# Implementing Network Attached Storage

Ken Fallon
Bill Bullers
Impact**data**

## Abstract

The Network Peripheral Adapter (NPA) is an intelligent controller and optimized file server that enables network-attached processors to unlock the potential of distributed data access while enjoying the benefits of centralized management and control. Optimized for high-performance and peer-level storage, the NPA is capable of linking virtually any workstation, high-performance computer and/or network with any type of storage device.

By connecting high-performance disk arrays and tape drives to high-speed networks, the NPA creates network storage and allows computers to access and share data. The NPA is embodied within the Distributed Storage Node Architecture (DSNA) of a network-centric computing and data storage enterprise. Initial integrations target Silicon Graphic Inc. (SGI) installations then integrate data sharing between SGI and CRAY systems.

## Introduction

The cutting edge of business, science, and industry, driving for computing, visualization, communications and information content are constantly creating challenges in data and information storage and retrieval processes. Context properties and intelligent formats for management agents are being applied to data. The ever increasing demand for managing larger quantities of data, and the growing requirement for rapid, distributed, global access is forcing the exploration of new approaches to handle and process data to be stored, searched, mined, and made available. This demand is sponsoring the evolution of data storage products that insulate users from storage location, media, and protective services, and is bringing about the concept of enterprise wide ***network computing***. Retrieval, archive, backup, hierarchical storage management applications, and distributed objects are evolving but, intelligence within the data storage devices and media has not been developed. As a result, there is an overwhelming demand for more sophisticated storage servers and intelligent storage media controllers. According to *Computer Client / Server Journal*, "A recent study of 125 companies performed by Strategic Research of Santa Barbara, California, showed that by recentralizing their data storage, companies were able to save an average of 55% over the cost of decentralized storage systems and procedures."

This paper describes an architecture developed by Impact**data** that provides intelligence at the storage device level and enables network attached processors to access and share data, unlocking the potential for distributed data access with central management and control.

## Distributed Storage Node Architecture

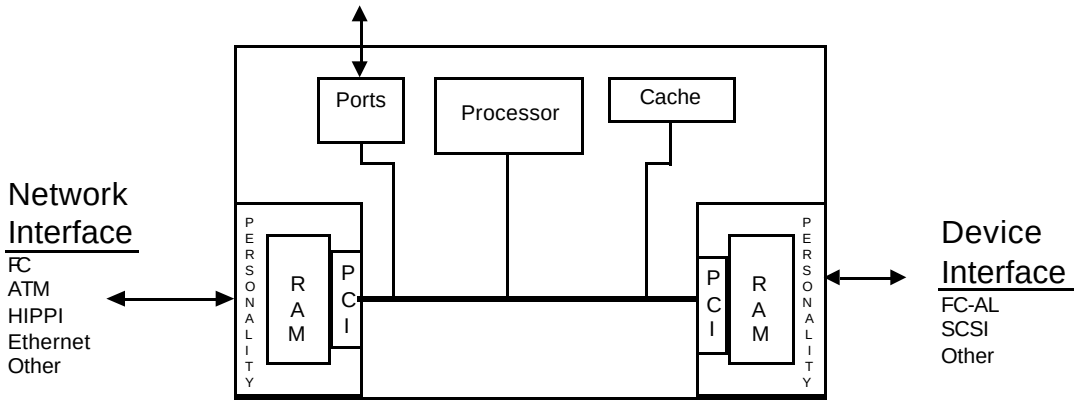
Distributed Storage Node Architecture (DSNA) was created to facilitate data management, data storage, data sharing, data archive, distributed objects, backup and retrieval. DSNA was developed with the flexibility to expand with the needs of high performance computing and high speed network users. Network file systems, like NFS, operate by copying data through multiple memory levels from the client to the file system, to cache, to user space, using small datagram block sizes. As a result they are inherently unable to provide high-performance network data flows. These systems typically apply stateless protocols, with unmanaged concurrency, that requires the full intelligence including file serialization to be provided by the client. NFS clients must be 'smart' because NFS servers are 'dumb'. DSNA is designed with an entirely different structure that integrates file system intelligence into the server and operates with peer-level data flows to deliver large blocks and avoid multiple memory transfers. DSNA is a solution to the limitations of sending large files at high speeds. Applying DSNA protocol, a set of drivers is implemented with a Common Peripheral Interface (CPI), to utilize high speed networks effectively. There are no application level client drivers required.

DSNA is structured to be a complete storage system solution that provides the following benefits:

- A Coherent and Cohesive Network Storage Environment
- Intelligent Storage and Data Mining
- The Management of Data Stores
- Handling Increased Network Bandwidth and Response Times
- Storage Servers and Controllers that Provide for Rapid Growth and Change
- Support for Evolving Distributed Object Storage

The implementation of Distributed Storage Node architecture is based on 'Personality Modules' that support distinct network and storage device interfaces. These Personality Modules are used to achieve fabric-independent integrations for high-performance and traditional networks. DSNA can accommodate interface modules for connection to HIPPI, Fibre Channel, ATM OC-3/12, SCSI, Gigabit Ethernet, and even 10-BaseT and 100-BaseT. Personality Modules are integrated in a Network Peripheral Adapter (NPA) and are connected together through a PCI data bus. High-rate data transfer is achieved by bus-mastering Personality Modules. Taking advantage of memory buffers built into the Personality Modules, the NPA processor reacts to data transfer commands by initiating Direct Memory Accesses (DMA) data flow between the Modules and eliminates multiple processor/cache read interactions with the data. By circumventing the processor, data-flows of up to 100 MB/s can be achieved across the 32 bit, 132 MB/s PCI bus.

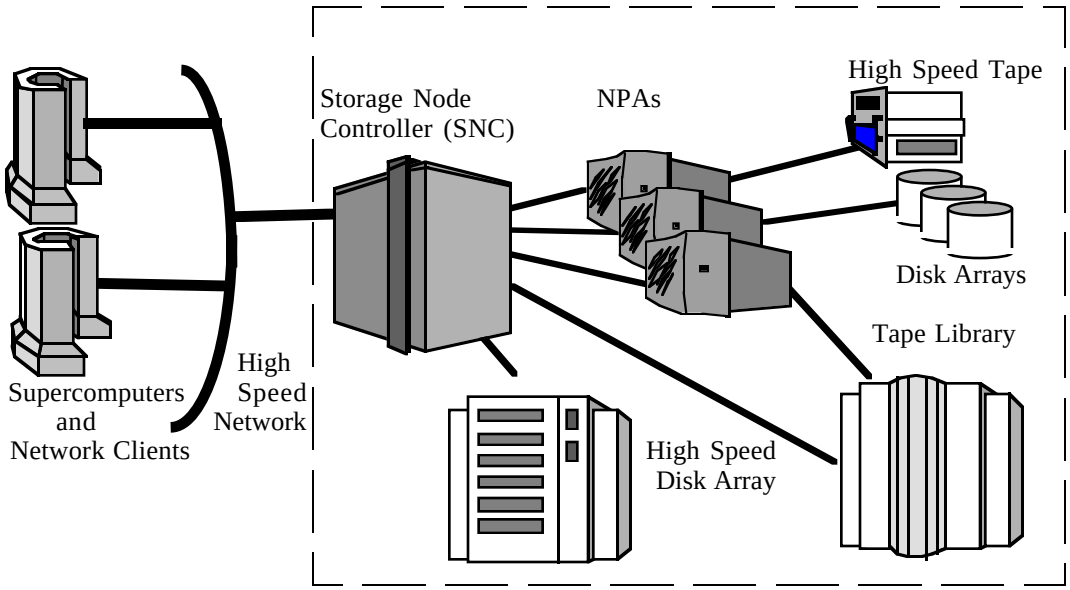
The simple diagram that follows shows the internal structure of the NPA.




The PCI bus-mastering technique achieves high-rate data-flow while significantly freeing the burden on the processor and operating system, and allows a low cost NPA implementation with 'Wintel' technology. The use of personality modules achieves significant flexibility for the DSNA to provide data storage in an open system context. The physical level operation and structure embodied in the NPA is a significant building block, but is only one of several major elements that enable DSNA to provide high-performance distributed data storage.

### Elements of DSNA

The Impact**data** implementation of DSNA creates a storage node with a central Storage Node Controller (SNC), several Network Peripheral Adapters (NPA's) and attached storage devices as shown.



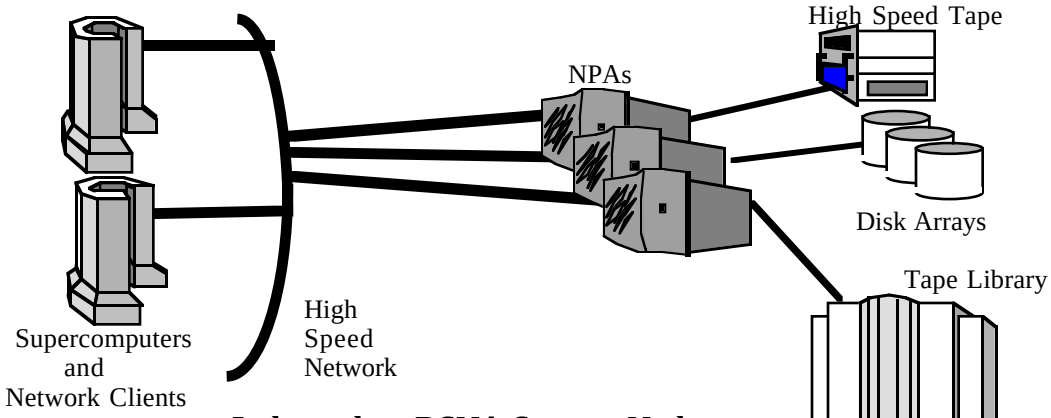

**Complete DSNA Storage Node**

The Storage Node Controller is a high-performance UNIX server used to increase the processing power, capacity, throughput and functionality of a DSNA node beyond the capability of a single NPA. The SNC supports the notion of distributed file management across NPAs and provides the capability for true managed file extent access concurrency across clients. The SNC can be setup as an applications server that provides Hierarchical Storage Management with data archive, backup, and disaster recovery processes. The SNC can provide integrated storage management as part of a 'Distributed Management' methodology in which any processor on the network can assert management functions. Each processor maintains its own management data, such as operational status, configuration records, file and media level storage statistics, alarms, accounting and billing, and software update facilities. Each processor can query every other processor and display this management information for each storage node. The SNC can also be setup to act as a security server to manage the security resources across a DSNA network. Operator access requires ID and password authentication with data access restrictions by ID. C2 security and POSIX compliance are provided in all processors. The SNC will include distributed object services with support for Object Request Brokers as defined through CORBA and ActiveX to promote open access to intelligent storage.

A DSNA storage node implementation can also be as simple as a single NPA that provides intelligent network connected shared data storage. The next diagram shows three device types connected as separate DSNA storage nodes.

The Common Peripheral Interface is middleware that enables UNIX (and Windows NT) clients to use and share network storage as if it were locally attached. Used with any application that reads and/or writes to locally attached devices, the CPI converts Input/Output (I/O) operations to DSNA network commands and the DSNA protocol. Users can take advantage of either a 'transparent' access to DSNA storage nodes or through an Application Programmers Interface (API), can apply a rich set of DSNA features including metadata and local file management. The DSNA protocol has as its foundation the proven IPI-3 storage protocol over which several extensions are specifically designed to accomplish high-performance network shared data storage.

Device drivers are provided for targeted systems that transform standard file I/O requests into DSNA protocol with CPI commands and messages. These CPI drivers provide a local file system connection through defined mount points to DSNA storage nodes with disk and tape media. Host application programs only require modifications if the rich DSNA feature set is to be fully exploited.




**Independent DSNA Storage Nodes**

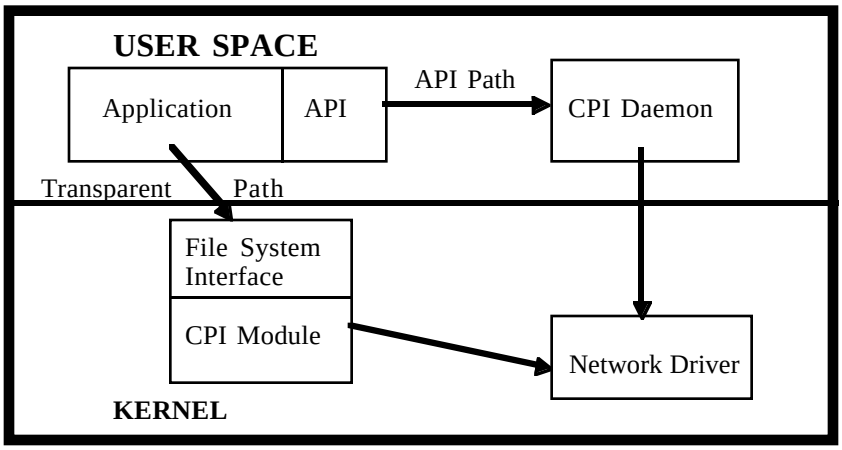
The DSNA protocol and the CPI integrate a metadata facility that collects useful information about the files, the storage devices, and the media. The DSNA Metadata is a vital component of the DSNA file system. The information is collected in real-time and is available to DSNA utilities that use the information to locate and manage directories and files, support file access at the block level, maintain file security and locking, and provide operational statistics.

### Common Peripheral Interface Operation

CPI drivers support two modes of operation, Transparent Mode, and API mode. Transparent Mode allows existing applications to apply DSNA without modifications and achieve high-performance capability. Transparent Mode does not give users the full range of DSNA features. API mode makes the full feature set available to the user. Applications that use Transparent Mode issue OPEN, CLOSE, IOCTL, READ and WRITE calls to a mount point. The mount point configuration/access table insures that the call is routed to the CPI file system interface where it is converted to CPI commands. Two classes of storage device are supported, tape and disk. The API mode is a programmatic interface to the CPI Driver that uses calls to OPEN, CLOSE, READ, WRITE, IOCTL and CPICMD. CPICMD makes Metadata and other DSNA capabilities available to the application. These capabilities include file metadata execution, and administrative data:

- Formatting and Configuration
- File Attribute Reports and Control
- Metadata Reports and Controls
- Operation and Execution Status
- Abort Execution Methods
- File Mark and Position Control
- Diagnostics and Error Logs
- Session Control
- Data Delivery Times

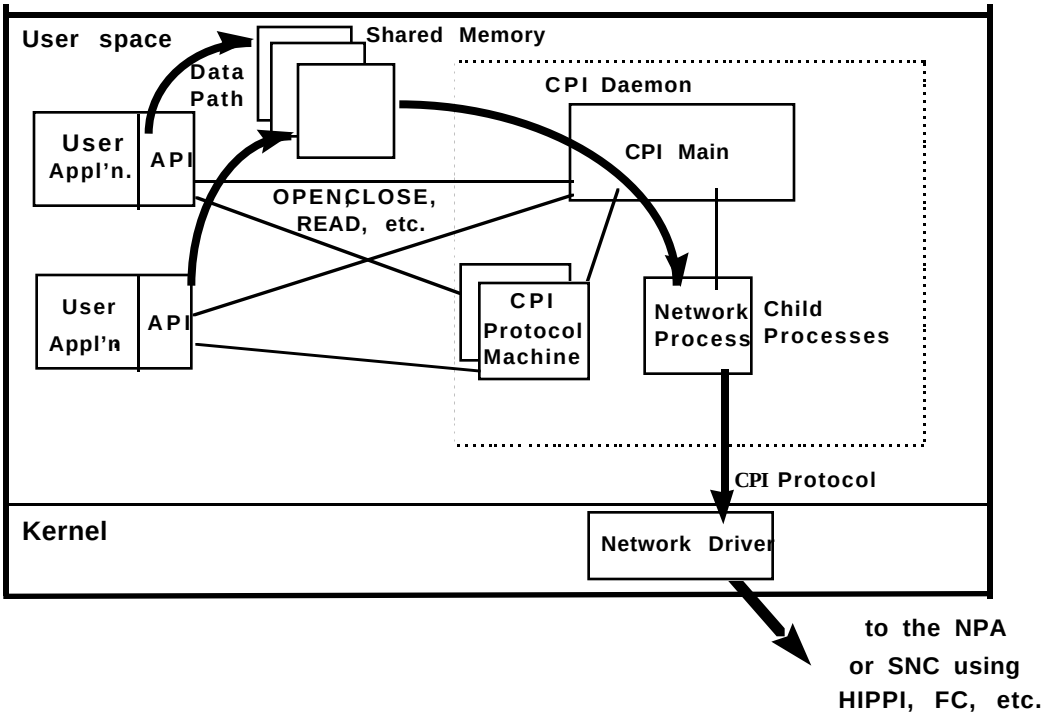
This diagram below shows the difference between how CPI Transparent Mode and API Mode operate within User Space (Application level) and Kernel space (CPI level).




**Common Peripheral Interface Operation**

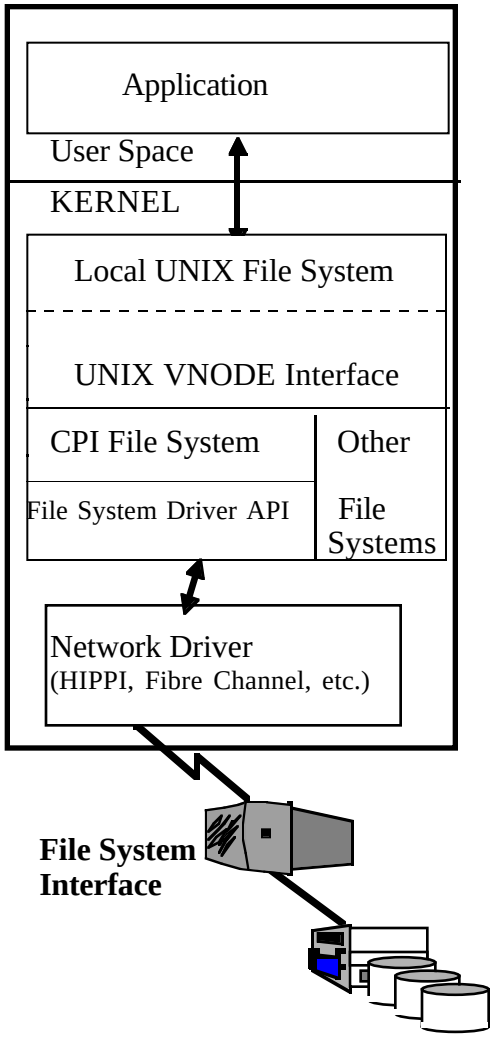
### API Mode

Details of the API Mode operation are presented in the next diagram. The CPI daemon process and the user application that calls the API are independent UNIX processes. The API code runs as part of the application and communicates with the CPI daemon through UNIX inter-process communication facilities. The CPI daemon can service multiple applications allowing several simultaneously accesses to the same NPA storage device. The API is a socket connection interface through which the CPI daemon and application communicate. The CPI daemon spawns a child process for each socket connection and returns the results to the application. The socket connection is used to communicate the API commands and responses. The transfer of data, defined in the API call, takes place through shared memory. The socket connection alerts the CPI daemon to associate an area of shared memory with the application that is used for the data transfer.




### Transparent Mode

The file system appears to be local to the UNIX client in the CPI Transparent Mode. UNIX commands operate normally even though files are maintained on the NPA and available through the network. The CPI Driver uses the mount point identifier to route the file request to the CPI file system. Through the CPI file system, requests are converted to CPI/DSNA protocol and issued to the network driver which then delivers them to the NPA. The NPA processes the request and delivers the data to the destination device through the bus-mastered interface. The CPI file system operates as a Virtual File System within UNIX and supports all the attributes of a local file system. The Local UNIX file system processes client application requests and routes the requests through the UNIX Vnode Interface to the CPI file system as shown in the diagram that follows. The CPI file system is architected into UNIX in the same way other file systems like NFS integrate.




The CPI file system supports features that are common to the Local File System. A configuration/access file is used to enforce system security through encrypted NPA address IDs, license keys, and passwords. Metadata maintained at the NPA, controls file access modes including locking, and maintains user ownership, group ownership, authorization, and privilege information that manages user files. Files that are shared by multiple clients can be locked at different levels including NO locking, READ/WRITE locking, and WRITE locking. The default is lock on WRITE.

CPI supports multiple concurrent accesses from different network attached clients but it is also envisioned to support multiple accesses from a single client. The initial implementation of the architecture is limited to a single shared file handle per client but future releases will enable separate file handles to be created for each of multiple file accesses made by a client. This capability will permit concurrent parallel processing applications to be performed on distinct extents of a file by a single client.

### Other File Systems

DSNA is structured to operate within several different file systems based on the mount point that is selected. The simplest format for operating with file systems other than the CPI file system, like NFS, and SGI's BDS, is to define and allocate specific disk partitions and tape volumes to each system. The first implementations of DSNA will default to this simple strategy. More general implementations of DSNA can be developed that eliminate the need for separate partitions and allow data to be shared across file systems. The Windows NT operating system supports an NT Redirector that allows Personality Module drivers to be written and installed that 'redirect' file accesses for selected mount points from the 'other file system' to CPI. A client application that chooses to operate through NFS or FTP, for example, is able to do that by defining specific mount points for NFS files to be managed through CPI. The NT Redirector diverts the file access control from NTFS to CPI in the NPA and allows the file to be opened at the NPA as a CPI file and at the client application as an NFS file. Clients attached to networks that support TCP/IP such as Gigabit Ethernet and future releases of the Essential Communications HIPPI Card can apply the network performance benefits of CPI to standard current applications.

### DSNA Integration

DSNA has initially been integrated as a high-performance network storage node in a High Performance Storage System (HPSS) at The Caltech Center for Advanced Computing Research by connecting a 72 GB FibreRAID through an NPA to the HIPPI network. The HPSS server provides native support for third-party IPI-3 disk which allows direct data transfers to be setup and executed from the HPSS server to create a data flow path directly between the disk and the client application and avoid the performance lag caused by staging the data through multiple memory read operations. A very similar configuration is planned for integration at the

Lawerence Livermore National Laboratory in May of 1997. These integrations do not initially explore the full capabilities of the DSNA because they are limited to direct IPI-3 connection and do not exploit the data sharing capability provided by the CPI file system.

The first applications that take advantage of CPI will be with SGI client systems. Integrations are currently planned at NASA Ames and at the University of Minnesota that will operate with CPI in the Transparent Mode and provide the capability for multiple SGI clients to share files. Plans are in the formulation stage for similar CPI integrations with HP Convex, SUN Systems, and CRAY. These implementations will provide shared-data operation across the multiple UNIX clients and the multiple client types that provide a Virtual File System Interface.

*Ken Fallon is Project Director for development of DSNA. Bill Bullers is Systems Engineering Director. Bill and Ken can be reached by email at the following: billb@impactdata.com and kenf@impactdata.com. Impact**data** is a wholly owned subsiderary of DATATAPE Incorporated.*