T A N D E M

SYSTEMS REVIEW

VOLUME 7, NUMBER 1 APRIL 1991

Fault-Tolerant NonStop Cyclone

NonStop-UX Operating System

Safeguard • TLAM

Pathway TCP Enhancements

Index

Volume 7, Number 1, April 1991

Editorial Director
Susan W. Thompson

Editor
Anne Lewis

Associate Editor
Steven Kahn

Technical Advisors
Mark Anderton
Terrye Kocher
Mike Noonan

Assistant Editor
Sarah Rood

Electronic Publishing
Marcy Cross

Cover Art
Niklas Hallin

Illustrations
Cynthia Moore

Circulation
Christina Cary

The *Tandem Systems Review* is published by Tandem Computers Incorporated.

Purpose: The *Tandem Systems Review* publishes technical information about Tandem software releases and products. Its purpose is to help programmer-analysts who use our computer systems to plan for, install, use, and tune Tandem products.

Subscription additions and changes: As of the March 1990 issue, customer subscriptions to the *Tandem Systems Review* must be approved by a Tandem representative. Complete the subscriber portion of the order form at the back of this copy and send the form to your local Tandem sales office. Anyone who does not have a Tandem representative should fill out the subscriber portion and follow the instructions on the form.

Comments: The editors welcome suggestions for content and format. Please send them to the *Tandem Systems Review*, LOC 216-05, 18922 Forge Drive, Cupertino, CA 95014.

Editor's Preface

Fault tolerance is becoming an increasingly important issue in the data processing market place. Companies of all sizes are basing the competitiveness of their enterprises on the timeliness and availability of information. These computer-based enterprises require a mix of systems that are not only geographically distributed but are also fault tolerant. In online transaction processing systems, successful fault tolerance includes both continuous system availability and reliably completed transactions. Tandem™ systems use both hardware and software methods to achieve complete fault tolerance.

The first two articles in this issue focus on the newest Tandem computers, the NonStop™ Cyclone™ and the Integrity S2™ systems. These products continue Tandem's development of fault-tolerant systems that address emerging computing needs in the data processing marketplace.

The NonStop Cyclone system is designed for high-volume processing, which includes simultaneous transaction, query, and batch processing. Because large-scale data processing systems are moving increasingly toward online databases, fault tolerance is a critical requirement. The opening article by Chan and Jardine describes the methods used to achieve fault tolerance in the NonStop Cyclone system. It describes the system architecture and its implementation of fault detection, fault containment, error recovery, and continuous operation from individual component failures.

The move toward UNIX has had an impact on all areas of computing, including the need for a fault-tolerant UNIX system. The Integrity S2 system combines the traditional features of fault tolerance with a standard implementation of UNIX. The article by Norwood discusses the Integrity S2 architecture, the NonStop-UX™ operating system, and the methods used to provide fault tolerance. It describes how triple modular redundancy, duplexed components, and self-checking circuitry support hardware fault tolerance. In addition, the article describes how the NonStop-UX operating system has been enhanced to increase system performance and functionality and improve the robustness of the UNIX kernel.

A comprehensive security plan is important for the complete protection of information. There are three forms of on-system protection: authentication, which permits the system to identify individual users; authorization, which restricts access to data files; and auditing, which records the actions of users and activities of programs. Tandem's Safeguard™ system protection software extends Guardian 90™ operating system protection by providing more extensive and general authentication, authorization, and auditing services. The article by Gaydos describes the basic elements of Safeguard and how they enhance Guardian 90 protection.

The article by MacKenzie discusses an enhancement to the Tandem LAN Access Method (TLAM) subsystem that enables the Expand™ data communications networking software to operate over standard local area network (LAN) media. As an additional communication interface to Expand, TLAM provides another tool for implementing Tandem's open-standards-based approach to networking.

The Pathway transaction processing system automatically manages the transaction workflow between terminal or devices and the database on NonStop systems. The final article, by Vannucci, describes the terminal control process (TCP) component of Pathway. The TCP provides the run-time environment for user-written SCREEN COBOL application requesters. Tandem has enhanced the TCP to support application requesters that require large data address spaces. This article discusses the TCP data address space limitations in earlier releases of Pathway and the enhancements introduced in later releases.

Finally, this issue includes an index of *Tandem System Review* articles. The index is a list of all articles, by subject and product, that have been published in this and each previous issue. If you would like to order back issues, submit the order form on the last page.

Susan W. Thompson
Editorial Director

Fault Tolerance in the NonStop Cyclone System

Commercial data processing systems are moving increasingly toward online databases and applications. These systems require continuous system availability, secure transactions, and error-free databases. The Tandem™ NonStop™ Cyclone™ system is a multiprocessor mainframe designed for the highest performance in simultaneous transaction, query, and batch processing. It uses various hardware and software techniques to perform fault detection and recovery, ensuring data integrity and continuous system operation for critical commercial applications.

To achieve system-level fault tolerance, the NonStop Cyclone system uses Tandem's proven system architecture. To achieve fault detection within an individual processor, the NonStop Cyclone system enhances the methods used by the Tandem NonStop VLX™ system.

This article surveys the methods used to achieve fault tolerance in the Tandem NonStop Cyclone system. It describes how the following design principles of fault-tolerant system operation apply through various levels of the NonStop Cyclone system:

- Fault detection by both hardware and software.
- Fail-fast and fault-containment designs that prevent corruption of user databases by faulty subsystems.
- High reliability through safe design and error-recovery features within subsystems.
- Continuous system operation achieved through mechanisms that recover from individual component failures.

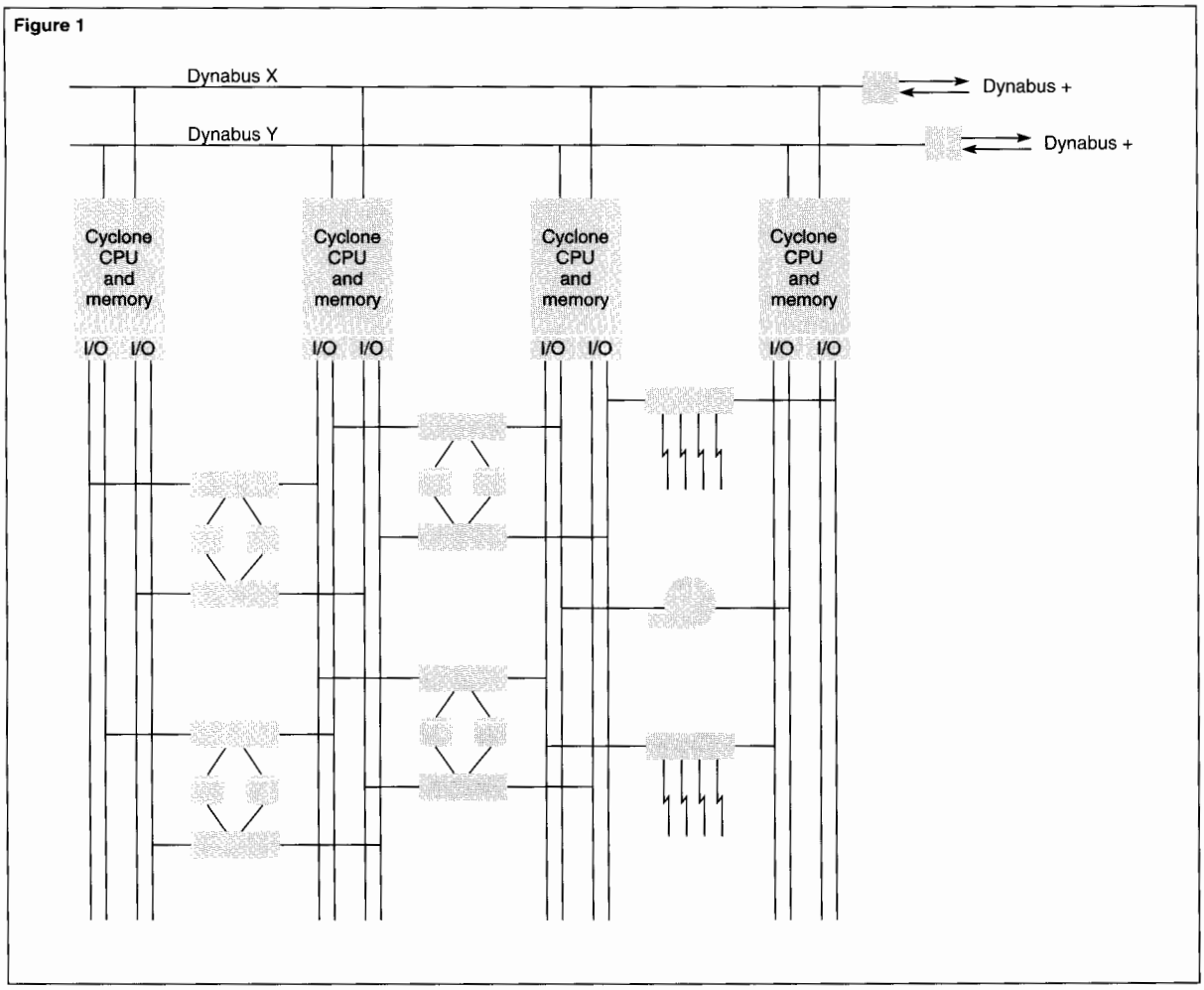


Figure 1.
NonStop Cyclone system architecture.

NonStop Cyclone System Design

The NonStop Cyclone system is the most powerful of Tandem's fault-tolerant, multiprocessor systems (Chan and Horst, 1989; Horst, Jardine, and Harris, 1990). Designed to support simultaneous transaction, query, and batch processing, NonStop Cyclone systems consist of two to sixteen processors loosely coupled by dual high-speed busses (the Tandem Dynabus). Figure 1 illustrates the NonStop Cyclone system architecture.

Sections of four processors are interconnected by fiber optic cables (the Tandem Dynabus+™) and can be physically separated up to 50 meters. Each processor has its own memory and controls two to four I/O channels. Fault detection is performed primarily by the hardware, and fault recovery is performed by Tandem's message-based Guardian™ 90 operating system. The system can tolerate a single fault in a processor, peripheral controller, power supply, or cooling system. Failed components can be serviced online without disrupting processing.

System-Level Fault Tolerance

The NonStop Cyclone system uses the proven system architecture of its predecessors, in which all major components of the system are replicated. (See Figure 1.) Guardian 90 manages these components and, if one fails, arranges for its function to be taken over by another component. The replicated components are not merely redundant or idle; they operate concurrently to enhance system performance while providing fault tolerance.

For example, in normal operation, both Dynabusses carry message traffic. If a single Dynabus failure occurs, all traffic is routed onto the remaining Dynabus. The new Dynabus+, the fiber-optic connection between processor sections, is configured in a dual-ring arrangement, allowing tolerance of selected multiple failures. Power supply and cooling blower loads are distributed so that the effect of a single failure is confined to at most a single processor or I/O controller.

I/O controllers are duplicated, lock-stepped microprocessors with self-checking comparison circuits. I/O controllers are dual-ported to separate processors, maintaining a path to the I/O device even when one processor or I/O channel is down. These replicated devices and busses ensure that no single failure will cause the entire system or any part of a database to be unavailable. Through mirroring, disk drives can also be configured to be tolerant of single faults.

At the software level, operating system processes are programmed as process pairs (Bartlett, 1981; Bartlett et al., 1990). A process pair consists of a primary process and a backup process, executing in different processors. The primary process performs the actual work of the process pair, occasionally sending the backup process a checkpoint message containing its current state. If the primary process fails (for example, if the processor in which it is executing fails to send its periodic *I'm Alive* message and is declared down), the backup process takes over and resumes processing at the point of the last checkpoint received. In addition to providing tolerance of single hardware faults, the message-based, process-pair structure of the software also provides tolerance of intermittent software faults, a feature not provided by hardware-only fault-tolerant systems.

Finally, the Tandem Transaction Monitoring Facility (TMF™) provides an even greater degree of protection and ease of programming (Bartlett et al., 1990). TMF allows an application to package its computations and database updates into atomic units. This protects the integrity of the database even in the face of multiple faults and prolonged power failures.

Fault Detection Within a NonStop Cyclone Processor

Once system-level fault tolerance is provided, the only requirement at the processor level is that the processor halt quickly after detecting a failure. This *fail-fast* principle has two benefits:

- It contains the error to the failing processor, preventing data corruption.
- It minimizes service delays while the backup processes take over.

The duplicate-and-compare method of fault detection, used in the Tandem NonStop CLX™ system (Lenoski, 1989), is not practical to implement on a processor as large as the NonStop Cyclone. Instead, a variety of hardware and software methods are used. Many of these methods are used in the NonStop VLX processor, although the NonStop Cyclone has much more extensive error identification and diagnostic capability.

The NonStop Cyclone uses parity checking extensively to detect single-bit errors. Parity is propagated through devices that do not alter data, such as memories, control signals, busses, and registers. Parity prediction is used on devices that alter data, such as arithmetic units and counters. Predicted parity is based strictly on a device's data and parity inputs; it does not rely on the device's outputs, which may be faulty. Thus, an adder might generate an erroneous sum, but the parity that accompanies the sum will correspond to the correct result. Parity checkers downstream will then detect the error.

A novel technique similar to recomputation with shifted operands (RESO) protects the hardware multiplier (Sohi et al., 1989). After each multiplication, a second multiplication is initiated with the operands exchanged and one operand shifted. Microcode compares the two results whenever the multiplier is needed again or before any data leaves the processor. Unlike other implementations of RESO, these checking cycles incur almost no performance penalty because they occur concurrently with unrelated execution steps.

Within the NonStop Cyclone processor, invalid-state checking or the duplication-and-comparison method is used in sequential logic circuits. Checksums protect multiple-word transmissions such as the interprocessor bus and I/O channel. Watchdog timers and microcode polling monitor operations that take many cycles to complete.

If the processor hardware detects a fault from which it cannot recover, the processor shuts itself down within two clock cycles, before it can transmit any corrupt data along the interprocessor bus or I/O channel. The error is flagged in one or more of the approximately 300 error identification registers, allowing quick fault isolation to any of the 500 hardware error detectors in each processor. (See Figure 2.)

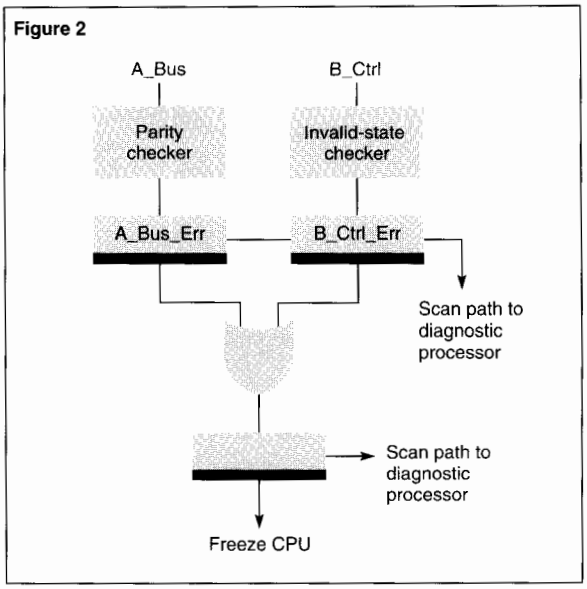
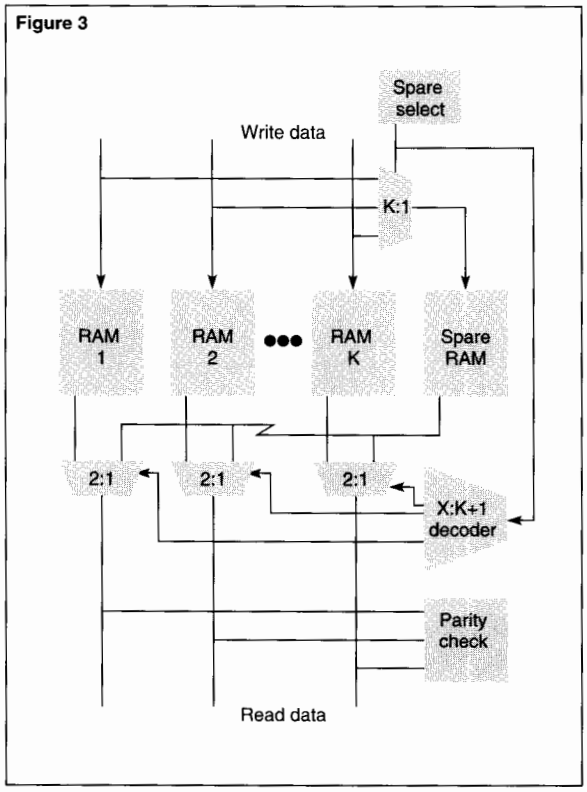


Figure 2.
NonStop Cyclone error identification registers.

Both the microcode and operating system perform numerous consistency checks such as invalid instruction detection and address bounds checking. Also, the microcode executes processor diagnostic routines during idle situations. If the microcode or operating system detects an unrecoverable error, it immediately executes a HALT instruction and transmits an error code to the Remote Maintenance Subsystem.

Figure 3.
NonStop Cyclone spare RAM mechanism.



Fault Recovery Within a NonStop Cyclone Processor

System-level fault tolerance can be achieved without incorporating any fault tolerance capabilities within the processors themselves. However, each NonStop Cyclone processor has numerous online recovery mechanisms that allow it to withstand certain types of hardware faults. This nearly doubles the calculated mean-time-to-failure of each processor and dramatically reduces customer service costs.

As in the NonStop VLX processor, large static random-access memory (RAM) arrays, such as data and instruction caches, main control store, and I/O subsystem control store, can recover from intermittent (soft) data errors by reloading from alternate copies. For example, a soft error in the data cache is corrected by refilling the block from main memory. In addition, the main control store and caches have spare RAMs that automatically replace hard-failed RAMs (Horst, 1989). (See Figure 3.)

A single-error-correcting, double-error-detecting code protects dynamic RAMs in main memory. This code incorporates both data and address bits, so that addressing failures are detected as well as RAM failures. An asynchronous microcode process periodically checks for correctable memory errors. These errors are logged, and the memory areas are scrubbed by Guardian 90. If checksum errors occur, the operating system retries interprocessor bus packets and I/O transfers.

Diagnostic Facilities

The NonStop Cyclone diagnostic facilities are based on those successfully developed in the NonStop VLX system (Allen and Boyle, 1987). Each processor has a dedicated microprocessor that executes quick diagnostics, scans the initial state into the processor, loads bootstrap microcode into the writable control stores, and initiates system cold load. It can generate and collect pseudo-random scan-test signatures for quick fault detection and isolation, and it serves as the interface to a system-level, fault-tolerant maintenance and diagnostic subsystem.

In addition, the maintenance subsystem has an extensive power and environmental monitoring facility. Sensors in all cabinets measure power supply voltages, air temperature, and cooling blower speeds.

The Tandem Maintenance and Diagnostic System (TMDS), a collection of software processes running in the Guardian 90 environment, monitors and logs events in a running system, supports diagnosis of failures anywhere in the system, and optionally dials out to report problems to a Tandem customer support center. The maintenance and diagnostic subsystem, the power and environmental monitoring facility, and TMDS software are compatible with NonStop VLX systems and allow the integration of NonStop Cyclone processor sections into existing NonStop VLX systems.

Conclusion

As commercial data processing systems move increasingly toward critical online databases and applications, features such as continuous system availability, secure transactions, and error-free databases become requirements. The NonStop Cyclone system has been developed to provide the highest levels of performance, system availability, and data integrity for today's commercial processing needs.

References

Allen, J. and Boyle, R. 1987. The VLX: A Design for Serviceability. *Tandem Systems Review.* Vol. 3, No. 1. Tandem Computers Incorporated. Part no. 83939.

Bartlett, J. 1981. A NonStop Kernel. *Proceedings of the Eighth Symposium on Operating System Principles.*

Bartlett, J. et al. 1990. *Fault Tolerance in Tandem Computer Systems.* Tandem Technical Report 90.5. Tandem Computers Incorporated. Part no. 40666.

Chan, S. and Horst, R. December 1989. Parallelism in the Instruction Pipeline. *High Performance Systems.*

Horst, R. 1989. Reliable Design of High-speed Cache and Control Store Memories. *Proceedings of the Nineteenth International Symposium on Fault Tolerant Computing.*

Horst, R., Jardine, R., and Harris, R. 1990. Multiple Instruction Issue in the NonStop Cyclone Processor. *Seventeenth International Symposium on Computer Architecture.* Also Tandem Technical Report 90.6. 1990. Tandem Computers Incorporated. Part no. 48007.

Lenoski, D. 1989. NonStop CLX: Optimized for On-line Transaction Processing. *Tandem Systems Review.* Vol. 5, No. 1. Tandem Computers Incorporated. Part no. 18662.

Sohi, G. et al. 1989. A Study of Time-Redundant Fault Tolerance Techniques for High-Performance Pipelined Computers. *Proceedings of the Nineteenth International Symposium on Fault Tolerant Computing.*

Note

A shorter version of this paper was previously published as Tandem Technical Report 90.7 (part no. 48008) and in the proceedings of the Spring Conference of the Institute of Electronic, Information, and Communication Engineers, Chuo University, Tokyo, Japan, March 1990.

Scott Chan was Project Lead Engineer for the NonStop Cyclone processor. He joined Tandem after graduating from Stanford University in 1982 and has contributed to the NonStop TXP, NonStop VLX, and several other Tandem processor designs.

Robert Jardine joined Tandem in 1984 to work on the NonStop Cyclone processor design and microcode. His prior experience included 12 years of design and implementation of compilers, operating systems, and processor microcode. He is currently contributing to the design of future Tandem products.

Overview of the NonStop-UX Operating System for the Integrity S2

The Tandem™ Integrity S2™ system combines high availability and data integrity, two traditional features of fault tolerance, with a standard implementation of the UNIX operating system. Integrity S2 meets current demands for fault-tolerant systems combined with an industry-standard operating environment.

The fault-tolerant capabilities of the Integrity S2 are realized through a combination of hardware and software. The hardware supports fault-tolerant operation through a variety of techniques, including triple modular redundancy, duplexed hardware, and self-checking circuitry. A hierarchical memory architecture takes full advantage of the reduced instruction set computing (RISC) processor technology used in the Integrity S2.

The NonStop-UX™ operating system, based on an AT&T UNIX V.3 kernel, has been enhanced in a number of ways to increase system performance and functionality, improve the robustness of the standard UNIX product, and support fault-tolerant system operation. Local and global memory is managed with a two-tiered design to maximize system performance. Additional software enhancements provide monitoring and diagnostic services to detect faulty components and perform appropriate recovery procedures.

This article describes the components of the hardware architecture and key features of the NonStop-UX operating system. It explains robustness enhancements made to the UNIX kernel. Finally, it discusses online serviceability and improvements made to ensure data integrity; these include failure detection, isolation, and recovery as well as the powerfail shutdown and automatic restart procedures activated by environmental failures.

The Integrity S2 Hardware Architecture

The Integrity S2 has a hardware architecture designed to support fault tolerance while supporting Tandem's implementation of UNIX System V operating system. The hardware design uses a redundant CPU architecture with a high-speed RISC microprocessor at the heart of each CPU. System expansion is possible by adding more memory or communication devices. Mass storage cabinets allow flexible system configuration. The fault-tolerant I/O system, replicated components and data paths, and self-checking circuitry are designed to prevent a single hardware failure from interrupting data processing. Figure 1 illustrates the components of the Integrity S2 hardware architecture.

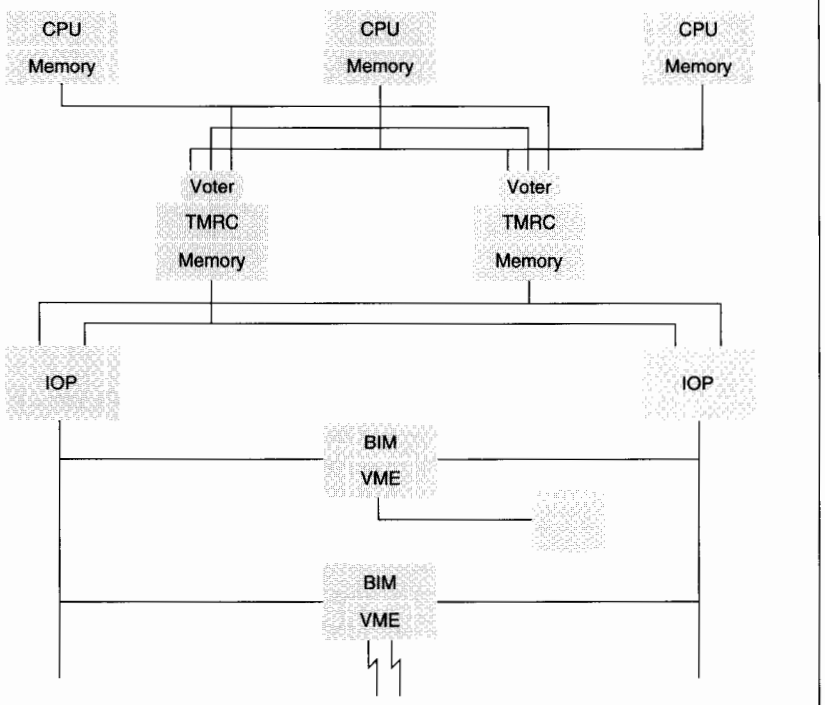
Triple Modular Redundancy

The Integrity S2 hardware architecture is based on triple modular redundancy (TMR). The Integrity S2 architecture design uses three CPU modules operating as one TMR logical processor. The three CPUs execute the same instructions, then compare and vote on the outputs. This design expedites isolation of a malfunctioning CPU and protects data integrity.

The most important architectural difference between the Integrity S2 and traditional TMR architectures is that Integrity S2 uses three independently clocked CPUs. While all three CPUs execute the same instruction stream, they do not necessarily execute the same instruction at the same time.

Each CPU has its own oscillator. If one of the CPUs has an oscillator that beats slightly faster than the rest, that CPU will move ahead of the others in the instruction stream. When an interrupt occurs, all CPUs must see the interrupt at the same point in the instruction stream, or the CPUs would take different paths in processing the interrupts. Therefore, the CPUs are synchronized whenever external interrupts are presented to the CPUs.

Figure 1



Hierarchical Memory Architecture

Memory for the Integrity S2 is organized hierarchically into local and global memory. Cache and disk are the other two components of the memory hierarchy, which is controlled by memory management software. Each CPU module contains 8 to 64 megabytes of high-speed local memory. The CPUs execute primarily from local memory. The processors can also access a somewhat slower, duplicated global memory by way of the reliable system bus (RSB). Global memory can be as large as 128 megabytes.

Figure 1.
The Integrity S2 hardware architecture uses a triple modular redundant design with replicated components and data paths.

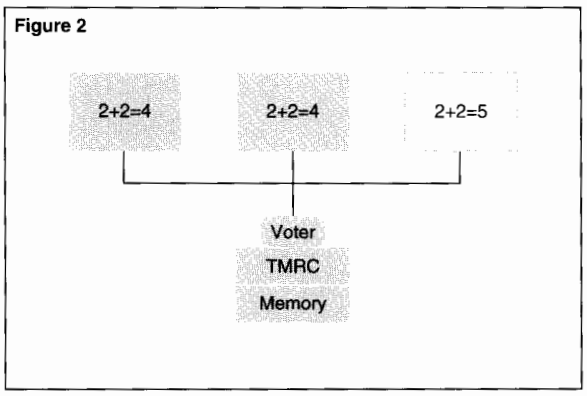


Figure 2.
The voter modules compare CPU output and vote all writes to global memories. CPU failures are reliably detected, and the voter modules isolate any errant CPU.

Integrating RISC Technology

The current CPU consists of a 16.67-MHz R2000 MIPS microprocessor using RISC technology and containing 64 kilobytes each of instruction and data cache. The hierarchical memory design takes full advantage of the RISC processor technology. Large caches and fast memories are required to keep RISC processors fed at rates that are fast enough to keep them from stalling. The global memory, used primarily as a fast swap device, ensures that the local memory has fast access to the active working sets of running processes.

The loose synchronization of the CPUs also enables the RISC microprocessors to run at high speed. At these high frequencies, it is difficult to lockstep multiple CPUs. Loose synchronization solves this problem without adversely impacting performance.

Voter Modules

Two self-checking voter modules connect to and monitor the CPU modules. Voting occurs when the global memory must be accessed or an interrupt needs to be processed. One voter compares the output of its CPU with that of the other two. If the outputs match, the CPUs are operating correctly. As shown in Figure 2, if a CPU has output that differs from the other two, the voter assumes it is faulty. The malfunctioning CPU is outvoted and isolated before it can corrupt any permanent data in the global memories of the system.

The Integrity S2 hardware design contains two boards called the TMR controllers (TMRCs). The two voter modules as well as the dual global memories reside on the TMRCs, one voter module and one memory module to each board. One TMRC is designated as the primary controller, and the other is the secondary. Data is always read from the primary TMRC. A process called the primary-secondary swapper periodically alternates the assignments of the primary and secondary TMRCs.

System Expansion

Memory can be increased to 192 megabytes. Slots for additional I/O controllers, which can provide connections for additional disk drives, tape drives, and data communication lines, are inside the system cabinet.

Up to four mass storage cabinets (MSC) can be added to one system. Each MSC houses seven slots for standard 5-1/4-inch small computer system interface (SCSI) devices. Alternately, six disk drives and one tape drive can be installed.

Duplexed Components in the Fault-Tolerant I/O Subsystem

The Integrity S2 I/O subsystem supports fault tolerance by providing redundant paths to peripheral and communications controllers. There are five major subsystem components:

- Dual reliable I/O buses (RIOBs).
- Dual I/O processors (IOPs).
- Dual NonStop V+™ buses.
- Bus interface modules (BIMs).
- Intelligent I/O controllers.

The dual RIOBs connect the three CPUs, two TMRCs, and three RSBs to the I/O subsystem. Each RIOB supports a bidirectional 32-bit data path plus 4-bit parity.

The dual IOPs, located between the TMRCs and the I/O controllers, verify data addresses from up to eight controllers before transferring the data to global memory. The IOPs are connected to the global memory modules, located on the TMRCs, through the RIOB. Each of the dual IOPs controls a single NonStop V+ bus.

The NonStop V+ bus is an industry-standard VMEbus that has been enhanced with parity and other fault detection and isolation properties to support a more robust I/O subsystem. Dual NonStop V+ buses provide a path for data transfer between VMEbus controllers and the active IOP for those controllers.

The dual NonStop V+ buses connect through BIMs to industry-standard VMEbus controllers. The BIMs allow a single controller to interface to either of the NonStop V+ buses, although only one connection is active at any one time. The processor can switch a controller from one bus to another if an IOP or a NonStop V+ bus fails. Up to eight VMEbus controllers can be integrated into the I/O subsystem.

The intelligent I/O controllers incorporate microprocessors to improve I/O performance. They manage most of the mass storage and communications processing requirements, freeing the CPUs to complete other tasks. The controllers are connected by way of redundant paths to the IOPs.

Self-Checking Circuitry

Self-checking designs are used throughout the architecture, along with other methods of error detection, to increase the fault detection coverage. A diagnostic subsystem reports the identification and isolation of the errors. Once isolated and identified, the user can remove it from the system and insert a new one without affecting the availability of the system.

The Integrity S2 Software Architecture

The principal goal of the software architecture was to provide a completely standard implementation of UNIX System V on the Integrity S2 as well as support the fault-tolerant hardware architecture. Additional objectives were to provide high availability, data integrity, and user serviceability.

These requirements are not usually associated with the UNIX operating system, which is known for its tendency to crash. Therefore, a simple port of UNIX to operate on the Integrity S2 hardware architecture would not satisfy all system software requirements. To achieve its goals, the NonStop-UX operating system contains enhancements to the basic implementation of UNIX System V while still maintaining System V Interface Definition (SVID) compliance.

Methodology for Extending UNIX to Provide Fault Tolerance

Tandem developers decided to base the NonStop-UX operating system on a standard implementation of UNIX after analyzing several unsuccessful versions of UNIX based on fault-tolerant hardware. Concluding that standard UNIX is not well-suited to a fault-tolerant environment, these other vendors rewrote the UNIX operating system. However, a proprietary version of UNIX can present serious problems for users.

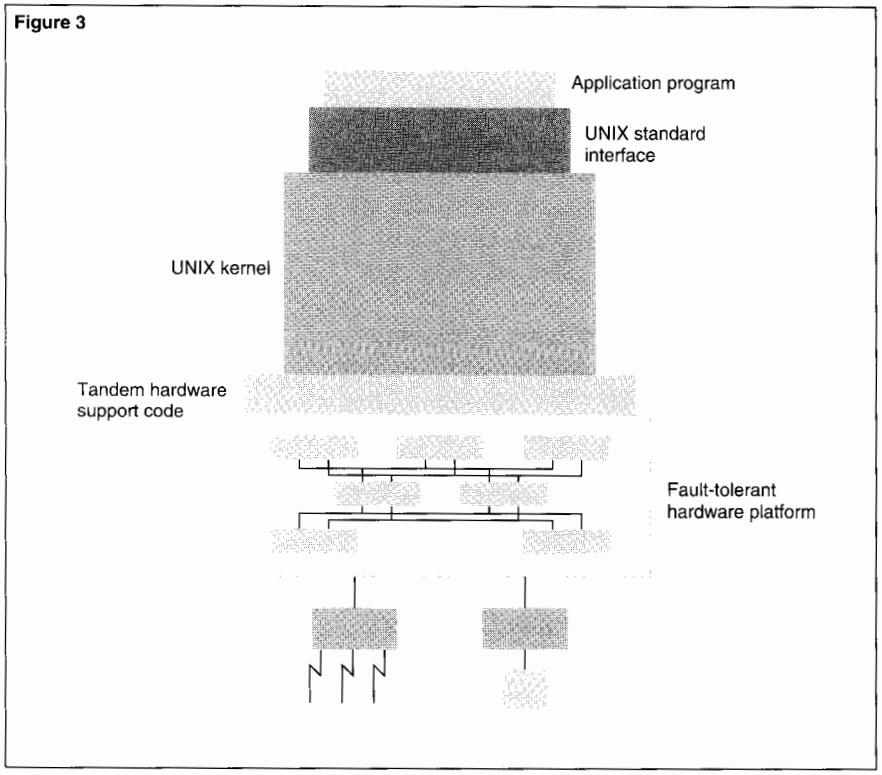


Figure 3.
The NonStop-UX operating system provides fault-tolerant capabilities through modular extensions to the kernel while maintaining full compliance with the SVID application interface.

First, many users have had trouble porting UNIX applications to a nonstandard UNIX operating system, even one intended to be an improvement of standard UNIX. Users are reluctant to port a large UNIX application to an operating system that deviates in any way from the UNIX interface definitions or, worse, does not obey UNIX semantics.

Second, users want to have access to the latest UNIX features. Users must wait longer for these features if they have to depend on the system provider to develop them. True UNIX semantics are hard to achieve by transforming another operating system into UNIX at the system call or library level. Implementing new internal features is much more difficult than simply porting the new release of UNIX to the hardware platform. By offering an emulation of UNIX, one loses one of its principal advantages, which is the technology of the UNIX operating system itself.

Third, users demand access not only to the UNIX operating system, but also to the UNIX software development environment. It is superior to development environments found on many proprietary systems. Also, it is available on a variety of hardware platforms. Finally, companies find it relatively easy to hire and retain developers trained in the UNIX environment.

Therefore, the NonStop-UX system was not designed as a layer that transformed a non-UNIX operating system into a SVID-compatible programming interface. Tandem chose to provide a UNIX implementation on a fault-tolerant hardware design based on the following guidelines:

- Start from a good standard port of UNIX System V.
- Whenever possible, introduce fault-tolerant features in a modular manner that are portable across releases of the operating system.
- Ensure that none of the work to add fault tolerance violates existing and emerging standards, such as X/OPEN or POSIX.[1]
- Ensure that no user-level application software changes are required to take advantage of fault-tolerant features.

Figure 3 illustrates the standard and modified elements that comprise the Integrity S2 system.

[1]X/OPEN and POSIX are two organizations that are defining standards important to UNIX.

Basic Features of the NonStop-UX Operating System

The NonStop-UX operating system is based on the AT&T System V, Release 3 kernel and provides excellent portability. It complies with the AT&T SVID Issue 2 and passes the System V Verification Suite. These tests verify conformance with SVID standards.

NonStop-UX provides improved system performance and networking capabilities. For example, NonStop-UX supports the improved Berkeley Software Distribution (BSD) Fast File System (FFS), which uses the File System Switch feature of UNIX System V. FFS allows NonStop-UX to have faster file access time.

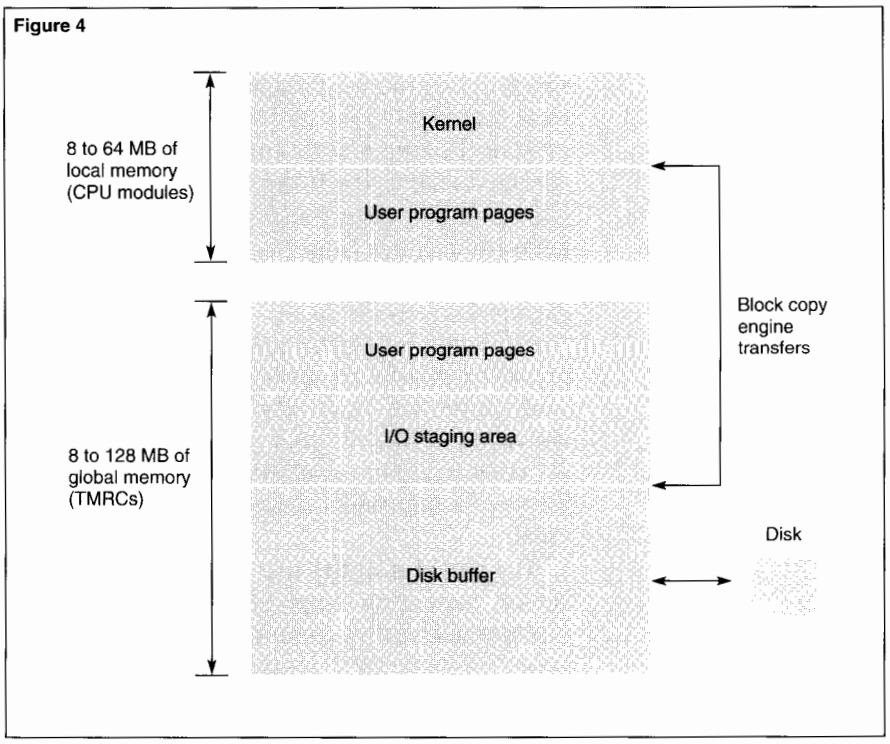
A MIPS optimizing compiler system is included to maximize RISC processing performance. The compiler system translates high-level languages[2] into machine code, optimized for the RISC microprocessor. The compiler stores data items among 32 registers in the RISC chip, making them accessible with the lowest number of operations and thus expediting program execution.

NonStop-UX integrates into local and wide area networks by supporting networking software. The BSD sockets library interface provides tools for developing distributed applications. The Sun Microsystems Network File System (NFS) allows an Integrity S2 system to access files and devices on other systems using NFS. Transmission Control Protocol/Internet Protocol (TCP/IP) and the X Window System are two of various protocols supported by NonStop-UX that enable communication with other UNIX systems.

Two-Tiered Memory Management

The two-tiered memory management design corresponds to the hierarchical local and global memory organization of the hardware architecture. This approach optimizes the demands of the RISC microprocessor and supports the hardware design. Figure 4 illustrates the two-tiered design.

Figure 4



Because the MIPS R2000 microprocessor, like all RISC processors, achieves its full performance potential when it is fed from memory at speeds that match its cycle time, the operating system must execute with a high cache hit rate and refill the cache quickly on misses.[3] Placing fast memory on the hierarchy's first tier improves the performance of local memory accesses. Local memory contains user program pages, kernel text, and kernel data. Global memory, less often accessed, is on the second tier. It contains disk buffer cache, I/O buffers, and user program pages.

Figure 4.
For optimum performance, system memory is organized in a two-tier hierarchy.

[2]Compilers are provided for the following languages: C, FORTRAN, Pascal, and COBOL. The C compiler is standard.

[3]A hit rate is the probability that a memory reference is stored in the cache. A miss occurs when a processor has to go to the next tier to fetch a memory reference.

The CPUs execute from local memory as much as possible. When the operating system schedules a process, it moves the process into local memory to execute. Each of the CPUs contains its own local memory, which can be accessed independently of the other CPUs. This allows rapid access to local memory and helps to ensure that the processor can continue to execute at relatively high speeds even when it encounters a cache miss.

Global memory is located on the TMR controller board. The global memory modules contain identical data to protect against TMRC and memory failure. Each global access triggers the voter modules, which monitor the CPUs, and the three independent instruction streams are voted.

The global memory acts as a fast swap device, increasing the performance of the system under heavy load. Processes that are aged first move from local to global memory. If needed again, they are quickly swapped back into local memory where they execute at full speed. Pages that continue to age are eventually swapped from global memory to disk.

The standard UNIX process, vhand, is responsible for swapping processes from global memory to disk. A new process manages swapping from local memory to global memory. It uses the same type of least-recently-used (LRU) scheme as the vhand process.

A block copy engine, a special-purpose direct memory access device, provides hardware assistance to move blocks of data across the reliable system bus (RSB), which connects the CPUs to global memory. The block copy engine can copy data much more quickly than the CPU can move it. The memory management code utilizes the faster block copy engine whenever it swaps pages back and forth between local and global memory.

The block copy engine is also used to improve the performance of bcopy() and bzero(), two frequently used kernel routines that copy and zero data buffers, respectively. The interface to the routines is unchanged, but the internals of the routine have been modified to make it aware of the option of using the block copy engine. Because these routines are used throughout the kernel, the performance improvement afforded by the block copy engine can be significant.

Fault Tolerance of the I/O Subsystem

The I/O subsystem of the Integrity S2 is composed of duplexed components that are checked so that they can be isolated in the event of a failure. These components provide redundant paths within the I/O subsystem and allow integration of VMEbus controllers into the Integrity S2 environment.

The hardware's self-checking logic detects any faults and allows isolation of a component before the fault corrupts data or other areas of the system. When a failure occurs, an interrupt is generated and the kernel's error recovery code takes the component offline. NonStop-UX is also responsible for rerouting any outstanding and subsequent I/O requests by way of an alternate path whenever a component has been moved offline.

The dual NonStop V+ bus, a Tandem implementation of a VMEbus, protects the data paths with the addition of parity. The bus interface module (BIM) and the NonStop V+ bus combine to check parity on data moving across the buses, to provide dual-port access to the buses, and to enhance fault tolerance by isolating the controllers from one another. Normally, a standard VMEbus passes signals through a string of controllers. The Tandem implementation uses a radial addressing scheme to route bus signals. Each of the controllers has an individual connection to the IOP and appears to be on a completely separate VMEbus from all the others. These changes are transparent to the controllers' hardware or firmware.

When a VMEbus controller is added to the system, the address mapping for that controller is kept on both IOPs. If the IOP being used for access to that controller is lost, the operating system can access the controller at the same address by way of the other IOP. This means that an IOP can fail while a device driver is in the process of accessing a controller, and the driver is transparently switched to accessing the controller through the other working IOP. This can be demonstrated on the system by starting up heavy I/O workloads and then pulling the active IOP. The BIM switches the controller to the other NonStop V+ bus and the system continues to operate uninterrupted.

The I/O subsystem uses two additional methods to preserve data integrity. Disk mirroring and disk checksums safeguard data being written to disk or reread from disk.

Disk Device Mirroring. Mirroring is a technique for protecting disk data by writing data to two different disk drives at the same time. Figure 5 illustrates how basic disk mirroring occurs.

On the Integrity S2, any disk partition can be mirrored to any other disk partition of the same size. Typically, disk data is written from memory to IOP_1, controller_1, disk_1 and simultaneously to IOP_2, controller_2, disk_2. All important data can be mirrored so that it is written to two different disk devices accessible through two completely separate paths. Mirrored disk partitions continue to provide access to data even if one disk fails.

Because writes to the two halves of the mirrored devices are done in parallel, the performance overhead for the mirroring is minimal. With disk mirroring, the performance of most applications improves. This improvement occurs because NonStop-UX also implements read optimization from the different disk partitions. Read optimization allows the operating system to select data from either of the two identical halves of a mirrored pair. It works by selecting the least busy device or by selecting the device on which the head is closest to the data to be read. As shown in Figure 6, the Integrity S2 system implements a number of different read optimization algorithms and allows any of the algorithms to be selected on a partition-by-partition basis.

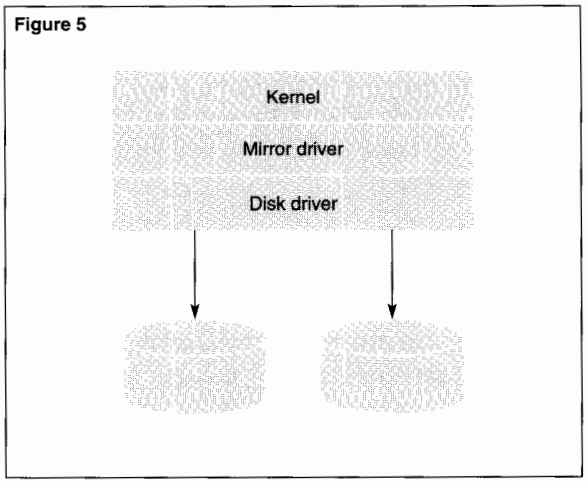


Figure 5.
Disk mirroring provides data integrity by protecting against disk and disk controller failure.

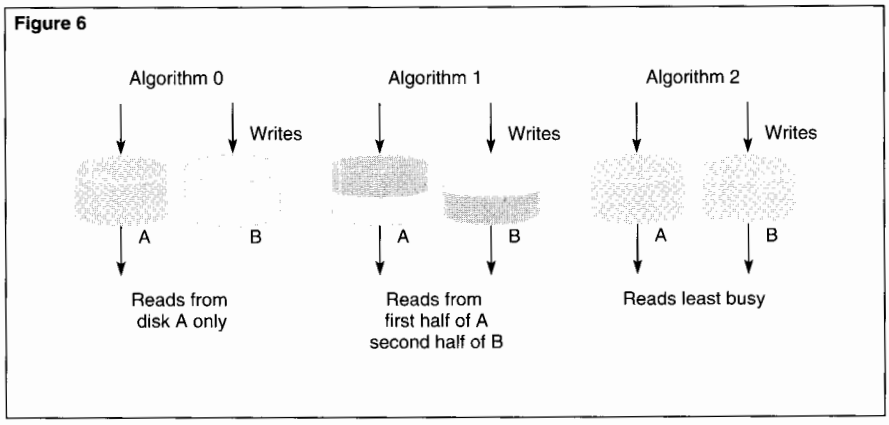


Figure 6.
Read optimization of mirrored disks is a technique used to improve system performance.

The disk device mirroring software is implemented in a manner independent of the device driver. This makes the mirroring software available to users who wish to integrate their own VME disk controllers and add their own standard drivers.

Disk Checksums. Although most of the I/O subsystem consists of self-checking components, the off-the-shelf VME controllers used in the Integrity S2 are not designed to contain their faults with self-checking logic. Therefore, it is possible for a controller that fails and corrupts data to go undetected in the system. The Integrity S2 protects itself against such failures by generating checksums on data that is written to disk. The checksums are checked when the data is reread from disk. This technique, called end-to-end checksums, ensures that the entire path from main memory to disk and back is checked. Any failure that compromises data integrity is identified. Because the data is mirrored, a correct version of the data can always be recovered.

Kernel reliability has been improved with enhanced error recovery.

Robustness Enhancements to UNIX

As with all standard UNIX implementations, the UNIX kernel is a single point of failure on the system. Although there are three processors executing the instruction stream, it is logically a single instruction stream. Therefore, a bug that causes the system to panic (shut down) or hang causes all three of the processors to panic or hang in exactly the same way.

To ensure the high availability and data integrity of the Integrity S2 system, the reliability of the kernel has been improved with a number of techniques. Most reliability has been gained through the evaluation and correction of many of the conditions that cause the kernel to panic. For further protection, the kernel is write-protected against errant or invalid processes. These enhancements have been implemented to the kernel while still maintaining compliance with SVID standards.

Error Recovery

UNIX is well known for the many calls to the panic() routine interspersed throughout the kernel. The panic() routine is often invoked after an assertion has been executed to determine if a dangerous condition exists. If something is amiss, the kernel usually chooses to invoke panic(), which flushes the buffer pool and shuts down the system.

Analyzing Panic Calls. Efforts to improve kernel reliability began with an extensive analysis of the panic conditions and assertions of the kernel. A database was created that includes information on over 800 panics and assertions in a standard System V Release 3 UNIX kernel.

A multidimensional value function was applied to determine the priority for implementing error recovery routines for the panics. The kernel was instrumented to determine how frequently the assertions were invoked, and the probability that each panic might occur was analyzed. These and several other metrics allowed prioritization of the list of panics and assertions. These tools can be applied to future UNIX releases and allow a quick identification of new panic conditions, which can be added to the database.

Choosing the Recovery Mechanism. Usually one can construct a recovery routine for an error condition. However, it may not always be wise to attempt recovery. An assertion that is satisfied is an indication that the system has been corrupted in some way. Although recovery from the error state is possible by, for example, removing a corrupted element from a linked list, the original cause of the data corruption may still exist.

This presents system users with a choice. Some users may value data integrity over availability; others may make the opposite choice. A configurable recovery mechanism (the /config file system, discussed below) was developed to be flexible enough to satisfy both sets of users.

Those users who prefer to preserve data integrity can configure the system to allow an immediate panic to be invoked after the first assertion that indicates a problem. This flushes the system state to disk, then enables a quick rebooting of the system on a fresh kernel image.

For those who need to maintain high availability, error recovery routines are invoked to allow the system to continue providing services even if it means a user process may be terminated to correct the problem. The system then enters a probationary state, where it lives for a period of time determined by the user. The system can be informed that if a specified number of error recoveries are attempted during the probationary period, the system is too unstable and a shutdown will occur so that the system can be quickly rebooted.

Subscription Services. Subscription services allow a subsystem to specify a recovery routine and request that the subsystem subscribe to it if an error occurs. Subscription services are available to any routines in the kernel. Multiple subsystem-specific recovery procedures can be defined for the same type of failure condition.

The NonStop-UX Panic Routine. The panic capability of the NonStop-UX operating system has been enhanced to ensure that data integrity is preserved if the system has to be shut down and rebooted. When a normal panic routine is executed, an attempt is made to flush the buffer pool and update the disks to a consistent state. This goal is often difficult to achieve because the panic routine is executing on a system that may have experienced some random memory corruption.

The NonStop-UX panic routine uses as little of the kernel as possible when it executes. A separate driver routine runs in polled rather than interrupt mode and uses data structures set aside for this purpose. As much data as possible is write-protected during the panic routine's execution. Key data structures, such as the superblocks, are checked for consistency before being written to disk to prevent writing corrupted data to the disks.

Hardware Protection of the Kernel

Hardware assistance was provided to protect portions of memory with write-protect random-access memory (RAM). The Integrity S2 uses special hardware memory-protection circuitry to prevent data loss or corruption of the kernel resulting from failures. In addition, because it is not uncommon for pointers in C to be used incorrectly, the write-protect RAM was used to protect kernel text from being overwritten. The write-protect hardware has a small enough granularity to be used to protect data structures as well as text, although the performance implications of write-protecting data often make this approach prohibitively expensive.

A unique feature of the Integrity S2 is its online serviceability.

If a user process is responsible for a write-protect violation, the process is terminated. In general, processes responsible for errant behavior are terminated while the system is kept available for the rest of the users.

Online Service

The ability to service the Integrity S2 online is one of the unique features of the system. One can easily service the Integrity S2 without losing availability.

The system's mechanical design facilitates the user serviceability of the system. All components and cables are accessible from the front. When the cabinet doors are opened, a user can replace any component without needing any tools. The term describing the components, customer replaceable units (CRUs), emphasizes that the Integrity S2 does not need trained service personnel.

Monitoring the Hardware and Software Subsystems

To facilitate online service, the Integrity S2 uses a pseudo-file system, called the /config file system, to provide status information for each hardware component and software subsystem running in the system. The system uses no disk space. It is maintained in memory and updated as device states and hardware configurations change.

The /config file system consists of one hardware and one software directory tree. The hardware directory contains files that correspond to each CRU in the system. The software directory contains files corresponding to performance statistics or software subsystems such as System V interprocess communications and the powerfail-autorestart subsystem.

One can obtain status information by calling a stat() routine or by sending an I/O control (ioctl) call[4] to the files in the /config file system. The real object corresponding to the file can also be operated on by opening any file and sending it an ioctl. The /config file system provides an elegant and flexible way to enhance the system's user serviceability in a way completely consistent with the UNIX model.

[4]An ioctl is a UNIX system call used to set certain attributes of I/O devices or to override a device's default settings.

Online Reintegration of System Components

The NonStop-UX operating system allows users to remove, replace, and reintegrate any of the active components within the Integrity S2 system. If any of the boards in the system fail, one can repair the system without scheduled downtime. CRUs can be replaced online while applications are running. The reintegration process is transparent to applications and system users.

Reintegration of CPUs. A CPU failure generates an event log message. Typically, the system administrator is notified of the failure by way of a status screen displaying the event log messages. The administrator can choose to enable a dial-out procedure[5] on an event-by-event basis. If enabled, the system automatically dials out for assistance when specified events occur.

Soft memory failures are corrected online by a software process called the memory scrubber. This process eliminates memory errors in dynamic random-access memory (RAM) by rewriting valid data to the failed address. However, if a hard memory failure occurs in a CPU, the CPU is voted offline and must be replaced.

The operating system manages the reintegration process of the module. First, the new CPU runs its power-on self-test (POST). Then the other CPUs receive notification that the new CPU has completed its POST. All three CPUs use the block copy engine to copy each local memory page out to global memory and back again. Because two of the CPUs still have valid data, that data replaces the bad data in the third CPU. Then all CPUs restart from the point in the processing stream where they were previously executing. This entire process takes approximately one second on an 8-megabyte local-memory CPU.

Reintegration of TMRCs. The memory scrubber process also protects both the local and global memories from transient soft RAM errors. As the scrubber copies data back and forth between local and global memory, it can uncover latent parity errors. Also, the primary-secondary swapper periodically swaps the primary and secondary TMRC so that the CPUs alternately read from both TMRCs and can uncover errors specific to one of the modes of operation.

The scrubber cannot correct a hard failure in a TMRC. If a global memory error occurs, the failed TMRC needs to be replaced.

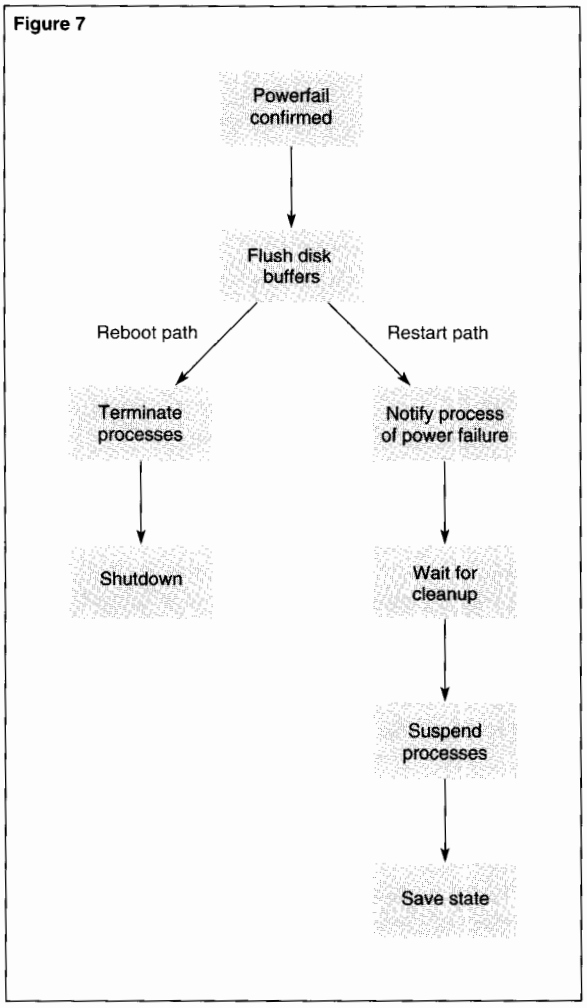
Once a new TMRC is installed, the valid data from the other TMRC must be copied to it. This procedure happens in the background while normal processing continues. Data is simply read from the primary TMRC and written back to both components. One can trade off performance with the speed of reintegration by varying the block size of the memory copy and the time interval between block copies.

Reintegration of IOPs. When an IOP fails, all of the controllers on that IOP are automatically switched by way of the BIM to the other IOP. Processing continues uninterrupted. When an IOP is replaced, the controllers that were connected to it are automatically moved back to connect with the new IOP. This restores balance to the system for both performance and fault tolerance.

The system can switch controllers back and forth between the IOPs because the address space used by a controller is always reserved on both IOPs. The tables that determine the address mappings of the VME controllers are kept in global memory.

[5]The diagnostic system dials out to the Tandem Online Support Center.

Figure 7.
This flow chart illustrates the steps taken by the kernel in response to a power outage.



Recovery From Environmental Failures

A power failure is the most common environmental failure, but flooding, earthquakes, tornadoes, and other natural disasters can also damage a computer system. Failures due to loss of power and overheating are the most manageable ones. Typically, power failures are transient, lasting at most a few minutes. Even a transient power failure will cause a non-fault-tolerant system to lose availability. If the system is running UNIX, chances are that data will also be lost because the disks are not kept in a consistent state.

The Integrity S2 supports continued operation through transient power failures. If power is lost for longer than a few minutes, data integrity is preserved by initiating a powerfail shutdown procedure. When power is resumed, an automatic restart procedure allows programs to be restarted where they had been stopped.

Powerfail Shutdown Procedure

Each of the cabinets in the Integrity S2 houses two bulk power supplies, which normally distribute power to the CRUs throughout the cabinet. Each cabinet also houses two batteries. If external power is lost, the two batteries immediately switch in to power the entire system. The batteries are effective for about 8 minutes.

When power is lost, the two bulk power supplies typically transition somewhat asynchronously from a good state to a bad state. The analog nature of the power supplies causes them to transition back and forth for a few milliseconds. A kernel powerfail process is notified of all transitions so that it can distinguish between a failing bulk and an external power loss. Once the powerfail process determines whether the failure is a permanent or a transient outage, the kernel decides which action to take.

If a single bulk has failed, a message is logged to the maintenance subsystem so that the bulk can be replaced. The system continues to operate in the event of a bulk failure. If power has been lost, the shutdown procedure is initiated. Figure 7 illustrates the steps in this procedure.

The administrator can mark certain processes to be terminated upon power failure, but the default action is for all processes to be saved so they will restart at exactly the same point in the instruction stream when power is resumed. No process needs to have special code to survive a power failure.

The system first notifies all system processes of the power failure by signalling them. This allows users to write applications that catch the powerfail signals and perform any necessary cleanup before a shutdown. It also allows applications to perform initialization routines, such as password verification, before resuming.

The buffer pool is flushed to disk in order to ensure that the disks are in a safe and consistent state when the system is shut down. Many of the VME controllers have memory that maintains important state information. This information is copied into global memory and written to a special powerfail partition on disk. The image of memory is then written to the same partition.

At this point, the kernel powerfail process turns off the system. The batteries have been sized so that this shutdown procedure can succeed with only one working battery. Because power failures are fairly common in most environments, the system considers them expected events rather than faults. The Integrity S2 can handle a power failure in the presence of a single fault, even if that fault is one of the batteries. The system can also withstand two consecutive power failures, an all too common occurrence.

Overheating is a second common cause of power failures. The Integrity S2 has bulk power supplies with internal heat sensors so they can notify the kernel if they begin to overheat. The Integrity S2 handles overheating with the same software technique it uses when managing a power failure.

Automatic Restart Procedure

When power is restored, the controller states are restored from the powerfail partition. The image of memory is then restored from disk to both local and global memory. Applications can then resume processing at the same point where they stopped when the power failure occurred.

A process can choose to catch the signal that is delivered when the power is restored. Then the application can execute some special logic. For example, the application can invoke its own security mechanism by prompting the user for a password when the power is restored. Power failures can last for several hours, and the people running certain applications may have left the area. Suppose the resumed application was waiting for input from a data entry screen. Such a security feature could prevent an unauthorized person from entering or retrieving data.

Conclusion

The NonStop-UX operating system provides a standard UNIX System V implementation with advanced features that support the fault-tolerant hardware, allow system networking, and improve the robustness of the UNIX kernel. Any user can perform online service. For environmental failures, the operating system manages shutdown and restart procedures to protect data integrity. The Integrity S2 is the most successful implementation to date to combine the availability and reliability of a high-performance fault-tolerant hardware architecture with a fully conforming UNIX environment.

Peter Norwood joined Tandem in 1983 as a software designer. He worked as both a designer and a manager on various portions of the Integrity S2 project. Peter holds a BA from Haverford College and an MSEE from the University of Texas at Austin. He is currently vice president of software development at Tivoli Systems in Austin, Texas.

Enhancing System Security With Safeguard

Each computer application is unique, with unique security requirements. To make a computer system secure, users need to implement their security policy in a manner that suits the unique requirements of each application. Safeguard™, the Tandem™ system protection product, can be the foundation of the system's security design. Safeguard enhances the security of Tandem systems in several areas, including authentication services, disk file security, protection of processes and devices, and auditing.

This article describes the basic elements of Safeguard, discusses the security enhancements offered by Safeguard, provides examples of the uses of Safeguard, and gives a brief overview of the Safeguard environment. The article assumes that readers are familiar with the security features provided by the Tandem Guardian™ 90 operating system. The article is based on the C22 release of Safeguard.

Developing an Application Security Plan

Users should develop the Safeguard configuration as part of a larger security plan, just as they would develop an application. Typically, security development involves four distinct phases: policy statement, assessment, design, and implementation.

Initially, the upper management of an organization issues a security policy, clearly stating the organization's security requirements and its degree of commitment to achieving the security goals it has outlined. This step plays an important role in eliminating organizational barriers later on.

During the assessment phase, a security administration team studies the application and its environment to identify which users need which forms of system access. The team also decides which forms of protection are appropriate and documents its conclusions in a security assessment report.

Next, the security team designs the security plan, including the Safeguard configuration. If possible, the team also tests the design. Finally, the team implements the security design, usually one part at a time, including provisions for addressing any unintended problems that may occur.

Basic Elements of Safeguard

A computer security system should provide three basic forms of protection. *Authentication* permits the system to identify individual users. *Authorization* allows the system to control which users are granted which access privileges. *Auditing* provides a way to trace illicit accesses, verify that no illicit accesses have occurred, and record events to assist in detecting fraudulent activity. Moreover, the computer security system should be flexible enough to protect the system without impairing the everyday work of users.

Safeguard has been designed to achieve these goals. For authentication services, Safeguard uses the Guardian 90 concept of the individual user ID, thereby ensuring full compatibility with systems that are not running Safeguard. In addition, Safeguard provides its own authorization and auditing services. Each functional area identified by Safeguard provides the authentication, authorization, and auditing services required by that function. For example, the *processes* function allows the security administrator to authorize and audit actions involving processes. Many Safeguard functions are based on Guardian 90 concepts such as disk files, users, devices, terminals, and processes. In addition, Safeguard introduces three new concepts to Guardian 90 systems:

- *Access control lists* (ACLs) can be assigned to existing system objects or groups of objects, providing both authorization and audit control.
- *Objecttypes* extend ACL functionality by allowing the security administrator to control the creation of new system objects such as disk files and processes.
- The *Groups* feature extends Safeguard's protection services to the management of the Safeguard system itself.

Access Control Lists

Safeguard uses ACLs to form a Lampson model of security. A Lampson model is a matrix in which subjects (users) form the rows and objects form the columns (Dynamic Protection Structures, 1969). Each entry in the matrix defines the types of access allowed for the given subject and object. A Safeguard ACL gives a column view of the Lampson model, defining which users are granted which types of access to an object.

Figure 1.
Subvolume security matrix.

Figure 1

Subvolumes

User IDs	$SYSTEM						$AUDIT			$TEST		$DATA			
	STARTUP	SAFE	SYSnn	SYSTEM	TANDEM SUBVOLS	<OTHER>	AUDIT	SAFE	<OTHER>	SAFE	<OTHER>	DATA	OBJ	SAFE	<OTHER>
SEC.ADMIN (1,255)	O	O	O	O	O	—	O	O	—	O	O	O	O	O	—
APPL.USER1 (2,1)	—	—	—	—	—	—	—	—	—	—	RW EP C	—	—	—	—
APPL.USER2 (2,2)	—	—	—	—	—	—	—	—	—	—	RW EP C	—	—	—	—
APPL.DATABASE (2,255)	—	—	—	—	—	—	—	—	—	—	—	RW EP C	RW EP C	—	—
SUPER.OPER1 (255,1)	RW EP C	—	RW EP C	RW EP C	RW EP C	—	RW EP C	—	—	—	—	—	—	—	—
SUPER.OPER2 (255,2)	RW EP C	—	RW EP C	RW EP C	RW EP C	—	RW EP C	—	—	—	—	—	—	—	—
SUPER.SYSMGR (255,100)	RW EP C	—	RW EP C	RW EP C	RW EP C	—	RW EP C	—	—	—	—	—	—	—	—
SUPER.SUPER (255,255)	RW EP CO	RW EP CO	RW EP CO	RW EP CO	RW EP CO	—	RW EP CO	RW EP CO	—	RW EP CO	RW EP CO	RW EP CO	RW EP CO	RW EP CO	—

"—" = no access

Table 1.
Possible forms of the user list in an access control entry.

Safeguard user list	Description
140,145	Local user 140,145 only
*.140,145	Remote and local user 140,145 only
140,*	Local group 140 only
.140,	Remote and local group 140 only
,	Any local user
.,*	Any user

Figure 1 shows a sample Lampson security matrix in a typical production system. In this example, the objects are disk volumes, subvolumes, and files. Each ACL may contain *access control entries*. Each access control entry has three parts: a user or group of users to which the entry applies, a list of access authorities, and, optionally, the DENY attribute.

User Lists. Users or groups of users can be listed in an access control entry in one of six ways. Table 1 shows examples of the six types of user lists.

Access Authorities. The types of access authorities include read (R), write (W), execute (E), purge or stop (P), create or start (C), and owner of the object (O). The meaning of each access authority may differ depending on the class of object being protected. For example, for both the *disk file* and *process* object classes, R and W mean the authority to open a file or process for read and write access and O the authority to alter or delete the Safeguard protection for the file. However, for *disk files*, C, P, and E refer to create, purge, and execute authorities, whereas for *processes*, C refers to creating a process, P to stopping a process, and E has no meaning.

DENY Attributes. All users but SUPER.SUPER are denied access to an object protected by Safeguard unless an access control entry specifically grants access. If the DENY attribute is included, the access control entry disallows access. Within a single ACL, if more than one access control entry refers to a single user ID and one of those entries contains the DENY attribute, the access is always denied.

Interpreting ACLs. The security of an object is determined by the combination of access control entries in that object's ACL. It is also possible in Safeguard for several ACLs to protect a single object.

Figure 2 shows a sample ACL with 14 access control entries. Each entry contains a user list (as described in Table 1), a list of access authorities, and, for three entries, the DENY attribute.

The ACL in Figure 2 is not typical, but it demonstrates all the possible types of access control entries. A single user ID can be affected by six different entries in an ACL. For example, the following entries affect user ID 160,027:

```
   160,027
\*.160,027
   160,*
\*.160,*
     *,*
  \*.*,*
```

Figure 2

User ID List	DENY Attribute	Access Authorities
001,001	DENY	R
002,255		W
160,027		O
*.000,040		C, O
*.020,033	DENY	R, E, O
*.160,027		W, C
001,*		R
103,*		R, E
160,*	DENY	W
.020,		R, E, O
.160,		P
.244,		R, W, E, P, C
,		R
.,*		E

Figure 2.
A sample access control list (ACL).

To determine the types of access user ID 160,027 is allowed in this ACL, one must combine all the access control entries that apply to this user ID, remembering that DENY entries take precedence. In this ACL, the composite security for 160,27 is (R,E,P,C,O). For more information on ACLs, see the *Safeguard Reference Manual*, the *Safeguard User's Guide*, and *The C10 Safeguard Primer*, Appendix C.

Objecttypes

An objecttype is a class of objects protected by Safeguard. Each objecttype has its own unique protection characteristics.

Users. In computer security terminology, users are usually called *subjects*. In Safeguard, users are also an objecttype.

Figure 3.
Sample user configuration.

Figure 3

```
GROUP.USER          USER-ID          OWNER       LAST-MODIFIED        LAST-LOGON        STATUS
APPL.USER2            2,2              1,255     23JAN91,   11:57    12MAY91,  12:19     THAWED

   USER-EXPIRES                        =      *  NONE  *
   PASSWORD-EXPIRES                    =  02JUN91,      0:00
   PASSWORD-MAY-CHANGE                 =  13MAY91,      0:00
   PASSWORD-MUST-CHANGE EVERY          =  30 DAYS
   PASSWORD-EXPIRY-GRACE               =  15 DAYS
   FROZEN/THAWED                       =  THAWED
   STATIC  FAILED LOGON COUNT          =       45

   AUDIT-ACCESS-PASS = REMOTE              AUDIT-MANAGE-PASS = ALL
   AUDIT-ACCESS-FAIL  = ALL                AUDIT-MANAGE-FAIL = NONE

   CI-PROG =  $SYSTEM.SYS03.TACL
   CI-NAME =  $CM25
   CI-SWAP =  $SWAP
   CI-CPU  =  2
   CI-PRI  =  150
   CI-PARAM-TEXT = "1"
```

Figure 3 shows an example of the Safeguard configuration for one user. It lists the settings of the authorization parameters for an individual user. One can obtain the list by using the INFO USER command in SAFECOM, the interactive tool for communicating with Safeguard. For a complete description of all the parameters listed in this figure, see the *Safeguard Reference Manual* and the *Safeguard User's Guide*.

Disk Files. The disk file objecttype includes any permanent Tandem disk files except optical files and SQL objects. In addition to the standard R, W, E, P, C, O access attributes, each disk file has three special attributes: Progid, Clearonpurge, and License.[1]

Volumes and Subvolumes. The volume objecttype applies to every non-optical disk in the system. The subvolume objecttype applies to every legal subvolume name on those disks.

[1]The attributes Progid, Clearonpurge, and License apply to Tandem disk files on all Guardian 90 systems. For more information, see the *File Utility Program (FUP) Manual* and the *Safeguard Reference Manual.*

The volume and subvolume objecttypes have two purposes. They control which users may create files on a disk or subvolume, and they provide an alternate way to protect disk files. Because of the second feature, a disk file can be protected simultaneously by a volume, subvolume, and disk file protection record.

Subvolume protection also allows users to secure data managed by NonStop™ SQL, Tandem's distributed relational database management system. By grouping NonStop SQL tables and views in separate subvolumes, users have the flexibility of specifying all of the functions provided by an ACL for each group of NonStop SQL objects.

Processes and Subprocesses. Safeguard provides two types of process protection. The first type controls which users may open a named process for read or write access. This protection can be implemented for a process (such as $S), a subprocess (such as $S.#PRINT2), or a process and subprocess simultaneously.

The second type of process protection controls the creating and stopping of processes. It can be implemented for an individual process name, all named processes (as a single entity), and all unnamed processes (as a single entity).

Devices and Subdevices. Device protection is similar to the type of protection that controls which users may open a named process. Using ACLs, the security administrator can control which users may open a device or subdevice such as a printer, terminal, or communications line.

Objecttype records. One can also configure an *objecttype protection record* for each of the objecttypes mentioned above. The objecttype ACL controls who may create protection records of that objecttype. For example, the OBJECTTYPE USER record determines which users may add new user IDs to the system.

Groups

A group is a list of user IDs that are granted certain Safeguard control capabilities. Safeguard allows two groups of users, SECURITY-ADMINISTRATOR and SYSTEM-OPERATOR. Users in the SECURITY-ADMINISTRATOR group may make changes to the Safeguard configuration. This permits control of certain Safeguard options, such as which actions to audit or the decision to encrypt user passwords, but does not affect control of the settings on individual ACLs. SYSTEM-OPERATOR privileges involve managing the audit trail files themselves (for example, managing their location and size).

Enhancements to User Security

Safeguard provides valuable enhancements to the user ID protection offered by Guardian 90. A Guardian 90 system without Safeguard offers basic user identification and authentication functionality. Safeguard uses the same user name and user ID conventions as Guardian 90, thus maintaining compatibility, but it offers several additional features such as user ID management, authentication protection, and password change control.

User ID Management

Safeguard offers added flexibility in assigning user management tasks. A Safeguard ACL controls which users may add new users. Also, each user ID is assigned an *owner*, who is allowed to modify or delete that user ID configuration.

A user ID can be configured with an expiration date, permitting safe use of temporary user IDs. In addition, a user ID can become *frozen*, preventing all logons by that user ID. Freezing is useful for protecting powerful user IDs (such as SUPER.SUPER) that are not needed for day-to-day use or for temporarily preventing access to a certain user ID.

Authentication Protection

In Guardian 90, three consecutive failed attempts to log on as one user ID causes the process attempting the logon to be suspended for 60 seconds. This protects against illicit logons through the trial-and-error method.

Safeguard makes this feature more flexible by allowing the security administrator to configure the number of attempts and the duration of the suspension through the global parameters AUTHENTICATE-MAXIMUM-ATTEMPTS and AUTHENTICATE-FAIL-TIMEOUT. Thus, one can configure a more or less strict penalty for failed logons, according to the security policy.

A third parameter, AUTHENTICATE-FAIL-FREEZE, provides additional protection. If this parameter is set to ON, a user ID will become frozen if the number of consecutive failed logon attempts for that user ID exceeds AUTHENTICATE-MAXIMUM-ATTEMPTS plus one. This parameter sharply limits the total number of invalid logon attempts that can be made against one user ID.

Assume, for example, that AUTHENTICATE-MAXIMUM-ATTEMPTS is 3, AUTHENTICATE-FAIL-TIMEOUT is 4 minutes and AUTHENTICATE-FAIL-FREEZE is OFF. If three consecutive invalid logon attempts are made against a given user ID, the process that made the third attempt is suspended for four minutes. Thereafter, any process that makes an invalid logon attempt against that user ID is also suspended for four minutes. This continues until a valid logon is made for that user ID, when the FAILED-LOGON-COUNT is reset to 0.

Table 2.
User ID settings in a typical system.

Function	Username	UserID	Password must change	Status
Security administrator	SEC. ADMIN	001,255	30 days	THAWED
Application developers	APPL.USER1	002,001	30 days	THAWED
	APPL.USER2	002,002	30 days	THAWED
Database owner	APPL.DATABASE	002,255	0 days	FROZEN
System operators	SUPER.OPER1	255,001	30 days	THAWED
	SUPER.OPER2	255,002	30 days	THAWED
System manager	SUPER.SYSMGR	255,100	30 days	THAWED
For emergency only	SUPER.SUPER	255,255	0 days	FROZEN

Now suppose AUTHENTICATE-FAIL-FREEZE is set to ON. After the third consecutive invalid logon attempt, the process is suspended. If the fourth logon attempt for this user ID is invalid, Safeguard automatically freezes the user ID. The user ID is frozen even if all logon attempts are entered at different terminals.

Password Change Control

In Guardian 90, passwords are optional for each user, and the password content has no restrictions. Safeguard offers significantly enhanced password control.

Password Content. Three parameters control the content of a user password. PASSWORD-ENCRYPT provides one-way Data Encryption Standard (DES) encryption of all passwords. PASSWORD-MINIMUM-LENGTH forces newly entered passwords to contain at least a set number of characters. PASSWORD-REQUIRED forces all users, including SUPER.SUPER, to enter the correct password when logging on to another user ID. Tandem recommends using these parameters to reduce the risk of users illicitly logging on to other user IDs.

Password Change. To reduce the risk of passwords becoming known to others, passwords should be changed periodically. Safeguard provides three parameters that, when used together, assure frequent password changes.

The PASSWORD-MUST-CHANGE parameter determines the lifetime of each password. For example, if this parameter is set to 30 DAYS and the user's password is not changed for 30 days, the password expires, and the user may not logon again until the password is changed.

The PASSWORD-MAY-CHANGE parameter determines the period of days before the password expiration date during which a password may be changed. Together with PASSWORD-MUST-CHANGE, this parameter defines a password-change window. For example, if PASSWORD-MAY-CHANGE is set to 20 DAYS and PASSWORD-MUST-CHANGE is 30 DAYS, the password may be changed only during the 20 days before the password expires. (Thus, the password cannot be changed for 10 days after a password change.) This prevents a user from changing the old password to a new password, then immediately changing it back to the old password.

PASSWORD-HISTORY specifies the number of past passwords saved for each user ID in the password history file. Safeguard does not accept any new password that matches a password for that user ID in the history file. This feature, when used with PASSWORD-MAY-CHANGE, prevents a user from reusing the same passwords.

Table 2 shows an example of the user IDs and password values in a typical production system. It illustrates how the PASSWORD-MUST-CHANGE and *status* features might be used, assuming PASSWORD-MAY-CHANGE and PASSWORD-HISTORY have been implemented on a global basis. In Table 2, two user IDs, APPL.DATABASE and SUPER.SUPER, are not in daily use by any user and are not needed for day-to-day production. Thus, they are kept frozen until needed. All other user IDs are thawed. Regular changing of passwords is enforced through the PASSWORD-MUST-CHANGE parameter. PASSWORD-MUST-CHANGE is not applied to the frozen user IDs because they offer no risk of an illicit logon.

Password Expiration. One can configure Safeguard to act as an authentication process at selected terminals. If so configured, Safeguard warns the user when he or she logs on of an impending password expiration. When the password has expired, Safeguard allows the user to change the password and proceed. Only a security administrator can correct an expired password on terminals not configured with this option.

Illicit Logon Detection. When Safeguard is configured as a terminal authentication process, it displays the time of the last logon at each new logon. This feature alerts users to unauthorized access to their user ID.

Subvolume-Based Disk File Security

Safeguard offers additional security advantages by enhancing Guardian 90 protection of disk files. In Guardian 90, every disk file is protected by a security string that includes read, write, execute, and purge (RWEP) attributes. Guardian 90 has no way to control file creation. Safeguard provides RWEP protection as well as file creation control on a disk volume, subvolume, or individual disk file basis.

With this flexibility, Safeguard offers many possible strategies for disk file protection. Because each strategy has advantages and disadvantages, it is important to select an appropriate strategy before designing a Safeguard configuration. This article discusses a subvolume-based method of disk file protection because such a strategy is easy to manage, protects all the files in the system, and aids other system management tasks.

In the subvolume-based approach, the security administrator needs to monitor only subvolume protection records because disk file records are controlled by individuals and disk volume records are static. Monitoring as few as a dozen subvolume records per disk is much easier than keeping track of hundreds or thousands of disk file records per disk. Also, users tend to change subvolume records less often than disk file records.

The subvolume-based approach can protect every file by permitting files to be created only in subvolumes with a Safeguard protection record. Moreover, this approach can make disk space management easier by controlling the number of subvolumes and associating each subvolume with a user ID. This simplifies disk space reports from the Tandem Disk Space Analysis Program (DSAP) and allows system managers to keep unauthorized files off of disks where space is critical.

Subvolume Security

In the subvolume-based security strategy, Safeguard is configured to protect disk files according to the following rules:

1. Safeguard checks to see if a disk file record exists for the file being accessed. If so, the ACL on that file applies.
2. If rule 1 yields no record, Safeguard checks for a subvolume record and applies that ACL.
3. If rules 1 and 2 yield no record, Safeguard checks for a volume record and applies that ACL.
4. If rules 1 through 3 yield no protection record and the Safeguard global parameter ACL-REQUIRED-DISKFILE is set to ON, access to the file is denied.

In the subvolume-based strategy, the setting in the subvolume record (if it exists) overrides the setting in the volume record, and the setting in the disk file record (if it exists) overrides the settings in both the subvolume and volume records.

To apply this strategy one could, for example, add a volume record to deny create access to all users. Next, one adds a subvolume record for each user in the system, granting create access for each user only to his or her own subvolume. (The subvolume security supercedes the volume security.) The subvolume records also determine the security of the files. Finally, one adds disk file records for files that require different security settings than the subvolume record settings.

Subvolume Security Matrix

The subvolume-based strategy assumes that one can group most disk files into subvolumes with identical security requirements. As shown in Figure 1, the security administrator can form a subvolume security matrix. Each entry in this figure shows the desired access authorities (if there are any) for a given user ID against a given subvolume. Without Safeguard's subvolume protection, each disk file would need to be listed separately.

The security administrator gives each subvolume listed in Figure 1 a Safeguard ACL to enforce the desired security settings. For subvolumes such as $SYSTEM.SYSTEM in which individual disk files require unique security settings, Safeguard disk file protection records are used in place of subvolume protection records.

Implementing the Subvolume Strategy

Although the subvolume-based strategy is a relatively simple concept, its implementation can be difficult if it does not suit the requirements of the application. Therefore, it is important to consider several disk file protection strategies before choosing one. Information about implementing the subvolume-based strategy and other disk file protection strategies (including important special considerations) appears in the *Safeguard User's Guide* and *The C10 Safeguard Primer.*

Protection of Pathway

Another way in which Safeguard protects the system is by protecting the processes that access the database. Assume, for example, that an application developed by the Tandem Pathway transaction processing system executes under its own user ID and the database files are secured for access only by this user ID (a common configuration). This configuration seems to secure the database, but in fact the data is vulnerable to unauthorized access.

Most user-written Pathway servers accept OPEN requests from any process or user ID and perform any operation if the message is in the correct format. This feature is good for flexibility but bad for security. It allows the data files to be read and modified by any user ID that sends the proper messages to the Pathway servers.

One can counteract the threat of indirect database access by using the process name protection offered by Safeguard together with the Pathway configuration. This is a two-step process:

1. Configure the Pathway process names, giving each server and terminal control process (TCP) a specific name.
2. Protect the process names. In Safeguard, secure each process name so that only the Pathway user ID may start or open any of the named processes.

In Safeguard, one can configure the process name protection so that all operators can stop and start the Pathway processes. This permits operators to perform necessary day-to-day functions without compromising system security.

Auditing With Safeguard

Safeguard allows the security administrator to audit security events on an object-by-object or system-wide basis. Moreover, Safeguard distinguishes between types of accesses (local versus remote, granted versus denied, and read versus update). The security administrator can use certain options that define the audit's effects on system operations.

Deciding Whether to Audit

Once the Safeguard configuration has been designed, one must decide which objects and subjects require auditing. One should consider the slight performance penalty that occurs when an action is audited and weigh that penalty against the loss of potentially valuable audit information when an action is not audited. The security administrator should make this choice in consultation with a system auditor and in accordance with the organization's security policy.

The Cost of Auditing. A performance cost is associated with writing an audit record to disk, but for most systems the cost is negligible except during system startup and shutdown. This is because the operations that Safeguard audits (opens, logons, process starts and stops, file creates and purges) are minimized when one tunes a system for performance. In a properly tuned system, there usually will be little writing of Safeguard audit records during normal operation.

The Benefits of Auditing. When tracking a possible security breach by using the audit trails, one often finds that the most innocuous piece of audit information is the key to identifying the source of the break-in. Whenever one excludes an object from audit, one risks that a vital piece of information will not be logged. Therefore, a good rule of thumb is to audit every action unless there is a good reason not to audit it.

Audit Service

The Audit Service adds several features to Safeguard. The Recovery feature allows the security administrator to define the relative importance of auditing to the system as a whole. The Recovery feature specifies the action Safeguard takes if the system fails to write audit (for example, if the audit trail is full). There are three Recovery options.

- The SUSPEND AUDIT option stops the Audit Service and sends error messages, but system operation is unaffected. This option is the default.
- The RECYCLE FILES option tries to write over the oldest unreleased audit trail. If this fails, Safeguard falls back to the SUSPEND AUDIT option. In both cases, Safeguard sends error messages.
- The DENY GRANTS option sends error messages and denies all actions that require audit except to members of the SECURITY-ADMINISTRATOR and SYSTEM-OPERATOR groups.

Most users will prefer one of the first two options. One should use the DENY GRANTS option only when having a potentially inaccessible application is preferable to having a running, unaudited system.

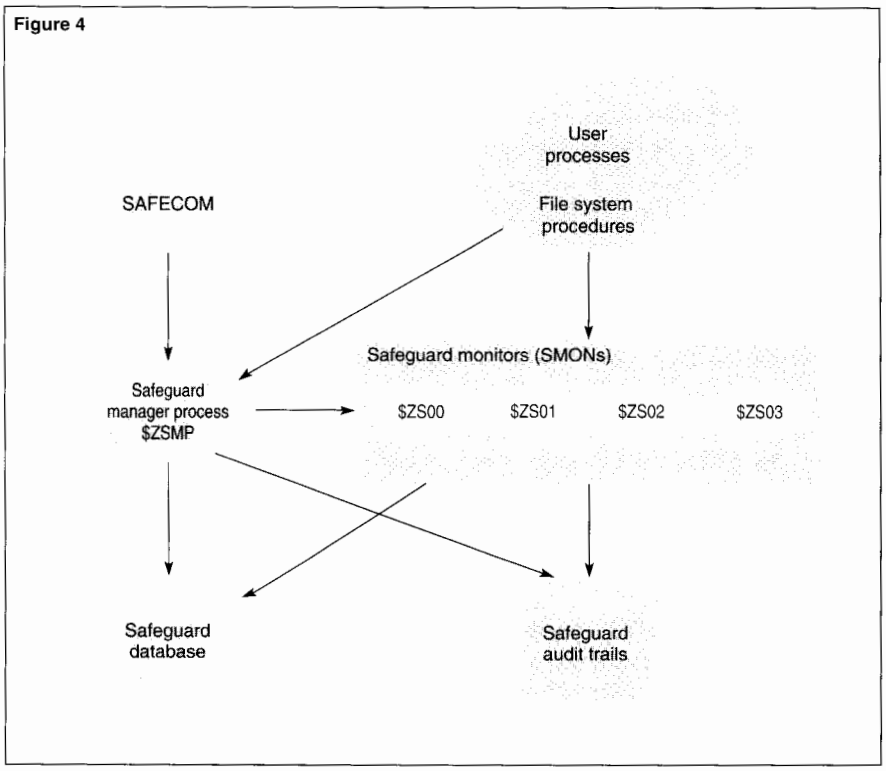


Figure 4.
Overview of Safeguard processes.

The Safeguard Environment

The Safeguard environment consists of a NonStop Security Manager Process (SMP), a Security Monitor (SMON) process in each processor, and a utility for communicating with the SMP called SAFECOM. SAFECOM is similar to other Tandem utilities such as PATHCOM, TMFCOM, or MEASCOM. The Safeguard configuration is kept in key-sequenced files on disk, and Safeguard audit is written to entry-sequenced files.

Figure 4 shows the interactions among the components of Safeguard. The following actions take place in the figure:

1. The SMP ($ZSMP) runs as a NonStop process, starts one SMON per CPU ($ZS00, $ZS01, and so on), sends each SMON a Safeguard configuration message, and restarts the SMON after a CPU reload.
2. The SMON receives authorization requests from user processes whenever one of these procedures is called: OPEN, NEWPROCESS, PURGE, CREATE, STOP, or RENAME.
3. The SMP receives authentication requests from user processes if the user process calls the procedure VERIFYUSER (to log on).
4. The SMP receives requests to read or modify the Safeguard configuration from SAFECOM processes. These messages may also come from backup, restore, and password processes.
5. The SMP and SMONs answer requests based on the information contained in the Safeguard database.
6. Depending on the configuration, the SMP and SMONs may log individual requests to the Safeguard audit trails.

In addition, a large part of Safeguard is embedded within the file system procedures and the rest of the Guardian 90 operating system. When Safeguard is running, the operating system will not complete an OPEN, PURGE, STOP, NEWPROCESS, RENAME, CREATE, CHANGE PASSWORD, or VERIFYUSER operation without first receiving an acknowledgment from Safeguard. Conceptually, one can view Safeguard as an operating system layer between the file system and the message system.

Figure 5 shows the relationship between Safeguard and Guardian 90. In a Tandem system, a message to a process must go through file system and message system procedures before arriving at its destination. When a file system request such as an OPEN is issued, Safeguard is consulted. If Safeguard rejects the request, it is returned to the user without being passed to the message system.

Safeguard users employ the SAFECOM utility to configure all the options described in this article. SAFECOM is the only utility new Safeguard users must learn. Because Safeguard operates at a layer below the file system, users do not need to change existing applications to run Safeguard.

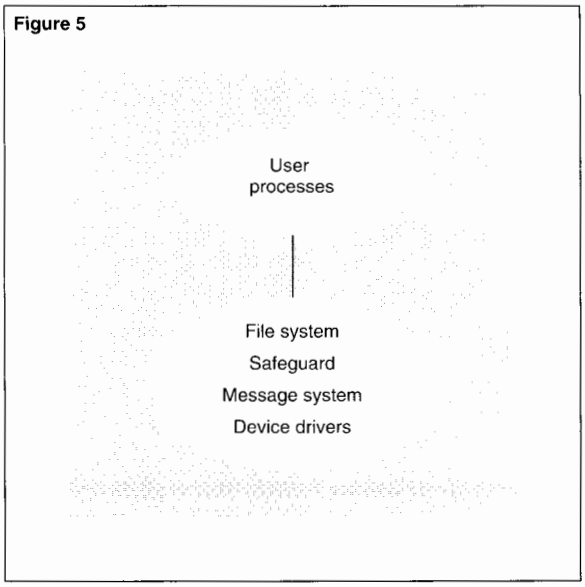


Figure 5.
Safeguard as a layer of the operating system.

Conclusion

Safeguard is a valuable tool for protecting a computer system. It offers authorization, authentication, and auditing features that enhance the security features of Guardian 90. The Safeguard features provide flexible control over system users, objects, and events.

One should remember that Safeguard is a security tool. It does not by itself constitute an implementation of a security plan. Once a security plan has been developed to meet the user's unique business requirements, Safeguard can play a critical role in implementing that plan.

Acknowledgments
I would like to thank Safeguard development and product management staff for their responsiveness and eagerness to help at all times; the European account analysts who have patiently worked with me to resolve Safeguard questions; and the members of Tandem's Security Special Interest Group, whose ideas, suggestions, and feedback provided a large portion of the material in this article.

Craig Gaydos is the Operating System section manager of the European Systems Support Group (ESSG). He has been with Tandem since 1984, originally as an account analyst and more recently as an ESSG specialist.

References

Dynamic Protection Structures. 1969. *Proceedings, AFIPS.* Vol. 35, pp. 27. Fall Joint Computer Conference.

File Utility Program (FUP) Manual. 1990. Tandem Computers Incorporated. Part no. 21664.

Safeguard Reference Manual. 1991. Tandem Computers Incorporated. Part no. 26191.

Safeguard User's Guide. 1991. Tandem Computers Incorporated. Part no. 26190.

The C10 Safeguard Primer. 1989. Support Note S89030. Tandem Computers Incorporated. Available from Tandem representatives upon request.

TLAM: A Connectivity Option for Expand

Expand™ data communications networking software connects Tandem™ systems in a geographically distributed network. Several Expand connectivity options support Expand services and provide connections to a number of networking standards and communication media, including 6700/6710 Fiber Optic Extension (FOX™/FOXII), leased lines, X.25 packet switching, and satellite.

A recent enhancement to the Tandem LAN Access Method (TLAM) subsystem now enables Expand software to share access to TLAM and operate over standard local area network (LAN) media. TLAM is an implementation of the Institute of Electrical and Electronic Engineers (IEEE) 802.2 Logical Link Control LAN standard. By providing a uniform interface between the physical LAN and upper-layer protocol products, TLAM eliminates the need to code applications for a specific type of LAN and provides another tool for implementing Tandem's open-standards-based approach to networking.

This enhancement, called Expand over TLAM, extends Expand connectivity most effectively to facilities where LAN technology is already, or is planned to be, installed. With no additional network modifications, Expand over TLAM is immediately available to connect Expand nodes over LAN lines. LAN technology and TLAM architecture provide full interconnectivity between system nodes, which simplifies network designs, obsoletes routing hops between intermediate nodes, and reduces equipment costs. TLAM can connect Expand nodes that have requirements not satisfied by other connectivity options, and it may offer improvements to systems using other types of Expand connections.

This article describes TLAM, the basic architecture of Expand, and how the two products relate functionally in an Expand environment to provide network connectivity. The article compares three other Expand connectivity options with Expand over TLAM to establish the most appropriate contexts for each option. Next, it discusses recent Expand enhancements that extend the fault tolerance and performance of Expand over TLAM. It describes the new features of the TLAM subsystem. Finally, the article presents Expand over TLAM configuration considerations and performance results from recent testing.

Expand Architecture

Expand software is an extension of the Guardian™ 90 operating system. Within an Expand network, any Guardian 90 system is referred to as an Expand *node*. Each node can support up to 63 Expand *paths* to other nodes in the network. An Expand path is a logical data path between two nodes that is controlled by an Expand process. An Expand network can accommodate up to 255 Expand nodes.

Expand connectivity options provide communication interfaces to the Expand network and support Expand operations. Three options available prior to TLAM to meet specific networking requirements are:

- FOX.
- Expand NetDirect.
- X.25 Access Method (X25AM).

These connectivity options are illustrated in Figure 1.

FOX and Expand NetDirect are direct connections to Expand nodes. Both X25AM and TLAM communicate with Expand NetNAM (Network Access Method), which enables an Expand network to exchange data with either a target LAN or X.25 network. Expand NetNAM directs an Expand process to send and receive data through a *network service provider,* an intermediate process that completes the connection between the Expand process and the target network. TLAM and X25AM are both network service providers.

Expand NetNAM is designed so that Expand can access standards-based networks (such as an IEEE 802.3 LAN or an X.25 network) without implementing the necessary protocols within the Expand process. Instead, the network service providers perform all protocol work required to pass the data through the standards-based network. However, using a network service provider requires data to travel through another process at each end of a connection. Compared to a direct connection (FOX and NetDirect), additional system resources are needed to move data between nodes. One must weigh these and other considerations when deciding where and how to use the different connectivity options.

The NetNAM connections between Expand processes and TLAM (or X25AM) are configured with SYSGEN, Dynamic System Configuration (DSC), or the Subsystem Control Facility (SCF). For detailed configuration information, see the *System Generation Manual for Expand*, the *Dynamic System Configuration (DSC) Manual*, the *SCF Reference Manual for Expand*, and the *Expand over TLAM Install Guide Support Note.*

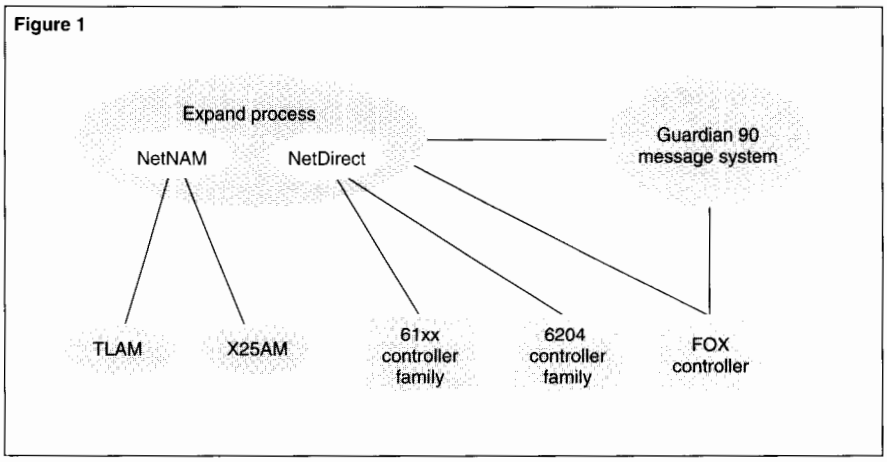
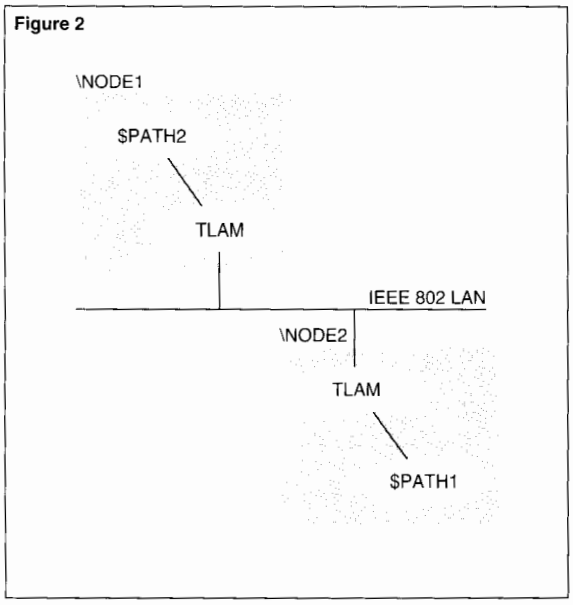


Figure 1.
The connectivity options available for Expand.

Figure 2.
A simple representation of two Guardian 90 nodes connected by way of TLAM. $PATH1 and $PATH2 are Expand processes.



Comparing TLAM With Other Expand Connectivity Options

When compared with other Expand connections, Expand over TLAM offers similar networking capabilities in some situations and superior benefits in others.[1] Figure 2 presents a simple Expand over TLAM configuration. This configuration is the model to keep in mind when comparing Expand over TLAM with the other connectivity options.

The FOX Option

The FOX connectivity option offers the most efficient and powerful way to connect local Expand nodes. The Guardian 90 message system and Expand combine to enable application and system processes to send data directly across dedicated optical fibers, bypassing the Expand process. FOX is well suited to configurations in which several Guardian nodes are located within the physical limits of the FOX cable.[2] Computer room facilities and clusters of buildings, or small "campuses," are good candidates for FOX connections.

The media bandwidth of FOX (40 megabits per second) is far greater than Expand over TLAM or any other connectivity option. Therefore, FOX connections offer the best throughput and response time. The maximum throughput of a FOX connection is 4.8 megabits per second. However, when comparing it with Expand over TLAM, one must weigh the higher cost of FOX hardware and consider whether the few technical limitations of FOX affect the intended installation. For example, FOX allows up to 14 nodes on a ring. FOX is not yet supported on the CLX™ line of processors. Also, a FOX connection is limited to the maximum distance of the cable, as it cannot be extended with bridge technology. Often, FOX and Expand over TLAM satisfy the same connectivity requirements for certain network designs. When FOX and Expand over TLAM are both appropriate options, FOX always offers better CPU utilization and response time.

[1]Performance varies with system configuration and host CPU type.

[2]The FOX cable distance limit is 1 kilometer for TXP CPUs and 2 kilometers for Cyclone CPUs. This limit applies to the distance between any two nodes in the FOX ring.

NetDirect Connectivity

Expand NetDirect has been the most common way to connect Expand nodes. The NetDirect option uses low-level Expand protocols that run directly over a designated controller (of the 6100 or 6204 controller family) and modem. Expand processes do their own point-to-point protocol over dedicated lines. In essence, nodes are viewed as having hard-wired connections.

NetDirect is designed to connect nodes that exchange large amounts of data and are located beyond the distance limitation of FOX. The maximum throughput of a NetDirect connection is 205 kilobits per second,[3] which is considered a midrange speed. Because NetDirect can use leased telephone line technology, there is virtually no limit to the distance between nodes.

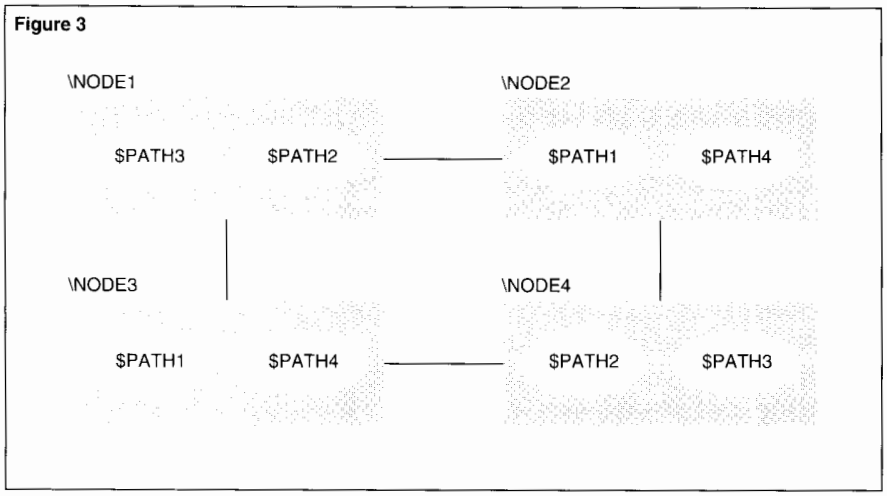
Expand over TLAM connections compare favorably with NetDirect connections in several areas:

- The LAN cabling requirements are less complex, more flexible, and less expensive.
- Expand over TLAM provides higher maximum throughput of bulk data transfers.
- The inherent shared bus topology eliminates any need for passthrough connections.
- A single physical TLAM connection can be shared by several Expand connections and with products other than Expand.

As shown earlier in Figure 2, a simple Expand over TLAM configuration consists of a LAN cable supporting multiple Expand nodes. Among the connected nodes, the Expand processes on one node have direct access to all the processes running in all other nodes without having to wire a separate physical connection to each node. Expand nodes can easily be connected to the LAN, disconnected, and reconnected to another LAN with a minimum of cabling, configuration, and expense.

A single NetDirect connection uses fewer CPU cycles and less memory to move data than a NetNAM connection. NetNAM requires an interprocess hop to reach the network service provider. However, because the physical layer of TLAM has a greater bandwidth than that of NetDirect, Expand over TLAM connections can achieve higher throughput. Also, TLAM becomes more efficient as it supports more connections, lowering the cost of CPU cycles TLAM uses to move each byte of data.

[3]This throughput was measured using Cyclone CPUs, 3604 controllers, and RS449 or V.35 null modems.



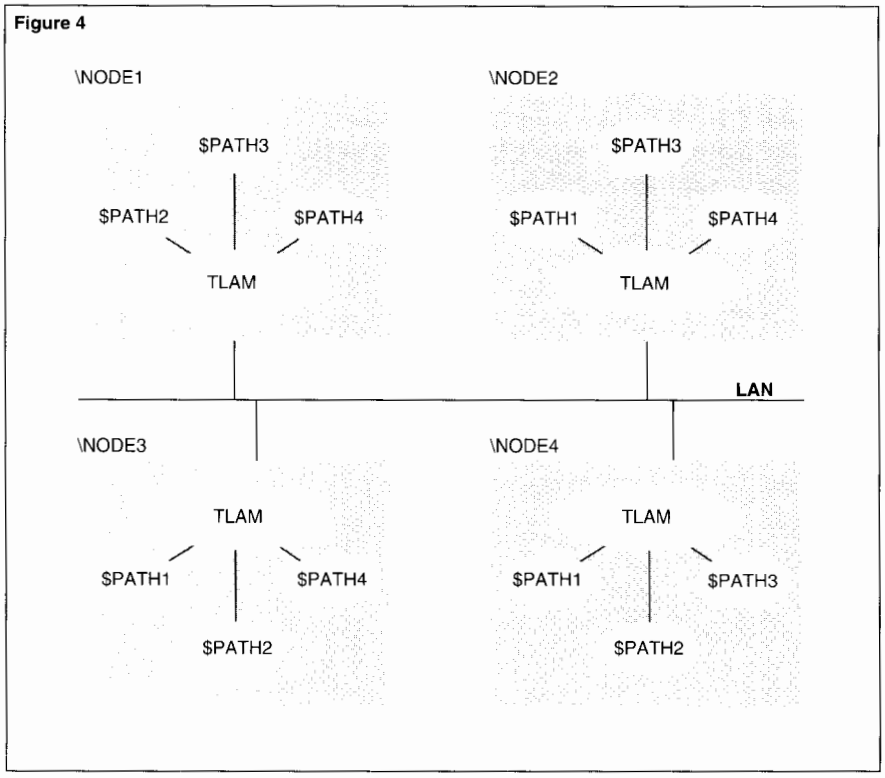


Figure 3.
These four Guardian 90 systems use Expand NetDirect.

Figure 4.
These four Guardian 90 systems use Expand over TLAM to create a fully interconnected network.

Figures 3 and 4 compare the fundamentals of NetDirect and Expand over TLAM configurations. Figure 3 shows a common four-node NetDirect configuration. Each node is physically connected to two other nodes and logically connected to the remaining node by way of passthrough hops. This network requires a minimum of 4 controllers, 8 modems, and 4 dedicated lines.

A passthrough hop literally passes data from one node through an intermediate node and on to the destination node. Routing is done at the intermediate node. Passthrough hops are necessary because the cost of using NetDirect to fully interconnect all nodes is prohibitive. The CPU cost of a passthrough hop is roughly the same as that of a direct connection. A NetDirect connection with one passthrough hop effectively doubles the CPU cycles and memory needed to move each byte of data from a source to a destination node. It also doubles the latency introduced by the network, resulting in lower throughput and higher response time.

Figure 4 shows a similar four-node network connected with Expand over TLAM. Each node has a single TLAM connection that enables communication with all other nodes attached to the LAN. There are four controllers, one LAN, and no passthrough hops.

The X25AM Option

X25AM, like Expand over TLAM, can be a network service provider. It is an intermediate process operating between an Expand process and an X.25 network. Like Expand over TLAM, X25AM connections are established by the NetNAM module of Expand, and in this case allow an Expand process to exchange data with an X.25 network. X25AM provides several of the same advantages as Expand over TLAM. For example, all the protocol work required to pass data through the X.25 network is performed by X25AM, instead of requiring the work to occur within the Expand process.

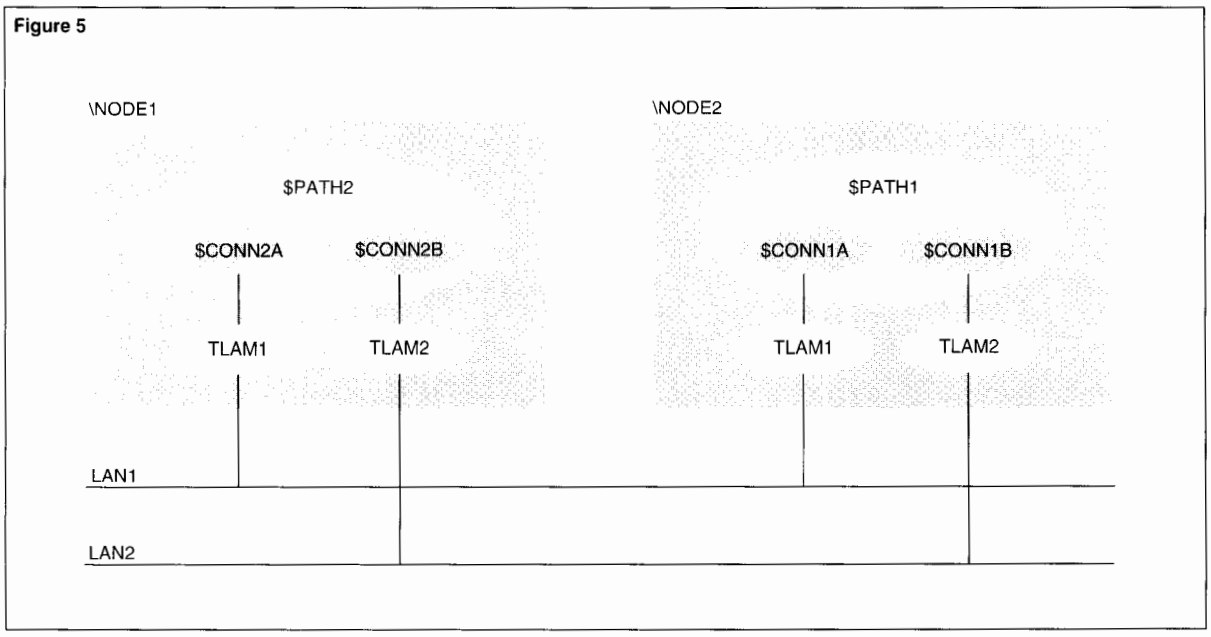


Figure 5.
These two Guardian 90 systems use multiple Expand over TLAM connections and two parallel LANs to increase throughput and fault tolerance.

One key consideration must be the efficiency of the target network. X.25 networks, for example, can perform very poorly; there may be numerous routing hops to be traversed inside the X.25 network. However, an Expand over X25AM connection is ideal for low-volume systems requiring high connectivity and reliability. An X.25 network offers a large number of addresses (as many as a sender is authorized to use) and charges for each packet sent. Because an X.25 subscriber does not need to pay to wire dedicated lines between numerous addresses, and if the amount of data being sent is modest, Expand over X25AM can be a very effective choice for connecting certain Expand networks.

Expand Enhancements Extend TLAM Potential

A recent enhancement to Expand called the multiline path feature[4] extends the value of Expand NetNAM. This enhancement provides improved fault tolerance and higher performance for Expand connections, such as Expand over TLAM and X25AM, that are based on Expand NetNAM.

Fault Tolerance and Higher Performance

The multiline path feature for Expand allows each Expand process to manage up to eight Expand NetNAM connections as one logical path. Fault tolerance improves and throughput increases when data destined for a peer node is allowed to pass through multiple Expand connections in parallel. Multiple Expand over TLAM connections can exist between two nodes, resulting in linear increases in throughput.

The multiline path feature has another important aspect. Each of the eight possible Expand connections can now travel a different physical route. For example, if so configured, the same Expand node can send data by both LAN and wide area network (WAN) communication lines. If one connection fails, Expand simply redistributes the load across the other connections in the path. Fault recovery is confined to the Expand path, requiring no intervention by the application.

Figure 5 presents a more fault-tolerant configuration with more throughput. It shows a path with two connections running over separate TLAM subsystems and LANs. If one connection fails, the Expand process continues to use the second connection for that path.

[4]The T9057AAZ IPM for Expand introduced the multiline path feature enhancement in November 1990.

Figure 6

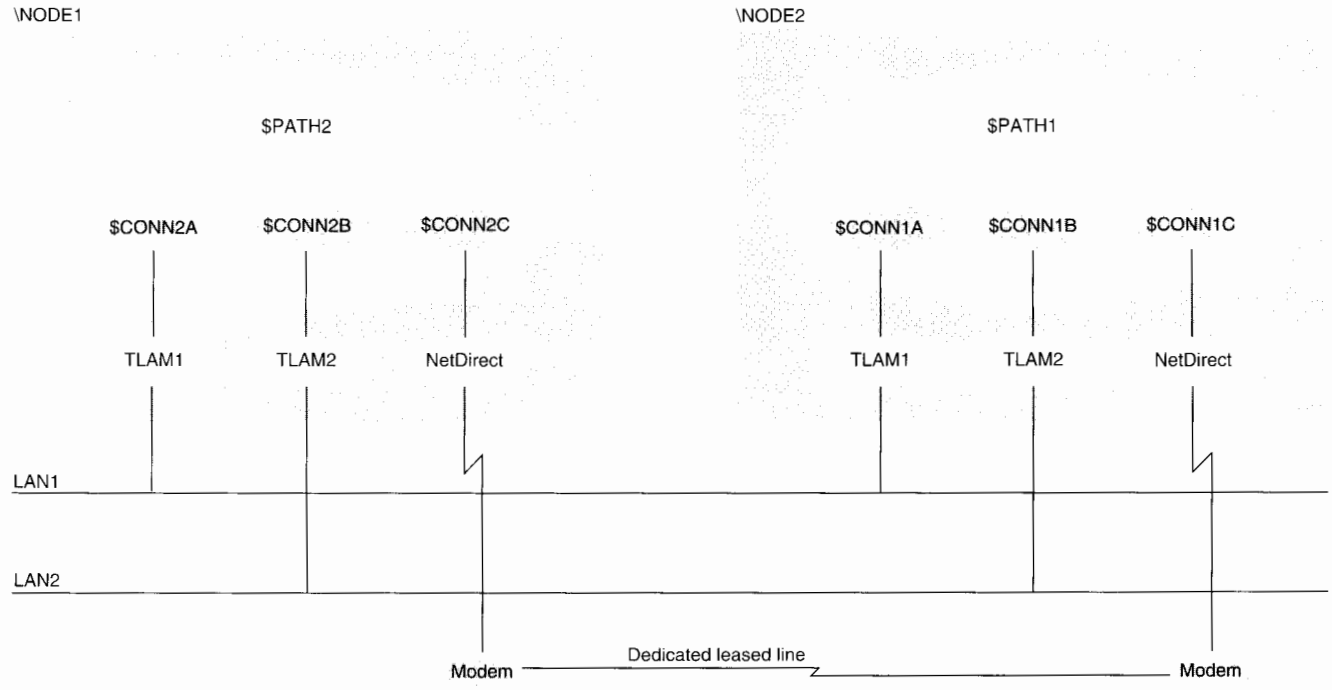


Figure 6.
Extra fault tolerance is built into this configuration with the additional Expand NetDirect connection. The two Guardian 90 nodes can use NetDirect if the Expand over TLAM connections fail.

As Figure 5 depicts, a path can be configured to achieve higher throughput by using multiple LANs and multiple TLAM subsystems to process data. Twice the throughput is available when using two LAN connections rather than one. Connecting to more than one LAN or configuring more than one TLAM subsystem results in better response time and helps to resolve network management problems such as queueing delays and collisions on heavily used LANs.

The multiline path feature allows one to choose the level of redundancy to be designed into a network. If multiple LANs are unavailable, a different configuration can still provide TLAM fault tolerance. Guardian systems can communicate over a single LAN through multiple Expand over TLAM connections. Like Figure 5, each system can have two Expand connections associated with two separate TLAM subsystems, but they would be sharing a single LAN. This modified configuration allows recovery if TLAM fails, but not if the LAN fails.

Figure 6 provides an example of mixing Expand connectivity options for additional robustness. In Figure 6, a connection using NetDirect is added to a path between two Expand nodes that are also linked by two Expand over TLAM connections. If the LANs fail, the NetDirect connection may continue to be operational. Even though the quality of the service would be degraded, communication is still possible until the LAN is again functional.

Extending LAN-Based Networks

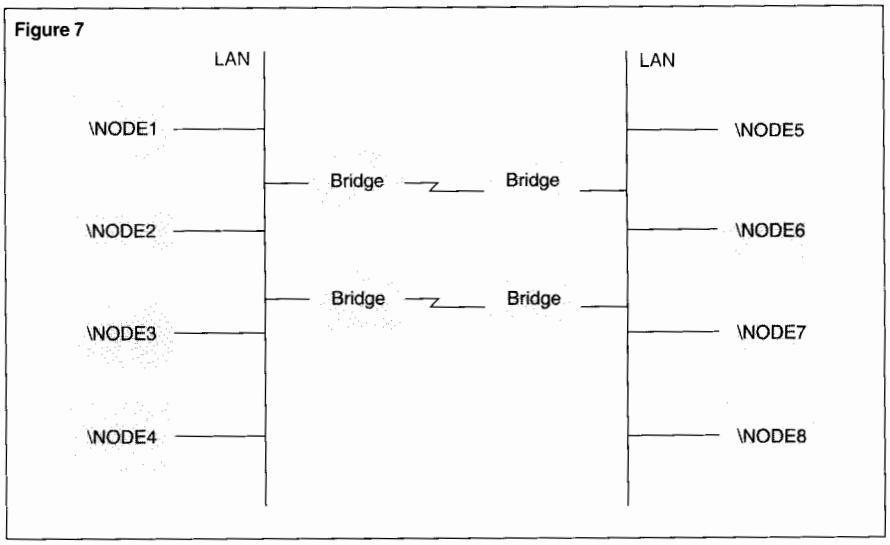
Bridges can link together separate LAN-based Expand networks into a distributed WAN. Media access control (MAC) bridge products make it possible to view geographically separate LANs as a single physical medium. Bridges can be fast; T1 (1.544 megabits per second) is a common speed for data transmission. Bridges are transparent to the senders and recipients on the linked LANs. Also, bridges used for Expand traffic can be shared by other network products: for example, networking clients such as Transmission Control Protocol/Internet Protocol (TCP/IP) and Open Systems Interconnection (OSI) or Multilan™ applications using NETBIOS.[5]

In addition to linking LANs, bridges can filter frames passed between the LANs they connect. The bridge can be configured to accept frames destined for particular addresses on the LAN. Only those frames are passed through the link. Multiple bridges may be used in parallel to increase bandwidth between the LANs. By allowing only a subset of data to pass through each parallel bridge, traffic can be balanced across the bridges.

Two or more geographically separate clusters of nodes that need to be connected into a single Expand network can make the most practical use of bridge technology. Each cluster can be LAN-based, with bridges transparently linking the LANs. Figure 7 shows two Expand sites connected by MAC bridges.

The capability of bridging separate LANs can be used to improve on some existing Expand network designs. Suppose a network utilizing a dual-site configuration (two FOX rings of four systems each) must be linked together. Without Expand over TLAM, the two sites would be linked with one or more NetDirect lines, possibly funneled through a T1 multiplexer. This hardware is costly, offers limited bandwidth, and relies on many passthrough hops to route data.

Figure 7



Switching to Expand over TLAM connections and bridge technology improves this configuration. A fully interconnected topology is achieved by installing a LAN at each site and connecting the LANs with a high-speed bridge. Expand processes can use the LAN to communicate with local and remote nodes with no passthrough hops. Improved fault tolerance and enhanced throughput can be achieved by adding more bridges to the configuration.

Figure 7.
Expand over TLAM, coupled with one or more bridges, allows two geographically separate sites to operate as one fully interconnected network.

[5]NETBIOS (Network Basic Input Output System) is a peer-to-peer standard application programming interface.

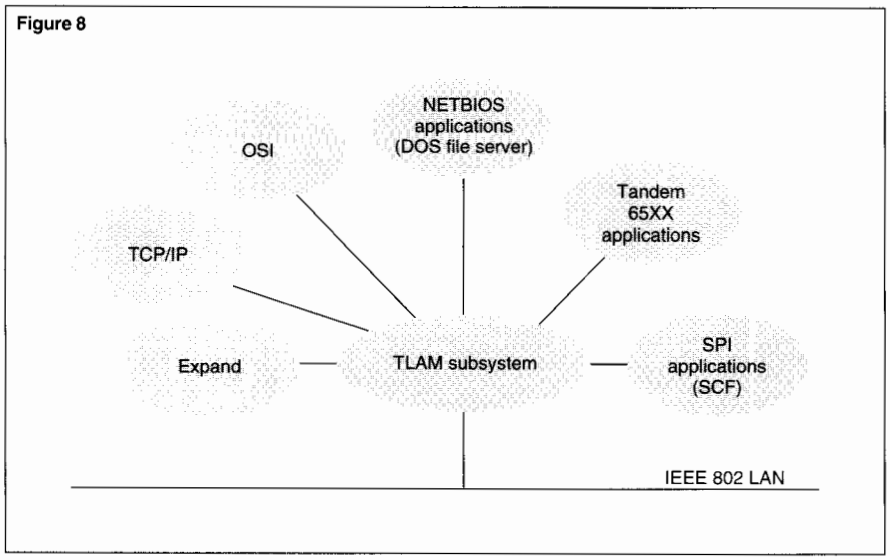


Figure 8.
One or more of these networking products can simultaneously share a TLAM subsystem.

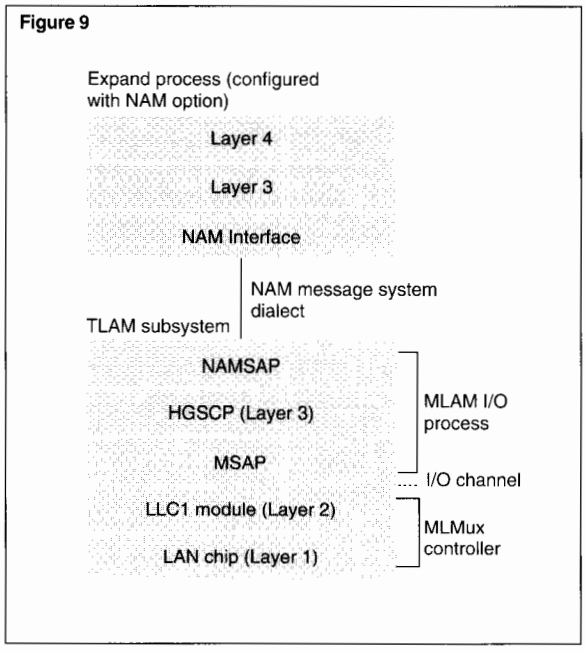


Figure 9.
This block diagram illustrates the software components used in an Expand over TLAM connection.

New Features of the TLAM Subsystem

The TLAM subsystem has been enhanced to act as a network service provider.[6] Figure 8 illustrates the many products, including Expand, that TLAM currently supports.

Network service requires the exchange of Network Access Method (NAM) message system data units between an Expand process and a network service provider. To enable TLAM to accomplish this work, three software modules were built. These modules comprise a new protocol stack in the TLAM I/O process (the MLAM) and allow the MLAM to process NAM system messages. Figure 9 presents a block diagram of the enhanced TLAM subsystem. The three new software modules are:

- The NAM service access point (NAMSAP).
- The home grown service control point (HGSCP).
- The message service access point (MSAP).

The high-level module, the NAMSAP, exchanges messages with the Expand process. The middle module, the HGSCP, performs Network Layer protocol services. The NAMSAP and HGSCP work as a pair, manipulating data in sequence as they service an Expand connection. The third, lower-level module, the MSAP, exchanges data with the TLAM controller. In addition, two Distributed System Management (DSM) objects, named CONN and MSAP, were added to TLAM DSM to give system administrators control over subsystem configuration.

[6]Expand over TLAM connectivity requires a C10 or later release of Guardian 90.

System Design

When configured with the NetDirect option, an Expand process performs, roughly, the work of Layers 1 through 4 of the OSI reference model.[7] In contrast, when an Expand process is configured with the NetNAM option (to use TLAM or X25AM), Expand only needs to do the work of Layers 3 and 4. The network service provider, either TLAM or X25AM, simulates the service provided by Layers 1 and 2 of Expand. (Figure 9 depicts the flow of data.) The data units, passed between Expand and TLAM by way of the NAM message system dialect, approximate those passed from Layer 3 to Layer 2 of Expand. These data units are designed to fit inside an Expand Layer 2 frame.[8]

The NAMSAP. Once passed from Expand, NAM messages are delivered to the MLAM, where they are routed to a connection's NAMSAP module. The NAMSAP handles the service access point work associated with the NAM message system dialect. This work involves the queueing and activation of outbound messages, delivering inbound messages, and mapping of NAM interface semantics to HGSCP semantics.

The HGSCP. When a message is activated for transmission to the destination node, the NAMSAP passes the message to the connection's HGSCP module. The HGSCP provides, roughly, a Layer 3 connection-oriented service. Using the IEEE 802 Data Link Layer Control type 1 (LLC1) services of the TLAM controller (MLMux), the HGSCP delivers NAM data to the destination node. The Layer 3 service connects the local NAMSAP to the remote NAMSAP. The HGSCP also performs an address resolution protocol that frees analysts or operators from managing MAC network addresses.

The data units passed between the MLAM software modules are of two types. The logical data units passed between peer protocol entities are considered *protocol data units* (PDUs). The data units that travel between a service provider and a service consumer are called *service data units* (SDUs).

The HGSCP handles both types of data units. Local and remote HGSCPs pass PDUs, and NAMSAP and HGSCP modules pass SDUs. When SDUs are larger than the media frame size,[9] HGSCP segments them on the sending side and reassembles them on the receiving side.

A small amount of duplicate work is performed by the HGSCP and Expand's Layer 3. Expand requires that it does its own routing and segmentation. The HGSCP does no routing. It deals only with a fully interconnected topology. HGSCP does segmentation if the segments generated by Expand's Layer 3 are not small enough to fit into an IEEE 802 LLC1 frame. Duplication of segmentation can be minimized by configuring Expand's framesize small enough to fit into an LLC1 frame (eliminating HGSCP segmentation), or making the framesize large enough to allow HGSCP to do most of the segmentation. The latter is more desirable for maximizing Expand over TLAM performance.

[7] OSI Layers 1 through 4 are the Physical, Data Link, Network, and Transport Layers, respectively.

[8] An Expand Layer 2 data frame has a default size of 132 words.

[9] The media framesize for Ethernet LANs is 1518 bytes.

The MSAP. This module provides an environment in which the HGSCP and NAMSAP modules can work without knowledge of the lower levels of the TLAM subsystem or LLC1 ports. The MSAP acts primarily as a multiplexer-demultiplexer. It also contains the code that supports the new DSM objects.

The MSAP performs a number of tasks required to link the environment offered by the base MLAM modules with the environment required by NAMSAP and HGSCP. It routes NAM system messages to the appropriate instance of NAMSAP as well as manages a pair of TLAM ports. The MSAP accumulates outbound SDUs from all connections into a single outbound LLC1 aggregate SDU. The MSAP also breaks inbound aggregate SDUs into individual LLC1 SDUs that are delivered to the appropriate instance of HGSCP.

Data Delivery. The MSAP interfaces with the existing lower levels of the TLAM subsystem to move data to or from the LAN media. The LLC1 module in the MLMux accepts outbound (from the MSAP) aggregate SDUs, breaks them into individual LLC1 PDUs, and issues them to the LAN hardware. Inbound (from the LLC1 module) LLC1 PDUs from the LAN hardware are bundled into aggregate SDUs and forwarded across the I/O channel to the MSAP.

The subsystem becomes more efficient as the load increases.

Any of three data forwarding parameters triggers the delivery of either outbound or inbound aggregate SDUs: time, SDU count, or aggregate SDU size. Aggregation of SDUs allows the MLAM to use one channel transfer to move multiple media data units. This lowers the per-media-data-unit overhead, as queueing and transferring single data units is avoided. The result is a subsystem that becomes more efficient as the data load increases. However, under light loads, the inherent latency that results from using these data forwarding parameters causes slower response times.

DSM Objects

Distributed System Management (DSM) is the global system management environment for Tandem. It is used to configure systems and subsystems (like TLAM), as well as give system operators access to statistics and diagnostic information. Two DSM objects, CONN and MSAP, were added to TLAM DSM to manage NAMSAP, HGSCP, and MSAP, the new MLAM software modules.

The CONN (connection) object represents the bundled HGSCP and NAMSAP instance-pair that services an Expand connection. CONN objects are automatically configured. This means that one becomes defined when an Expand process requests a connection from the MSAP and becomes undefined when that connection is terminated. There can be up to 64 connections associated with a TLAM subsystem. Operators can obtain status and statistics information from any of those connections.

The MSAP object represents the MSAP module. It is permanently configured, which means that the operator cannot add or delete it. The operator can use commands to start, stop, abort, and alter the MSAP as well as to obtain information, status, and statistics. For more detailed information on these DSM objects, see the *SCF Reference Manual for TLAM*.

Performance Overview

Elements that influence network performance results include data length, physical media bandwidth, CPU speed, system latency, and, particularly for Expand networks, framesize.[10] Framesize significantly impacts system considerations because this parameter must be the same for all nodes on an Expand network. Nodes added later to the network must also use that predetermined value. This can impact performance as the network size increases or when different physical media are installed.

In general, the TLAM subsystem exhibits better performance when processing bulk data transfers rather than smaller message-based requests. Performance differences can be characterized by noting the response times for two common system commands used by operators. These commands, entered at a Tandem Application Command Language (TACL™) command interpreter, request file information and file transfer, respectively. They are:

- FILENAMES \NODE1.$SYSTEM.SYS01.*
- FUP DUP \NODE1.$SYSTEM.SYS01.*, *, SOURCEDATA, PURGE

The FILENAMES request often takes longer than expected for a system attached to a LAN. This is due to the message-intensive way that this command searches a remote directory. However, the impact of many operators from different systems doing the same request simultaneously creates little difference in response time. The FUP DUP command, in contrast, is a file utility command used to duplicate and transfer files. Response time for this command is generally very good, as this command involves Expand over TLAM in a bulk data transfer.

[10]Framesize is an alterable Expand attribute. It is measured in words rather than bytes.

Expand Over TLAM Configuration Considerations

In Expand over TLAM test configurations, CPU costs for a single application message varied substantially with framesize. This is true for other Expand connectivity options (except FOX). However, because the other options are more limited by physical media bandwidth than by processor speed, Expand over TLAM configurations benefitted most when the framesize value was increased.

Approximately 20 percent of the cost of running Expand over TLAM can be eliminated by configuring the Expand and TLAM components in the same processor. However, this is not necessarily recommended because it could interfere with load balancing.

Expand Over TLAM Performance Results

Performance tests were conducted in a laboratory environment using best-case configurations. With Cyclone CPUs, maximum throughput for a single Expand over TLAM connection was measured at 1.3 megabits per second.

Tests were also run to analyze Expand over TLAM performance using more common configurations. Under these conditions, the performance of one Expand over TLAM connection was compared to that of a four-connection NetDirect path. Results indicated that in some situations, the throughput of the Expand over TLAM connection tripled that of the four-connection NetDirect path.

Expand framesize figured prominently in these performance results. The default framesize of 132 words (256 user data bytes) increases CPU overhead, as many interprocess messages must be passed between TLAM and Expand for each application request. On both the CLX 700 and the VLX processors, throughput and message costs generally improved when the framesize was increased to 516 words.

On the CLX 700, with a framesize configuration of 132 words, Expand over TLAM provided equivalent throughput. It consumed about twice the CPU cycles as the four-connection NetDirect path. When framesize was increased to 516 words, throughput was about double that of the NetDirect path. Even so, the CPU cost per message on the CLX 700 was still as much as 60 percent more than that of a NetDirect connection.

The framesize parameter figured prominently in performance results.

A VLX processor with a framesize of 132 words running Expand over TLAM showed competitive throughput and a similar trend in message costs. Throughput on the VLX was equivalent to a NetDirect configuration having four lines running at 56 kilobits per second. Message costs, about 60 percent more than a NetDirect connection, required up to 15 percent of a VLX processor. Increasing the framesize parameter to 516 words on the VLX improved the performance and reduced the message costs. Throughput was two to three times higher than a NetDirect configuration with four 56-kilobit lines. Message costs equalled those of the NetDirect connections.

Performance testing explored the effect of adding one or more Expand connections within an Expand over TLAM configuration. Performance improvements depended on whether the additional connections were configured to share the same TLAM subsystem or use a separate one.

A second or third Expand connection configured through the same TLAM subsystem resulted in a slight increase in throughput. The capacity of a single TLAM connection is shared among all connections that use it. However, configuration of a second Expand connection through a different TLAM subsystem and LAN offered twice the throughput of a single TLAM connection. Configuration of a third Expand connection over a third TLAM subsystem and LAN offered another 100 percent additional throughput. This linear increase is achieved only with the first three connections.

Tests with CLX 800 CPUs determined that an Expand framesize of 2047 words and an application message size of 16 kilobytes resulted in a single Expand over TLAM connection throughput of 1.352 megabits per second. Expand over TLAM is designed to handle framesizes up to 32,000 words. It is also designed to be more efficient as framesize increases over 750 words (1500 bytes) because HGSCP fragments these messages into segments that fit into IEEE 802 LLC1 frames.

Early research indicates that more than 20 Cyclone systems could communicate through one TLAM subsystem. The throughput of each Expand connection could equal a 56-kilobit NetDirect connection.

Conclusion

Now able to support Expand services, TLAM provides an additional connectivity option for Expand networks. Enhancements to the TLAM subsystem enable it to function as an Expand network service provider, passing data between the Expand and IEEE 802 LAN networks. Additionally, the Expand multiline path enhancement allows parallelism to be incorporated into LAN-based networks, which improves both fault tolerance and performance.

Expand over TLAM is ideally suited for connecting new Expand networks where clusters of Expand nodes are to be installed and where IEEE 802 LANs already exist or are required for other applications. For situations suited for Expand over TLAM, this connectivity option offers a cost-effective, simple, relatively high-speed, and flexible method of connecting Expand nodes over LANs. With LAN bridges, Expand over TLAM can easily connect geographically distributed systems into a single Expand network.

References

Dynamic System Configuration (DSC) Manual. 1990. Tandem Computers Incorporated. Part no. 31219.

Expand over TLAM Install Guide. 1991. Tandem Computers Incorporated. Support Note S90013A. Available from Tandem representatives upon request.

Expand Reference Manual. 1990. Tandem Computers Incorporated. Part no. 22242.

SCF Reference Manual for Expand. 1990. Tandem Computers Incorporated. Part no. 18100.

SCF Reference Manual for TLAM. 1990. Tandem Computers Incorporated. Part no. 31419.

SCF Reference Manual for TLAM, Update 1. 1990. Tandem Computers Incorporated. Part no. 46512.

System Generation Manual for Expand. 1990. Tandem Computers Incorporated. Part no. 32496.

Acknowledgments

I would like to thank the reviewers of this article for providing their technical expertise. Special thanks go to John Friedenbach, John Marsh, and Mike Noonan.

Kirk MacKenzie joined Tandem in 1983 as a networking software engineer. He graduated in 1983 with a master's degree in computer science from California State University at Chico. Kirk has worked eight years in networking product development.

Pathway TCP Enhancements for Application Run-Time Support

The chief component of the Tandem™ Pathway transaction processing system is the terminal control process (TCP). The TCP provides the run-time environment for user-written SCREEN COBOL (SCOBOL) application requesters. Tandem has enhanced the TCP in order to support application requesters that require large data address spaces. These enhancements give application developers greater flexibility in the design of Pathway applications.

The TCP enhancements were developed to further support the emerging client-server model between terminals or workstations and Tandem's Guardian™ 90 operating system. Also, they allow Pathway applications to better exploit Tandem hardware products, such as the NonStop™ Cyclone™ computer system together with the larger I/O configurations it supports.

This article describes how the current TCP has evolved from earlier versions of the TCP. It discusses the TCP data address space limitations in the pre-C10 releases of Pathway and explains the impact of those limitations on Pathway application design. Next, it describes the enhancements introduced in the C10 and C11 releases of Pathway. Finally, the article discusses the advantages of using extended memory I/O operations, which improve the performance and integrity of TCP checkpointing and other operations.

The article assumes that readers are familiar with Pathway configuration management and the requester-server architecture of Pathway applications. To be concise, the article omits certain details about the implementation of the TCP enhancements. This article uses the terms *terminal task, user task, application task*, and *application thread* synonymously.

Removing TCP Data Address Space Limitations

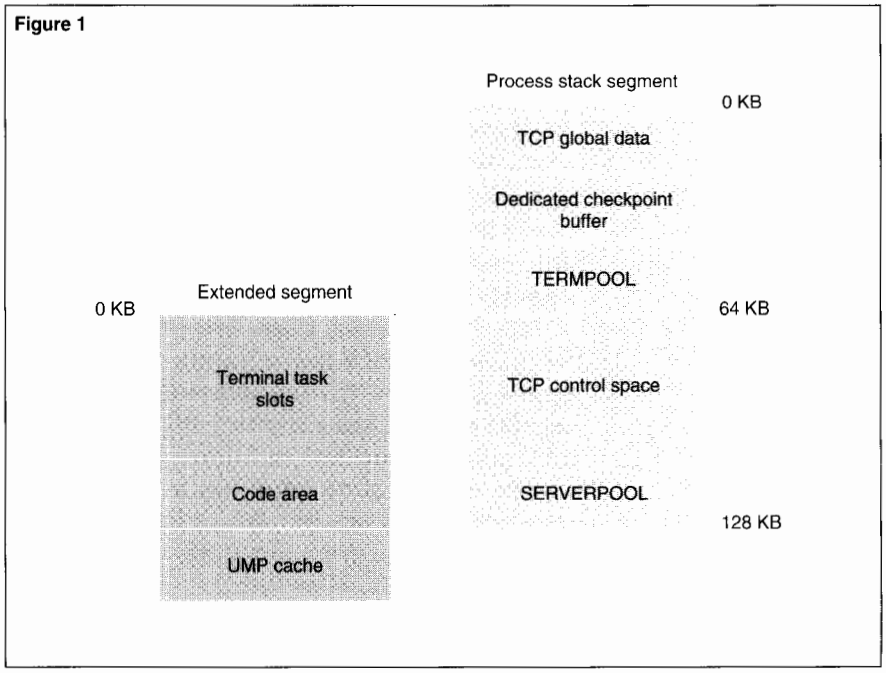
In the C10 and C11 releases of Pathway, two limitations on TCP data address space were removed, allowing the TCP to take full advantage of its extended memory. First, the TCP's I/O buffer pool space (TERMPOOL and SERVERPOOL) was moved from the TCP process stack segment[1] to its extended memory segment. The size limit of the process stack segment (128 kilobytes) limited the size of the I/O buffers, which restricted I/O operations to and from the TCP. Now I/O operations can be as large as 32,000 bytes, the interprocess message limit of the Tandem Guardian 90 operating system.[2] Figures 1 and 2 show the changes in memory organization in the TCP.

Second, the data address space assigned to an individual task in the TCP, formerly limited to 32 kilobytes, can now use as much of the TCP's extended memory as the task requires. The practical size of a task's data address space is limited only by the available Guardian 90 disk swap space, which supports the TCP's extended memory.

These enhancements allow a single TCP to manage a greater number of terminal tasks without sacrificing the response times of any individual task. The TCP can perform data transfers as large as 32,000 bytes to and from terminals or intelligent devices and server processes. The TCP can support application tasks that require large and deep call history stacks. Finally, each task can include individual SCOBOL requesters that require large Working Storage data items or large aggregates of Working Storage space.

[1]A process stack segment is the area of main memory that holds the information a process needs to perform its work.

[2]The size of the Guardian 90 message limit, 32,000 bytes, is slightly smaller than the number commonly associated with the term *32 kilobytes* (32,768 bytes).



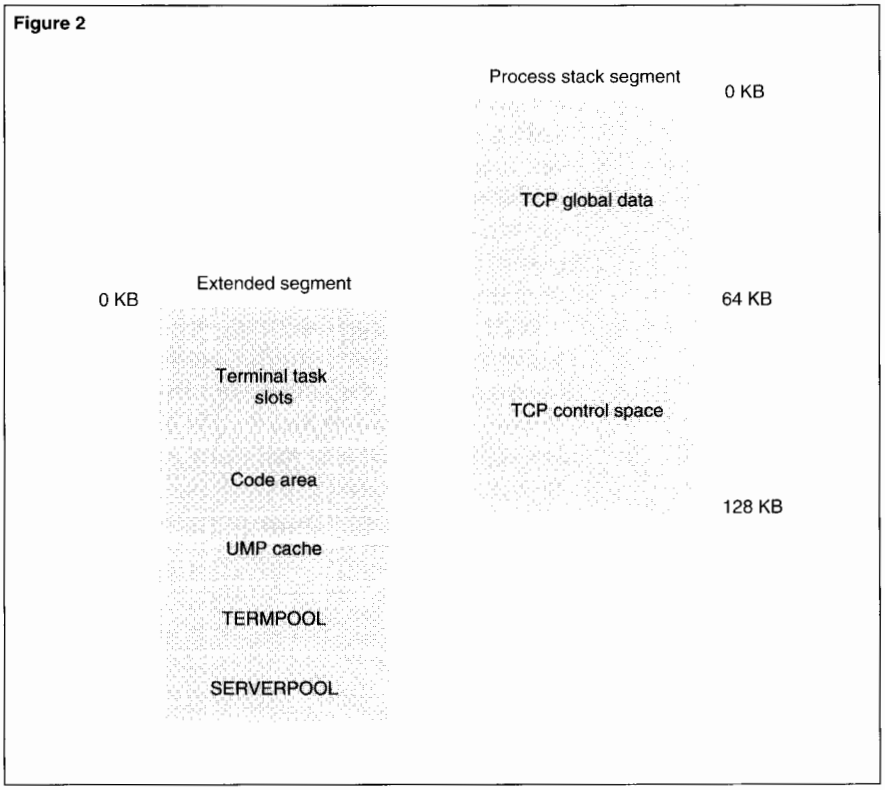


Figure 1. *Memory organization of the TCP in the pre-C10 releases of Pathway.*

Figure 2. *Memory organization of the TCP in the C10 release of Pathway.*

Evolution of the TCP Architecture

In a Pathway application, a requester is written in SCOBOL program code. The TCP interprets the SCOBOL-compiled program code, executing it as a terminal (user) task running in the TCP's process environment. The TCP provides services (such as checkpointing) for the terminal task, communicating with terminals and intelligent devices as well as server processes.

A TCP can provide a run-time environment for many terminal tasks (requesters) simultaneously; thus, the TCP is multithreaded. To execute multiple tasks concurrently, the TCP must not allow one task to block (impede) another task. To accomplish this, the TCP must perform all I/O operations nowaited.

Tandem's early processors did not offer extended memory. Therefore, the original TCP design confined the data address space for the entire TCP process to the TCP process stack segment. The maximum size of a process stack segment is 64K words (128 kilobytes).

All tasks managed by the original TCP (TCP1) contended for a reserved area in the TCP process stack segment. When a task was made ready to execute, TCP1 checked the reserved area to see if the task context was present. If required, portions of other tasks were swapped out, and the current task context was swapped in from a disk swap file explicitly maintained for that task context.

TCP1 was restructured in the E07 release of Pathway (the B00 release of Guardian 90). Known as TCP2, the new design used extended memory. *Task context spaces*, the data address areas allocated for terminal tasks managed by the TCP, were moved from the TCP process stack segment to an extended segment.

By using the extended segment, TCP2 made all task context directly accessible as one linear space. It eliminated the need to swap portions of a task context between the process stack segment and an explicit disk swap file. (To manage CPU memory in general, the Guardian 90 memory manager would continue to swap data, when necessary, at the CPU level.) By reducing disk I/O operations, TCP2 simplified task management and improved TCP performance. The architectural enhancements of TCP2 are described in detail in Wong, 1984.

Checkpointing was also improved in TCP2. TCP1 performed checkpointing of a task context by writing explicitly to disk (by writing to the backup TCP's copy of the task context swap file). This method transferred the task context indirectly from the primary TCP to the backup TCP. When the task context was moved to the extended segment in TCP2, the checkpointing method also changed. TCP2 transferred the task context directly from the primary process to the backup process (Wong, 1984). In the C00 release of Pathway, support for TCP1 was discontinued, and TCP2 was called *the TCP.*

TCP Limitations in Pre-C10 Releases

Even after the task context spaces were moved into extended memory, the TCP continued to have certain limitations in the pre-C10 releases of Pathway. Two architectural areas of the TCP restricted the TCP run-time environment: the size of buffer pool space and task context space.

The TCP maintains two buffer pools, one for I/O operations to terminals (TERMPOOL) and one for I/O operations to server processes (SERVERPOOL). A limited buffer pool space restricts both the number of terminal tasks the TCP can support and the amount of data it can send to a single terminal or server process.

The task context space determines the amount of data an application task can access at any given moment during its execution. A limited task context space restricts the size of Working Storage and can influence the complexity of the application structure.

Buffer Pool Space

In the pre-C10 releases of Pathway, TERMPOOL and SERVERPOOL were left in the TCP process stack segment because, at the time, Guardian 90 could not perform I/O to and from extended memory. (Facility for performing extended I/O became available in the C00 release of Guardian 90, which introduced file system extended I/O routines such as READX, WRITEX, and WRITEREADX.) Because of other demands on the process stack segment, the combined size of TERMPOOL and SERVERPOOL had to be much smaller than the process stack maximum of 64K words (128 kilobytes).

Ideally, users should configure a TCP to manage as many tasks as possible while ensuring that it is not queueing internally for buffers in TERMPOOL and SERVERPOOL. That is, at least one task should always be ready to execute.

Because the TCP is multithreaded and can support task execution concurrency, it is possible, and desirable, for the TCP to have multiple I/O operations outstanding. Therefore, buffer pool space must be available so that the TCP can support I/O operations for every task in the TCP.

In the pre-C10 releases of Pathway, users had to restrict demands on the TCP because of its limited buffer pool space. Users could control demands on the buffer pools by reducing the number of tasks supported by a TCP. This necessitated adding TCPs to their Pathway environment to support the required number of tasks.

The buffer pool space limitation also affected the design of SCOBOL requesters. Application developers had to work around the I/O limits on their SCOBOL requesters to ensure that they did not burden the buffer pool space. Otherwise, tasks might have to queue for buffer pool space, which would degrade task concurrency.

Task Context Space

In the original TCP design, the task context space was limited to 32 kilobytes (Wong, 1984). When the task context space was moved into the extended segment, its size remained at 32 kilobytes.[3]

A portion of the task's slot serves as a *pseudo stack*. The pseudo stack is to a task what the process stack segment is to a Guardian 90 process. By definition, an active task must have at least one SCOBOL *program unit* (PU), the executing version of a SCOBOL requester. The TCP maintains the local context of each PU in the task's pseudo stack. Each PU's Working Storage data space resides in the local context area of the PU.

When the task requires nested PU calls (successive calls from one PU to another), the pseudo stack grows. The amount of PU call history that a stack can accommodate depends on the stack's size, the maximum being MAXTERMDATA. Exceeding MAXTERMDATA during a sequence of PU calls can cause the pseudo stack to overflow, which can cause the task to abort.

[3]Users can define the size of the task context space by using MAXTERMDATA, a parameter of the Pathcom SET TCP command.

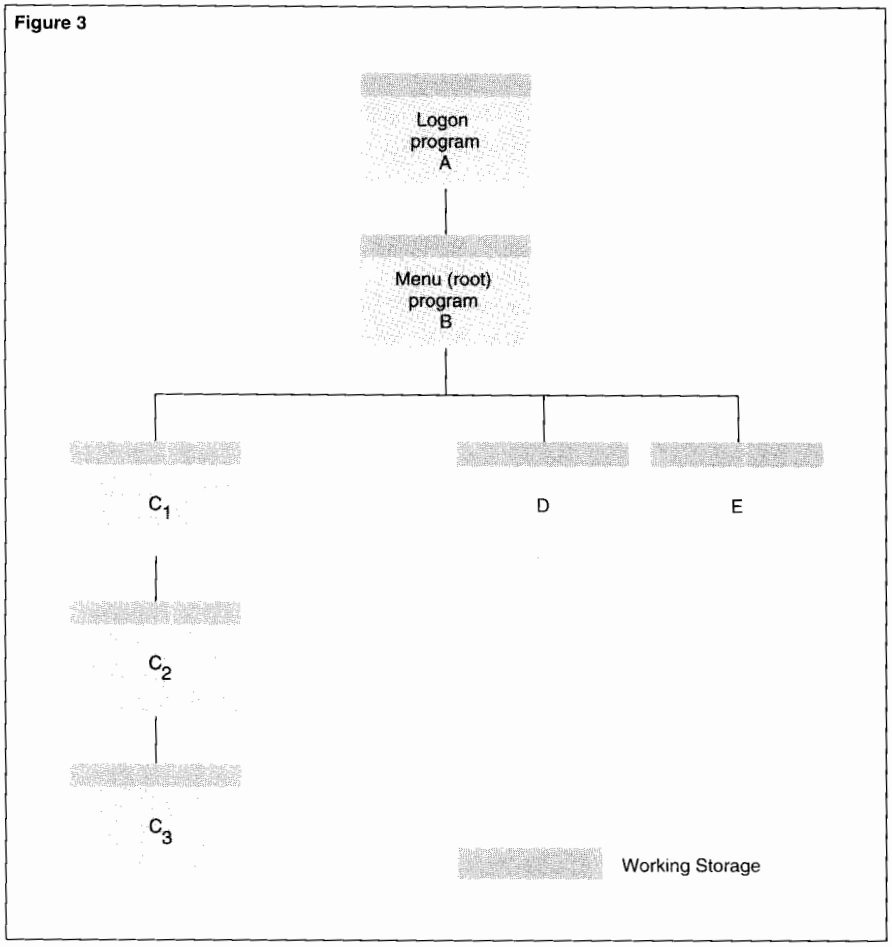


Figure 3.
A structured application call hierarchy.

Thus, the size of the task context space can determine whether or not the task will execute successfully. The task may abort if the PU call history sequence is too deep or if the Working Storage in one or more of the PUs involved in the call sequence is too large. Consequently, an application's design is constrained by the limits of the task context space in which it must operate.

Also, during certain SCOBOL operations such as sending data to servers, the TCP performs internal operations that add to the use of the task's pseudo stack. If the pseudo stack is nearly at MAXTERMDATA when the SCOBOL operation begins, it can cause the pseudo stack to overflow and the task to abort. Therefore, the Working Storage space permitted in each PU had to be smaller than the 32-kilobyte limit. The limited Working Storage space could affect the capabilities of a SCOBOL requester.

Impact on Application Design

The 32-kilobyte limit on task context space could influence the design of Pathway applications by restricting the choices available to application developers. The limitation could make it more difficult to use a structured application call hierarchy, which follows the application logic naturally but requires a deep call history stack. Figure 3 shows a structured call history sequence in which the logon program calls the menu program, which calls C_1, which calls C_2, which calls C_3.

One way to cope with the limitation is to design the application to be hierarchically *flat.* In one kind of hierarchically flat design, the root program calls all other PUs directly, and control always returns to the root program before going on to another PU. Developers have devised various PU routing schemes to keep an application's hierarchy flat.

There are legitimate reasons to use a flat application call sequence. For example, it can provide random (dynamic) screen navigation. However, this method of limiting task context space can introduce unnecessary complexity into the application's design, implementation, and maintenance.

Figure 4

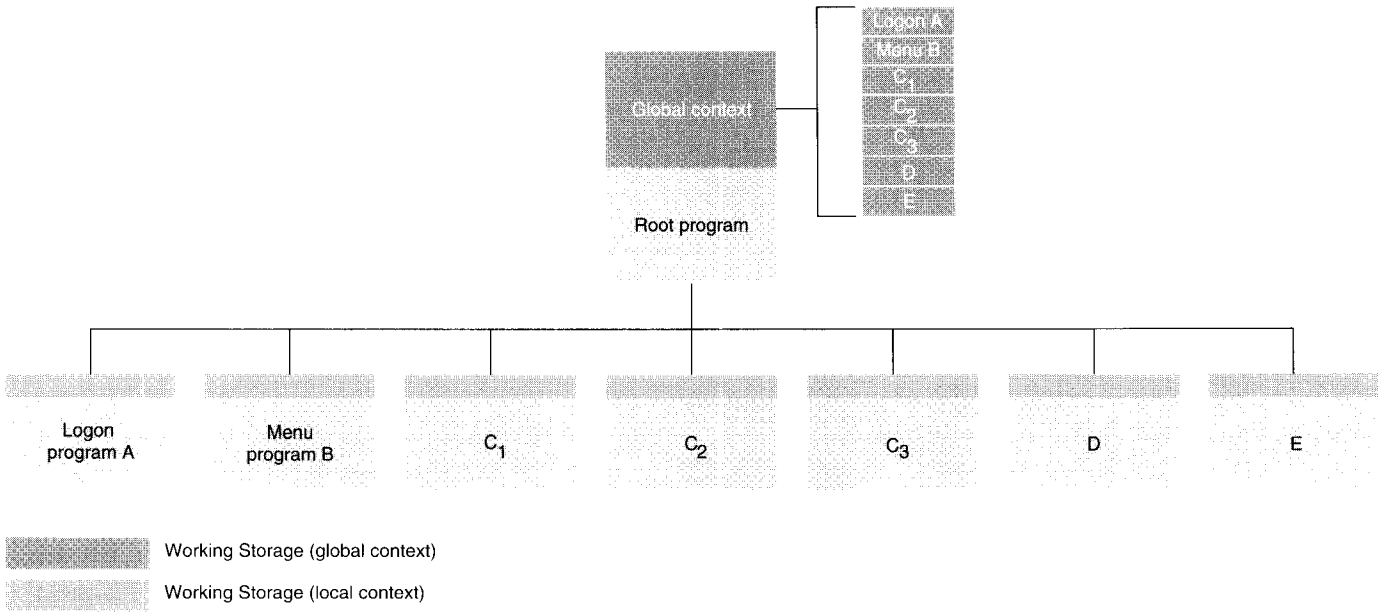


Figure 4 shows an example of a flat application call sequence. To complete its processing, program C_1 informs the root program (through its Linkage Section) that it should call program C_2 the next time it gets control. Program C_1 exits to the root program, which calls program C_2. The same call sequence occurs to execute program C_3.

The example in Figure 4 economizes on task context space by declaring a large Working Storage space in the initial (root) PU, which is shared by called PUs through references in their Linkage Sections. The Working Storage in a PU can be kept relatively small if global data structures are identified in the shared areas of the root PU. The PU's Working Storage can be limited to local, transitory data.

Figure 4.
A flat application call sequence. Programs such as C_1, C_2, and C_3 use the global context (shared data area) in the root program by references in their Linkage Sections.

Figure 5

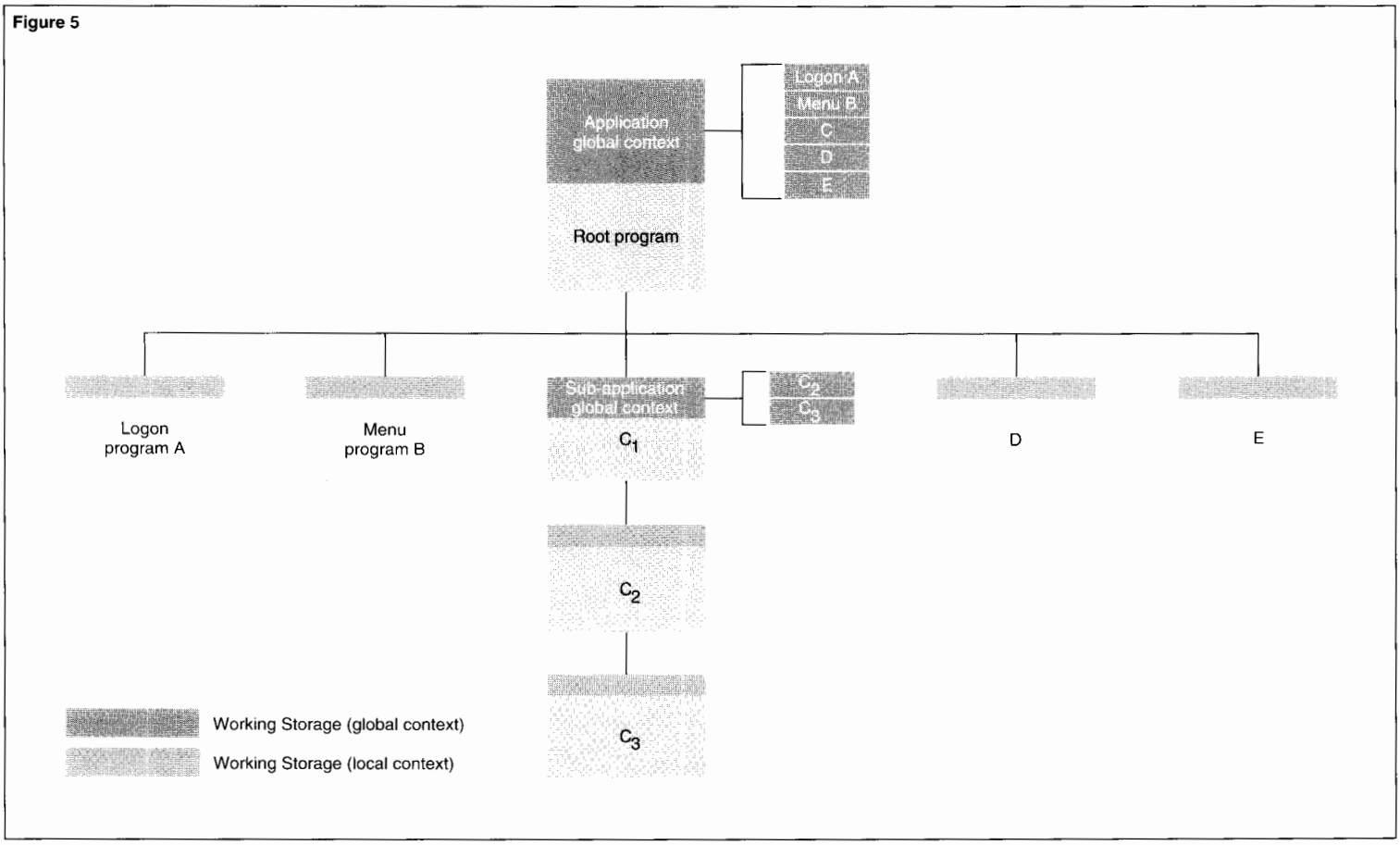


Figure 5.
A combination of structured and flat application call sequences. Called programs use the application global context (shared data area) in the root program. Programs C_2 and C_3 also use the sub-application global context in program C_1.

The application can avoid running out of task context space even when it needs to perform relatively deep call sequences from one SCOBOL PU to another. Figure 5 shows an example of this approach, which blends a flat call sequence with a hierarchical call sequence. The Working Storage in the root program contains data structures global to the entire application, and program C_1's Working Storage contains data structures global to the routines performed by programs C_1, C_2, and C_3. With this method, the application maintains a call history stack smaller than one required by a structured hierarchy and minimizes the overall size of the task context space.

Implementing the Enhancements

The size limits of the TCP buffer pool space and task context space were removed in the C10 release of Pathway. The size limit of the program context (Working Storage) for individual PUs was removed in the C11 release of Pathway.

Removing these limitations allows the TCP to support a larger number of tasks requiring large I/O data transfers. Individual SCOBOL requesters can use their Working Storage size limit potential and can now maintain deep call history stacks. Also, application thread design is no longer constrained by a limited data address space.

Task Context Space

In the C10 release of Pathway, the references to the task context space for an individual task in a TCP were increased from 16-bit to 32-bit addressing ranges. A 32-bit value can refer to a 2-gigabyte data address space (set by the MAXTERMDATA parameter). Thus, changing

the addressing ranges to 32 bits effectively removed the size limit of TCP task context space. Because a Guardian 90 disk swap file supports the TCP extended segment, the available disk space determines the practical size limit of the task context space.

Program Context (Working Storage)

In the pre-C10 and C10 releases of Pathway, SCOBOL programs had to operate within a smaller local address space than the 32-kilobyte maximum. The practical size limit was approximately 28 kilobytes. Also, to keep the aggregate size of Working Storage data items below the 32-kilobyte limit, individual Working Storage data items had an artificial limit of 12 kilobytes. Because Message Section field items map to Working Storage data items, the same size limits implicitly applied to them. All these limitations restricted the design of SCOBOL programs.

In the C11 release of Pathway, the SCOBOL compiler was changed so that SCOBOL programs could support 32-bit addressing ranges to PU context (Working Storage) space. This enhancement extends the size limit of an individual 01-level Working Storage data item from 12,288 bytes to 32,000 bytes (the maximum Guardian 90 interprocess message size). Also, Message Section field items have been extended so that they can map to the new size limits of Working Storage data items.

Table 1 shows the evolution of the size limits of PU context space and task context space in the pre-C10, C10, and C11 releases of Pathway. The numbers in Table 1 indicate the addressing ranges of the data address spaces.

Figure 6 shows the relationships between individual PU context spaces and the task context space. The figure shows three call history stacks for a task executing in the different Pathway releases. The task context comprises a sequence of PU calls. PU_1 calls PU_2, which calls PU_3; the call sequence continues until it reaches PU_n.

In the pre-C10 releases (column a), each individual PU context space had a limit of 32 kilobytes (16-bit addressing range), of which the PU could exploit approximately 28 kilobytes. However, all the PU context spaces had to fit into a task context space of at most 32 kilobytes.

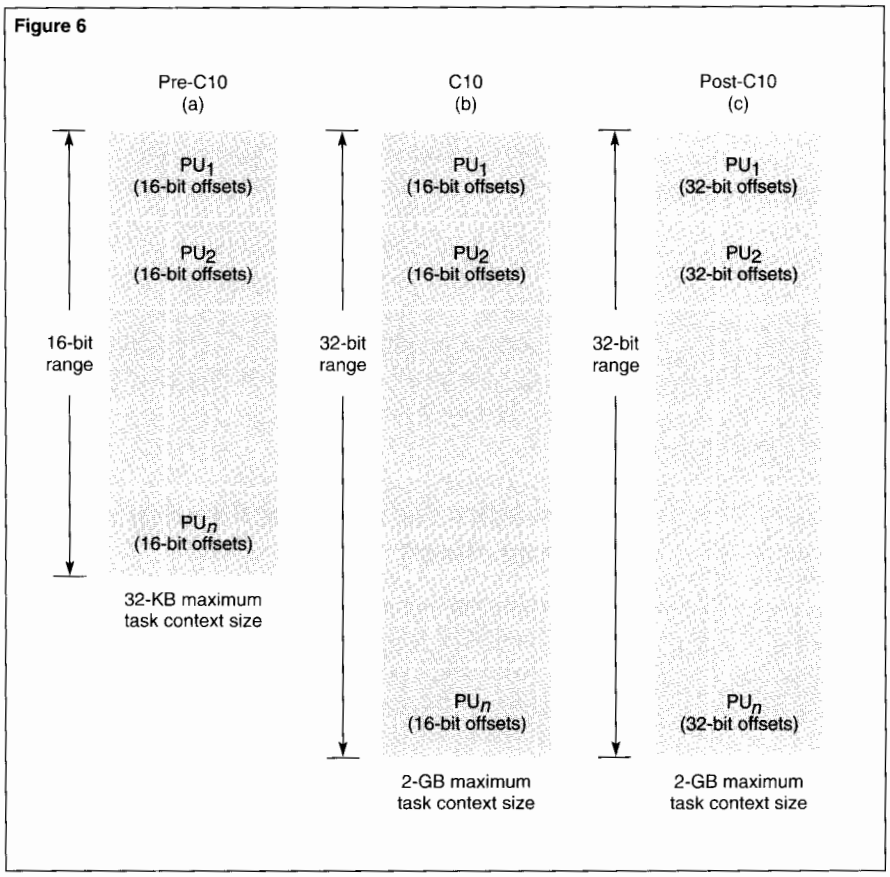


Figure 6.
Relationship of program unit (PU) size and task context size.

Table 1.
Addressing ranges for SCOBOL program units (PUs) and task context spaces in the pre-C10, C10, and post-C10 releases of Pathway.

	Addressing range values (in bits)		
	Pre-C10	**C10**	**Post-C10**
Program unit (SCOBOL compiler)	16	16	32
Task context space (TCP)	16	32	32

In the C10 release (column b), the individual PU context space still had a limit of 32 kilobytes, but the task context space could be as large as 2 gigabytes (32-bit addressing ranges).

In the post-C10 releases (column c), the PU context space can be as large as 2 gigabytes. The task context space also can be as large as 2 gigabytes.

In each new release of Pathway, the TCP continues to support SCOBOL pseudo code compiled in the previous releases. However, it is recommended that users running a post-C10 release of Pathway recompile SCOBOL requesters developed in the C10 and pre-C10 releases. Existing SCOBOL pseudo code executes more efficiently if it is recompiled because it can take advantage of the improvements in the post-C10 SCOBOL compiler and TCP. (The post-C10 SCOBOL compiler produces 32-bit addressing ranges for the TCP's direct access to the Working Storage in the PU).

Benefits for I/O and Checkpointing Operations

In addition to removing the size limits of TCP data address spaces, Tandem has improved the efficiency of checkpointing operations performed by the TCP. Because of the new Guardian 90 I/O capabilities, the TCP can perform checkpointing directly from the extended memory of the primary TCP to the extended memory of the backup TCP. The new checkpointing method enhances both performance and integrity of checkpointing.

Extended Memory I/O

The TCP now performs most I/O operations to and from extended memory. The TCP uses the extended options of the Guardian 90 file system routines (such as READX, WRITEX, WRITEREADX, and READUPDATEX). Specifically, the TCP performs the following I/O operations to and from extended memory: terminal I/O, server I/O, SCOBOL pseudo code caching, and task context checkpointing.

Improvements in Checkpointing

A NonStop TCP process performs checkpointing operations to maintain a current backup version of the TCP and its tasks. *Task context checkpointing* saves the data context of the currently executing task in the backup TCP process.

Both the primary and backup TCPs allocate two areas of task context space in extended memory: SLOT0 and SLOT1. In the primary TCP, SLOT0 maintains the current (active) context for the task. SLOT1 holds a backup copy of SLOT0. If necessary, the TCP uses SLOT1 to restore the current task context to a previous state (Wong, 1984).

The TCP performs its checkpointing operations nowaited. It cannot use Guardian 90 checkpointing routines such as CHECKPOINT and CHECKMONITOR because they are waited. Instead, the primary TCP uses the Guardian 90 WRITEREADX routine to exchange information with the backup TCP; the backup TCP uses the READUPDATEX and REPLY routines. In this type of checkpointing, in which the TCP performs its own I/O operation, the backup TCP is an *active* backup process.

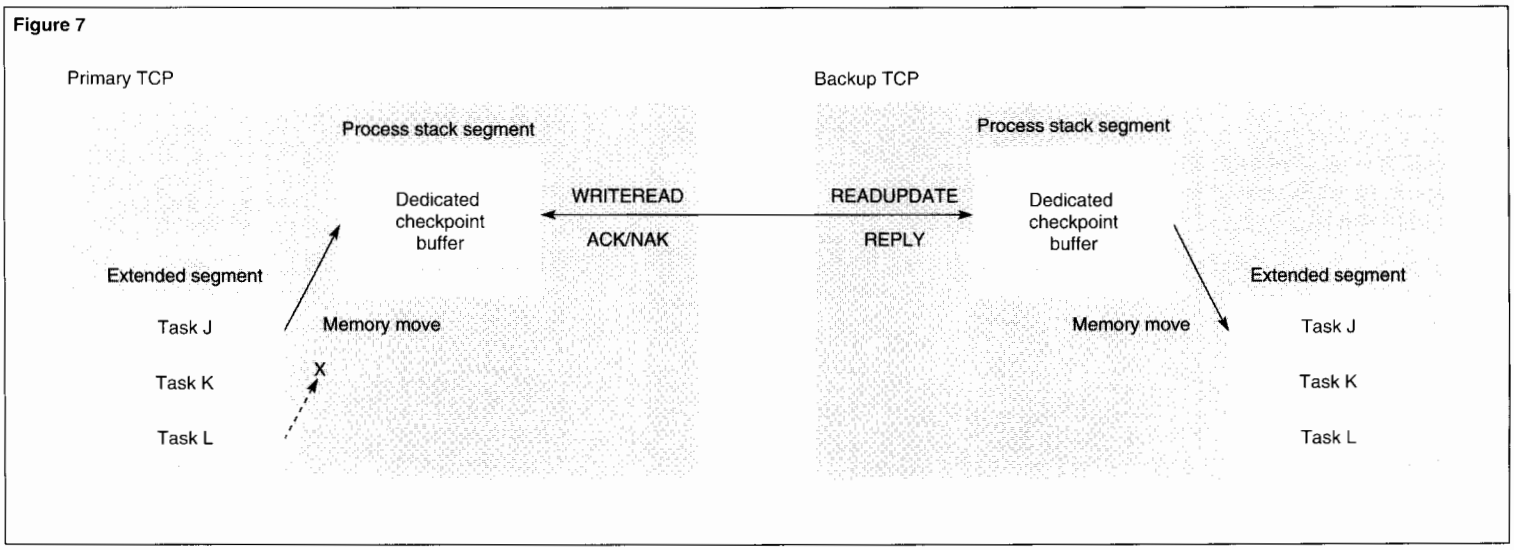


Figure 7.
TCP task context checkpointing in the pre-C10 releases of Pathway.

Figure 7 shows the task context checkpointing method for pre-C10 releases of Pathway. First, the primary TCP stages Task J's context data from its slot in extended memory to the checkpoint buffer in the process stack segment. Next, it performs the I/O operation, sending the task context to the backup TCP. The backup TCP receives the task context in the checkpoint receive buffer in its process stack segment. Finally, the backup TCP stages the task context from the process stack segment to Task J's slot in extended memory. (Only the portion of a slot that is currently in use is checkpointed, not the entire slot.)

Figure 7 also illustrates that the TCP's checkpointing facility is *single-threaded*. That is, an in-progress checkpoint must complete before another one can begin. In Figure 7, Task J's checkpoint must complete before Task L's checkpoint can begin.

As of the C10 release of Pathway, the primary TCP transfers task context data directly from its source in extended memory to the extended memory of the backup TCP. The TCP does not have to stage the task context to a buffer in its process stack segment before transferring it to the backup TCP. If the staging method were used, an I/O transfer size would be limited to the size of the buffer in the process stack segment, something less than 32,000 bytes. (The size of the staging buffer would depend on other storage demands made on the process stack segment.)

In contrast, the new method permits I/O transfers of maximum size and eliminates the memory move from the extended segment to the process stack segment. Similarly, the backup TCP receives all checkpoint data in an extended memory buffer. It does not have to stage the task context to a buffer in the process stack segment before moving it to its destination in the extended segment. This eliminates a second memory move (in the backup TCP). All slots are of equal size and are allocated and deallocated in their entirety.

Figure 8.
Step 1 of TCP task context checkpointing in the C10 and post-C10 releases of Pathway. The backup TCP allocates the extra slot as the checkpoint receive buffer.

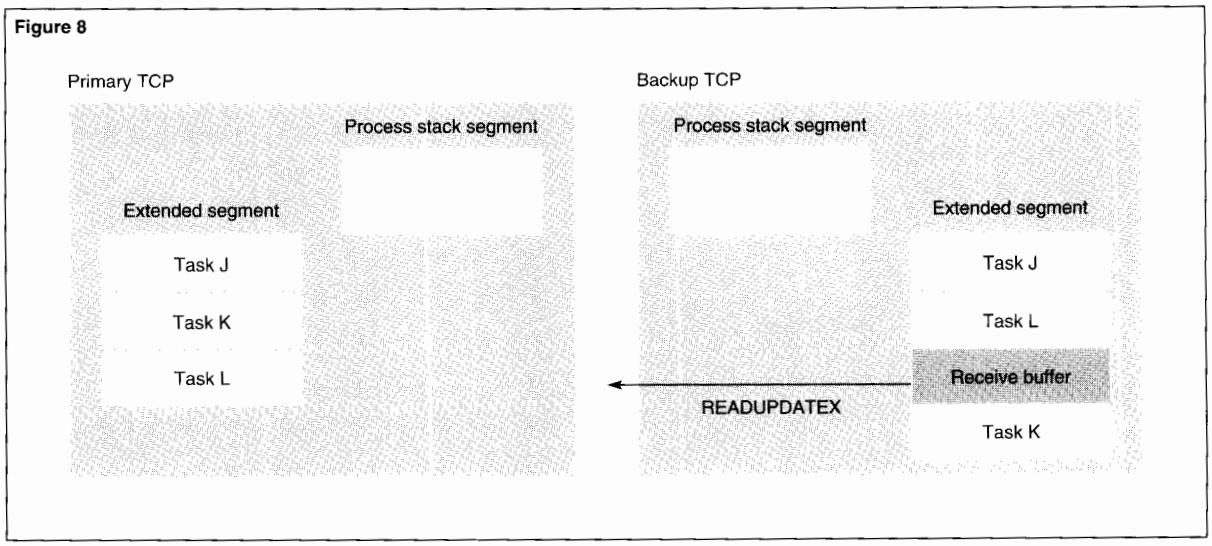


Figure 9.
Step 2. The primary TCP sends checkpoint context data for Task J to the checkpoint receive buffer.

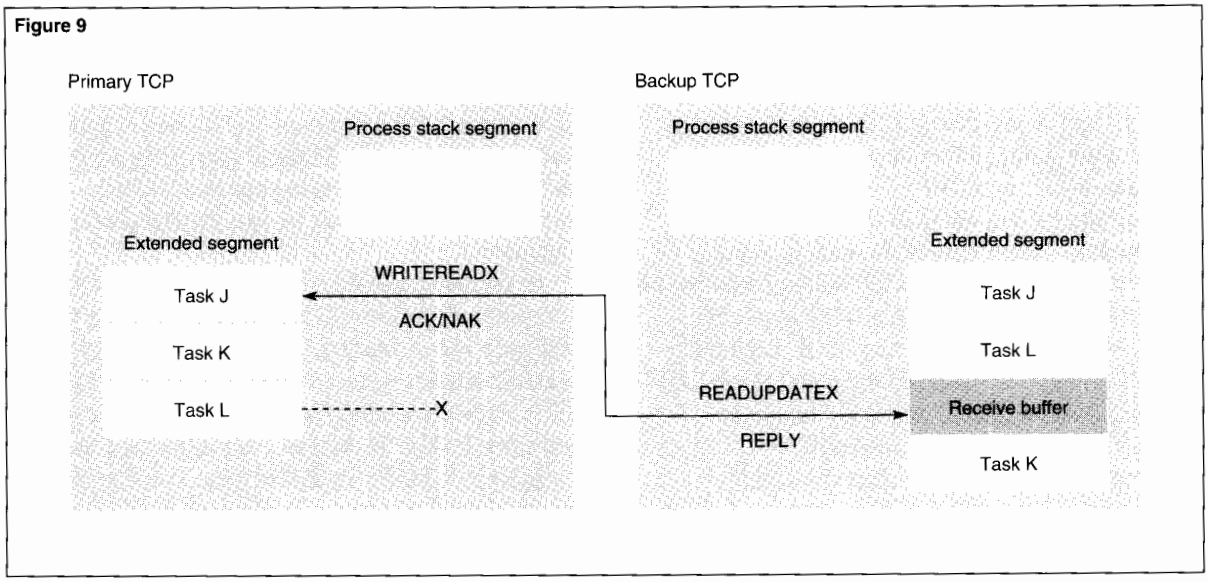


Figure 8 shows the first step in a task context checkpointing operation performed in a C10 (or later) release of Pathway. The TCP maintains a pool of slot areas. The backup TCP dynamically allocates one of the slots in this pool for its checkpoint receive buffer. (The TCP implicitly configures one extra slot so that a slot is always available for this purpose.) Next, it issues a READUPDATEX message and waits for the primary TCP to send checkpoint data.[4]

Figure 9 shows the second step in the task context checkpointing operation. The primary TCP sends task context data for Task J from its extended segment to the checkpoint receive buffer. The operation does not involve the process stack segment in either TCP. As in Figure 7, Task L is also ready to transfer its task context. It waits for Task J's checkpointing operation to complete before beginning its own checkpointing.

[4]If the received checkpoint involves a TCP control checkpoint (one involving the TCP task's own status or activity but not its data context), the checkpoint is processed and the checkpoint receive buffer is reused.

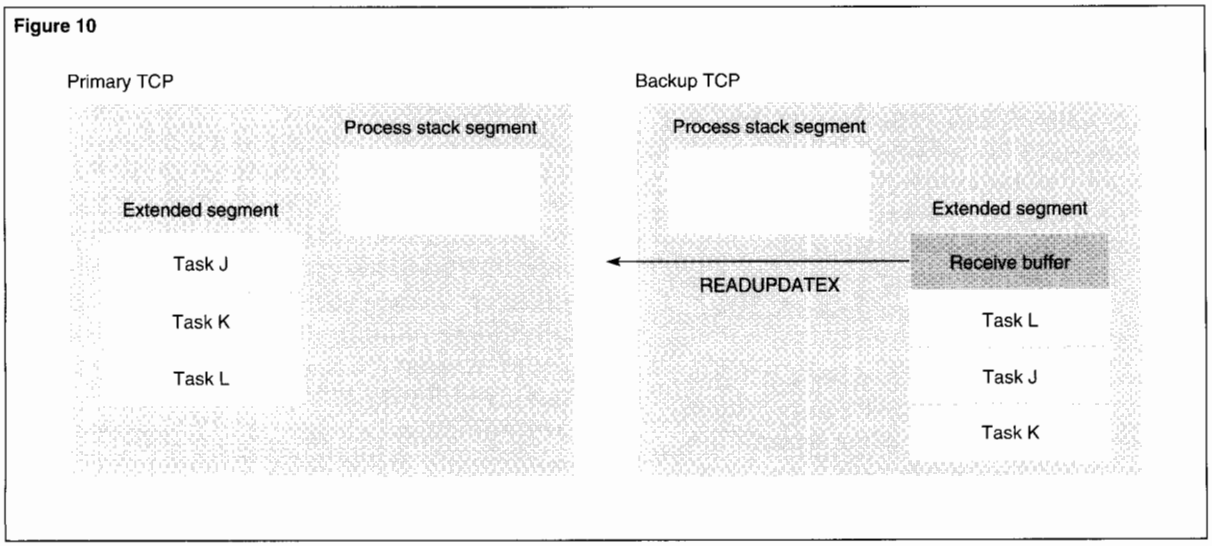


Figure 10.
Step 3. The backup TCP commits the completed checkpoint to Task J. The slot that held the old task context of Task J becomes the new checkpoint receive buffer. Task L, which was waiting, can now transfer its task context to the backup TCP.

Figure 10 shows the third step in the task context checkpointing operation. The backup TCP commits the completed task context checkpoint to Task J by assigning to it the location of the current checkpoint receive buffer. The backup TCP deallocates the slot that held the old task context for Task J and returns it to the slot area pool, making it available for later use. Now Task L can perform its task context checkpointing operation.

Because task contexts can now differ vastly in size, and because TCP checkpointing is single-threaded, it is possible that the task context checkpointing operations of one task could indirectly influence the uniformity of response times of its neighboring tasks in a TCP. When this issue is a possibility, users should consider assigning terminal tasks with similar task context sizes to the same TCP.

Checkpoint Integrity

The checkpointing method introduced in the C10 release of Pathway significantly improves the integrity of task context checkpointing. This is especially true when a checkpoint involves multiple I/O operations.

Checkpoint messages are prefixed with a descriptor containing control information. For a task context checkpoint message, the descriptor contains the length (in bytes) of the context being transferred. The backup TCP can determine from this length if it will require multiple I/O operations to transfer the task's context.

In the pre-C10 releases of Pathway, a backup TCP receiving a checkpoint that required multiple I/O operations was vulnerable to task context slot contamination. All checkpoints, including those that required multiple I/O operations, were staged from the checkpoint receive buffer in the process stack segment directly to the task's *live* task context space (slot area). If the primary TCP failed and a multiple I/O checkpointing operation was not completed, the affected backup task context, and therefore the task itself, became invalid.

In the C10 (and later) releases of Pathway, the checkpoint is staged (concatenated) directly into the current checkpoint receive buffer in extended memory. The backup TCP does not commit the new task context to the task until all the I/O operations for the checkpoint have completed successfully. The previous version of the task context, located in its current slot, remains valid until the current checkpoint is committed.

Backup TCP Is Not an Exact Mirror

The primary TCP pre-allocates slots consecutively from the slot area pool. The effect is that a true primary TCP (one that has never been a backup) always has adjacent slots in extended memory for a task's current context space (SLOT0) and backup context space (SLOT1). In contrast, the backup TCP dynamically assigns the slots in extended memory to various tasks as their context checkpoints arrive from the primary TCP.

Therefore, the primary and backup TCPs do not mirror one another with respect to the allocations of their task context spaces. This difference has no effect on the operation of the TCP. Moreover, this method of allocating slots dynamically is the basis of the improved integrity of task context checkpointing.

Conclusion

In the C10 and post-C10 releases of Pathway, the size limitations of the TCP buffer pool space, task context space, and individual SCOBOL program unit space have been removed. Removing these limitations allows the TCP to handle a greater number of terminal tasks that require large I/O operations while maintaining task execution concurrency. In addition, the TCP can support large Working Storage data contexts as well as applications with deep call history stacks. Finally, the TCP has enhanced the performance and integrity of its checkpointing operations by performing extended memory I/O operations between the primary and backup TCPs.

References

Wong, R. 1984. A New Design for the PATHWAY TCP. *Tandem Journal*. Vol. 2, No. 2. Tandem Computers Incorporated. Part no. 83932.

Acknowledgments

Special thanks are due to Mike Noonan for his many thoughtful contributions to this article.

Robert Vannucci works in the Pathway software development group and has been at Tandem eight years. He is responsible for the enhancements described in this article and has been a key developer of intelligent device support (IDS) for the TCP.

Tandem Systems Review Index — April 1991

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

Volume 1, Number 1	Fall 1983	Part no. 83930
Volume 2, Number 1	Winter 1984	Part no. 83931
Volume 2, Number 2	Spring 1984	Part no. 83932
Volume 2, Number 3	Summer 1984	Part no. 83933

As of this issue, 15 issues of the *Tandem Systems Review* have been published:

Volume 1, Number 1	February 1985	Part no. 83934
Volume 1, Number 2	June 1985	Part no. 83935
Volume 2, Number 1	February 1986	Part no. 83936
Volume 2, Number 2	June 1986	Part no. 83937
Volume 2, Number 3	December 1986	Part no. 83938
Volume 3, Number 1	March 1987	Part no. 83939
Volume 3, Number 2	August 1987	Part no. 83940
Volume 4, Number 1	February 1988	Part no. 11078
Volume 4, Number 2	July 1988	Part no. 13693
Volume 4, Number 3	October 1988	Part no. 15748
Volume 5, Number 1	April 1989	Part no. 18662
Volume 5, Number 2	September 1989	Part no. 28152
Volume 6, Number 1	March 1990	Part no. 32986
Volume 6, Number 2	October 1990	Part no. 46987
Volume 7, Number 1	April 1991	Part no. 46988

The articles published in all 19 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	Season or month and year	Part number
Application Development and Languages					
Ada: Tandem's Newest Compiler and Programming Environment	R. Vnuk	TSR	3,2	Aug. 1987	83940
A New Design for the PATHWAY TCP	R. Wong	TJ	2,2	Spring 1984	83932
An Introduction to Tandem EXTENDED BASIC	J. Meyerson	TJ	2,2	Spring 1984	83932
Debugging TACL Code	L. Palmer	TSR	4,2	July 1988	13693
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
PATHWAY IDS: A Message-level Interface to Devices and Processes	M. Anderton, M. Noonan	TSR	2,2	June 1986	83937
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 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

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
Customer Support					
Customer Information Service	J. Massucco	TSR	3,1	March 1987	83939
Remote Support Strategy	J. Eddy	TSR	3,1	March 1987	83939
Tandem's Software Support Plan	R. Baker, D. McEvoy	TSR	3,1	March 1987	83939
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
Changes in FOX	N. Donde	TSR	1,2	June 1985	83935
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
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
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
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
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

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
Manuals/Courses					
B00 Software Manuals	S. Olds	TSR	1,2	June 1985	83935
C00 Software Manuals	E. Levi	TSR	4,1	Feb. 1988	11078
New Software Courses	M. Janow	TSR	1,2	June 1985	83935
New Software Courses	J. Limper	TSR	4,1	Feb. 1988	11078
Subscription Policy for Software Manuals	T. McSweeney	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,2	June 1986	83937
Operating Systems					
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
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 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	83935
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
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
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
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
Improved Performance for BACKUP2 and RESTORE2	A. Khatri, M. McCline	TSR	1,2	June 1985	83935
MEASURE: Tandem's New Performance Measurement Tool	D. Dennison	TSR	2,3	Dec. 1986	83938
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 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
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

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
Performance and Capacity Planning					
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
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
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
Processors					
Fault Tolerance in the NonStop Cyclone System	S. Chan, R. Jardine	TSR	7,1	April 1991	46988
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
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
The VLX: A Design for Serviceability	J. Allen, R. Boyle	TSR	3,1	March 1987	83939
Security					
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					
Building Open Systems Interconnection with OSI/AS and OSI/TS	R. Smith	TSR	6,1	March 1990	32986
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
The OSI Model: Overview, Status, and Current Issues	A. Dunn	TSR	5,1	April 1989	18662

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
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
Introducing TMDS, Tandem's New On-line Diagnostic System	J. Troisi	TSR	1,2	June 1985	83935
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
Network Statistics System	M. Miller	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 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
Utilities					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939

Index by Product

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	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					
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
Ada					
Ada: Tandem's Newest Compiler and Programming Environment	R. Vnuk	TSR	3,2	Aug. 1987	83940
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
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
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

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
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
Overview of DSM	P. Homan, B. Malizia, E. Reisner	TSR	4,3	Oct. 1988	15748
Network Statistics System	M. Miller	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
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
FASTSORT					
FASTSORT: An External Sort Using Parallel Processing	J. Gray, M. Stewart, A. Tsukerman, S. Uren, B. Vaughan	TSR	2,3	Dec. 1986	83938
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
GUARDIAN 90					
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
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
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	83935
The Tandem Global Update Protocol	R. Carr	TSR	1,2	June 1985	83935

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
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
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 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
PS MAIL					
Enhancements to PS MAIL	R. Funk	TSR	3,1	March 1987	83939
SAFEGUARD					
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

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
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
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
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

Article title	Author(s)	Publication	Volume, Issue	Season or month and year	Part number
Miscellaneous[1]					
A Performance Retrospective	P. Oleinick	TSR	2,3	Dec. 1986	83938
Batch Processing in Online Enterprise Computing	T. Keefauver	TSR	6,2	Oct. 1990	46987
Buffering for Better Application Performance	R. Mattran	TSR	2,1	Feb. 1986	83936
Capacity Planning Concepts	R. Evans	TSR	2,3	Dec. 1986	83938
Configuring Tandem Disk Subsystems	S. Sitler	TSR	2,3	Dec. 1986	83938
Credit-authorization Benchmark for High Performance and Linear Growth	T. Chmiel, T. Houy	TSR	2,1	Feb. 1986	83936
Data-window Phase-margin Analysis	A. Painter, H. Pham, H. Thomas	TSR	2,2	June 1986	83937
Estimating Host Response Time in a Tandem System	H. Horwitz	TSR	4,3	Oct. 1988	15748
Getting Optimum Performance from Tandem Tape Systems	A. Khatri	TSR	2,3	Dec. 1986	83938
Network Design Considerations	J. Evjen	TSR	5,2	Sept. 1989	28152
New Software Courses	M. Janow	TSR	1,2	June 1985	83935
New Software Courses	J. Limper	TSR	4,1	Feb. 1988	11078
Optimizing Batch Performance	T. Keefauver	TSR	5,2	Sept. 1989	28152
Optimizing Sequential Processing on the Tandem System	R. Welsh	TJ	2,3	Summer 1984	83933
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
Peripheral Device Interfaces	J. Blakkan	TSR	3,2	Aug. 1987	83940
Predicting Response Time in On-line Transaction Processing Systems	A. Khatri	TSR	2,2	June 1986	83937
Remote Support Strategy	J. Eddy	TSR	3,1	March 1987	83939
Streaming Tape Drives	J. Blakkan	TSR	3,2	Aug. 1987	83940
Subscription Policy for Software Manuals	T. McSweeney	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,1	Feb. 1986	83936
Tandem's New Products	C. Robinson	TSR	2,2	June 1986	83937
Tandem's Software Support Plan	R. Baker, D. McEvoy	TSR	3,1	March 1987	83939
Terminal Connection Alternatives for Tandem Systems	J. Simonds	TSR	5,1	April 1989	18662
The Performance Characteristics of Tandem NonStop Systems	J. Day	TJ	1,1	Fall 1983	83930
The Role of Optical Storage in Information Processing	L. Sabaroff	TSR	3,2	Aug. 1987	83940

[1]This category is composed of articles that contain product information but are not specifically product-related.

TANDEM SYSTEMS REVIEW ORDER FORM

Use this form to request or renew a subscription, change subscription information, or order back copies.

☐ If you are a Tandem customer, complete Part A of this form and send it to your Tandem representative. Your request is subject to approval.

☐ For other subscribers, complete Part A of the form and send it to the address below. You will receive an invoice for the subscription and back copies that you order. The cost is $40 for a one-year subscription and $15 for each back issue.

Part A. To be completed by the subscriber.

Subscription Information

☐ New subscription

☐ Subscription renewal

☐ 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 Order Requests

Number of copies

Tandem Systems Review

_____ Vol. 1, No. 1, Feb. 1985
_____ Vol. 1, No. 2, June 1985
_____ Vol. 2, No. 1, Feb. 1986
_____ Vol. 2, No. 2, June 1986
_____ Vol. 2, No. 3, Dec. 1986
_____ Vol. 3, No. 1, March 1987
_____ Vol. 3, No. 2, Aug. 1987
_____ Vol. 4, No. 1, Feb. 1988
_____ Vol. 4, No. 2, July 1988
_____ Vol. 4, No. 3, Oct. 1988
_____ Vol. 5, No. 1, April 1989
_____ 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. 1, Winter 1984
_____ Vol. 2, No. 2, Spring 1984
_____ Vol. 2, No. 3, Summer 1984

Tandem Application Monographs

_____ *Developing TMF-Protected Application Software,* 3/83
_____ *Designing a Tandem Word Processor Interface,* 3/83
_____ *Application Database Design in a Tandem Environment,* 8/83
_____ *Capacity Planning for Tandem Computer Systems,* 10/84
_____ *Sociable Systems: A Look at the Tandem Corporate Network,* 5/85

Tandem customers should send this form to their Tandem representative.

Other subscribers send this form to:

Tandem Computers, Incorporated
Tandem Systems Review, Loc 216-05
18922 Forge Drive
Cupertino, CA 95014-0701

Part B. To be completed by the Tandem representative.

Subscription Processing

Please complete this portion of the form to approve your customer's subscription. Your department will be charged $40 per year per subscription. Incomplete requests will be returned for resubmittal.

Back Order Processing

If your customer requests back issues, you must order them through Courier. The menu sequence is:

Marketing Information → Literature Orders → Tandem Systems Review → Back Orders

Your department will be charged $15 for each back issue.

NAME

TITLE | DEPARTMENT NUMBER

LOC | TELEPHONE NUMBER

CUSTOMER NUMBER | SYSTEM NUMBER

SIGNATURE

Send completed approvals to:

Tandem Computers Incorporated
Tandem Systems Review, Loc 216-05
18922 Forge Drive
Cupertino, CA 95014-0701

Process back order requests through Courier.

For our tracking purposes, please indicate the date you submitted the back order request:

_____

Comments:

TANDEM SYSTEMS REVIEW CUSTOMER SURVEY

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

1. How useful is each article in this issue?

Fault Tolerance in the NonStop Cyclone System

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

Overview of the NonStop-UX Operating System for the Integrity S2

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

Enhancing System Security With Safeguard

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

TLAM: A Connectivity Option for Expand

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

Pathway TCP Enhancements for Application Run-Time Support

17 ☐ Indispensible 18 ☐ Very 19 ☐ Somewhat 20 ☐ Not at all

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

21 ☐ Overview discussions of new products and enhancements.
22 ☐ Performance and tuning information.
23 ☐ High-level overviews on Tandem's approach to solutions.
24 ☐ Application design and customer profiles.
25 ☐ Technical discussions of product internals.
26 ☐ Other _____

3. Your title or position:

27 ☐ President, VP, Director 28 ☐ Systems analyst 29 ☐ System operator
30 ☐ MIS manager 31 ☐ Software developer 32 ☐ End user
33 ☐ Other _____

4. Your association with Tandem:

34 ☐ Tandem customer 35 ☐ Tandem employee 36 ☐ Third-party vendor 37 ☐ Consultant
38 ☐ Other _____

5. Comments

_____

_____

_____

_____

NAME

_____

COMPANY NAME

_____

ADDRESS

_____

▶ 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 216-05
TANDEM COMPUTERS INCORPORATED
19333 VALLCO PARKWAY
CUPERTINO, CA 95014-9862

▶ FOLD

▶ FOLD

TANDEM

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

MARC BRANDIFINO
LOC NUM 50-00
NEW YORK NY DISTRICT

400110 4/91 Printed in USA

Part No. 46988