call center has a promotional sale item and customers are asked to call a certain phone number to order the item. One can configure the application to route all the phone calls to this number to a particular group of agents who have been alerted to the sale. When another sales item is advertised with another phone number, one can configure the application to send those calls to another group of agents. The application can easily handle these dynamic changes in the call center. However, if the switch were to handle the call routing rather than the application, the routing changes would require more time to configure, and there would be downtime for the phones affected by these changes.

Benefits of Automating a Call Center With Tandem Computers

Telephone switches generally have been built to be reliable, with minimal down time. However, it does no good to have the telephones working if agents can't access the information in the computer because of a computer failure. Tandem's NonStop computers and software environment complement the reliability of the telephone switches by providing continuous availability and data integrity. Thus, Tandem systems are ideally suited for call center applications.

Third-Party Solutions

Tandem has partnerships, through its Alliance programs, with several companies that write call center application software. Different Alliance partners focus on addressing different call center needs. For example, some partners specialize in interacting with a VRU. Other partners specialize in telephone delivery system software, which provides solutions for customer service, automated sales support, and account management. By forming these partnerships, Tandem can provide a total business solution to a call center's requirements.

For information about Tandem partners who provide call center solutions, refer to the *Tandem Alliance 1994 Solutions & Services Directory* (1993), or its most current version.

Conclusion

By providing a simple, uniform API command set that handles computer-telephone integration (CTI), CAM can speed up the development of call center applications, thereby reducing the cost of automating a call center. With CAM, users can choose a variety of call center architectures that incorporate features such as VRUs, call routing dynamically controlled by the application, and the simultaneous delivery (or transfer) of customer information with the customer's phone call. CAM also provides switch independence, giving the application flexibility when it comes to choosing, adding, or changing telephone switches.

Thus, CAM can make it possible for a call center application to provide enhanced services to customers. In addition, CAM reduces the time it takes to handle each call, which allows fewer agents to serve the same number of customers and thus substantially lowers the cost of running a call center.

Reference

Tandem Alliance 1994 Solutions & Services Directory. 1993. Tandem Computers Incorporated. Part no. 102324.

William Choi is the project leader of the CAM development group and was one of two initial developers of the CAM software. William joined Tandem in 1989 and worked on the Integrated Services Digital Network (ISDN) project before starting on CAM.

TandemSystemsReview*Index*

The *Tandem Journal* became the *Tandem Systems Review* in February 1985. Four issues of the *Tandem Journal* were published:

Volume 1, No. 1	Fall 1983	Volume 2, No. 2	Spring 1984
Volume 2, No. 1	Winter 1984	Volume 2, No. 3	Summer 1984

As of this issue, 25 issues of the *Tandem Systems Review* have been published:

Volume 1, No. 1	Feb. 1985	Volume 5, No. 1	April 1989	Volume 9, No. 2	Spring 1993
Volume 1, No. 2	June 1985	Volume 5, No. 2	Sept. 1989	Volume 9, No. 3	Summer 1993
Volume 2, No. 1	Feb. 1986	Volume 6, No. 1	March 1990	Volume 9, No. 4	Fall 1993
Volume 2, No. 2	June 1986	Volume 6, No. 2	Oct. 1990	Volume 10, No. 1	Jan. 1994
Volume 2, No. 3	Dec. 1986	Volume 7, No. 1	April 1991	Volume 10, No. 2	April 1994
Volume 3, No. 1	March 1987	Volume 7, No. 2	Oct. 1991		
Volume 3, No. 2	Aug. 1987	Volume 8, No. 1	Spring 1992		
Volume 4, No. 1	Feb. 1988	Volume 8, No. 2	Summer 1992		
Volume 4, No. 2	July 1988	Volume 8, No. 3	Fall 1992		
Volume 4, No. 3	Oct. 1988	Volume 9, No. 1	Winter 1993		

The articles published in all 29 issues are arranged by subject below. (*Tandem Journal* is abbreviated as TJ and *Tandem Systems Review* as TSR.) A second index, arranged by product, is also provided.

Index by Subject

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
APPLICATION DEVELOPMENT AND LANGUAGES					
A New Design for the PATHWAY TCP	R. Wong	TJ	2,2	Spring 1984	83932
An Overview of Client/Server Computing on Tandem Systems	H. Cooperstein	TSR	8,3	Fall 1992	89803
An Introduction to Tandem EXTENDED BASIC	J. Meyerson	TJ	2,2	Spring 1984	83932
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Automating Call Centers With CAM	W. Choi	TSR	10,2	April 1994	104398
Basic Uses and New Features of Extended GDS	A. Hotea	TSR	10,1	Jan. 1994	104396
Debugging TACL Code	L. Palmer	TSR	4,2	July 1988	13693
Designing and Implementing a Graphical User Interface	S. Wolfe	TSR	9,3	Summer 1993	89806
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
Extending the Client/Server Model With Object-Oriented Technology	T. Rohner	TSR	10,1	Jan. 1994	104396
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
New TAL Features	C. Lu, J. Murayama	TSR	2,2	June 1986	83837
PATHFINDER–An Aid for Application Development	S. Benett	TJ	1,1	Fall 1983	83930

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
APPLICATION DEVELOPMENT AND LANGUAGES *(cont.)*					
PATHWAY IDS: A Message-level Interface to Devices and Processes	M. Anderton, M. Noonan	TSR	2,2	June 1986	83937
The RESPOND OLTP Business Management System for Manufacturing	H. Bolling, W. Bronson	TSR	9,1	Winter 1993	89804
State-of-the-Art C Compiler	E. Kit	TSR	2,2	June 1986	83937
TACL, Tandem's New Extensible Command Language	J. Campbell, R. Glascock	TSR	2,1	Feb. 1986	83936
Tandem's New COBOL85	D. Nelson	TSR	2,1	Feb. 1986	83936
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804
The ENABLE Program Generator for Multifile Applications	B. Chapman, J. Zimmerman	TSR	1,1	Feb. 1985	83934
TMF and the Multi-Threaded Requester	T. Lemberger	TJ	1,1	Fall 1983	83930
Writing a Command Interpreter	D. Wong	TSR	1,2	June 1985	83935
CLIENT/SERVER					
An Overview of Client/Server Computing on Tandem Systems	H. Cooperstein	TSR	8,3	Fall 1992	89803
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Client/Server Availability	A. Wood	TSR	10,2	April 1994	104398
Designing and Implementing a Graphical User Interface	S. Wolfe	TSR	9,3	Summer 1993	89806
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
Extending the Client/Server Model With Object-Oriented Technology	T. Rohner	TSR	10,1	Jan. 1994	104396
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804
DATA COMMUNICATIONS					
An Overview of SNAX/CDF	M. Turner	TSR	5,2	Sept. 1989	28152
A SNAX Passthrough Tutorial	D. Kirk	TJ	2,2	Spring 1984	83932
Basic Uses and New Features of Extended GDS	A. Hotea	TSR	10,1	Jan. 1994	104396
Changes in FOX	N. Donde	TSR	1,2	June 1985	83935
Connecting Terminals and Workstations to Guardian 90 Systems	E. Siegel	TSR	8,2	Summer 1992	69848
Expand High-Performance Solutions	D. Smith	TSR	9,3	Summer 1993	89806
Introduction to MULTILAN	A. Coyle	TSR	4,1	Feb. 1988	11078
Overview of the MULTILAN Server	A. Rowe	TSR	4,1	Feb. 1988	11078
SNAX/APC: Tandem's New SNA Software for Distributed Processing	B. Grantham	TSR	3,1	March 1987	83939
SNAX/HLS: An Overview	S. Saltwick	TSR	1,2	June 1985	83935
TLAM: A Connectivity Option for Expand	K. MacKenzie	TSR	7,1	April 1991	46988
Using the MULTILAN Application Interfaces	M. Berg, A. Rowe	TSR	4,1	Feb. 1988	11078

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DATA MANAGEMENT					
A Comparison of the B00 DP1 and DP2 Disc Processes	T. Schachter	TSR	1,2	June 1985	83935
An Overview of NonStop SQL Release 2	M. Pong	TSR	6,2	Oct. 1990	46987
Batch Processing in Online Enterprise Computing	T. Keefauver	TSR	6,2	Oct. 1990	46987
Concurrency Control Aspects of Transaction Design	W. Senf	TSR	6,1	March 1990	32968
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
DP1-DP2 File Conversion: An Overview	J. Tate	TSR	2,1	Feb. 1986	83936
Determining FCP Conversion Time	J. Tate	TSR	2,1	Feb. 1986	83936
DP2's Efficient Use of Cache	T. Schachter	TSR	1,2	June 1985	83935
DP2 Highlights	K. Carlyle, L. McGowan	TSR	1,2	June 1985	83935
DP2 Key-sequenced Files	T. Schachter	TSR	1,2	June 1985	83935
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
High-Performance SQL Through Low-Level System Integration	A. Borr	TSR	4,2	July 1988	13693
Improvements in TMF	T. Lemberger	TSR	1,2	June 1985	83935
NetBatch: Managing Batch Processing on Tandem Systems	D. Wakashige	TSR	5,1	April 1989	18662
NetBatch-Plus: Structuring the Batch Environment	G. Earle, D. Wakashige	TSR	6,1	March 1990	32986
NonStop SQL: The Single Database Solution	J. Cassidy, T. Kocher	TSR	5,2	Sept. 1989	28152
NonStop SQL Data Dictionary	R. Holbrook, D. Tsou	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Basic Concepts	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Query Optimization and User Influence	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Reliability	C. Fenner	TSR	4,2	July 1988	13693
Online Information Processing	J. Viescas	TSR	9,1	Winter 1993	89804
Online Reorganization of Key-Sequenced Tables and Files	G. Smith	TSR	6,2	Oct. 1990	46987
Optimizing Batch Performance	T. Keefauver	TSR	5,2	Sept. 1989	28152
Overview of NonStop SQL	H. Cohen	TSR	4,2	July 1988	13693
Parallelism in NonStop SQL Release 2	M. Moore, A. Sodhi	TSR	6,2	Oct. 1990	46987
The NonStop SQL Release 2 Benchmark	S. Englert, J. Gray, T. Kocher, P. Shah	TSR	6,2	Oct. 1990	46987
The Outer Join in NonStop SQL	J. Vaishnav	TSR	6,2	Oct. 1990	46987
The Relational Data Base Management Solution	G. Ow	TJ	2,1	Winter 1984	83931
Tandem's NonStop SQL Benchmark	Tandem Performance Group	TSR	4,1	Feb. 1988	11078
The TRANSFER Delivery System for Distributed Applications	S. Van Pelt	TJ	2,2	Spring 1984	83932
TMF Autorollback: A New Recovery Feature	M. Pong	TSR	1,1	Feb. 1985	83934
DECISION SUPPORT SYSTEMS					
Online Information Processing	J. Viescas	TSR	9,1	Winter 1993	89804
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804
The RESPOND OLTP Business Management System for Manufacturing	H. Bolling, W. Bronson	TSR	9,1	Winter 1993	89804
OBJECT-ORIENTED TECHNOLOGY					
Extending the Client/Server Model With Object-Oriented Technology	T. Rohner	TSR	10,1	Jan. 1994	104396

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
OPERATING SYSTEMS					
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
Highlights of the B00 Software Release	K. Coughlin, R. Montevaldo	TSR	1,2	June 1985	83935
Increased Code Space	A. Jordan	TSR	1,2	June 1985	83935
Managing System Time Under GUARDIAN 90	E. Nellen	TSR	2,1	Feb. 1986	83936
Migration Planning for D-Series Systems	S. Kuukka	TSR	9,2	Spring 1993	89805
New GUARDIAN 90 Time-keeping Facilities	E. Nellen	TSR	1,2	June 1985	83935
New Process-timing Features	S. Sharma	TSR	1,2	June 1985	83935
NonStop II Memory Organization and Extended Addressing	D. Thomas	TJ	1,1	Fall 1983	83930
Overview of the C00 Release	L. Marks	TSR	4,1	Feb. 1988	11078
Overview of the D-Series Guardian 90 Operating System	W. Bartlett	TSR	9,2	Spring 1993	89805
Overview of the NonStop-UX Operating System for the Integrity S2	P. Norwood	TSR	7,1	April 1991	46988
Robustness to Crash in a Distributed Data Base: A Nonshared-memory Approach	A. Borr	TSR	1,2	June 1985	83935
The GUARDIAN Message System and How to Design for It	M. Chandra	TSR	1,1	Feb. 1985	83934
The NonStop Himalaya K10000 Interprocessor Bus	R. Jardine, S. Hamilton, K. Krishnakumar	TSR	10,2	April 1994	104398
The Tandem Global Update Protocol	R. Carr	TSR	1,2	June 1985	83935
PERFORMANCE AND CAPACITY PLANNING					
A Performance Retrospective	P. Oleinick, P. Shah	TSR	2,3	Dec. 1986	83938
Buffering for Better Application Performance	R. Mattran	TSR	2,1	Feb. 1986	83936
Capacity Planning Concepts	R. Evans	TSR	2,3	Dec. 1986	83938
Capacity Planning With TCM	W. Highleyman	TSR	7,2	Oct. 1991	65248
C00 TMDS Performance	J. Mead	TSR	4,1	Feb. 1988	11078
Credit-authorization Benchmark for High Performance and Linear Growth	T. Chmiel, T. Houy	TSR	2,1	Feb. 1986	83936
Debugging Accelerated Programs on TNS/R Systems	D. Cressler	TSR	8,1	Spring 1992	65250
DP2 Performance	J. Enright	TSR	1,2	June 1985	83935
Estimating Host Response Time in a Tandem System	H. Horwitz	TSR	4,3	Oct. 1988	15748
Expand High-Performance Solutions	D. Smith	TSR	9,3	Summer 1993	89806
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938
Getting Optimum Performance from Tandem Tape Systems	A. Khatri	TSR	2,3	Dec. 1986	83938
How to Set Up a Performance Data Base with MEASURE and ENFORM	M. King	TSR	2,3	Dec. 1986	83938
Implementing a Systems Management Improvement Program	J. Dagenais	TSR	9,4	Fall 1993	89807
Improved Performance for BACKUP2 and RESTORE2	A. Khatri, M. McCline	TSR	1,2	June 1985	83935
Improving Performance on TNS/R Systems With the Accelerator	M. Blanchet	TSR	8,1	Spring 1992	65250
MEASURE: Tandem's New Performance Measurement Tool	D. Dennison	TSR	2,3	Dec. 1986	83938
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Message System Performance Enhancements	D. Kinkade	TSR	2,3	Dec. 1986	83938
Message System Performance Tests	S. Uren	TSR	2,3	Dec. 1986	83938
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
NonStop NET/MASTER: Configuration and Performance Guidelines	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop VLX Performance	J. Enright	TSR	2,3	Dec. 1986	83938
Optimizing Sequential Processing on the Tandem System	R. Welsh	TJ	2,3	Summer 1984	83933
Pathway TCP Enhancements for Application Run-Time Support	R. Vannucci	TSR	7,1	April 1991	46988

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
PERFORMANCE AND CAPACITY PLANNING *(cont.)*					
Performance Benefits of Parallel Query Execution and Mixed Workload Support in NonStop SQL Release 2	S. Englert, J. Gray	TSR	6,2	Oct. 1990	46987
Performance Considerations for Application Processes	R. Glasstone	TSR	2,3	Dec. 1986	83938
Performance Measurements of an ATM Network Application	N. Cabell, D. Mackie	TSR	2,3	Dec. 1986	83938
Predicting Response Time in On-line Transaction Processing Systems	A. Khatri	TSR	2,2	June 1986	83937
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
The ENCORE Stress Test Generator for On-line Transaction Processing Applications	S. Kosinski	TJ	2,1	Winter 1984	83931
The PATHWAY TCP: Performance and Tuning	J. Vatz	TSR	1,1	Feb. 1985	83934
The Performance Characteristics of Tandem NonStop Systems	J. Day	TJ	1,1	Fall 1983	83930
Sizing Cache for Applications that Use B-series DP1 and TMF	P. Shah	TSR	2,2	June 1986	83937
Sizing the Spooler Collector Data File	H. Norman	TSR	4,1	Feb. 1988	11978
Tandem's 5200 Optical Storage Facility: Performance and Optimization Considerations	S. Coleman	TSR	5,1	April 1989	18662
Tandem's Approach to Fault Tolerance	B. Ball, W. Bartlett, S. Thompson	TSR	4,1	Feb. 1988	11078
Understanding PATHWAY Statistics	R. Wong	TJ	2,2	Spring 1984	83932
PERIPHERALS					
5120 Tape Subsystem Recording Technology	W. Phillips	TSR	3,2	Aug. 1987	83940
An Introduction to DYNAMITE Workstation Host Integration	S. Kosinski	TSR	1,2	June 1985	83935
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Data-Encoding Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937
Data-Window Phase-Margin Analysis	A. Painter, H. Pham, H. Thomas	TSR	2,2	June 1986	83937
Introducing the 3207 Tape Controller	S. Chandran	TSR	1,2	June 1985	83935
Peripheral Device Interfaces	J. Blakkan	TSR	3,2	Aug. 1987	83940
Plated Media Technology Used in the XL8 Storage Facility	D.S. Ng	TSR	2,2	June 1986	83937
Streaming Tape Drives	J. Blakkan	TSR	3,2	Aug. 1987	83940
Terminal Selection	E. Siegel	TSR	8,2	Summer 1992	69848
The 5200 Optical Storage Facility: A Hardware Perspective	A. Patel	TSR	5,1	April 1989	18662
The 6100 Communications Subsystem: A New Architecture	R. Smith	TJ	2,1	Winter 1984	83931
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
The DYNAMITE Workstation: An Overview	G. Smith	TSR	1,2	June 1985	83935
The Model 6VI Voice Input Option: Its Design and Implementation	B. Huggett	TJ	2,3	Summer 1984	83933
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	3,2	Aug. 1987	83940
The V8 Disc Storage Facility: Setting a New Standard for On-line Disc Storage	M. Whiteman	TSR	1,2	June 1985	83935

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
PROCESSORS					
Fault Tolerance in the NonStop Cyclone System	S. Chan, R. Jardine	TSR	7,1	April 1991	46988
A Hardware Overview of the NonStop Himalaya K10000 Server	C. Kong	TSR	10,1	Jan. 1994	104396
NonStop CLX: Optimized for Distributed On-Line Transaction Processing	D. Lenoski	TSR	5,1	April 1989	18662
NonStop VLX Hardware Design	M. Brown	TSR	2,3	Dec. 1986	83938
Overview of Tandem NonStop Series/RISC Systems	L. Faby, R. Mateosian	TSR	8,1	Spring 1992	65250
The High-Performance NonStop TXP Processor Transaction Processing	W. Bartlett, T. Houy, D. Meyer	TJ	2,1	Winter 1984	83931
The NonStop Himalaya K10000 Interprocessor Bus	R. Jardine, S. Hamilton, K. Krishnakumar	TSR	10,2	April 1994	104398
The NonStop TXP Processor: A Powerful Design for On-line Transaction Processing	P. Oleinick	TJ	2,3	Summer 1984	83933
The VLX: A Design for Serviceability	J. Allen, R. Boyle	TSR	3,1	March 1987	83939
SECURITY					
Dial-In Security Considerations	P. Grainger	TSR	7,2	Oct. 1991	65248
Distributed Protection with SAFEGUARD	T. Chou	TSR	2,2	June 1986	83937
Enhancing System Security With Safeguard	C. Gaydos	TSR	7,1	April 1991	46988
SYSTEM CONNECTIVITY					
Basic Uses and New Features of Extended GDS	A. Hotea	TSR	10,1	Jan. 1994	104396
Building Open Systems Interconnection with OSI/AS and OSI/TS	R. Smith	TSR	6,1	March 1990	32986
Connecting Terminals and Workstations to Guardian 90 Systems	E. Siegel	TSR	8,2	Summer 1992	69848
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
Terminal Connection Alternatives for Tandem Systems	J. Simonds	TSR	5,1	April 1989	18662
Terminal Selection	E. Siegel	TSR	8,2	Summer 1992	69848
The OSI Model: Overview, Status, and Current Issues	A. Dunn	TSR	5,1	April 1989	18662
SYSTEM MANAGEMENT					
Configuring Tandem Disk Subsystems	S. Sitler	TSR	2,3	Dec. 1986	83938
Data Replication in Tandem's Distributed Name Service	T. Eastep	TSR	4,3	Oct. 1988	15748
Enhancements to TMDS	L. White	TSR	3,2	Aug. 1987	83940
Event Management Service Design and Implementation	H. Jordan, R. McKee, R. Schuet	TSR	4,3	Oct. 1988	15748
Implementing a Systems Management Improvement Program	J. Dagenais	TSR	9,4	Fall 1993	89807
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Network Statistics System	M. Miller	TSR	4,3	Oct. 1988	15748
NonStop NET/MASTER: Configuration and Performance Guidelines	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Management Architecture	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Processing Costs and Sizing Calculations	M. Stockton	TSR	9,4	Fall 1993	89807
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
SCP and SCF: A General Purpose Implementation of the Subsystem Programmatic Interface	T. Lawson	TSR	4,3	Oct. 1988	15748
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
SYSTEM MANAGEMENT *(cont.)*					
Tandem's Subsystem Programmatic Interface	G. Tom	TSR	4,3	Oct. 1988	15748
Using FOX to Move a Fault-tolerant Application	C. Breighner	TSR	1,1	Feb. 1985	83934
Using the Subsystem Programmatic Interface and Event Management Services	K. Stobie	TSR	4,3	Oct. 1988	15748
VIEWPOINT Operations Console Facility	R. Hansen, G. Stewart	TSR	4,3	Oct. 1988	15748
VIEWSYS: An On-line System-resource Monitor	D. Montgomery	TSR	1,2	June 1985	83935
Writing Rules for Automated Operations	J. Collins	TSR	7,2	Oct. 1991	65248
UTILITIES					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939

Index by Product

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
3207 TAPE CONTROLLER					
Introducing the 3207 Tape Controller	S. Chandran	TSR	1,2	June 1985	83935
5120 TAPE SUBSYSTEM					
5120 Tape Subsystem Recording Technology	W. Phillips	TSR	3,2	Aug. 1987	83940
5200 OPTICAL STORAGE					
Application Profile: Storing Macintosh Graphics on the Tandem 5200 Optical Storage Facility	D. Broyles	TSR	9,3	Summer 1993	89806
Tandem's 5200 Optical Storage Facility: Performance and Optimization Considerations	S. Coleman	TSR	5,1	April 1989	18662
The 5200 Optical Storage Facility: A Hardware Perspective	A. Patel	TSR	5,1	April 1989	18662
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	4,1	Feb. 1988	11078
6100 COMMUNICATIONS SUBSYSTEM					
The 6100 Communications Subsystem: A New Architecture	R. Smith	TJ	2,1	Winter 1984	83931
6530 TERMINAL					
The Model 6VI Voice Input Option: Its Design and Implementation	B. Huggett	TJ	2,3	Summer 1984	83933
6600 AND TCC6820 COMMUNICATIONS CONTROLLERS					
The 6600 and TCC6820 Communications Controllers: A Performance Comparison	P. Beadles	TSR	2,3	Dec. 1986	83938
BASIC					
An Introduction to Tandem EXTENDED BASIC	J. Meyerson	TJ	2,2	Spring 1984	83932
C					
State-of-the-art C Compiler	E. Kit	TSR	2,2	June 1986	83937
CAM					
Automating Call Centers With CAM	W. Choi	TSR	10,2	April 1994	104398
CIS					
Customer Information Service	J. Massucco	TSR	3,1	March 1987	83939
CLX					
NonStop CLX: Optimized for Distributed On-Line Transaction Processing	D. Lenoski	TSR	5,1	April 1989	18662
COBOL85					
Tandem's New COBOL85	D. Nelson	TSR	2,1	Feb. 1986	83936
COMINT (CI)					
Writing a Command Interpreter	D. Wong	TSR	1,2	June 1985	83935
CYCLONE					
Fault Tolerance in the NonStop Cyclone System	S. Chan, R. Jardine	TSR	7,1	April 1991	46988
DAL SERVER					
The DAL Server: Client/Server Access to Tandem Databases	W. Schlansky, J. Schrengohst	TSR	9,1	Winter 1993	89804

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
DP1 AND DP2					
A Comparison of the B00 DP1 and DP2 Disc Processes	T. Schachter	TSR	1,2	June 1985	83935
Determining FCP Conversion Time	J. Tate	TSR	2,1	Feb. 1986	83936
DP1-DP2 File Conversion: An Overview	J. Tate	TSR	2,1	Feb. 1986	83936
DP2 Highlights	K. Carlyle, L. McGowan	TSR	1,2	June 1985	83935
DP2 Key-sequenced Files	T. Schachter	TSR	1,2	June 1985	83935
DP2 Performance	J. Enright	TSR	1,2	June 1985	83935
DP2's Efficient Use of Cache	T. Schachter	TSR	1,2	June 1985	83935
Sizing Cache for Applications that Use B-series DP1 and TMF	P. Shah	TSR	2,2	June 1986	83937
DSM					
Data Replication in Tandem's Distributed Name Service	T. Eastep	TSR	4,3	Oct. 1988	15748
Event Management Service Design and Implementation	H. Jordan, R. McKee, R. Schuet	TSR	4,3	Oct. 1988	15748
Instrumenting Applications for Effective Event Management	J. Dagenais	TSR	7,2	Oct. 1991	65248
Measuring DSM Event Management Performance	M. Stockton	TSR	8,1	Spring 1992	65250
Network Statistics System	M. Miller	TSR	4,3	Oct. 1988	15748
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
SCP and SCF: A General Purpose Implementation of the Subsystem Programmatic Interface	T. Lawson	TSR	4,3	Oct. 1988	15748
Tandem's Subsystem Programmatic Interface	G. Tom	TSR	4,3	Oct. 1988	15748
Using the Subsystem Programmatic Interface and Event Management Services	K. Stobie	TSR	4,3	Oct. 1988	15748
VIEWPOINT Operations Console Facility	R. Hansen, G. Stewart	TSR	4,3	Oct. 1988	15748
Writing Rules for Automated Operations	J. Collins	TSR	7,2	Oct. 1991	65248
DYNAMITE					
An Introduction to DYNAMITE Workstation Host Integration	S. Kosinski	TSR	1,2	June 1985	83935
The DYNAMITE Workstation: An Overview	G. Smith	TSR	1,2	June 1985	83935
ENABLE					
The ENABLE Program Generator for Multifile Applications	B. Chapman, J. Zimmerman	TSR	1,1	Feb. 1985	83934
ENCOMPASS					
The Relational Data Base Management Solution	G. Ow	TJ	2,1	Winter 1984	83931
ENCORE					
The ENCORE Stress Test Generator for On-line Transaction Processing Applications	S. Kosinski	TJ	2,1	Winter 1984	83931
ENSCRIBE					
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
EXPAND					
Expand High-Performance Solutions	D. Smith	TSR	9,3	Summer 1993	89806
FASTSORT					
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
FOX					
Changes in FOX	N. Donde	TSR	1,2	June 1985	83935
Using FOX to Move a Fault-tolerant Application	C. Breighner	TSR	1,1	Feb. 1985	83934
FUP					
Online Reorganization of Key-Sequenced Tables and Files	G. Smith	TSR	6,2	Oct. 1990	46987
GDS					
Basic Uses and New Features of Extended GDS	A. Hotea	TSR	10,1	Jan. 1994	104396
GUARDIAN 90					
Application Code Conversion for D-Series Systems	K. Liu	TSR	9,2	Spring 1993	89805
B00 Software Manuals	S. Olds	TSR	1,2	June 1985	83935
C00 Software Manuals	E. Levi	TSR	4,1	Feb. 1988	11078
Highlights of the B00 Software Release	K. Coughlin, R. Montevaldo	TSR	1,2	June 1985	83935
Improved Performance for BACKUP2 and RESTORE2	A. Khatri, M. McCline	TSR	1,2	June 1985	83935
Increased Code Space	A. Jordan	TSR	1,2	June 1985	83935
Managing System Time Under GUARDIAN 90	E. Nellen	TSR	2,1	Feb. 1986	83936
Message System Performance Enhancements	D. Kinkade	TSR	2,3	Dec. 1986	83938
Message System Performance Tests	S. Uren	TSR	2,3	Dec. 1986	83938
Migration Planning for D-Series Systems	S. Kuukka	TSR	9,2	Spring 1993	89805
New GUARDIAN 90 Time-keeping Facilities	E. Nellen	TSR	1,2	June 1985	83935
New Process-timing Features	S. Sharma	TSR	1,2	June 1985	83935
NonStop II Memory Organization and Extended Addressing	D. Thomas	TJ	1,1	Fall 1983	83930
Overview of the C00 Release	L. Marks	TSR	4,1	Feb. 1988	11078
Overview of the D-Series Guardian 90 Operating System	W. Bartlett	TSR	9,2	Spring 1993	89805
Robustness to Crash in a Distributed Data Base: A Nonshared-memory Multiprocessor Approach	A. Borr	TSR	1,2	June 1985	83935
Tandem's Approach to Fault Tolerance	B. Ball, W. Bartlett, S. Thompson	TSR	4,1	Feb. 1988	11078
The GUARDIAN Message System and How to Design for It	M. Chandra	TSR	1,1	Feb. 1985	83934
The Tandem Global Update Protocol	R. Carr	TSR	1,2	June 1985	83935
HIMALAYA					
A Hardware Overview of the NonStop Himalaya K10000 Server	C. Kong	TSR	10,1	Jan. 1994	104396
The NonStop Himalaya K10000 Interprocessor Bus	R. Jardine, S. Hamilton, K. Krishnakumar	TSR	10,2	April 1994	104398
INTEGRITY S2					
Overview of the NonStop-UX Operating System for the Integrity S2	P. Norwood	TSR	7,1	April 1991	46988
MEASURE					
How to Set Up a Performance Data Base with MEASURE and ENFORM	M. King	TSR	2,3	Dec. 1986	83938
MEASURE: Tandem's New Performance Measurement Tool	D. Dennison	TSR	2,3	Dec. 1986	83938
MULTILAN					
Introduction to MULTILAN	A. Coyle	TSR	4,1	Feb. 1988	11078
Overview of the MULTILAN Server	A. Rowe	TSR	4,1	Feb. 1988	11078
Using the MULTILAN Application Interfaces	M. Berg, A. Rowe	TSR	4,1	Feb. 1988	11078

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
NETBATCH-PLUS					
NetBatch: Managing Batch Processing on Tandem Systems	D. Wakashige	TSR	5,1	April 1989	18662
NetBatch-Plus: Structuring the Batch Environment	G. Earle, D. Wakashige	TSR	6,1	March 1990	32986
NONSTOP NET/MASTER					
NonStop NET/MASTER: Configuration and Performance Guidelines	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Management Architecture	M. Stockton	TSR	9,4	Fall 1993	89807
NonStop NET/MASTER: Event Processing Costs and Sizing Calculations	M. Stockton	TSR	9,4	Fall 1993	89807
NONSTOP SQL					
An Overview of NonStop SQL Release 2	M. Pong	TSR	6,2	Oct. 1990	46987
Concurrency Control Aspects of Transaction Design	W. Senf	TSR	6,1	March 1990	32986
Converting Database Files from ENSCRIBE to NonStop SQL	W. Weikel	TSR	6,1	March 1990	32986
Gateways to NonStop SQL	D. Slutz	TSR	6,2	Oct. 1990	46987
High-Performance SQL Through Low-Level System Integration	A. Borr	TSR	4,2	July 1988	13693
NonStop SQL Data Dictionary	R. Holbrook, D. Tsou	TSR	4,2	July 1988	13693
NonStop SQL: The Single Database Solution	J. Cassidy, T. Kocher	TSR	5,2	Sept. 1989	28152
NonStop SQL Optimizer: Basic Concepts	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Optimizer: Query Optimization and User Influence	M. Pong	TSR	4,2	July 1988	13693
NonStop SQL Reliability	C. Fenner	TSR	4,2	July 1988	13693
Overview of NonStop SQL	H. Cohen	TSR	4,2	July 1988	13693
Parallelism in NonStop SQL Release 2	M. Moore, A. Sodhi	TSR	6,2	Oct. 1990	46987
Performance Benefits of Parallel Query Execution and Mixed Workload Support in NonStop SQL Release 2	S. Englert, J. Gray	TSR	6,2	Oct. 1990	46987
Tandem's NonStop SQL Benchmark	Tandem Performance Group	TSR	4,1	Feb. 1988	11078
The NonStop SQL Release 2 Benchmark	S. Englert, J. Gray, T. Kocher, P. Shah	TSR	6,2	Oct. 1990	46987
The Outer Join in NonStop SQL	J. Vaishnav	TSR	6,2	Oct. 1990	46987
OSI					
Building Open Systems Interconnection with OSI/AS and OSI/TS	R. Smith	TSR	6,1	March 1990	32986
The OSI Model: Overview, Status, and Current Issues	A. Dunn	TSR	5,1	April 1989	18662
PATHFINDER					
PATHFINDER--An Aid for Application Development	S. Benett	TJ	1,1	Fall 1983	83930
PATHWAY					
A New Design for the PATHWAY TCP	R. Wong	TJ	2,2	Spring 1984	83932
PATHWAY IDS: A Message-level Interface to Devices and Processes	M. Anderton, M. Noonan	TSR	2,2	June 1986	83937
Pathway TCP Enhancements for Application Run-Time Support	R. Vannucci	TSR	7,1	April 1991	46988
The PATHWAY TCP: Performance and Tuning	J. Vatz	TSR	1,1	Feb. 1985	83934
Understanding PATHWAY Statistics	R. Wong	TJ	2,2	Spring 1984	83932
POET					
Designing Client/Server Applications for OLTP on Guardian 90 Systems	W. Culman	TSR	8,3	Fall 1992	89803
PS MAIL					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
RDF					
RDF: An Overview	J. Guerrero	TSR	7,2	Oct. 1991	65248
RDF Synchronization	F. Jongma, W. Senf	TSR	8,2	Summer 1992	69848
RESPOND					
The RESPOND OLTP Business Management System for Manufacturing	H. Bolling, W. Bronson	TSR	9,1	Winter 1993	89804
RSC					
Implementing Client/Server Using RSC	M. Iem, T. Kocher	TSR	8,3	Fall 1992	89803
SAFEGUARD					
Dial-In Security Considerations	P. Grainger	TSR	7,2	Oct. 1991	65248
Distributed Protection with SAFEGUARD	T. Chou	TSR	2,2	June 1986	83937
Enhancing System Security With Safeguard	C. Gaydos	TSR	7,1	April 1991	46988
SNAX					
An Overview of SNAX/CDF	M. Turner	TSR	5,2	Sept. 1989	28152
A SNAX Passthrough Tutorial	D. Kirk	TJ	2,2	Spring 1984	83932
SNAX/APC: Tandem's New SNA Software for Distributed Processing	B. Grantham	TSR	3,1	March 1987	83939
SNAX/HLS: An Overview	S. Saltwick	TSR	1,2	June 1985	83935
SPOOLER					
Sizing the Spooler Collector Data File	H. Norman	TSR	4,1	Feb. 1988	11078
TACL					
Debugging TACL Code	L. Palmer	TSR	4,2	July 1988	13693
TACL, Tandem's New Extensible Command Language	J. Campbell, R. Glascock	TSR	2,1	Feb. 1986	83936
TAL					
New TAL Features	C. Lu, J. Murayama	TSR	2,2	June 1986	83837
TCM					
Capacity Planning With TCM	W. Highleyman	TSR	7,2	Oct. 1991	65248
TLAM					
TLAM: A Connectivity Option for Expand	K. MacKenzie	TSR	7,1	April 1991	46988
TMDS					
C00 TMDS Performance	J. Mead	TSR	4,1	Feb. 1988	11078
Enhancements to TMDS	L. White	TSR	3,2	Aug. 1987	83940
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
TMF					
Improvements in TMF	T. Lemberger	TSR	1,2	June 1985	83935
TMF and the Multi-Threaded Requester	T. Lemberger	TJ	1,1	Fall 1983	83930
TMF Autorollback: A New Recovery Feature	M. Pong	TSR	1,1	Feb. 1985	83934
TNS/R					
Debugging Accelerated Programs on TNS/R Systems	D. Cressler	TSR	8,1	Spring 1992	65250
Improving Performance on TNS/R Systems With the Accelerator	M. Blanchet	TSR	8,1	Spring 1992	65250
Overview of Tandem NonStop Series/RISC Systems	L. Faby, R. Mateosian	TSR	8,1	Spring 1992	65250

Article title	Author(s)	Publication	Volume, Issue	Publication date	Part number
TRANSFER					
The TRANSFER Delivery System for Distributed Applications	S. Van Pelt	TJ	2,2	Spring 1984	83932
TXP					
The High-Performance NonStop TXP Processor	W. Bartlett, T. Houy, D. Meyer	TJ	2,1	Winter 1984	83931
The NonStop TXP Processor: A Powerful Design for On-line Transaction Processing	P. Oleinick	TJ	2,3	Summer 1984	83933
V8					
The V8 Disc Storage Facility: Setting a New Standard for On-line Disc Storage	M. Whiteman	TSR	1,2	June 1985	83935
VIEWSYS					
VIEWSYS: An On-line System-resource Monitor	D. Montgomery	TSR	1,2	June 1985	83935
VLX					
NonStop VLX Hardware Design	M. Brown	TSR	2,3	Dec. 1986	83938
NonStop VLX Performance	J. Enright	TSR	2,3	Dec. 1986	83938
The VLX: A Design for Serviceability	J. Allen, R. Boyle	TSR	3,1	March 1987	83939
XL8					
Data-encoding Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937
Plated Media Technology Used in the XL8 Storage Facility	D. S. Ng	TSR	2,2	June 1986	83937

TandemSystemsReview*OrderForm*

Use this form to order new subscriptions, change subscription information, and order back issues.

☐ I am a Tandem customer. My Tandem sales representative is ______________________.

☐ I am not a Tandem customer and am enclosing a check or money order for the requests indicated on this form. (Subscriptions are $75 per year and each back issue is $20. Make checks payable to Tandem Computers Incorporated.)

Subscription Information

☐ New subscription

☐ Update to subscription information
Subscription number: ________________

Your subscription number is in the upper right corner of the mailing label.

COMPANY

NAME

JOB TITLE

DIVISION

ADDRESS

COUNTRY

TELEPHONE NUMBER (include all codes for U.S. dialing)

Title or position:

☐ President/CEO
☐ Director/VP information services
☐ MIS/DP manager
☐ Software development manager
☐ Programmer/analyst
☐ System operator
☐ End user
☐ Other: ________________

Your association with Tandem:

☐ Tandem customer
☐ Third-party vendor
☐ Consultant
☐ Other: ________________

Back Issue Requests

Number of copies *Tandem Systems Review*

____	Vol. 1, No. 1, Feb. 1985	____	Vol. 7, No. 1, April 1991
____	Vol. 1, No. 2, June 1985	____	Vol. 7, No. 2, Oct. 1991
____	Vol. 2, No. 1, Feb. 1986	____	Vol. 8, No. 1, Spring 1992
____	Vol. 2, No. 2, June 1986	____	Vol. 8, No. 2, Summer 1992
____	Vol. 2, No. 3, Dec. 1986	____	Vol. 8, No. 3, Fall 1992
____	Vol. 3, No. 1, March 1987	____	Vol. 9, No. 1, Winter 1993
____	Vol. 3, No. 2, Aug. 1987	____	Vol. 9, No. 2, Spring 1993
____	Vol. 4, No. 1, Feb. 1988	____	Vol. 9, No. 3, Summer 1993
____	Vol. 4, No. 2, July 1988	____	Vol. 9, No. 4, Fall 1993
____	Vol. 4, No. 3, Oct. 1988	____	Vol. 10, No. 1, Jan. 1994
____	Vol. 5, No. 1, April 1989	____	Vol. 10, No. 2, April 1994
____	Vol. 5, No. 2, Sept. 1989		
____	Vol. 6, No. 1, March 1990		
____	Vol. 6, No. 2, Oct. 1990		

Tandem Journal

____	Vol. 1, No. 1, Fall 1983	____	Vol. 2, No. 2, Spring 1984
____	Vol. 2, No. 1, Winter 1984	____	Vol. 2, No. 3, Summer 1984

For questions or ordering information, call 800-473-5868 in the U.S. and Canada or +1-408-285-0665 in other countries.

Send this form to:
Tandem Computers Incorporated
Tandem Systems Review, Loc 208-65
10400 Ridgeview Court
Cupertino, CA 95014-0723
FAX: +1-408-285-0840

Tandem employees must order their subscriptions and back issues through Courier.

Menu sequence: Marketing Information → Literature Orders → Technical Marketing Pubs (TSR)

▲ FOLD

▲ FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM SYSTEMS REVIEW
LOC 208-65
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95014–9862

▼ FOLD

▼ FOLD

TandemSystemsReview*ReaderSurvey*

The purpose of this questionnaire is to help the *Tandem Systems Review* staff select topics for publication. Postage is prepaid when mailed in the United States. Readers outside the U.S. should send their replies to their nearest Tandem sales office.

1. How useful is each article in this issue?

Product Update

01 ☐ Indispensable	02 ☐ Very	03 ☐ Somewhat	04 ☐ Not at all

The NonStop Himalaya K10000 Interprocessor Bus

05 ☐ Indispensable	06 ☐ Very	07 ☐ Somewhat	08 ☐ Not at all

Client/Server Availability

09 ☐ Indispensable	10 ☐ Very	11 ☐ Somewhat	12 ☐ Not at all

Automating Call Centers With CAM

13 ☐ Indispensable	14 ☐ Very	15 ☐ Somewhat	16 ☐ Not at all

2. I specifically would like to see more articles on (select one):

17 ☐ Overview discussions of new products and enhancements	18 ☐ Performance and tuning information
19 ☐ High-level overviews on Tandem's approach to solutions	20 ☐ Application design and customer profiles
21 ☐ Technical discussions of product internals	22 ☐ Strategic information and statements of direction

23 ☐ Other ____________________

3. Your title or position:

24 ☐ President, VP, Director	25 ☐ Systems analyst	26 ☐ System operator
27 ☐ MIS manager	28 ☐ Software developer	29 ☐ End user

30 ☐ Other ____________________

4. Your association with Tandem:

31 ☐ Tandem customer	32 ☐ Tandem employee	33 ☐ Third-party vendor	34 ☐ Consultant

35 ☐ Other ____________________

5. Comments

____

____

____

____

NAME ____________________

COMPANY NAME ____________________

ADDRESS ____________________

____

4/94

▲ FOLD

▲ FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 482 CUPERTINO, CA U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE

TANDEM SYSTEMS REVIEW
LOC 208-65
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95014–9862

▼ FOLD

▼ FOLD

TANDEM

Tandem Computers Incorporated
19333 Vallco Parkway
Cupertino, CA 95014-2599

Allen Goldin
LOC NUM 128-00
Jericho Ny Long Island Bran

4/94 Printed in USA

Part No. 104398