CHAPTER 1 UltraNet Target Parameters

1.1 Overview of the UltraNet

The UltraNet is a high speed network capable of rates up to one gigabit per second. It is a hub based network with four optical fiber links connecting each hub. Each link can carry up to 256 megabits of data, and the hub backplane is capable of one gigabit aggregate throughput. Host connections to the hub may be fiber, coax, or channel based. Bus based machines have adapter boards that connect to transceivers in the hub, while channel based machines use a personality module in the hub. One way that the UltraNet achieves its high transfer rates is by off-loading the protocol processing from the hosts to special purpose protocol engines in the UltraNet hubs. In addition, every hub has a PC connected to it by StarLAN for network management purposes.

Although there is hub resident and PC resident UltraNet software, this document treats only the host resident UltraNet software.

1.2 UltraNet Platforms

The following NAS platforms have hosts with connections to the UltraNet: Amdahl, Connection Machine (CM), Convex, Cray, and SGI. Suns will be connected to the UltraNet in the near future.

1.3 Host Connections to the UltraNet

There are a variety of ways in which a host can be connected to the UltraNet; what type of connection is chosen depends in large part on the I/O architecture of the host. The type of connection a host has to the UltraNet affects the UltraNet software for that host, as software for a particular I/O type tends to come from common source tree[1].

Table 1.1 Type of UltraNet Connection

Amdahl	Convex	Cray	CM	SGI	Sun
Striped BMC	VME	HSX	HIPPI	VME	VME

The Amdahl and the Crays are channel based machines. On the Amdahl, data is striped across multiple Block Multiplexed Channels (BMCs) and out to the UltraNet. On the Crays, the UltraNet is connected to the Cray high speed channel. VME is a standard bus architecture for bus based machines. HIPPI is a standards based high speed data channel.

1.3.1 UltraNet Addressing

The UltraNet requires two IP network or subnet numbers, because there are two distinct way of accessing it. In addition, a file which maps IP addresses (or hostnames) to UltraNet addresses is needed because the current release of the UltraNet software does not support any type of address resolution protocol.

The two ways to access the UltraNet are called the Ultra native path and the Ultra IP path. These are often referred to simply as the native and IP paths. Because the UltraNet requires special software to achieve the data rates it promises, UltraNet Technologies has emulated the BSD sockets library and provided a replacement, the Ultra sockets compatibility library. Programs which have been linked with the Ultra library can run over the UltraNet in native mode. The IP path allows access to the UltraNet by programs which have not been relinked. It provides binary compatibility with existing network applications, but does not afford the performance advantages of the native path.

1. As some releases come from UltraNetwork Technologies and others come from the system vendor, this is not a hard and fast rule.

UltraNet adapters include a protocol processor (PP) which implements the network and transport protocol layers which are usually implemented in each host's kernel network subsystem. When a connection is established over the Ultra native path, data is copied directly from user process memory to the adapter, bypassing the kernel networking subsystem. This copying is done using the direct memory access (DMA) facilities provided by the hardware when it is available (typically on bus based machines), and simulated DMA when it is not (typically on channel based machines). In contrast, when data is sent over a normal IP path, it must be copied from user space to kernel space and then passed off between the transport network and interface layers before it makes it onto the network.

1.3.2 Ultra Names and IP Addresses[2]

The Ultra native path and the Ultra IP paths form two IP networks. A host with an UltraNet interface gets two new addresses and two new names composed of the host's canonical hostname[3] and a suitable suffix. In the nameserver, a host on the UltraNet will have at least three address records associated with its canonical hostname: the host's non-Ultra address, the host's Ultra native path address and the host's IP over Ultra address. In addition, there will be address records which single out each path by associating each address with a name consisting of the canonical hostname and an appropriate suffix.

Table 1.2 Ultra IP Networks and Naming Conventions

Network	Address	Netmask	Suffix
Ultra native	192.12.102.0	255.255.255.0	-u
Ultra IP	192.5.64.0	255.255.255.0	-uip

1.3.3 UltraNet Hardware Addresses and the *Unetstations* File

The ***unetstations*** file provides a mapping between hostnames and UltraNet addresses. UltraNet addresses consist of a 16 bit group address and a 4 bit unit address.

2. See UltraNet Technologies. 1990. Assigning Network Addresses. Part Number 06-0020-001, Revision A, UltraNet Technologies.

3. The canonical hostname is the left-most component of the host's fully qualified domain name. It can be determined by executing the command ***'hostname | sed 's/\..*//'*** on the host.

Table 1.3 Format of the *Unetstations* File Entries for a Host on the UltraNet

Hostname	UltraNet Address
fully-qualified-hostname-u.	group-number/unit-number
fully-qualified-hostname-uip.	group-number/unit-number

The name must be fully qualified because there are hosts in the arc.nasa.gov domain on the UltraNet. The trailing dot (".") is needed because each name is resolved by the nameserver, and without the trailing dot, each name will generate 3 queries, e.g., hostname-u.nas.nasa.gov.nas.nasa.gov, hostname-u.nas.nasa.gov.nasa.gov, and hostname-u.nas.nasa.gov. This behavior significantly slows down the loading of the ***unetstations*** file.

Table 1.4 NAS Group Numbers (16 bits)

Hub	Group Numbers
1	1-49
2	51-99
3	101-149
4	151-199
5	201-249
6	251-299
7	301-349
8	351-399
9	401-449

Group numbers are assigned to hubs in blocks of 50, starting at one. The values 0, 50, 100 etc. are not used. The first two values in a block are assigned to the link adapters for the hub. The values 1000-1023 are reserved.

Table 1.5 NAS Unit Numbers (4 bits)

Range	Use
0-31	Reserved for UltraNet Technologies usage
32	For hosts connected to the UltraNet
33-62	Unused
63	For link adapters

Using the UltraNet Paths

The NAS UltraNet is an isolated network, and every host on the UltraNet has at least one other network connection. The implication of this is that every host on the UltraNet has at least three names, three IP addresses and three logical paths, and at least two physical paths. The naïve way to handle multiple IP addresses for a given host is to use a special name to connect to it over each path. The disadvantage is that special names, **host-u** and **host-uip**, must be used to connect to a host over its Ultra native and Ultra IP paths.

The NAS configuration does not require the use of special names. As noted above, address records are supplied by the nameserver associating every interface of a host with its canonical hostname. The order in which a host's address records are returned from the nameserver is controlled globally by a parameter in the nameserver's boot file. Because the UltraNet is unreachable from many hosts, NAS nameservers are configured to return UltraNet addresses after a host's other addresses. A means of sorting the UltraNet native address of a destination host ahead of its other addresses is supplied for programs which have been linked to run over the Ultra native path and which are installed on a host with an UltraNet connection. A NAS addition to the Ultra sockets compatibility library, ***sortaddrs,*** sorts the address returned by the nameserver. With the presence of the ***sortaddrs*** code and a configuration file, ***/etc/netpref,*** which specifies the network sorting order, the Ultra paths can be sorted ahead of other paths to a destination. For programs linked with the Ultra compatibility library, running on a host with an Ultra connection, the Ultra native path is always the preferred path. For programs such as ***nfsmount,*** which can benefit from the additional bandwidth of the UltraNet but cannot use the Ultra native path, running on a host with UltraNet, the Ultra IP path is the preferred path. See *Sortaddrs Addition to Gethostbyname* for more details.

1.4 UltraNet Host Software

1.4.1 Current UltraNet Software Releases

The UltraNet requires special host software including kernel device drivers, startup scripts, maintenance commands, user applications and programming libraries. This software is provided by UltraNetwork Technologies for some systems and by the system vendor for others. UltraNetwork Technologies uses the following scheme for assigning release numbers:

Figure 1.1 UltraNetwork Technologies Release Numbers

```
Category: H=host, A=adapter, D=diagnostics, N=netmanager
|  Version (Major.Minor) (feature set, including target platform)
|  |        Revision (Alpha/Beta/Pre-release/FirstGA/BugFix, etc.)
|  |        |   Spin; Only appears in internal ID Strings (New)
|  |        |   |    Fix Release (patch) level (optional, fix releases only)
|  |        |   |    |     Tag (Optional); designates Alpha, Beta, Pre-Release
|  |        |   |    |     |
C M.mm .rr .ss [Fnn] [T]
```

This document only discusses host software. The following table gives the UltraNet Technologies host release number that the host software on the given platform is running.

Table 1.6 UltraNet Software Releases Installed at NAS

Amdahl	Convex	Cray	CM	SGI	Sun
H4.00.04	H2.5	H3.01.03	H3.50	H3.51F01	H3.50.04

The Sun and SGI releases are provided by UltraNetwork Technologies in San Jose. Binary releases for the SGI are built by the NAS workstation development team. Sun releases are currently binary only and provided by UltraNetwork Technologies. Cray releases are provided by UltraNetwork Technologies at Cray Research. The Cray UltraNet source is part of the UNICOS source tree and is built and installed by Cray analysts at NAS. The Amdahl release is a joint effort between Amdahl and UltraNetwork Technologies. Currently, Amdahl builds and installs the binaries from source at NAS. The NAS ISPC development team may assume that role in the future. Thinking Machines provides a binary release for the CM. Convex provides a binary release which is integrated into the Convex operating system.

The next major release for most systems is release 4. The Amdahl is already at release 4. Release 4 is a major release and includes a mode which is incompatible with release 3 operation. This mode consists of running the TCP/IP protocol stack in the adapters instead of the OSI protocol stack. Although the Amdahl release is a port of Ultra release 4, currently only the mode which is compatible with release 3 is supported.

SGI, Sun and Amdahl provide a file which contains the current version of the Ultra software. It is called ***ultravers***; it can be found in ***/etc*** on the Amdahl and in ***/usr/etc*** on Suns and SGIs. This file must be provided on all platforms in the target.

NAS workstation development has converted the Ultra software release into the standard SGI release format. These changes must be submitted to Ultra for incorporation into their future SGI releases.

1.4.2 Ultra Startup and Shutdown Scripts

The bootstrap process is the most idiosyncratic part of every Unix platform. As a result, there are significant differences between the startup and shutdown scripts on the various NAS platforms. Two conflicting goals for NAS UltraNet installations are: first, to have the UltraNet software well integrated into the vendor's operating system, and second, to have uniformity across vendor platforms. For the sake of integration, uniformity must be sacrificed in the case of startup scripts. In Table 1.7, target values which differ from the existing are listed in <u>***underlined bold italics***</u>.

Table 1.7 Scripts to Startup and Shutdown the UltraNet

	Amdahl	Convex	Cray	CM	SGI	Sun
Startup	rc.ultra	rc.ultra	unetup	unet_startup	ultra	rc.ultra
Shutdown	rc.ultra	none	unetdown	unet_shutdown	ultra	rc.ultra
Directory	/etc	/etc	/usr/lbin/ultra	/usr/local/etc	/etc/config	/etc
Target	/etc	/etc	<u>***/etc***</u>	<u>***/etc***</u>	/etc/config	/etc

1.4.2.1 Startup Script Parameters

Every vendor has a slightly different set of parameters that need to be set correctly in their startup scripts. Some of these are specific to the platform, others are similar across all systems. The following tables contain the correct settings for NAS systems. Target values which differ from the existing values are shown in <u>***underlined bold italics.***</u> On many systems, the parameters for the system startup are contained in a separate configuration file which can be found in the same directory as the startup and shutdown scripts. Commands and pathnames are always shown in ***bold italics*** in the descriptions.

Table 1.8 *Ultra* Parameters for SGIs (Contained in *Ultra_conf)*

Variable	Value	Description
ULTRADIR	${ULTRADIR=/usr/etc}	Directory where Ultra tools and administrative commands are kept.
ULTRA_STATIONS	/etc/unetstations	Physical <-> IP address mapping.
HOSTNAME	NOT SET	Name of host. This gets set by the ***hostname*** command when ***ultra*** is executed.
ULTRANET_EXTENSION	-u	Ultra native suffix
ULTRANET_IP_EXTENSION	${ULTRANET_EXTENSION}ip	Ultra IP suffix
ULTRANET_NETMASK	0xffffff00	Netmask used for ***unetcf radd*** command. Every logical IP network which flows over the UltraNet physical network must be given a route with the ***unetcf radd*** command.
ULTRANET_IF_NETMASK	0xffffff00	Netmask used for ***ifconfig*** command.
ULTRANET_ADAPTER_CHARACTER_DEVICES	uvm0	UltraNet Character Device
ULTRANET_ADAPTER_BLOCK_DEVICES	buvm0	UltraNet Block Devices
ULTRANET_FAMILIES	NOT SET	UltraNet Families. NAS does not use adapter families.
DUMPDIR	NOT SET	Adapter dump directory.
ADAPTER_UCODE	${ULTRADIR}/uvm_ucode	Directory containing adapter code.
ULTRANET_DEBUG_LEVEL	'error uerror err_rb'	Level of error reporting. Set for a normally functioning system. Change if needed when troubleshooting.
ULTRANET_KERNEL_TRACING	off	Kernel tracing. Set for a normally functioning system. Change if needed for troubleshooting.

Table 1.9 *Rc.ultra* Parameters for Suns (Contained in *Rc.ultra_conf)*

Variable	Value	Description
RC_ULTRA_VERSION	400	Version number
ULTRADIR	NOT SET	Directory where Ultra tools and administrative commands are kept.
ULTRA_STATIONS	/etc/unetstations	Physical <-> IP address mapping.
HOSTNAME	NOT SET	ULTRANET_BROADCAST="ones"
ULTRANET_EXTENSION	-u	Ultra native suffix
ULTRANET_IP_EXTENSION	${ULTRANET_EXTENSION}ip	Ultra IP suffix
ULTRANET_NETMASK	0xffffff00	Netmask used for ***unetcf radd*** command. Every logical IP network which flows over the UltraNet physical network must be given a route with the ***unetcf radd*** command.
ULTRANET_IF_NETMASK	0xffffff00	Netmask used for ***ifconfig*** command.
ULTRANET_BROADCAST	***ones***	Use ones for the host part of the broadcast address.

ULTRANET_DEFAULT_ROUTER	NOT SET	Default router. Not used as NAS does not route UltraNet traffic.
ULTRANET_DEFAULT_ROUTER_METRIC	NOT SET	Metric for default router. Not used as NAS does not route UltraNet traffic.
ULTRANET_FAMILIES	NOT SET	UltraNet Families. NAS does not use adapter families.
DUMPDIR	NOT SET	Adapter dump directory.
ADAPTER_UCODE	${ULTRADIR}/uvm_ucode	Directory containing adapter code.
ULTRANET_DEBUG_LEVEL	'error uerror err_rb'	Level of error reporting. Set for a normally functioning system. Change if needed when troubleshooting.
ULTRANET_KERNEL_TRACING	off	Kernel tracing. Set for a normal functioning system. Change if needed for troubleshooting.

Table 1.10 *Unetup* parameters for Crays (Navier and Reynolds)

Variable	Navier Value	Reynolds Value	Description
IDEV	/dev/hsx/ultra_in	/dev/hsx/i00	Input device name
ODEV	/dev/hsx/ultra_out	/dev/hsx/o00	Output device name
SFX1	***u***	***u***	Ultra native path suffix[4]
SFX2)	uip	uip	Ultra IP path suffix
MODE	multiplexed	multiplexed	Use multiplexed daemons[5]
NETMASK	0xffffff	0xffffff	Netmask[6]

4. Currently set to ***ul***.
5. The alternative is to use daemons which only work on the Ultra native path. There is no reason to use the non-multiplexed daemons in a production environment.
6. The Crays use the same netmask for the ***unetcf radd*** and ***ifconfig*** commands

Table 1.11 *Rc.Ultra* Parameters for the Convex

Variable	Value	Description
UHOST	'hostname'-u	Native UltraNet host name
UIPHOST	'hostname'-uip	IP UltraNet host name
IFCONFIG	/etc/ifconfig	Interface configuration.
UNETCF	/usr/etc/unetcf	Pathname of ***unetcf*** command
UNETD	/usr/etc/unetd	Pathname of ***unetd*** command
UNETDCONF	/usr/etc/unetd.conf	Pathname of ***unetd.conf*** file
UVMLOAD	/usr/etc/uvmload	Pathname of ***uvmload*** command
UVMRESET	/usr/etc/uvmreset	Pathname of ***uvmreset*** command
NETMASK	0xffffff00	Netmask
UV	/dev/uv0	Block device name for host adapter device driver
RUV	/dev/ruv0	Raw device name for host adapter device driver
UCODE	/usr/lib/firmware/uv/uvm32.ult	Host adapter microcode

Figure 1.2 Contents of *Unet-Startup* File on the Connection Machine

```
#!/bin/sh
set -x
PATH=${PATH}:/usr/local/etc:/usr/local/etc/diag:/usr/local/etc/ultra
unetcf adel
unetcf rdel
unetcf mdel
hippi_coldboot
connect -u 0x14; accept -u
read_ctl -p; write_ctl -p
unetcf madd -f /usr/local/etc/ultra/unetstations
unetcf aadd unet-hppi 192.12.102.108 /dev/bhioc -h4
unetcf radd ultra unet-hppi 0xffffff00 192.12.102.108
unetwd
unetdb error uerror err_rb
[ -f /usr/local/etc/socket-server ] & {
    /usr/local/etc/socket-server &
    (echo -n ' socket-server') >/dev/console
}
[ -f /usr/local/etc/cmftpd ] & {
    /usr/local/etc/cmftpd &
    (echo -n ' cmftp') >/dev/console
}
```

Figure 1.2 includes the entire contents of ***unet-startup*** because there are no parameters and the script is short. The pathname for the ***unetstations*** file will need to be changed to ***/etc/unetstations*** when the ***unetstations*** file is moved into its target location.

Table 1.12 *Rc.ultra* Parameters for the Amdahl (Contained in *Rc.ultra_conf*)

Variable	Value	Description
ULTRANET_DEBUG_LEVEL	default	UltraNet debugging. Set for normal operation.
ULTRANET_KERNEL_TRACING	off	UltraNet kernel tracing. Should be off for normal operation, but may be enabled in case of problems. Trace information is accessed using the ***unettb*** command.
HOSTNAME	prandtl	Name of NAS Amdahl. Needed because the output of the ***hostname*** command is the fully qualified domain name, ***prandtl.nas.nasa.gov.***
ULTRANET_ADAPTERS=chan usd usd0 busd '-c -d -s -h2'		This variable describes Amdahl's adapter configuration. ***Usd*** is the name of the UltraNet adapter, and ***usd0, busd0*** and ***busd1*** are the character and block device names. The flag ***"-c "***means "enable copydata mode" which enables the UltraNet to use the channels more efficiently; the ***-d*** flag enables DMA word coalescing; the ***-s*** flag indicates striped channels; and the ***-h2*** flag causes the adapter to send the host a heartbeat every two seconds.
ULTRANET_STRIPEGROUPS=		
usd0 '-d' 8k 'bubmc02c1 bubmc02d1 bubmc02e1 bubmc02f1'		
uds1 '-d' 8k 'bubmc02c0 bubmc02d0 bubmc02e0 bubmc02f0'		This is an important parameter which needs to be tuned carefully. The striped BMCs must form two groups, the input stripe group and the output stripe group. Data is striped over all the channels in a stripe group, and the goal is to get data flowing over all channels in a stripe group simultaneously. This configuration has 8k stripes flowing over 4 channels at once. 16K stripes are more efficient, but only 32K of data can be sent at a time. This parameter needs experimentation and tuning. It may be possible to send 64K of data per transfer, which could then be striped as 4 16K stripes. See *Table 1.30, UltraNet Parameters.*
ULTRANET_NFSMOUNTS_WANTED	off	Indicates whether to attempt NFS mounts over the UltraNet as part of the ***rc.ultra*** script. As ***rc.ultra*** generally runs before the script which brings up the TCP/IP networking, this must be set to off.
HOST_BROADCAST_ENABLED	yes	Enable broadcasts over the UltraNet.
ULTRANET_BROADCAST_HOSTPART	ONES	An obsolete form of the broadcast address uses zeros for the host part. The correct form is to use ones for the host part, as is done at NAS.

1.4.3 Additional Configuration Files

Silicon Graphics has a utility, ***chkconfig,*** for displaying, enabling and disabling additional system software. NAS development has converted the Ultra software release from UltraNet Technologies to use this system. Table 1.13 lists the required files and their contents.

Table 1.13 ***Chkconfig*** **Configuration Files**[7]

File	Location	Contents With Ultra	Contents Without Ultra	Description
ultra.ip	/etc/config	on	off	Enable/disable UltraNet IP path
ultra.nfs	/etc/config	on	off	Enable/disable NFS over UltraNet
ultra.unetd	/etc/config	on	on	Enable/disable ***unetd***

Amdahl has a similar system for tracking installed software packages. The file ***ultra*** must be present in the directory ***/etc/feature,*** and must contain a single line reading ***"feat_ultra=1"*** with no leading or trailing blanks.

1.4.4 Utilities for Managing the UltraNet

The following commands are used by startup scripts to initialize the UltraNet. They can also be used for troubleshooting. Only VME based hosts need to download code into their adapters, and only VME based hosts allow the administrator to troubleshoot the adapter. The commands must be available in the ***/usr/etc*** path, possibly as symbolic links. The microcode may be located wherever it is installed by default. It is desirable for the microcode to be located in the same directory on all VME based systems, but as it is loaded as part of the Ultra startup script, it is not a major concern. Target locations which differ from the existing location are listed in <u>***underlined bold italics***</u> in Table 1.14. Existing locations are listed in Table 1.15.

7. These files are used by the SGI startup scripts and installation scripts. See ***chkconfig***

Table 1.14 Administrative Commands and Microcode

File	Target Location[8]	Description
unetcf	***/usr/etc***	Configure UltraNet
unetdb	***/usr/etc***	Control UltraNet kernel printout
unetnm	***/usr/etc***	UltraNet network monitor
unettb	***/usr/etc***	Format UltraNet kernel trace buffer
unetwd	***/usr/etc***	UltraNet Watchdog Daemon
VME Adapter Commands (SGI, Sun and Convex Only)		
uvmdiag	/usr/etc	Perform adapter diagnostics
uvmdump	/usr/etc	Dump adapter memory
uvmload	/usr/etc	Load Microcode
uvmreset	/usr/etc	Reset Adapter
Microcode	**SGI and Sun Location**	**Convex Location**
uvm32.ult	/usr/etc/uvm_ucode	/usr/lib/firmware/uv
uvm39.ult	/usr/etc/uvm_ucode	/usr/lib/firmware/uv
uvm44.ult	/usr/etc/uvm_ucode	/usr/lib/firmware/uv

Table 1.15 Existing Locations of Administrative Commands

Amdahl	Convex	Cray	CM	SGI	Sun
/etc	/usr/etc	/usr/lbin/ultra	/usr/local/etc/ultra	/usr/etc	/usr/etc

1.4.5 Services Offered over the UltraNet

A subset of the services available over the usual IP networks are available over the UltraNet native path as the NAS UltraNet is intended for bulk data transfers. Each service consists of a server, started by ***unetd***, and a client. The server must have an entry in the ***unetd.conf*** file, and must not have an entry in the ***inetd.conf*** file. The standard services provided are file transfer and remote copy. The NAS program ***plot3d*** is also provided.

The Ultra superserver, ***unetd***, and the servers it spawns must be found in ***/usr/etc***. The server names must begin with the prefix ***ultra.*** The servers in ***/usr/etc*** may be symbolic links to the Ultra servers in the existing Ultra directories. For example, ***/usr/etc/ultra.ftpd*** may be

8. This directory may contain a symbolic link to the existing pathname of the command.

a link to ***/usr/lbin/ultra/ftpd*** on the Crays. The original servers must be preserved. Target locations are shown in Table 1.16 in ***underlined bold italics*** to indicate that they differ from the existing location on at least one platform. Existing locations are listed in Table 1.17.

Table 1.16 Target UltraNet Servers (Daemons) and Target Clients (User Programs)

Target Filename	Target Location	Description
unetd	***/usr/etc***	Ultra Version of ***inetd,*** the Internet services daemon.
ultra.ftpd	***/usr/etc***	Ftp daemon.
ultra.rshd	***/usr/etc***	Remote shell daemon. ***Rcp*** connects to this daemon.
ftp	***/usr/ucb***	File Transfer Protocol (FTP) client
remsh	***/usr/ucb***	Remote shell client (Cray only, for ***plot3d***) Although remote login sessions must not go over the UltraNet, the NAS command ***plot3d*** depends on the ***remsh*** client.
rcp	***/usr/ucb***	Remote copy client.

Table 1.17 Existing Locations

	Amdahl	Convex	Cray	SGI	Sun
Unetd	/etc	/usr/etc	/usr/lbin/ultra	/usr/etc	/usr/etc
Servers	/etc	/usr/etc/un.*[9]	/usr/lbin/ultra	/usr/etc	/usr/etc
Clients	/usr/amdahl/bin	/usr/ucb	/usr/lbin/ultra	/usr/ucb[10]	/usr/ucb

1.4.5.1 *Unetd* and *Inetd*

Inetd and ***unetd*** listen for incoming requests and spawn servers to handle them. ***Unetd*** is used for services offered over both the usual network paths and the Ultra native path. ***Unetd*** differs from ***inetd*** only in that it has been linked with the Ultra sockets compatibility library instead of the usual BSD sockets library. ***Unetd.conf*** is the configuration file for ***unetd.*** The target location for both ***unetd*** and ***unetd.conf*** is ***/usr/etc***. These may be symbolic links to the existing location.

9. The prefix ***un*** is used instead of ***ultra*** by default.
10. ***/usr/ucb*** is a symbolic link to ***/usr/bsd*** on the SGIs.

Figure 1.3 Target Contents of *Unetd.conf* File[11]

ftp	*stream*	*tcp*	*nowait*	*root*	*/usr/etc/ultra.ftpd*	*ultra.ftpd*[12]
shell	*stream*	*tcp*	*nowait*	*root*	*/usr/etc/ultra.rshd*	*ultra.rshd*[13]

1.4.5.2 Connection Machine Services

The CM does not run ***unetd.*** A single service, ***cmftpd,*** is supported and it runs all the time on the Connection Machine front end. ***Cmftpd*** is currently found in ***/usr/local/etc.*** The CM does not support any Ultra client programs. BSD sockets support is provided on the CM by running a special server, ***/usr/local/etc/socket-server,*** on the CM front end. A set of library routines, including calls such as ***cmfswritefilealways,*** is provided for writing programs on the front end which manipulate data on the CM and move it between the front end, the datavault, and the CM. Ultranet has been incorporated into this library so that it is now possible to use the same routines to move data between the CM and another remote machine.

1.4.5.3 Future Issues

Our current policy is to support primarily bulk data transfers over the UltraNet. Tests done to date indicate that the Ethernet performs *better* than the UltraNet for small packet, interactive traffic such as ***telnet*** or ***rlogin*** traffic on Silicon Graphics workstations.[14] This topic warrants further investigation. It is possible that we may in the future desire to offer all network services over the UltraNet. To do this would require linking all services with the Ultra sockets compatibility library. In this case, the need for two superservers, ***inetd*** and ***unetd***, would be eliminated. The forthcoming Ultra release 4 sockets compatibility library contains many fixes for bugs found in the release 3 version. Consequently, the relinking of all applications must not happen until Ultra release 4, and is not recommended at this time.

1.4.6 NFS and UltraNet

As NFS is implemented in the kernel, it cannot be linked with the Ultra libraries to enable it to run over the Ultra native path. Hence, NFS traffic on the UltraNet must use the Ultra IP path. The perfor-

11. All daemons which appear in the ***unetd.conf*** file must be commented out of the ***inetd.-conf*** file.

12. The current name and path for this is different on each architecture.

13. The current name and path for this is different on each architecture.

14. See Krystynak, John, 1991, *UltraNet Application Development: A Case Study,* RND Technical Report RND-91-016, NAS Project, NASA/Ames Research Center.

mance advantage for running NFS traffic over the UltraNet is not as great as is desired; this is a result of both the fact that NFS traffic must run over the Ultra IP path and the fact that the NFS protocol is not an efficient protocol. It is, however, useful to run NFS traffic over the UltraNet to avoid congestion on other paths. The target UltraNet NFS configuration is detailed in *Section 1.4.6.2.*

1.4.6.1 Running NFS over the UltraNet

There are three ways NFS mounts are initiated at NAS: the system administrator may create an NFS mount entry in a system's ***/etc/fstab*** file, which causes the filesystem to be mounted as part of the boot process; the system administrator may issue the mount command interactively; or a user may use the ***nfsmount*** command to mount a filesystem over NFS. In all these instances, the UltraNet is the preferred path if it is available. The normal way to specify the UltraNet path when mounting is to use the ***-uip*** name for the host as one of the parameters to the ***mount*** command instead of its canonical name. This is acceptable when a system administer is setting up an NFS mount. However, users must get the ***-uip*** path for their mount when it is available just by specifying the canonical hostname. In order to accomplish this, the local program ***nfsmount*** must be modified.

1.4.6.2 Target UltraNet NFS Configuration

The two Crays, Reynolds and Navier, use the UltraNet for all NFS cross-mounting. This will continue as the target configuration.

The four Support Processor Subsystems, Orville, Wilbur, Amelia, and Fred, will continue to use the UltraNet for cross-mounts.

The command ***nfsmount*** will modified so that users may transparently cross-mount other workstations over the UltraNet. The required modifications are described in *Section 1.4.6.3.*

Other NFS mounts may use the UltraNet or another network at the discretion of the lead system analyst.

1.4.6.3 Target Modifications to the *Nfsmount* Command

The ***nfsmount*** command must be modified so that the Ultra IP path to a host is used when it is available, even when a host is specified to ***nfsmount*** using the host's canonical name. To accomplish this, the ***nfsmount*** command must be linked with the Ultra library, ***ulsock,*** to get the NAS modified version of ***gethostbyname,*** which includes ***sor-***

taddrs. (See *1.4.7.1, Sortaddrs Addition to Gethostbyname.*) It must also include a call to the ***putenv*** library routine to set the environmental variable ***NETPREF*** to ***/etc/netpref-uip.*** This results in the Ultra IP path being sorted ahead of other paths; therefore the filesystem is mounted over the Ultra IP path.

1.4.6.4 Future Work

Major reconfigurations to the local and wide area networks have taken place in the last year. When the networks have been running without problems in their final configurations for three months, the NFS issue must be revisited. Measurements must be made by RND of NFS performance over the UltraNet and other paths. The UltraNet NFS target configuration must be revisited and modified if appropriate.

In the long term, better support for distributed filesystems over the UltraNet is needed. One possibility is for NAS to modify NFS locally to use the UltraNet more efficiently, and then work with the vendors to get the changes incorporated. Another possibility is to consider alternative distributed filesystems which offer better performance, and make operation over the Ultra native path a requirement.

1.4.7 Ultra Libraries

UltraNet Technologies provides a replacement for the BSD sockets library as well as a couple of other libraries. Programs linked with the Ultra multiplexed sockets library will run over the Ultra native path and over normal IP networks; programs linked with the Ultra native path library will only run over the Ultra native path. The native-only library is only used for program debugging and diagnostic tools. See *Tables 1.18, 1.19, 1.20, 1.21, 1.22, and 1.23.*

Table 1.18 Ultra Libraries (in */usr/lib)*

Library	Description
libulsock.a	Ultra multiplexed sockets compatibility library
libulosock.a[15]	Ultra native path only sockets compatibility library
libuutil.a[16]	Ultra utility routines -- emulation of BSD library functions

15. Does not use namservice for hostname resolution.
16. Only on SGIs.

Table 1.19 System Calls and Library Routines Emulated in *libulsock.a*

accept	bind
close	connect
dup	execve
fcntl	getpeername
getsockname	getsockopt
ioctl	listen
recv	recvfrom
select	send
sendto	setsockopt
shutdown	socket

Table 1.20 Nonfunctional System Calls and Library Routines Included in *libulsock.a*

sendmsg	socketpair

Table 1.21 NAS Additions to *libulsock.a*

Amdahl	Convex	Cray	CM	SGI	Sun
ghostnmadr.o[17]	gethostnamadr.o	gethcompat.o gethinfo.o ghostnamed.o	NA[18]	gethostwrap.o	gethostent.o
sortaddrs.o	sortaddrs.o	sortaddrs.o	NA	sortaddrs.o	sortaddrs.o

Table 1.22 UltraNet Internal Routines (Called from Routines in *libulsock*.a and *libulosock.a*)

putaddr	getaddr
SockClose	RedoSocket
ListenOnBoth	GetMuxSocket
SockSetDupSock	PrintSockInfo

17. There are two versions of this file: one which uses hosttable and one which uses the nameserver. This object file must be built from a modified version of the file ***/usr/src/win/usr/lib/net/named/ghostnmadr.c***

18. There are no clients for the CM.

Table 1.23 UltraNet Utility Routines to Emulate BSD Library Routines in *libuutil.a*

strcasecmp	herror
rcmd	login
logout	logwtmp
getenv	setenv
strtok	strpbrk
strtol	glob
clocktime	valloc

1.4.7.1 *Sortaddrs* Addition to *Gethostbyname*

Sortaddrs is a routine called by ***gethostbyname.*** When ***gethostbyname*** returns more than one address for a name, the order of those addresses is important and is dependent on runtime conditions. ***Sortaddrs***, with the help of a configuration file, sorts the addresses into an order appropriate for the application. It is used at NAS to provide access to the Ultra native path or Ultra IP path without using the ***-u*** or ***-uip*** name of a host.

1.4.7.2 *Sortaddrs* Configuration Files

Sortaddrs has the name of its configuration file hard-coded to ***/etc/netpref***. However, there is an environmental variable, ***NETPREF***, which can be set to change the name and path of the configuration file. Every NAS system must have an ***/etc/netpref*** file.[19] See *Figure 1.4*. NAS systems on the UltraNet must also have an ***/etc/netpref-uip*** file. The ***netpref-uip*** file is used by the ***nfsmount*** program to allow user to transparently NFS mount workstations over the UltraNet. See *Figures 1.5 and 1.6 and Section 1.4.6, NFS and UltraNet*.

Figure 1.4 Contents of *letc/netpref* for a Host on the UltraNet

```
# Network Preferences File
# for subnet 129.99.XX.0 and 192.102.12.0
# Network Netmask Preference
192.12.102.0      255.255.255.0     1    # Ultranet is best,
129.99.XX.0       255.255.255.0     2    # next, my own subnet
129.99.144.0      255.255.255.0     3    # next, the FDDI
```

19. Currently, NAS hosts do not have ***netpref*** files

Figure 1.5 Contents of *letc/netpref-uip* for a Host on the UltraNet

```
# Network Preferences File
# for subnet 129.99.XX.0 and 192.102.12.0
# Network Netmask Preference
192.5.XX.0        255.255.255.0     1    # Ultra IP path is best,
129.99.XX.0       255.255.255.0     2    # next, my own subnet
129.99.144.0      255.255.255.0     3    # next, the FDDI
```

Figure 1.6 Contents of *letc/netpref* for a Host with One Ethernet Interface

```
# Network Preferences File
# for subnet XX hosts not on the ultranet
# Network Netmask Preference
129.99.XX.0       255.255.255.0     1    # prefer my own subnet
129.99.144.0      255.255.255.0     2    # next, the FDDI
192.12.102.0      255.255.255.0     0    # unreachable network
192.5.64.0        255.255.255.0     0    # Ultra IP -- unreachable
```

1.4.8 UltraNet Kernel Software

The UltraNet kernel software consists of four distinct parts.

The ***ifun*** driver allows access to the UltraNet from the BSD sockets subsystem of the kernel. See *Table 1.24.*

The ***unet*** driver handles the communication between the host and the protocol processor (PP) in the Ultra Adapter by means of a proprietary Request Block (RB) protocol. It also contains the kernel entry points from the Ultra sockets library and UltraNet memory management routines. See *Table 1.25.*

The ***usd*** driver sits between the ***unet*** driver and the I/O driver. Its function is to stripe data across multiple channels. The Amdahl is the only NAS platform which uses the ***usd*** driver. See *Table 1.27.*

The I/O driver handles the movement of data between the host and the adapter. This includes the copying of data directly from user processes' memory to the adapter where practical. Direct memory access, DMA, is used on VME systems. On channel based systems streaming the data from user space to the channel may not make sense due to the setup time required by the channel. So, channel based machines may operate in **copydata mode** instead. Deciding what is appropriate for a particular machine type and channel type is an important tuning issue[20]. This decision is typically made as part of porting UltraNet to a new platform. The I/O driver is machine specific and depends on the type of connection to the

UltraNet: VME, HSX, HIPPI and striped BMC are used at NAS. The I/O driver is part of the Ultra software release for VME based systems, but is part of the generic operating system otherwise. See *Table 1.26 and Table 1.28.*

Table 1.24 ***Ifun*** **and** ***MACUNET*** **Files**

File	Description
ifun.o	UltraNet to sockets interface
macunet.o	UltraNet to STREAMS interface (Amdahl only)

This module makes the UltraNet a recognized interface to the BSD networking software. When the Ultra IP path is used, this module is used. Note that the Amdahl implementation is STREAMS based, and so is very different from the implementations on other systems.

1.4.8.1 Future Work Needed for the BSD Network Interface Layer

On systems whose network implementation is BSD derived, small buffers called ***mbufs*** are used to communicate between different protocol layers in the networking software. There are two types of ***mbufs:*** 128 byte ***mbufs*** and ***mbufs*** containing a pointer to a larger data buffer, typically 1024 bytes. The latter type of ***mbufs*** are important for good performance, but are implemented differently by every vendor. Hence the mainline code only uses the 128 byte ***mbufs***, unless the software has been ported to use the local variety of larger ***mbufs***. On the SGIs, the use of the large ***mbufs*** is toggled by defining the preprocessor variable ***CLUSTERED_MBUFS_AVAILABLE*** in ***ifun.c***. This variable is not defined for the SGIs; hence, they only use the smaller, 128 byte ***mbufs*** for the IP path. Fixing this could improve Network File System (NFS) performance over the Ultra IP path, as NFS cannot run over the native path. The Cray version of this module is very different from the SGI version, and does not appear to have this problem. As the SGI code is a port of the Sun code, it appears from examining the SGI code that the Suns will use the larger mbufs. The Amdahl uses ATT ***STREAMS*** instead of BSD ***mbufs***, the CM does not implement the IP path, and the Convex requires further investigation.

20. See UltraNet 1991, *Host Software Porting Guide. Part Number 06-0045-001*, UltraNet Technologies, Appendix A.

Table 1.25 *Unet* Files

File	Description
unet.o	Unet driver -- interface between the Ultra sockets compatibility library and the I/O driver
unetbf.o	UltraNet *buf* header pool manager
unetdu.o	UltraNet driver debug dump routines
unetkv.o	Configuration sensitive kernel variables[21]
unetrb.o	Manages the Request Block Pool
unetrt.o	Manages Ultra routing, adapter, and family tables.
unetxm.o	UltraNet kernel memory allocator

Unet maintains four tables: Transport State Table (TST), Adapter Table (ADTE), the Routing Table (RTTE) and the Address Mapping Table (AMAP).

The Transport State Table contains the per connection (per socket) information.

The Adapter Table contains statistics and variables associated with each UltraNet adapter. A host may have more than one adapter on the UltraNet.

The Routing Table keeps routing information for each logical IP network on the UltraNet. At NAS we use two logical networks.

The Address Mapping Table keeps track of the mapping between hosts and their Ultra group/unit number addresses.

The sizing of these tables must be done locally, based on local configurations. This module is very important for tuning, as it is where the Ultra protocol is really implemented; hence many of the parameters in the file ***unetparm.h***, the file into which Ultra gathers tunable parameters, are related to this module.

Table 1.26 I/O Driver Files

File	Description
uvm.o	UltraNet VME adapter driver[22]
uvmad.o	UltraNet VME loopback adapter simulator
ubmc.o	UltraNet BMC channel driver

21. The variables in this file depend on values set in the file ***unetparm.h.*** Parameters must be changed in ***unetparm.h,*** not in this file.

Unet calls these routines to do the actual data transfers. On VME machines this most often consists of using the hardware direct memory access, DMA, to transfer data between the user process memory space and the adapter.

SGIs lack a ***minphys*** function in their implementation of ***physio.*** ***Physio*** is the way the kernel implements DMA. The function of ***minphys*** is to break user data into managable chunks if the user asks for DMA for too big a chunk at once. This results in reads or writes of over 4 megabytes to the Ultra native path in stream mode failing, even though where they will not for other paths. The reason that only the Ultra native path fails is that the sockets interface to the other paths will break the data into managable chunks for transmitting, while the DMA path that the Ultra native mode uses cannot do so.

Table 1.27 Stripe I/O Driver Files (Amdahl only)

File	Description
usd.o	Converts single write requests from the ***unet*** driver into multiple write requests which are multiplexed over multiple data paths to an UltraNet hub. It can be configured to stripe over at most eight channels. NAS stripes over eight channels.

Table 1.28 Channel Driver Files

File	Description
ulbad.o	UltraNet Channel loopback adapter simulator

The Cray uses the HSX driver, which is part of the Cray system software, for its I/O driver.

1.4.9 UltraNet Tunable Parameters

On most platforms, the file ***unetparm.h***, in the directory ***/usr/include/ultra*** contains Ultra tunable parameters. The Amdahl uses the file ***unetmd1.h*** for some parameters and ***unetparm.h*** for the rest. On the Convex, the tunable parameters are only provided with the source distribution, hence ***unetparm.h*** is found in the directory

22. The Ultra VME driver, ***uvm,*** is heavily based on the skeleton device driver discussed at great length in Sun Microsystems, 1990, *Writing Device Drivers,* Part Number: 800-3851-10, Revision A of 27, Sun Microsystems. It appears as though Ultra wrote the Sun device driver and has subsequently ported it to other VME based machines.

/usr/src/ultra/sys/netif. Parameters exist for several purposes. The only parameter which is for application programming is ***UNET_-GOOD_BUF_SIZE***. Other parameters are for tuning table sizes, memory pools, packet size and other purposes.

Table 1.29 lists the parameters found in ***unetparm.h,*** in the order in which they appear in the file. The page number indicates their location in *Table 1.30*. *Table 1.30* describes each variable, its dependencies on other variables, and gives the existing, target and range of values for each platform. The Amdahl UltraNet release is not available at NAS, so the parameters and values for the Amdahl come from Amdahl documentation rather than inspection of the appropriate files.[23] In many cases, experimentation is needed to determine the optimal value and range for the parameter., Target values which differ from the existing values are shown in ***<u>underlined bold italics</u>*****.**

Table 1.29 Overview of UltraNet Parameters

Parameter	Page
NUNET	Page 24
UNET_MAX_TPDU	Page 26
UNET_DEF_TPDU_BIT	Page 27
UNET_DGRAM_EXTRA	Page 27
UNET_QOS	Page 28
UNET_OUT_TP_TGT	Page 28
UNET_OUT_TP_ACP	Page 28
UNET_IN_TP_TGT	Page 28
UNET_IN_TP_ACP	Page 28
UNET_MAX_RXMITS	Page 28
UNET_MAX_ADAP	Page 29
UNET_MAX_CRS	Page 28
UNET_MAX_ACTIVE	Page 29
UNET_MAX_AFTE	Page 29
UNET_MAX_RTTE	Page 29
UNET_MAP_AMAP	Page 30
UNET_NBUFS	Page 30
UNET_COPYDATA_NBUF	Page 31
UNET_COPYDATA_BSIZ	Page 32
ULBAD_NUM	Page 32

23. See Amdahl, 1991, *UTS UltraNet Network Administrator Guide, Release 2.1.3,* Publication Number ML-143252-001, Amdahl Corporation.

Table 1.30 UltraNet Parameters

Parameter	Discussion
NUNET	Number of simultaneous Ultra connections. This is used to size the Transport State Table (TST), the initial *buf* header pool, and the two loopback tables. **Range: 32-128 or more. The default value of 32 must not shrink, and must be increased on the larger machines. The number of simultaneous connections as reported by the command *netstat -f inet* gives a rough estimate, assuming UltraNet usage is similar to other network usage. Note that the number of */dev/ultra/unetX* devices must equal *NUNET*, so if NUNET is increased, more devices must be made as well. *NUNET* must always exceed the total number of UltraNet connections by a bit**.

Platform	Existing	Target
Amdahl	Unknown	***128***
Cray	128	128
Convex[24]	64	128
CM	32	32
SGI SPS	32	***128***
SGI Workstation	32	32
Sun Fileserver	32	***128***
Sun Workstation	32	32

Parameter	Discussion
UNET_MAX_TPDU	Maximum packet size. **TPDU** is an OSI term which has a similar meaning to the IP term **MTU**. The impact of varying this value is of interest. Must be specified as a power of 2. A 64K TPDU on the Amdahl allows for 8 16K stripes, which improves the performance of the BMC channel striping. It is desirable to have all systems using the same TPDU. **Range: 32678 or 65536, possibly higher** **Existing: 65536 on Crays, 32678 elsewhere** **Target for All Platforms: 65536**

24. Found in the file */usr/sys/ultra/sysgen/unet.h.*

UNET_DEF_TPDU_BIT	If ***UNET_MAX_TPDU*** is 2^N, then this value must be N+1. **Range: if TPDU == 32678, TPDU_BIT is 16.** **If TPDU == 65536, TPDU_BIT is 17, and similarly for higher TPDU values.** **Target for All Platforms: 17**
UNET_DGRAM_EXTRA	Extra bytes above ***UNET_MAX_TPDU***. The reason for needing extra bytes, and the right number of bytes must be determined. It is likely to depend on the underlying hardware or device driver doing the data transfers. This does not appear to be a crucial performance parameter. **Range: 256 or 4096** **Target and Existing for Cray 2: 4096** **Target and Existing for All Other Platforms: 256**

The following parameters are included for completeness, but are currently used only as placeholders. As the UltraNet is treated as a device, the ***ioctl*** system call is used for interaction between the Ultra sockets library and the UltraNet. The header file ***unetioct.h*** defines ***uioc_req,*** a union of structures, which is used to pass and receive commands to and from the ***unet*** layer of the UltraNet device driver. The following six parameters are defaults which are passed as part of the Ultra ***listen*** and ***connect*** system call replacements. They are used to fill in values in a transport layer request block. (Request blocks are used for communication between ***unet*** and the UltraNet adapter.) The definition of the structure which holds a transport layer request block, ***struct t_conn_rb,*** can be found in the header file ***unetrb.h.*** It contains a parameters structure, ***par,*** which has entries corrosponding to the following six parameters, among others.

UNET_QOS

Quality of service. See comment above.

Range, Existing and Target: 0

UNET_OUT_TP_TGT

Target output in megabytes per second. See comment above.

Range, Existing and Target: 255 megabytes per second. The UltraNet is only capable of 125 megabytes per second, so this is effectively infinity.

UNET_OUT_TP_ACP

Acceptable output in megabytes per second. See comment above.

Range, Existing and Target:
See UNET_OUT_TP_TGT.

UNET_IN_TP_TGT

Target input in megabytes per second. See comment above.

Range, Existing and Target:
See UNET_OUT_TP_TGT.

UNET_IN_TP_ACP

Acceptable input in megabytes per second. See comment above.

Range, Existing and Target:
See UNET_OUT_TP_TGT.

UNET_MAX_CRS

Connection retries before retransmit. See comment above.

Range, Existing and Target for all Platforms:0

UNET_MAX_RXMITS

Maximum number of retransmissions. See comment above.

Range, Existing and Target for all Platforms:0

UNET_MAX_ADAP

Maximum number of defined adapters. This is used to size the table of adapters and the table of per adapter statistics. On all platforms except the Convex, memory pool requirements are sized by ***UNET_MAX_ACTIVE***. The Convex does define ***UNET_MAX_ACTIVE***; it uses ***UNET_MAX_ADAP*** instead.

Range: Maximum number of adapters each hardware platform can support.The size of an entry in the adapter table is 108 bytes, and the size of an entry in the per adapter statistics table is 300 bytes per adapter.

Existing and Target for All Platforms except Convex: 8

Convex Existing: 4

Convex Target: *1*

UNET_MAX_ACTIVE

The number of adapters the system actually has configured. The Convex uses ***UNET_MAX_ADAP*** instead.

Platform	Existing	Target
Amdahl	8	8
Convex	N/A	N/A
All Others	2	***1***

UNET_MAX_AFTE

An adapter family is one way of grouping multiple adapters on a single host to perform as a single logical adapter with a single IP address. It allows for load sharing on a per connection basis. ***UNET_MAX_AFTE*** is used to size the adapter family table. Each family uses 144 bytes of memory. This parameter is not available on the Convex. NAS does not use adapter families.

Range[25]: 0 or 1 to *UNET_MAX_ADAP*

Existing for All Platforms: 8

Target for All Platforms: 0 or 1

UNET_MAX_RTTE

An Ultra route is defined for each IP network which runs over the UltraNet. ***UNET_MAX_RTTE*** is used to

25. It has not been determined whether zero is a valid value.

size the route entry table, and each route uses 176 bytes. NAS uses two IP networks for the UltraNet.

Range: 1 to 32 or more (more than 2 IP networks requires the use of adapter families)

Convex Existing: 10

Existing for All Other Platforms: 32

Target for All Platforms: 2

UNET_MAX_AMAP

Maximum number of address mappings. This number determines the size of the address mapping table, the size of the ***unetstations*** file, and the maximum number of hosts that can have UltraNet connections. Because each host has two IP addresses for the UltraNet, each host must have two entries in the address map table, so the maximum number of hosts on the UltraNet is ***UNET_MAX_AMAP***/2. Note that if this parameter is exceeded, the ***unetstations*** file cannot be loaded until its size is reduced, which will prevent that host from talking to new hosts on the UltraNet. However, it will still be able to talk to old hosts if its ***unetstations*** file is shortened and reloaded. This parameter may become obsolete when Ultra implements the Address Resolution Protocol[26] in release 4.

Range:

```
(anticipated number of Ultra hosts)
* (number of IP networks used for
the UltraNet).
```

Must be less than one thousand.

Target for All Platforms: *400*

UNET_NBUFS

The size of the initial ***buf*** header pool. This pool is statically allocated. If it is exhausted, more headers are allocated from the ***unet*** driver memory pool. If this happens, the pool may become fragmented because once memory is allocated to ***buf*** headers, it is never returned. ***Buf*** headers are used to describe buffers used for I/O operations.

Range: > = 4 * NUNET. The idea is that 3 reads and one write may be pending. The Amdahl formula is different. Increase if machine runs out of Ultra buf

26. Plummer, D, 1982, *An Ethernet Address Resolution Protocol,* Defense Data Network Request for Comments 863, SRI Information Center.

headers, or if the value of UNET_COPYDATA_N-BUF is increased.

Convex Existing and Target: 8 * NUNET

Amdahl Formula:

```
3 * NUNET + UNET_COPYDATA_NBUF *
NADAP + UNET_COPYDATA_NBUF * NCHANSG
+ (NMOUTR1 + 2) * 2 + ... + (NMOUTRn
+2) * 2.
```

There are two major differences between the Amdahl formula and other platforms: first, as writes are striped, one buffer is needed for each stripe as well as for each adapter. Second, on the Amdahl, extra buffers are needed for each host IP path.

Existing and Target for All Other Platforms:
4 * NUNET

UNET_COPYDATA_NBUF

Number of buffers per adapter, for the **copydata mode** on channel based machines. Channel based systems work differently than backplane systems, and depending on the specific timing characteristics of each hardware platform, it is often better to copy data into kernel memory and then out the data channel. In this case, buffers are needed to hold the in-flight data. As each of these buffers requires a ***buf*** header, the value of ***UNET_NBUFS*** may need to be increased for large values of ***UNET_COPYDATA_N-BUF***. In **copydata mode**, the kernel always tries to have three reads outstanding against each adapter.

Range[27]:

```
N * (number of adapters)
```

It is clear that this parameter is linearly dependent on the number of adapters in use. However, it is not clear what the value of N must be.

Platform	Existing	Target
Amdahl	Unknown	27 = (8 + 1) * 3, for eight striped physical adapters plus one virtual adapter with 3 outstanding reads per

27. It appears that for the BMC striping on the Amdahl, each adapter which is part of the stripe group counts as one adapter, and the group counts as an additional adapter.

		adapter. This may need further refinement.
Cray	6	TBD
SGI, Sun, Convex, CM	Not Applicable	

UNET_COPYDATA_BSIZ

Size of a copydata buffer. Sized to hold a maximum datagram, including slop and a request block (in adapter format rather than host format). This parameter must be changed if ***UNET_MAX_TPDU*** or ***UNET_DGRAM_EXTRA*** changes.

Range:

```
UNET_MAX_TPDU + UNET_DGRAM_EXTRA +
sizeof(struct xrb).
```

Note that on the Crays, this formula is used in the definition of UNET_COPYDATA_BSIZ.

Platform	Existing	Target
Amdahl	33152	65920
Cray 2	69761	69761
Cray Y/MP	65920	65920
Convex, CM, Sun, SGI	N/A	N/A

ULBAD_NUM

The ***ulbad*** module simulates a channel oriented adapter as a loopback device.

Range:

1 == In-kernel simulation only

2 == User space simulation only

3 == Both

Existing and Target Value for Cray and Amdahl: 3

Other Platforms: Not applicable

ULBAD_NRT

Ulbad rendezvous table size. It is used to store received request blocks (RBs) while awaiting the other half of a pair. ***Ulbad*** needs this because it is simulating an adapter.[28]

Existing and Target for Amdahl and Cray: NUNET + 4

Other Platforms: Not applicable

UVMAD_NRT

Uvmad rendezvous table. The ***uvmad*** module simulates a VME based adapter as a loopback device.

Existing and Target for SGI, Sun and Convex:

`NUNET + 4`

Other Platforms: Not applicable

IFUNDEBUG

The ***ifun*** module makes the UltraNet accessible to the Berkeley sockets kernel path. ***IFUNDEBUG*** controls the amount of debugging reported by this module.

Range:

1 == Modest error reporting

2 == Extensive error reporting

3 == No error reporting

CM and Amdahl Existing and Target:

Not applicable

Other Platforms Existing and Target: 1

IFUN_RD_MAX

Number of datagram receives (DRCVs) to have outstanding.

Range: > = 3 The most likely reason this is set to three is to have three outstanding reads on each adapter for copydata mode. Data structures are sized using this parameter and NIFUN, which comes from the header file *lusr/include/ultra/ifun.h*; this file is generated by the config process and is the number of Ultra adapters available to the host resident TCP/IP stack.

Existing and Target for All Platforms except Amdahl: 3

Existing and Target for Amdahl: Not applicable. Amdahl uses a different host stack implementation.

28. See UltraNet, 1991, *Host Software Porting Guide,* Part Number 06-0045-001, UltraNet Technologies.

IFUN_WR_MAX

Maximum number of datagram sends (DSNDs) to have outstanding. This parameter needs further investigation.

Range: 2 or more

Platform	Existing	Target
Cray, SGI, Sun	2	TBD
Amdahl, CM	N/A	N/A
Convex	3	TBD

IFUN_MTU

The Internet Protocol maximum transmission unit (MTU) for this interface. Since the kernel protocol stack must be used for NFS traffic, the MTU must be optimized for NFS. So the MTU must be big enough to hold a disk block and some slop.

Range: 576 - 65535 in theory, 32767 in practice, because most TCP/IP implementations store the MTU in a *short*.

Platform	Existing	Target
Amdahl	Unknown	8400
Cray[29]	112 * 32 * 1024	112 *32 * 1024
CM	N/A	N/A
Sun and SGI	8400	8400
Convex[30]	16 * 1024	16 * 1024

UNET_GOOD_BUF_SIZE

Size of application read and write buffers. This is used by Ultra applications to set up their read and write buffers, so that they will then issue reads and writes of ***UNET_GOOD_BUF_SIZE***. The comments indicate that ***UNET_GOOD_BUF_SIZE*** is set for good, if not optimal transfer rates on the UltraNet. It is used by Ultra applications such as ***ftp***, and developers must be encouraged to use ***UNET_GOOD_-BUF_SIZE*** as well. If a new value for ***UNET_GOOD_BUF_SIZE*** is chosen, then all those applications will benefit from the choice when they are recompiled. The existing value may have some-

29. Set in the file ***/usr/src/uts/tcp/conf/ioconf.c.*** Cray uses a 32K disk block.
30. Convex uses a 16K disk block.

thing to do with the ***minphys*** subroutine of ***physio***, which some systems use as part of their DMA.

Range: 1 - 3.5 Megabytes. The SGIs have a bug in their driver which prevents reads and writes of greater than about 3.5 Megabytes.

Existing and Target for All Platforms[31]: 128K

UNET_TBUF_SIZE

Number of words in the trace buffer.

Range: 4096, 8192, 16384 etc. Must be a power of 2. The maximum value has not been established. The value represents the number of 4-byte words in the trace buffer, and each entry in the trace buffer is 4 words long. This buffer is accessed by the *unettb* command. If not enough trace information is being collected to diagnose a problem, build a new kernel with a higher value for this parameter.

Platform	**Existing**	**Target**
Amdahl	8192	8192
Cray	4096	4096
CM	4096	4096
Convex	TBD	TBD
SGI	4096	4096
Sun	4096	4096

UNET_MIN_READ

The smallest read to send to the adapter.

Range: Not Established

Platform	**Existing**	**Target**
Amdahl	TBD	TBD
Cray	256	256
CM[32]	256	256
Convex	256	256
SGI	256	256
Sun	256	256

31. The optimal size for Convex is really 384K, however, it is left at 128k for compatibility with the rest of the world.

32. The comments in ***unetparm.h*** on the CM indicate that this must be set to zero for the CM's HIPPI interface, but the preprocessor statements define it to be 256 anyway.

UNET_RARB_RECV_SIZE

Size of the read-ahead buffer for a connection oriented transport service (COTS). Used to size buffers for anticipatory reads on an adapter, so that data can be received before a user process exists to collect it.

Range: Not Established

Existing and Target for All Platforms: 256

UNET_RARB_DRCV_SIZE

For anticipatory datagram reads on an adapter.

Range, Existing and Target for All Platforms:

```
UNET_MAX_TPDU + UNET_DGRAM_EXTRA
```

UNET_RARB_DRCV_NUM

Determines how many reads can be outstanding on an adapter. If more than ***UNET_RARB_DRCV_NUM*** processes issue reads for an adapter, the other processes are put to sleep while waiting for that adapter.

Platform	**Existing**	**Target**
Amdahl	TBD	TBD
Cray	2	2
Convex	TBD	TBD
CM	1	1
SGI	2	2
Sun[33]	2	1

UNET_TST_DG_MAX

Maximum number of bytes Ultra will deposit in a buffer associated with a particular socket. It is used in case data arrives from the adapter and can't be given to the application right away. The comments in ***unetparm.h*** say "maximum dg_data bytes/TST". TST is the transport state table, and there is one entry per connection.

Range: 2 or more maximum sized packets, including slop.

Existing and Target for All Platforms:

```
2 * UNET_RARB_DRCV_SIZE.
```

33. ***Unetparm.h*** contains the following comment: "on Sun uCray, if max TPDU size is large we need to shrink the number used to avoid problems with "sys point too small" on booting." Small appears to mean 32K. Since the target TPDU is 64K, the target for ***UNET_RARB_DRCV_SIZE*** must be 1. It is not clear if the SGI or other platforms have similar constraints, although this appears to be Sun specific.

UNET_SOCK_MAX_KBUF

Maximum bcount for kernel buffering of socket writes. This is only used in one place, and the following comment from *unet.c* describes its use:

```
"SOCKET KLUDGE #1: If not too much is being written
(<=U_sock_max_kbuf), buffer output in kernel to allow
write operation to return immediately (unless of course
it is already asynchronous i/o...). The temporary data
buffer allocated here is freed by SOCKET KLUDGE #2 in
unetintr as a result of having the flag RB_WRITEBEHIND
set. Unetintr also frees the new RB allocated here
(RB_FREEABLE is set), which frees the dummy "master buf
hdr" gotten here as a side effect".
```

Note that U_sock_max_kbuf == UNET_SOCK_-MAX_KBUF.

Range, Target and Existing for All Platforms: 256.

UNET_SOCK_MAXPORT

Highest TCP/UDP port number.

Range, Target and Existing for All Platforms: 0xfffe

UNET_RB_HOT_MAX

Maximum number of request blocks to keep on the "hot" list. This parameter determines how many request blocks to allocate from the **unet** driver memory pool when the **unet** driver is initialized. If more request blocks are needed, they are allocated and freed from the **unet** driver pool as needed.

Range, Target and Existing for All Platforms: 20.

The following two parameters are very closely related. The file ***unetparm.h*** contains this comment regarding them:

```
The following specify the number of times to probe or sec-
onds to tick remote stations that have been declared
either DOWN or UNKNOWN in a call to unetam_bad(). DOWN
stations are those that we cannot get any response from
(RB_STAT_ER_LOC_TIMEOUT) and UNKNOWN stations are those
that we cannot establish a connection to (RB_STAT_ER_CON-
N_REJECT). (The idea behind UNKNOWN stations is that the
remote application may have posted ALSNs only in some
adapters in a group, and we just want to force uneta-
m_out() to try them all. DOWN adapters are really not
there, so we want to give them time to come back up before
retrying.)
```

UNETAM_BAD_DOWN

See comment above.

Range, Target and Existing for All Platforms: 60.

UNETAM_BAD_UNKN

See comment above.

Range, Target and Existing for All Platforms: 5.

UNET_SEND_TMOUT

Sun VME send timeout. Unused by other platforms.

Range, Target and Existing for Sun: 1.

Range, Target and Existing for All Other Platforms: 0 or N/A.

UNET_RECV_TMOUT

Sun VME send and recv timeouts. Unused by other platforms.

Range, Target and Existing for Sun: 2.

Range, Target and Existing for All Other Platforms: 0 or N/A.

The following four parameters are only found in ***unetparm.h*** on the Sun platform. They are specific to the way that the Sun does DMA.

UNET_ADJ_INTVL

Seconds between checks.

Sun Range, Existing and Target: 3.

UNET_ADJ_NTOS

Timeouts to trigger adjustments.

Sun Range, Existing and Target: 1.

UNET_ADJ_MAXB

Maximum value of ***U_mx_bsize***, for ***minphys***.

Sun Range, Existing and Target: *UNET_MAX_BSIZE*.

UNET_ADJ_MINB

Minimum Value of ***U_mx_bsize***, for ***minphys***.

Sun Existing Value: 8192.

Sun Range and Target: The comments indicate that the current value is a "random guess at a good value". So the range and target is yet to be determined. Obviously, it should be less than *UNET_MAX_BSIZE*.

UNET_MAX_BSIZE

Maximum blocksize for ***physio*** (.5 megabytes). Used as part of DMA process.

Range: Depends on the *physio* implementation. Must be the largest value the implementation will handle gracefully.

Platform	Existing	Target
Amdahl	TBD	TBD
Cray	0x80000	0x80000
Convex	TBD	TBD
CM	0x80000	0x80000
SGI	0x80000	0x80000
Sun	0x20000	0x20000

UNET_XMEM_POOL_WORDS

Kernel memory allocation pool. The ***unet*** driver needs memory for several purposes: request blocks, read buffers, the anticipatory datagram receive buffer, read and write buffers for the IP path and ***buf*** headers for the buffers. Request blocks and ***buf*** headers are preallocated; if the driver runs out, more are allocated from the pool. Allocating ***buf*** headers from this pool can cause it to become fragmented as they are never returned to the pool. The read buffers are for running in ***copydata*** mode: they hold incoming data before it is copied to user space. The anticipatory datagram receive buffer is needed because the UltraNet adapter always tries to have two anticipatory datagram receives pending. Memory is also needed for the interface with the IP layer of the host protocol stack.

Note: The Cray and SGI UltraNet memory managers are identical; it is likely that this part of the driver is remarkably similar across platforms.

Range: On bus-based machines which do not employ *copydata* mode, the *unet* driver does not need to buffer much data, so the memory requirements are relatively small. In fact, for the SGI, CM and Sun, it is considerably less than the constant Amdahl uses in its formula for "other small blocks of memory used by the *unet* driver." For channel-based machines, performance could be significantly affected by sizing this correctly; for bus--based machines it may be less critical. In all cases, if the low low water mark is ever reached in non-error conditions, the memory pool needs to be increased.

Platform	Existing	Target
Amdahl	1,171,888	1,171,888 (see below)
Cray	See below	See below
Convex	TBD	TBD
CM	5000[34]	TBD
SGI	10000[35]	***20000***
Sun	5000 (See Below)	***20000***

The Amdahl, and the Cray use formulas to size their memory pool. The Sun appears to use the Cray formula as well, although the comments imply that the formula is only used when the Sun is operating in uCray mode.

The exact Cray Formula is:

```
UNET_MAX_ACTIVE * ((UNET_COPYDATA_NBUF *
UNET_COPYDATA_BSIZ + UNET_RARB_DRCV_NUM *
UNET_RARB_DRCV_SIZE) /NBPW + 20000))
```

where

UNET_MAX_ACTIVE represents the number of active adapters.

UNET_COPYDATA_NBUF * UNET_COPYDATA_BSIZ is the total memory needed to hold data copied from user space to the **unet** driver kernel space when the system is operating in **copydata mode.**

UNET_RARB_DRCV_NUM * UNET_RARB_DRCV_SIZ is the total memory needed to hold data transferred from the adapter to the **unet** driver, but not yet delivered to any user process.

NBPW is the number of bytes per word. It is a conversion factor.

20000 is the additional memory needed per adapter for other overhead.

The formula can be reinterpreted as shown in Figure 1.7:

34. There is some experimental striped driver code on the CM. It uses a formula for the memory pool which is similar to the Cray or the Amdahl.

35. There is a comment in ***unetparm.h*** on the SGIs indicating that this value needs to be tuned for the various models of SGI.

Figure 1.7 Cray Memory Pool Size

$$\text{unet\_memory\_pool} = (\#\text{ adapters}) \times \left(\frac{(\text{copydata memory}) + (\text{received memory})}{\frac{\text{bytes}}{\text{word}}} + 20000 \right)$$

The exact Amdahl formula is:

```
NADAP *(STDREAD_SIZE + DRCV_SIZE) + NCHANSG *
STDREAD_SIZE + (NMUOUTR1 +2) * MTUSIZE1 + (NMUOUTR2 +2) *
MTUSIZE2 + ... + (NMOUTRn +2)MTUSIZEn + NUCHANS * UBMREQS
* (UNET_COPYDATA_NBUF + 3) + UNET_TBUF_SIZE * 4 + 20000
```

where

NADAP is the number of active adapters.

STDREAD_SIZE = UNET_COPYDATA_NBUF + UNET_COPYDATA_BSIZ, the total memory needed to hold data copied from user space to the **unet** driver kernel space when the system is operating in **copydata** mode.

DRCV_SIZE = UNET_RARB_DRCV_SIZE * UNET_RARB_DRCV_NUM, the total memory needed to hold datagrams received by the **unet** driver but not yet read. Amdahl cautions against modifying this value.

NCHANSG is the total number of channels in a stripe group.

NMUOUTR1 is the number of outstanding reads the first Ultra IP interface has outstanding. The two is a fudge factor for the buffers used for writes. Similarly, **NMUOUTR2 ... NMUOUTRn** are the numbers of outstanding reads for each additional Ultra IP interface. Each adapter can have a separate IP interface associated with it, but a striped adapter only has a single IP interface associated with it.

MTUSIZE1 is the MTU for the first Ultra IP interface. Similarly for the additional Ultra IP interfaces.

UNET_TBUF_SIZE is the size of the trace buffer.

NUCHANS is the maximum number of simultaneously active UltraNet channels.

UBMREQS is the size of the ***ubm_request*** structure, which is about 300 bytes for a **TPDU** of 32K. The size of ***ubm_request*** blocks must be increased if the **TPDU** is increased. ***Ubm_request*** blocks hold all information related to an UltraNet channel I/O request.

The Amdahl formula can be reinterpreted as shown in Figure 1.8:

Figure 1.8 Amdahl Memory Pool Size

$$\text{unet memory pool} =$$

$$(\#\ \text{adapters}) \times (\text{copydata memory} + \text{received memory}) +$$

$$(\#\ \text{stripes}) \times (\text{copydata memory}) +$$

$$(\#\ \text{stripes}) \times (\text{number of copydata reads and writes}) \times \left(\frac{\frac{\text{striping overhead}}{\text{stripe}}}{\text{copydata read or write}} \right) +$$

$$\sum_{i=1}^{n} (\text{outstanding reads for Ultra IP interface i}) + (\text{trace buffer}) + 20000$$

UNET_XMEM_POOL_HI_LO

Turn new activity back on when avail > hi_lo.

Range: 20% of the total pool. It is likely that the memory management algorithm which is used operates very poorly below a certain threshold. It is most likely that the choice for this value and for UNET_XMEM_POOL_LO_LO have been chosen to keep the management algorithm operating within the range it was designed for. These values must not be modified without careful examination of the algorithm.[36]

Existing and Target for All Systems:

`UNET_XMEM_POOL_WORDS/5.`

UNET_XMEM_POOL_LO_LO

Turn off all new activity when avail < lo_lo.

Range: 10% of the total pool. See comment under UNET_XMEM_POOL_HI_LO above.

Existing and Target for All Systems:

`UNET_XMEM_POOL_WORDS/10.`

UNET_RD_NBUF

Convex specific parameter which appears in Number of reads per adapter. Used by Convex to allow

36. See Knuth, D. E. 1973, *The Art of Computer Programming, Volume One,* Addison Wesley Series in Computer Science and Information Processing. Addison Wesley, algorithm A from page 437, as modified on pages 438, 442 and the solution of exercise 6 on page 597.

multiple request blocks to be read back from the VIOP with a corresponding increase in performance.

Range: Since this is not a tunable parameter on the Convex, appropriate values for the Convex have not been established. Must be set to one for all other platforms.

Platform	Existing	Target
Amdahl	N/A	N/A
Cray	N/A	N/A
Convex	TBD	TBD
CM	1	1
SGI	1	1
Sun	1	1

1.4.10 Ultra Devices

Unlike networks which are accessed through the kernel network subsystem, UltraNet uses a different device for each connection. Ultra devices used for connections are called ***/dev/ultra/unet0, /dev/ultra/unet1, ..., /dev/ultra/unetN***, where **N = NUNET,** the number of simultaneous UltraNet connections defined in ***unetparm.h***. When a new connection is requested, ***unet0*** is opened, a free device is found and opened, and ***unet0*** is closed. Hence ***unet0*** plays a multiplexing role. On VME based machines, there are also devices to talk directly to the adapter to download microcode, gather adapter statistics, and other functions. These devices are called ***uvm0*** and ***uvm1*** for the character devices and ***buvm*** and ***buvm1*** for the block devices. The parameter UNET_MAX_ACTIVE in ***unetparm.h*** determines how many Ultra VME devices are needed on VME systems. Similarly, the Amdahl requires I/O devices ***bubmc*** and ***ubmc***.

1.4.11 UltraNet Header Files

All UltraNet header files can be found in the directory ***/usr/include/ultra.*** The file most likely to be of interest is ***unetparm.h,*** which contains the kernel tunable parameters as well as the best choice data buffer size for application programming.

1.5 References

1. Amdahl. 1991. *UTS UltraNet Network Administrator Guide, Release 2.1.3.* Publication Number ML-143252-001, Amdahl Corporation.

2. Knuth, D. E. 1973. *The Art of Computer Programming, Volume One.* Addison Wesley Series in Computer Science and Information Processing. Addison Wesley.

3. Krystynak, John. 1991. *UltraNet Application Development: A Case Study.* RND Technical Report RND-91-016, NAS Project, NASA/Ames Research Center.

4. Silicon Graphics, 1990. *Irix System Administrator's Reference Manual, Version 5.0.* Document Number 007-0604-060, Silicon Graphics, Inc.

5. Sun Microsystems, 1990. *Writing Device Drivers*. Part Number: 800-3851-10, Revision A of 27, Sun Microsystems.

6. Plummer, D, 1982. *An Ethernet Address Resolution Protocol.* Defense Data Network Request for Comments 863, SRI Information Center.

7. UltraNetwork Technologies. 1990. *Ultra Product Overview.* Part Number 06-0017-001, Revision A.

8. UltraNetwork Technologies. 1990. *Assigning Network Addresses.* Part Number 06-0020-001, Revision A, UltraNet Technologies.

9. UltraNetwork Technologies. 1991. *Host Software Porting Guide.* Part Number 06-0045-001, UltraNet Technologies.