A Brief Survey of Current Work on Network Attached Peripherals

Rodney Doyle Van Meter III
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
rdv@ISI.Edu

January 19, 1996

Abstract

Work on network-attached peripherals (NAPs) can be divided into essentially three areas – device interfaces and protocols, multimedia use and mass storage use. This paper is an extended abstract reviewing some of the current work and provides references and WWW pointers to many of the projects. The impact of this technological advance on operating systems is discussed. The primary purpose of this paper is to broaden understanding of the advantages and pitfalls of NAPs and encourage further research in the design and use of network-attached peripherals and NAP-capable systems. This paper[1] and an extended abstract are available on the web or from the author. [2]

Note: This is a preliminary version of in-progress, unreviewed and incomplete work. Data, conclusions and verbiage may all change. Not yet for public distribution.

1 Introduction

In the past five years, network-attached peripherals have gone from being a research topic in supercomputing environments to production use in a wide variety of areas. Only now, however, is the necessary operating systems support beginning to fall into place. This paper seeks to present the issues involved as well as the current state of the art for NAPs and NAP-capable OSes, in order to familiarize systems engineers whose lives have not yet but will soon be impacted by this new technology.

The focus is primarily storage and multimedia peripherals using new, high-speed interconnects. The information on NAP research and principles is relatively complete; references to related topics such as multimedia operating systems, authentication, distributed file systems, etc. are representative rather than comprehensive. The information on NAP products is also representative due to the rapidly changing state of the market.

The next section defines the characteristics of network attached peripherals. The three major areas of network-attached peripherals work are covered in the following three sections of this paper. Next is a discussion of the relevance of NAPs to operating systems research and development. Following that is a brief, incomplete list of existing network attached peripherals, then the conclusions and references.

2 Characteristics of Network Attached Peripherals

A network-attached peripheral (NAP) is (tautologically) a computer peripheral that communicates via a network rather than a traditional I/O bus, such as SCSI. Typical NAPs will have several characteristics that distinguish them from traditional bus-attached peripherals. These characteristics may be present in varying degrees, depending on the physical interconnect and features of the environment for which they are designed.

[1] http://www.isi.edu/~rdv/netstation/nap-research/index.html

[2] This research was sponsored by the Advanced Research Projects Agency under Contract No. DABT63-93-C-0062. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of ARPA, the U.S. Government, or any person or agency connected with them.

- The physical interconnect is usable over at least computer-room distances, and possibly campus or wide area networks, and connecting potentially very large numbers of nodes. Thus, resource discovery and network routing may be problematic.

- There is no physically-defined **owner** for the device. It may be owned by a single remote system or shared among several, adding complexity to the device controller as well as the software using it.

- The interconnect is capable of carrying general-purpose network traffic, including host-to-host communications. This introduces significant security concerns and may change performance characteristics due to the shared nature of the network.

- Latencies tend to be significantly higher. This affects the command protocols that can be used.

- Data delivery may become subject to traditional network problems, such as packetization and checksumming overhead, fragmentation, out-of-order data delivery, and/or transfer size limitations.

- NAPs are typically capable of talking directly to other NAPs, with only limited supervision by a host computer, and without consuming host resources such as bus bandwidth. This is known as *third party transfer*, and affects many aspects of the system architecture.

- More powerful processors are typically required.

These are the characteristics that distinguish bus-attached peripherals from network-attached peripherals. At the other end of the spectrum, it becomes difficult (and sometimes irrelevant) to distinguish NAPs from network hosts that provide certain services. The obvious example is a special-purpose network node that provide NFS (Network File System)[58, 13] services only – no general-purpose computing facilities. Examples include the Parity Systems Etherstore[48], Auspex NS7000[3], Network Appliance[4] [31] and the Maximum Strategy proFILE XL RAID array[1]. However, the high-level protocol spoken by these is NFS, which provides file-oriented service, an operating system dependent interface. Thus, they would qualify as file servers rather than network-attached peripherals.

A disk NAP would typically provide a block-oriented protocol, such as SCSI or IPI-3, and allow the host operating system to define its own structures on top of the block structure[5]. These structures may be raw partitions, swap space, database partitions, Unix-like file systems (FFS, log-structured, journalled, striped) or file systems with nothing in common with Unix-like file systems (VMS, PCs, mainframes). Katz [37] distinguishes the two as *block servers* and *file servers*.

Some storage subsystems may in fact provide both sorts of interfaces, file and block, allowing the system to be configured flexibly. Notably the MaxStrat proFILE XL is essentially the same hardware as the GEN XL; the former is a file server and the latter is a block server. See section 3.2 for additional discussion of the merits of each.

A common example of a NAP is a tape drive with a HiPPI interface, such as the Sony/TriPlex ID-1 tape drive. It presents an interface like a standard tape drive, except that it is directly available across a network without the interference of a host operating system.

Another example is a network-attached display, which, when running as a NAP, would use a protocol that allows data to be written directly to the frame buffer, rather than a higher-level protocol such as the X protocol, placing the burden of managing the space on the host. Thus, the host can choose to send raw video-like data, still images, X windows, or any other data, without paying the overhead of the X protocols and window management or restricting itself to X semantics.

3 NAP Device Interfaces

The work on device interfaces consists of several areas:

- physical interconnects

[3] http://www.auspex.com/

[4] http://www.netapp.com/

[5] True pedants will argue that the SCSI protocol can be viewed as a file system protocol with fixed-size files (the blocks) and numeric file names (their addresses). While technically true, SCSI provides a simpler interface, "closer" to the hardware, missing much of the functionality we typically associate with file systems – human-readable names, variable sizes, protections, understanding of the concept of users, etc.

- upper-level command protocols
- networking (especially transport) layers
- third-party transfers
- security, authentication, resource discovery and other issues not relevant to normal bus-attached devices

These are detailed in the subsections below.

Terminology held over from bus systems obscures the issues somewhat. SCSI, for example, has historically been used to refer to a system consisting of the SCSI physical interconnect, the SCSI networking (transport) protocols, and the SCSI command syntax and semantics (that is, the RPC interface). These are now on separate standardization tracks, and can be used independently.

3.1 Physical Interconnects

There are several interfaces which are currently in development or being used. Excellent WWW sources of information include CERN's High Speed Interconnect[6] page and LLNL's Standards page[7]. See also Sachs et al[53] for a summary of the network-related issues facing interconnect developers that did not affect channel developers. Interconnects in use include:

- HiPPI (High Performance Parallel Interface)
- P1394 (Firewire, or Serial Bus)
- SSA (Serial Storage Architecture)
- Fibre Channel
- ATM
- Myrinet
- more common, low-performance networks such as ethernet and FDDI
- special-purpose interconnects such as the VAX-cluster CI, UltraNet, or Storage Crossbar

HiPPI[8], which runs at 100 or 200 MBytes/second, is probably the most commonly used NAP attachment to date, though its expense, speed and heavy, short cables have largely limited its use to supercomputers. A parallel copper connection can be switched over computer-room distances, or used as a simple channel. Often used in conjunction with an ethernet for a back channel or control network, as HiPPI interfaces are unidirectional[4, 55]. HiPPI can also carry TCP/IP network as a high-speed LAN. There is some discussion now of improving transfer rates to 1 GB/s with a growth path to 10 GB/s.

IBM's Serial Storage Architecture (SSA) is a relatively new interface. There is a brief, informal history[9] on Micropolis' WWW server, and a collection of pages at the SSA Industry Association[10]. SSA is targetted primarily at dedicated intracabinet and intra-machine-room I/O networks attached to a single processor. It has excellent bandwidth and robustness properties and the apparent advantage of being simpler than Fibre Channel. It is interesting to note that ongoing discussions in SSA standardization include traditional networking topics such as routing, out-of-order delivery and event ordering in distributed systems (e.g., what behavior is appropriate if an `abort` for a particular command arrives before the command itself does?).

Fibre Channel[11] is the other main new interface competing to be the new de facto standard. The Fibre Channel Association[12] maintains an excellent collection of materials. FC can be used for interhost networks as well as I/O. It can be used in physical configurations including a simple channel, an arbitrated loop, or a fully switched fabric.[18]

SBCON[13] is the standardization effort for what started as IBM's ESCON (Enterprise System Connect) for their mainframes. They are trying to take advantage of the Fibre Channel work as well. However, I believe ESCON is treated primarily as a channel.

Serial Bus, known as P1394[14] or as FireWire, comes originally from Apple [61]. It behaves as a system bus, using the IEEE 1212 Control and Status Register low-level address assignments or SCSI as a high-level protocol.

Variants of ATM[54] networks are in use for the Viewstation and Desk Area Network research.

[6] http://www1.cern.ch/HSI/

[7] http://www.cmpcmm.com/cc/

[8] http://www.esscom.com/hnf/

[9] http://www.microp.com/SSA.html

[10] http://www.ssaia.org/

[11] http://www1.cern.ch/HSI/fcs/fcs.html

[12] http://www.amdahl.com/ext/CARP/FCA/FCA.html

[13] http://www.amdahl.com/ext/CARP/SBCON/SBCON.html

[14] http://firewire.org/

Our own Netstation research is using the ATOMIC[15] high-speed switched local area network[22], originating in an interconnect technology for massively parallel computers and being commercialized as Myrinet[16].

The Digital VAXcluster CI and star coupler [39] and the UltraNet are currently in only limited use, due to the aging of the technology. Solflower Computer's Storage Crossbar[56] is a newer technology with some similarities to the VAXcluster. These all suffer from the drawback of being non-standard interconnects.

SSA and P1394 arguably do not qualify as network-attached peripheral interconnects, since they do not carry general-purpose network traffic and are oriented toward a single-host environment. Firewire in particular, which arbitrates and does resource discovery like a bus, has more in common with a bus than a network.

Most of these network technologies are intended to be used primarily in homogeneous environments, although some level of interoperation is supported. For example, both the HiPPI framing protocol[3] (which defines the packet format) and ATM have defined mappings on top of a Fibre Channel physical connection. Thus, care must be taken when referring to the higher-level protocols to distinguish them from the physical interconnects. Figure 1 shows some possible methods of using an IPI-3 upper-level protocol. HiPPI framing protocol can be used over either HiPPI-PH or via a mapping to Fibre Channel, or IPI can use its own direct mapping to Fibre Channel services. Using HiPPI-FP over Fibre Channel is only likely in the event of a heterogeneous interconnect involving both.

The network structure for these interconnects tends, rather than directly following the ISO 7-layer model[59], to have a flatter structure. The upper-level protocols often are involved in packet framing and flow control, which may complicate use of heterogeneous interconnects.

A list of the email reflectors concerning many of these interfaces can be found in the Usenet comp.arch.storage newsgroup Frequently Asked Questions (FAQ).[17].

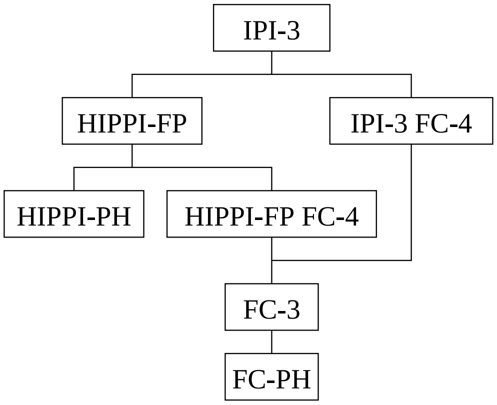

Figure 1: IPI-3 Over HiPPI and Fibre Channel

3.2 Upper Level Command Protocols

Maintaining compatability with directly attached peripherals and retaining existing device and system firmware have been key goals of the NAP effort to date. The implementors of the devices themselves have been "coming up from the bottom", that is, working from an existing base of channel-attached devices and working up toward full network-attached peripherals. Thus, the protocols used at the devices themselves have been drawn mostly from SCSI and IPI-3. Historically IPI-3 has been preferred, but SCSI is becoming increasingly common.

SCSI as a command protocol (SPC[7], the SCSI Primary Commands, are common to all SCSI devices, plus each device type has a type-specific set such as the SCSI Block Commands for disk drives) runs over SSA, Fibre Channel and Serial Bus. IPI-3 is commonly used over HiPPI, and runs over Fibre Channel as well. These are all block-oriented protocols. IPI-3 provides transport up through application layers in the ISO networking model. SCSI relies on the lower-level network interfaces to provide some of these services. SCSI grew into what is known as SCSI-3[6] partially as a result of the desire to use SCSI for NAPs.

It has been suggested[66] that certain aspects of the SCSI model have shortcomings from a networking point of view. Notably, the syntax of SCSI RPCs (remote procedure calls), including small fixed-size fields, the low upper limit on RPC control block sizes (sixteen bytes), and restricted opcode space (256, broken up somewhat and occassionally reused) are bothersome. Restriction to big-endian syntax also becomes unnecessary if a lower-level network layer

[15] http://www.isi.edu/div7/atomic2/
[16] http://www.myri.com
[17] http://alumni.caltech.edu/~rdv/comp-arch-storage/FAQ-2.html

can provide any necessary translation. These are holdovers from SCSI's history as a bus interconnect.

The larger problems with SCSI are semantic. There is no provision in SCSI for unreliable (datagram) unacknowledged RPCs, a feature considered to be useful in network systems but not generally used with devices. Identifiers for nodes are limited-length numbers[18]; in some cases human-readable identifiers (probably internet-style domain names) may be easier to manage. A key shortcoming, as with IPI-3, is SCSI's simple security model. Access can generally be controlled for concurrency, but not protection, and it is desirable to be able to restrict the execution of individual commands, especially management commands such as adding/deleting access capabilities for other nodes.

Most distributed file systems, such as NFS, are built on an RPC interface built on datagram network services such as UDP[52]. The file system semantics may be either stateless (NFS) or stateful (Sprite[?], Spring[47]) Some NAPs, such as the 2nd-generation LLNL RAID array, use TCP[51] to set up a connection for each transfer as it is initiated. A disadvantage of this is the potentially long latency to begin a transfer. Others use IPI-3 or a particular SCSI protocol (such as the Fibre Channel Protocol, FCP [5] as the transport protocol to insure the fidelity and ordering of data as it is sent through the network. This is especially important for transfers that exceed the network maximum transfer unit (MTU).

The key point is that the protocol presented should be low-level enough to allow the host operating system to define any structure it desires without paying a significant penalty in overhead for unused functionality.

Maximum Strategy, as part of Cray's Shared File System effort[45], has augmented their Gen 5 storage array, which uses the IPI-3 command set, to include support for semaphores at the device. Their first two implementations used a separate semaphore server, at first a custom hardware device and later a dedicated Sun SPARC. The current version can use either the semaphore server or semaphores at the array. Note that the storage array itself attaches no semantic meaning to the semaphores; cooperating clients running the SFS must agree on the meanings of the semaphores. Thus, one rogue client can still compromise the file system. In addition to the HiPPI array, SFS has been tested with SCSI devices in conjunction with the semaphore server. It may be possible to build a semaphore server or equivalent mechanism on top of SCSI by appropriate use of extent reservations, linked commands, and perhaps `READ BUFFER` and `WRITE BUFFER`.

Some researchers have proposed that the SCSI block-oriented approach is too low-level, while recognizing the limitations of the NFS server approach. They have proposed more object-oriented semantics for the network node, rearchitecting the file system by moving the file system/device level boundary to take advantage of the strengths of NAPs. One possibility is to have the disk drive store *objects*, reached via the triple `<objectid,offset,length>` [25], reminiscent of but more advanced than `<count,key,data>` mainframe disk drives.

The protocols for supporting third-party transfer are complex; see section 3.4 for a discussion.

3.3 Security

The issues of security are only now beginning to be addressed; to date the assumption seems to be that networks used for storage peripherals are secure either physically or due to constraints imposed by the lower networking levels.

One difficulty is that some of these are hard to do efficiently, and above all a network peripheral is useless if it isn't fast.

The concerns of security can be divided into several parts, well-known to programmers of distributed systems[42], but not common issues for peripherals:

- authentication of authority to execute a given command
- authentication of source of data and command status
- integrity of data
- privacy of data

Another important element in the security of the data on the disk drive in a system is that it must be impossible for user processes on the host to directly access the disk drive. Existing systems typically do not restrict outgoing network traffic with respect to protocol and port or destination address, but without such limits user processes could send arbitrary commands to the disk drive, bypassing the normal file

[18] Architecturally, 64 bits, but the number of bits implemented varies with the physical interconnect. 64 bits is smaller than the 128 bit addresses in IPv6, a potential drawback.

system protection mechanisms and reading or modifying any files as well as the file system metadata. Thus, the disk drive cannot validate requests only on source address, even in an environment where address spoofing (as is possible with IP) is impossible. Such protection is inadequate; the drive must confirm that the request is from the system kernel or file manager process (or a party authorized by them). A cryptographic exchange to confirm the identity of requestor may be necessary.

The Zebra striped network file system[27] achieves its level of protection by allowing writes to the storage server to append to the log without authentication. However, the written blocks do not become part of the visible file system until the file metadata has been updated by the file manager process, which performs appropriate permission checks. The worst effect unauthorized writes may have is to cause the storage server's garbage collector to run more frequently. Unauthorized reads of files could be prevented by allowing reads only on presentation of an appropriate key which can only be obtained from the file manager.

3.3.1 Protection Semantics

When devices existed in a relatively benign environment, consisting of one or a few cooperating initiators and protected by the operating system from direct access by user processes, the only need was for concurrency control, not validation of permission to execute commands. Access rights were assumed, and the semantics of, for example, the SCSI RESERVE command[6] are of the shared/exclusive read/write variety. Building access rights equivalent to "read only access for blocks 1–10, and no access to other blocks" requires several reservations (which may all be executable via one RESERVE command) reserving exclusive access to all areas outside the permitted area, reserving write access to the desired area to a "safe" initiator, and reserving read access to the desired area to the desired initiator.

In addition to read/write concurrency control for data blocks, it is possible to imagine a number of other protection semantics that might be useful:

- disk read only.
- disk write only (or write before read allowed). This would allow the system to prevent reading data from "deleted" files, for example, without executing a time-consuming erase or overwrite operation.
- tape append only. Thus, multiple users' data can be added to one tape safely.
- limit volume of data written to a tape.
- initialization of disks and tapes should be well-controlled.
- management functions should be protected (network address and other transfer parameters, device ownership, cacheing or prioritizing customization, operating mode, RAID array data layout configuration, etc.).

It is clear from this list that virtually every request should be pre-approved in some fashion; arbitrary nodes in the network (which may be the Internet itself) cannot be allowed to execute almost anything!

3.3.2 LLNL NAP

At Lawrence Livermore National Labs, research is underway on their second generation network-attached storage device[19] (a RAID susbsystem). Security is currently provided by physical isolation of the control network for the device. The data network is a HiPPI network, which allows transfers to a number of client machines. The control network is a dedicated ethernet, which only storage peripherals and the storage server are attached to. The clients make requests of the server machine, which then instructs the NAP to execute the transfer. The storage devices are assured that the requests are legitimate because the only host capable of sending them is the server, and no user processes that might generate suspect requests are allowed on the server.

This solution, however, is recognized to be inadequate in the long term. A dedicated network for control of the peripherals is an expensive solution that does not translate well outside the machine room. Plans call for some sort of cryptographic verification scheme to determine the validity of commands.

3.3.3 Fibre Channel

As an example, Fibre Channel supports protection of devices based on their position in the network topology. However, this is a side effect of the desire to simplify implementation at Arbitrated Loop devices,

[19] http://www.llnl.gov/liv_comp/siof/siof-nap.html

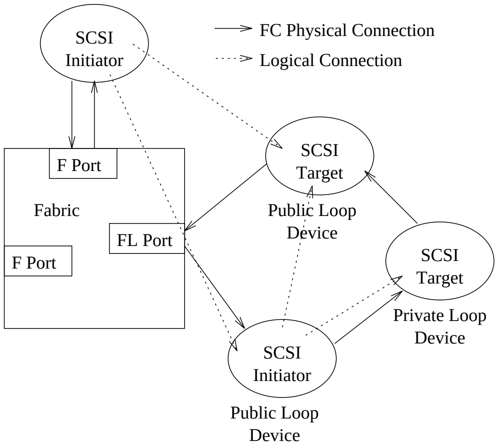

Figure 2: Fibre Channel Private Loop

rather than a policy decision to provide a protection mechanism.

The current choice for implementing security is to put the SCSI peripherals either on a network that is physically isolated, or as private loop ports on an arbitrated loop[65] (AL) that contains no "dangerous" untrusted ports. In figure 2, an arbitrated loop is connected to a fabric. The dotted arrows indicate access that is allowed. The SCSI Initiator external to the loop can access the device configured as *public loop*, but not the device configured as *private loop*. The AL might be physically inside the case of a workstation. The SCSI initiator would be the system CPU, the private device might be a disk drive, and the public device might be a camera. The connection to the FC fabric may be used as the system's LAN connection for general-purpose traffic.

The fabric itself might also contribute to the security of the private loop device by refusing (or being unable) to carry external traffic bound for the private loop device.

There appears to currently be no means of protecting peripherals in a fabric. There is no means of authenticating specific requests (with respect to allowed initiators or allowed parameter values) or modifying the set of initiators that can access devices. Access is all or nothing, based on position inside or outside the arbitrated loop. This is assuming it is impossible to spoof Fibre Channel communications so that communications external to the loop appear to be internal.

3.4 Third-Party Transfers

Third party transfer is functionally perhaps the most interesting new feature provided by NAPs. While in theory this capability has been included in the SCSI command set (the `COPY` command) for some time, it is only now becoming widely used. The operating systems support for third-party I/O is still in its infancy, and even the protocols to support this have been problematic.

Simply put, third-party transfer is a request from a party for a data transfer in which that party is neither the source nor the sink for the data. A third party transfer might involve, for example, a host computer instructing a disk drive to transfer data to a tape drive (or vice-versa) or to a frame buffer. Data does not have to transit the host's bus or be copied by the host; it transfers directly from the disk drive to the tape drive through the network.

Hyer et all[33] discuss the hazards of getting a NAP to cooperate with an existing system, pointing out flaws in IPI-3 that make it unsuitable for third-party use, especially the lack of an authentication mechanism. Using the IPI-3 `COPY` command results in the **mover**[20] for one of the devices being left out of the command loop, making device management more complex. Their solution involves creating a variant of IPI-3 third party transfer in which the initiating device itself sends a request to the responder's mover rather than directly from the device itself. The host directly transfers the first and last blocks of a long transfer that doesn't fall on block boundaries, because IPI-3 does not support arbitrary offsets.

Third-party transfer is generally considered to be a high-overhead operation, and as such is only useful for large transfers. The Livermore group has identified a sequence of 23 steps necessary to execute an authenticated third-party transfer, including authorizing the transfer to the endpoints and providing them with means of recognizing communication from the other endpoint (e.g. using cryptographic methods to authenticate the source of the RPC and data).

3.5 Continuous Media Services

One goal, especially for the multimedia-oriented systems, is to provide real-time delivery of data, such

[20] In the OSSI[35], the mover is the lowest-level entity controlling the device, responsible for moving data. In a bus-attached system, the mover may be the device driver. For a NAP, however, part or all of the mover's functionality might be implemented at the NAP itself.

as graphics data. While research is being conducted on such topics within non-NAP systems [57] and for networked systems, I know of no work specifically related to using NAPs for real-time services. The Fibre Channel community has considered isochronous classes of service but the work is low priority. The ATM standard supports isochronous transfers.

This issue is tightly bound to the issues of data characteristics and networking protocols for bandwidth reservation, performance guarantees and quality of service (QOS).

3.6 Network Parallelism

Most system buses, such as SCSI, support only a single concurrent transfer, since they are broadcast buses. The same is true of networks such as ethernet, Fibre Channel arbitrated loop, and FireWire. Most of the other interconnects under discussion here, including Myrinet, HiPPI, Fibre Channel fabrics and ATM, are switched networks that allow multiple full-bandwidth transfers to execute concurrently. SSA even in its loop form supports some *spatial reuse* [28]. Transfers that do not pass through the same loop node can run independently.

In addition to supporting parallel file systems (see section 6 below) via concurrent transfers from separate devices to separate compute nodes, it is possible to transfer data in parallel between two endpoints, if two or more paths between the nodes exist[64]. The Parallel Transport Protocol proposal[9] provides a means for specifying logically concurrent transfers between groups of NAPs and mapping data from N sources to M sinks. Jain et al propose graph coloring as a means for optimizing the use of sources and sinks in concurrent transfers[36]. It is also possible (Zebra) to utilize multiple servers for a single client, similar to a parallel file system or distributed RAID array. The TickerTAIP [14] distributed array transfers to and from the network-attached disks in parallel.

4 NAP Multimedia Research

Several research projects concerning using network-attached peripherals in multimedia workstations are ongoing in various universities. The canonical example of the uses for NAPs in multimedia is the desire to transmit data directly from a camera to a frame buffer without passing through the system's backplane, where it unproductively consumes bandwidth. Capture of video to disk and playback from disk are similar.

- Netstation[21] – Greg Finn's group at ISI[23, 66].
- The ViewStation work is being done by David Tennenhouse's Telemedia, Networks, and Systems Group[22] at MIT[43, 32, 2].
- The Desk Area Network work is being done at Cambridge[8, 40, 29]. Some of this work has now been commercialized by Nemesys[23].
- Symphony[26] is concerned with intra-node hardware and software architectures to support real-time network protocols.
- At Los Alamos National Laboratory, an experimental system that drives a HiPPI frame buffer from a farm of Alpha workstations has been built.[62]

5 NAPs in Mass Storage

"Mass storage" in the context of this section means systems that support very large numbers of files and users, typically with total data volumes in excess of a terabyte. The key conferences on this topic are hosted by the IEEE Computer Society [15] and NASA's Goddard Space Flight Center [38].

NAPs in mass storage are used in hierarchical storage management (HSM) systems, as well as with channel extenders for remote copying of data. NAPs have been used in hierarchical storage systems for a number of years (primarily HiPPI disk arrays), but the increasing speed and sophistication of both the peripherals and the HSM software has truly brought NAPs to the forefront recently. See Coleman and Watson[16] for a good introduction to HSM (and good references on the history of network-attached peripherals).

The SSSWG[24] is the IEEE's Storage Systems Standards Working Group. The SSSWG's Open Storage Systems Interconnection reference model[35] defines a structure for a set of standards relating to mass storage, and (implicitly) incorporates NAPs.

The High Performance Storage System[25] (HPSS)[68] being de-

[21] http://www.isi.edu/div7/netstation/netstation-home.html

[22] http://www.tns.lcs.mit.edu/tns-www-home.html

[23] http://www.nemesys.uk/

[24] http://www.arl.mil/IEEE/ssswg.html

[25] http://www.ccs.ornl.gov/HPSS/HPSS.html

veloped at the National Storage Lab[26] uses NAPs. This is quite probably the most advanced work on network attached peripherals, and has been tackling the important issues of security and parallel transfers. The Parallel Transport Protocol proposal[9] is related to HPSS. There is some related work in the Sequoia 2000[27] project on the network aspects[28] of mass storage and especially operating system I/O. SIOF[29], the Scalable I/O Facility is targetting I/O for massively parallel supercomputers. SIOF will also use HPSS.

Channel extenders, such as the CHANNELink[30] from CNT and the Symmetrix Remote Data Facility[31], are used by some mainframe systems to create remote copies of disks (remote mirroring) as a disaster recovery measure. Early systems used dedicated fibre or telephone lines and ran proprietary communications protocols. Newer systems from CNT are capable of communicating over general-purpose wide-area networks, thus saving the costs of the dedicated lines. Typically the controller is transparent to the mainframe, and copies the data to the local disk as well as the remote disk, buffering the data as necessary for the network transfer. The subsystem can be configured so that the data must be committed to both disks or only the local disk before the command is reported as complete. There are also channel extenders such as the ChannelHIway[32] from Essential Communications for HiPPI and others for SCSI, which do not copy a disk but do allows devices to be used over significant distances.

Katz[37] compares different hardware approaches to the problem of networked storage. He discusses the distinction between "block servers" (NAPs) and "file servers". The DEC VAXcluster HSC, Control Data disk array controller, Auspex NS5000, Maximum Strategy HiPPI-2 array controller, and Berkeley's own RAID-II are covered in detail. The HSC, CDC, and MaxStrat controllers are clearly block servers, and the Auspex clearly is a file server; the RAID-II is more of a hybrid system.

The RAID-II system developed at UC Berkeley[21] blurs the distinction between network-attached RAID array and file server. The host system behaves as a typical NFS server to most clients, transferring data first from the disks to the server's memory, then across the network. Thus, the data passes twice across the server's memory bus. For client applications linked with the UltraNet networking library, however, data can transfer directly across the high-speed XBUS to the UltraNet and to the client *without* passing through the server's memory. In this case, the server manages the data and initiates transfers, but need not be in the data path, a canonical example of third-party transfers and the uses of network-attached peripherals.

The Swift distributed RAID array[12, 44] was the first project to propose striping of data across multiple network connections as an alternative to striping on local disks. Their approach involves creating *transfer plans* to support the striping.

The TickerTAIP distributed RAID array[14] is composed of network-attached disks. It represents important work in calculation and management of distributed parity, especially for small writes.

As covered in section 3.2, Cray has implemented a Shared File System for a HiPPI RAID array. They have achieved read rates through the file system, which involves setting shared-read semaphores at the semaphore server, of 12 to 84 megabytes per second, as transfer size varies from 64KB to 16 MB[33].

The Solflower Computer Storage Crossbar [56] provides direct high-bandwidth access to SCSI disks to up to sixteen Sun workstations. Using a custom file system, also known as *Shareable File System* (SFS), that links into the kernel at the *vnode*, access to the disks is coordinated to prevent metadata corruption. Although details of the implementation are proprietary, in principle it seems to have some similarity to VAXClusters[39], providing buffering and multiple device control, and optionally acting as a processor-to-processor communications path. The Storage Crossbar may have some of the concurrency control and internal security problems common to NAPs, but resource discovery problems and external threats should not be present.

6 Operating Systems

A system using NAPs is a heterogeneous distributed system. The various nodes in the system provide different services. A processor node provides compute services to the users of the system. A disk node pro-

[26] http://www.llnl.gov/liv_comp/nsl/nsl.html

[27] http://s2k-ftp.cs.berkeley.edu:8000/

[28] http://www-cse.ucsd.edu:80/users/pasquale/Projects/Sequoia.html

[29] http://www.llnl.gov/liv_comp/siof.html

[30] http://www.cnt.com/products/clnk/clnk2.htm

[31] http://www.emc.com/symmdoc.htm

[32] http://www.esscom.com/

[33] the logical block size of the array is 64KB

vides stable storage, typically managed by the operating system of a host node.

As a distributed system, the existing body of research on issues such as resource control and deadlock, naming, caching, etc. is all relevant. Especially important is the work on distributed file systems[41, 67, 47, 24, 11, 67].

An important realization is that the resources are truly distributed. A disk drive that "belongs" to no processor may contain a file system that is shared by multiple clients, necessitating a new synchronization policy between clients. This may make distributed systems such as Amoeba[60] or Plan 9[50] more efficient and more easily location-transparent. The boundaries for which nodes are and are not technically part of "my" system become less clear, as well.

Also relevant is the work on operating systems for distributed-memory multicomputers[63], such as the Intel Touchstone Delta/Paragon[34] family, which has some nodes dedicated to I/O and others to computation, and the IBM SP and Cray T3D machines. Numerous studies on I/O performance[10] and file system design [46, 20, 17, 19] have been done. Some of this work includes, for example, distributed file block layout and synchronization mechanisms that may prove useful for NAP file systems. A key resource for research in this area is David Kotz's excellent page on parallel I/O[35].

If the host operating system device driver structure is carefully layered, it should be possible to replace the transport layer below, for example, the SCSI disk driver, with the code necessary to reach the NAP via the network, and run the system transparently as though the disk were connected to a local SCSI bus. This is the approach taken for SCSI disks on a Fibre Channel or SSA network.

Research into significant changes in the I/O paradigm presented to applications programmers, such as the work on *containers*[49], is beginning to address ways of making the system efficient. Containers separates the actions of causing an I/O to occur and mapping the resulting data into the process' address space. When the mapping is executed, the return value includes a pointer to the data, rather than the buffer to be filled being an argument on input. Thus, the operating system determines the placement and alignment of data. These two features make containers an excellent candidate for integrating third-party I/O (where mapping the data is inappropriate) and network I/O (possibly eliminating a data copy to the user's supplied buffer) into the standard I/O model.

The operating system for the Cambridge DAN work is known as Pegasus[40]. Pegasus[36] is intended to support transfers to and from multimedia peripherals at appropriate data rates. The system has one large, shared address space for all nodes and processes, similar to shared-virtual-memory multicomputers such as the KSR-1. Files are memory-mapped and managed in conjunction with virtual memory, as in Multics or Plan 9.

Existing systems sharing file systems on NAPs, such as Cray's SFS, DEC's VAXclusters, and Solflower's SFS, depend heavily on correct behavior from all clients. In each case the file system code of the host operating system has been significantly modified so that metadata updates are consistent, and file writes do not cause problems. Non-cooperating clients (or unauthorized requests originating at normally cooperating clients) could potentially read or modify any directory or file or file system metadata.

Cray SFS, for example, uses semaphores maintained at a centralized semaphore server so that multiple nodes can be reading (and caching) a file or a single node writing it. It is robust against failure of all (but one) hosts, but not failure of the semaphore server (or, obviously, the NAP itself). These semaphores may, when interpreted, lock an object as small as an inode, but that convention is enforced by the hosts rather than the NAP.

VMS for VAXclusters uses a distributed lock manager that attempts to be robust against failures of hosts and to distribute the workload for managing locks. The lock manager performs numerous functions, but its principal role is in the file system. To prevent partitioning of the cluster and its potentially disastrous consequences, failure of only up to half of the nodes can be tolerated, but there is no single point of failure for the lock manager, and the lock manager supports disks distributed arbitrarily around the cluster, attached to other host nodes or to one or more Hierarchical Storage Controllers.

There is a large body of work on network and operating systems support for multimedia, notably the workshops on digital audio and video [30]. Also see papers such as [57] . These have not focussed specifically on NAPs, but the principles are important.

[34] http://www.ssd.intel.com/

[35] http://www.cs.dartmouth.edu/pario.html

[36] http://www.pegasus.esprit.ec.org/papers/pegpapers.html

7 Existing Network Attached Peripherals

Numerous network-attached peripherals using IPI-3 over HiPPI already exist and are in production use, primarily in supercomputing environments. The use of other interfaces, such as Fibre Channel, is still largely in testing phases, though items such as Sun's Fibre Channel disk array (early versions of which treated the FC interface as a channel only) have been in use in restricted environments for some time.

The following list is *very* incomplete, and is intended to give only a flavor of the range of peripherals that are available. Consulting the web servers for the various interface standardization efforts mentioned above leads to various lists of vendors. There may also be additional information on this topic in the comp.arch.storage Frequently Asked Questions (FAQ)[37].

- Maximum Strategy[38] makes high-speed supercomputer RAID arrays that speak either IPI-3 block protocol or NFS over HiPPI, ATM or Fibre Channel[1].
- HiPPI tape drives available include the Datatape ID-1 and Sony ID-1 with a Triplex interface. These also use IPI-3 as the transport and application layers.
- Storage Tek's Redwood tape drive and IBM's Magstar tape drive will have SBCON interfaces.
- More than a half dozen companies, including Quantum, Conner and Seagate, have or will soon have Fibre Channel disk drives[39]. These will likely be used only on private arbitrated loops in the near term.
- Micropolis, IBM, Conner and others have or will soon have SSA disks.
- Companies such as PsiTech and Avaika make HiPPI frame buffers.

8 Conclusions

I have shown various facets of the state of the art in network-attached peripherals. Areas that remain ripe for research include improved security (internal and external, and especially flexible), syntax and semantics of RPC and lower-level networking protocols, real-time use, lower-overhead third-party use, and especially changes in operating system I/O paradigms to support the efficient use of third-party transfers.

[37] http://alumni.caltech.edu/~rdv/comp-arch-storage/FAQ-1.html

[38] http://www.maxstrat.com/

[39] http://www1.cern.ch/HSI/fcs/storage.html

References

[1] File server manages HPC networks. *Parallel and Distributed Technology*, 2(4):90, 1994. Maximum Strategy product announcement in *New Products* section.

[2] J. F. Adam, H. H. Houh, M. Ismert, and D. L. Tennenhouse. Media-intensive data communications in a "desk-area" network. *IEEE Communications*, pages 60–67, Aug. 1994.

[3] ANSI. High-performance parallel interface – framing protocol (HIPPI-FP). Technical Report X3T9.3/89-013 rev 4.2, June 1991.

[4] ANSI. High-performance parallel interface – mechanical, electrical and signalling protocol specification (HIPPI-PH). Technical Report X3T9.3/88-023 rev 8.1, June 1991.

[5] ANSI. Information systems – fibre channel protocol for SCSI (FCP). Technical Report TR X3T10-993D R010, ANSI, Sept. 1994.

[6] ANSI. Information technology – SCSI-3 architecture model. Technical Report TR X3T10-994D R17, ANSI, June 1995.

[7] ANSI. Information technology – SCSI-3 primary commands. Technical Report TR X3T10-995D R6, ANSI, Mar. 1995.

[8] P. Barham, M. Hayter, D. McAuley, and I. Pratt. Devices on the desk area network. *J. Selected Areas in Communications*, 13(4):722–732, May 1995.

[9] L. Berdahl. Parallel transport protocol proposal. Lawrence Livermore National Labs, January 3, 1995. Draft. ftp://svr4.nersc.gov/pub/Pio-1-3-95.ps.

[10] R. Bordawekar, A. Choudhary, and J. M. del Rosario. An experimental performance evaluation of touchstone delta concurrent file system. In *International Conference on Supercomputing*, pages 367–376, 1993.

[11] U. M. Borghoff. Design of optimal distributed file systems: A framework for research. *ACM Operating Systems Review*, 26(4):30–61, Oct. 1992.

[12] L.-F. Cabrera and D. D. E. Long. Swift: Using distributed disk striping to provide high i/o data rates. *Computing Systems*, 4(4):405–436, 1991.

[13] B. Callaghan, B. Pawlowski, and P. Staubach. NFS version 3 protocol specification, June 1995.

[14] P. Cao, S. B. Lim, S. Venkataraman, and J. Wilkes. The TickerTAIP parallel RAID architecture. In *Proc. 20th Annual International Symposium on Computer Architecture*, pages 52–63, May 1993.

[15] S. Coleman, editor. *Twelfth IEEE Symposium on Mass Storage Systems*, Apr. 1993.

[16] S. S. Coleman and R. W. Watson. The emerging paradigm shift in storage system architectures. In *Proceedings of the IEEE*, pages 607–620, Apr. 1993.

[17] P. F. Corbett, S. J. Baylor, and D. G. Feitelson. Overview of the vesta parallel file system. *Computer Architecture News*, pages 7–14, Dec. 1993.

[18] R. Cummings. System architectures using fibre channel. In Coleman [15], pages 251–256.

[19] E. P. DeBenedictis and J. M. del Rosario. Modular scalable i/o. *J. Parallel and Distributed Computing*, 17:122–128, 1993.

[20] P. C. Dibble and M. L. Scott. Beyond striping: The Bridge multiprocessor file system. *Computer Architecture News*, 19(5), September 1989.

[21] A. L. Drapeau, K. W. Shirrif, J. H. Hartman, E. L. Miller, S. Seshan, R. H. Katz, K. Lutz, D. A. Patterson, E. K. Lee, P. H. Chen, and G. A. Gibson. RAID-II: a high-bandwidth network file server. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 234–244, 1994.

[22] R. Felderman, A. DeSchon, D. Cohen, and G. Finn. ATOMIC: A high speed local communication architecture. *J. High Speed Networks*, 3(1):1–29, 1994.

[23] G. Finn. An integration of network communication with workstation architecture. *ACM Computer Communication Review*, Oct. 1991. Available online at ftp://venera.isi.edu/atomic-doc/ATOMIC.Netstation.ps.

[24] M. Fridrich and W. Older. Helix: The architecture of the XMS distributed file system. *IEEE Software*, pages 21–29, May 1985.

[25] G. Gibson. Secure distributed and parallel file systems based on network-attached autonomous disk drives. White paper, Sept. 1995.

[26] R. Gopalakrishnan and A. D. Bovopolous. A protocol processing architecture for networked multimedia computers. *ACM Operating Systems Review*, 27(3):19–33, July 1993.

[27] J. H. Hartman and J. K. Ousterhout. The zebra striped network file system. *ACM Trans. Comput. Syst.*, 13(3):274–310, Aug. 1995.

[28] A. Hawes. Serial storage architecture: A low-cost, high-speed serial connection for disk subsystems. Technical report, IBM, June 1994.

[29] M. Hayter and D. McAuley. The desk area network. *ACM Operating Systems Review*, 25(4):14–21, Oct. 1991.

[30] R. G. Herrtwich. Summary of the second international workshop on network adn operating system support for digital audio and video. *ACM Operating Systems Review*, 26(2):32–59, Apr. 1992.

[31] D. Hitz. An NFS file server appliance. Technical Report 3001 Rev. B, Network Appliance, Dec. 1994.

[32] H. H. Houh, J. F. Adam, M. Ismert, C. J. Lindblad, and D. L. Tennenhouse. The vunet desk area network: Architecture, implementation and experience. *J. Selected Areas in Communications*, 13:710–721, May 1995.

[33] R. Hyer, R. Ruef, and R. W. Watson. High-performance data transfers using network-attached peripherals at the national storage laboratory. In Coleman [15], pages 275–284.

[34] IEEE. *Proc. Fourteenth IEEE Symposium on Mass Storage Systems*, Sept. 1995.

[35] IEEE P1244. *Reference Model for Open Storage Systems Interconnection – Mass Storage System Reference Model Version 5*, Sept. 1994.

[36] R. Jain, K. Somalwar, J. Werth, and J. Browne. Scheduling parallel i/o operations in multiple bus systems. *J. Parallel and Distributed Computing*, 16:352–362, 1992.

[37] R. H. Katz. High-performance network and channel based storage. *Proc. IEEE*, 90(8):1238–1261, Aug. 1992.

[38] B. Kobler and P. Hariharan, editors. *Third NASA Goddard Conference on Mass Storage Systems and Technologies*, Oct. 1993.

[39] N. P. Kronenberg, H. M. Levy, and W. D. Strecker. Vaxclusters: A closely-coupled distributed system. *ACM Trans. Comput. Syst.*, 4(2):130–146, May 1986.

[40] I. M. Leslie, D. McAuley, and S. J. Mullender. Pegasus – operating system support for distributed multimedia systems. *ACM Operating Systems Review*, 25(1):69–78, Jan. 1993.

[41] E. Levy and A. Silberschatz. Distributed file systems: Concepts and examples. *ACM Comput. Surv.*, 22(4):322–374, Dec. 1990.

[42] A. Liebl. Authentication in distributed systems: A bibliography. *ACM Operating Systems Review*, 27(4):31–41, Oct. 1993.

[43] C. J. Lindblad, D. J. Wetherall, W. F. Stasior, J. F. Adam, H. H. Houh, M. Ismert, D. R. Bacher, B. M. Philips, and D. L. Tennenhouse. Viewstation applications: Implications for network traffic. *J. Selected Areas in Communications*, 13:768–778, May 1995.

[44] D. D. E. Long, B. R. Montague, and L.-F. Cabrera. Swift/RAID: A distributed RAID system. *Computing Systems*, 7(3):333–359, 1994.

[45] K. C. Matthews. Implementing a shared file system on a HIPPI disk array. In IEEE [34], pages 77–88.

[46] E. L. Miller and R. H. Katz. Rama: A file system for massively-parallel computers. In Coleman [15], pages 163–168.

[47] M. N. Nelson, Y. A. Khalidi, and P. W. Madany. The spring file system. Technical Report SMLI TR-93-10, Sun Microsystems Laboratories, Inc., Feb. 1993.

[48] Parity Systems. Etherstore product info, 1995.

[49] J. Pasquale and E. Anderson. Container shipping: Operating system support for i/o-intensive applications. *IEEE Computer*, 27(3):84–93, Mar. 1994.

[50] R. Pike, K. Thompson, and H. Trickey. Plan 9 from bell labs. In *Proc. Summer 1990 UKUUG Conf.*, pages 1–9, July 1990.

[51] J. Postel. DoD standard transmission control protocol, Jan. 1980. RFC 761.

[52] J. Postel. User datagram protocol, Aug. 1980. RFC 768.

[53] M. W. Sachs, A. Leff, and D. Sevigny. LAN and I/O convergence: A survey of the issues. *IEEE Computer*, pages 24–32, Dec. 1994.

[54] K.-Y. Siu and R. Jain. A brief overview of ATM: Protocol layers, LAN emulation and traffic management. *ACM SIG Communications*, 25(2):6–20, Apr. 1995.

[55] N. P. Smith. HIPPI – and beyond. *Supercomputing Review*, pages 69–79, Nov. 1991.

[56] Solflower Computer. A storage crossbar for unix workstations, Jan. 1995. white paper.

[57] R. Steinmetz. Analyzing the multimedia operating system. *IEEE Multimedia*, 2(1):68–84, 1995.

[58] Sun Microsystems Inc. NFS: Network file system protocol specification, 1989.

[59] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, 2 edition, 1988.

[60] A. S. Tanenbaum, R. van Renesse, H. van Stavaren, G. J. Sharp, S. J. Mullender, J. Jansen, and G. van Rossum. Experiences with the amoeba distributed operating system. *Commun. ACM*, 33(12):46–63, Dec. 1990.

[61] M. Teener. A bus on a diet – the serial bus alternative. In *Proc. IEEE CompCon 1992*, Feb. 1992. An updated version can be obtained at ftp://ftp.apple.com/pub/standards/p1394.

[62] D. Tolmie. An experimental workstation farm with ATM interconnections and an hdtv frame buffer. email communication from det@lanl.gov, 1994.

[63] A. R. Trapathi and N. M. Karnik. Trends in multiprocessor and distributed operating system design. *J. Supercomputing*, 9(1/2):23–50, 1995.

[64] C. B. S. Traw and J. M. Smith. Striping within the network subsystem. *IEEE Network*, pages 22–32, July 1995.

[65] H. Truested. Fibre channel arbitrated loop direct attach SCSI profile (private loop) version 2.0. Technical report, Aug. 1995.

[66] R. Van Meter. Nxs: X on a network-attached frame buffer. Oct. 1995. in preparation.

[67] R. Y. Wang and T. E. Anderson. XFS: A wide area mass storage file system. Available at http://cs-tr.cs.berkeley.edu/TR/UCB:CSD-93-783 or http://now.berkeley.edu/, Dec. 1993.

[68] R. W. Watson and R. A. Coyne. The parallel i/o architecture of the high-performance storage system (HPSS). In IEEE [34], pages 27–44.