**FORSCHUNGSZENTRUM JÜLICH GmbH**
**Zentralinstitut für Angewandte Mathematik**
**D-52425 Jülich, Tel. (02461) 61-6402**

Interner Bericht

# KFAnet/INTERNET Installation and Operating of the IBM-HiPPI-Interface

***R. Niederberger, F.J. Schoenebeck***

KFA-ZAM-IB-9407

Februar 1994
(Stand 15.02.94)

# Table of contents

**List of Figures** . . . . . **v**

**List of Tables** . . . . . **vii**

**1 Introduction** . . . . . **1**
1.1 Motivation . . . . . 1

**2 Network environment** . . . . . **3**
2.1 The computer network KFAnet . . . . . 3
2.2 The workstation concept . . . . . 3
2.3 UltraNet . . . . . 6
2.4 HiPPI . . . . . 7
2.5 UltraNet HiPPI network processor . . . . . 8
2.6 HiPPI for MVS at KFA . . . . . 8

**3 Hardware Installation of the HiPPI-Interface** . . . . . **11**
3.1 The IBM-HiPPI-interface . . . . . 11
3.2 The Ultra-HiPPI-board . . . . . 11

**4 Software Installation and Bugs** . . . . . **13**
4.1 The Software incompatibilities . . . . . 13
4.2 Software Bugs . . . . . 14

**5 Test environment and results** . . . . . **15**
5.1 Introduction to test series . . . . . 15
5.2 TCPSPRAY . . . . . 17
5.2.1 TCPSPRAY (Ultra+FDDI) . . . . . 18
5.2.2 TCPSPRAY( Ultra) . . . . . 19
5.3 FTP . . . . . 20
5.3.1 FTP (Ultra+FDDI) . . . . . 21
5.3.2 FTP (Ultra) . . . . . 22
5.4 TSOCK . . . . . 23
5.4.1 TSOCK (Ultra TCP-communication) . . . . . 25
5.4.2 TSOCK (Ultra TP4-communication) . . . . . 26

**6 Conclusion** . . . . . **27**

**7 Appendix** . . . . . **29**
7.1 MVS/ESA Definitions . . . . . 29
7.1.1 MVS-Ultra Definitions . . . . . 29
7.1.1.1 Ultra HiPPI Profile (UPROFILH) . . . . . 29
7.1.1.2 Ultra Stations file (STATIONH) . . . . . 30
7.1.2 TCP/IP Version 2.2 (Definitions) . . . . . 31
7.1.2.1 PROFILE TCPIP . . . . . 31

7.2 Participating Systems . . . . . 33
7.3 Man page TCPSPRAY . . . . . 35
7.4 Man page FTP . . . . . 37
7.5 Man page TSOCK . . . . . 43
7.6 Literature . . . . . 52

# List of Figures


Figure 1 KFAnet/Internet configuration . . . . . 5
Figure 2 UltraHub1000 internals at KFA Juelich . . . . . 6
Figure 3 HiPPI Architecture . . . . . 7
Figure 4 The HiPPI test scenario . . . . . 9
Figure 5 HiPPI, Ultra and TCP/IP software components . . . . . 13
Figure 6 Usage of TCPSPRAY command . . . . . 17
Figure 7 Usage of FTP command . . . . . 20
Figure 8 TSOCK processing logic . . . . . 23
Figure 9 Usage of TSOCK command . . . . . 24

# List of Tables


| | | |
|---|---|---|
| Table 1 | LPAR weighting at ES9000/620 | 15 |
| Table 2 | Abbreviations used in throughput tables | 16 |
| Table 3 | Throughput TCPSPRAY (Ultra + FDDI) | 18 |
| Table 4 | Throughput TCPSPRAY (Ultra) | 19 |
| Table 5 | Throughput FTP (Ultra + FDDI) | 21 |
| Table 6 | Throughput FTP (Ultra) | 22 |
| Table 7 | Throughput TSOCK (Ultra TCP-communication) | 25 |
| Table 8 | Throughput TSOCK (Ultra TP4-communication) | 26 |

# 1 Introduction

## 1.1 Motivation

In the last decade network research and development led to more and more powerful computer networks. Starting with local area networks with 10 Mbit/s bandwidth like Ethernet, followed by FDDI with 100 Mbit/s and not finally HiPPI with 800 esp. 1600 Mbit/sec, the transmission speeds of local networks grew in fantastic steps. Though HiPPI, the High Performance Parallel Interface, is not the last step in this evolution, this interface seems to be the fastest interface, that can be managed by the computation power of currently available mainframe cpu's, taking into account, that this mainframe is not used for communication purposes only.

In 1990 a new concept for a cooperative computing in the KFA was introduced, the so named workstation concept. This concept includes the communication of low bandwidth attached PC's with high bandwidth attached supercomputers as well as the high speed communication of CRAY supercomputers and IBM mainframes. Taking into account this requirements the KFA decided to install an IBM HiPPI interface for the IBM ES9000/620 and to analyse the opportunities available with this interface.

# 2 Network environment

## 2.1 The computer network KFAnet

KFAnet/Internet, the high speed computer network, is an open offering to the technical scientific staff of the KFA to solve the communication problems of distributed processing in a heterogen computer system area.

Main transport mechanisms are FDDI with 100 Mbit/s to KFA institutes with high bandwidth requirements and Ethernet with 10 Mbit/s otherwise. The CRAY supercomputers and the IBM ES9000/620 (MVS/ESA) are connected to an UltraHub 1000 and via a Cisco router to the KFA wide FDDI network. The default path to the IBM mainframe is currently a FDDI connection via two IBM-RS6000/350. This two gateways connect the operating systems VM/ESA and MVS/ESA to the local network. For hardware backup purposes an IBM 8232 communication controller can be used.

The local area network KFAnet is connected to the German WiN, Wissenschafts-Netz, and across this to the worldwide Internet via a serial Cisco router interface with 2 MBit/s bandwidth.

The physical KFAnet/Internet configuration at KFA is shown in the picture on the next page.

The offered network services are the standard TCP/IP applications, developed by DARPA and Berkley. Main protocols used are FTP, Telnet, Electronic Mail, REXEC, Line Printing and Sun's NFS. Further applications have been developed based on the BSD socket interface. For software backup purposes the WDSF, Workstation Data Save Facility, and later on DFDSM, Data Facility Distributed Storage Management, now called ADSM, Adstar Distributed Storage Management, implemented by IBM, have been tested.

## 2.2 The workstation concept

Introducing the TCP/IP protocol family the KFA offered the possibility to use powerful workstations in connection with the existing mainframes, a future additional dimension of effective computing resources will come true. The workstation concept realizes the following objects in view:

- additional user productivity
- improvement of technical and psychological aspects; the offerings to the users are *state of the art*
- approving innovative computing concepts
- increasing effectivity and economy of central services and distributed data processing by balancing expenses and usage

Realizing the above objects a concept has to distribute computing into a central part and a local part. The central part can be done by servers on special mainframes or a batch system. The local part has to be realized by workstations.

With the TCP/IP protocol family the ZAM has offered a possibility to use the central servers, but this servers have to be accessed with an aggregate bandwidth. This can be done only by specialized interfaces and well matched networks.

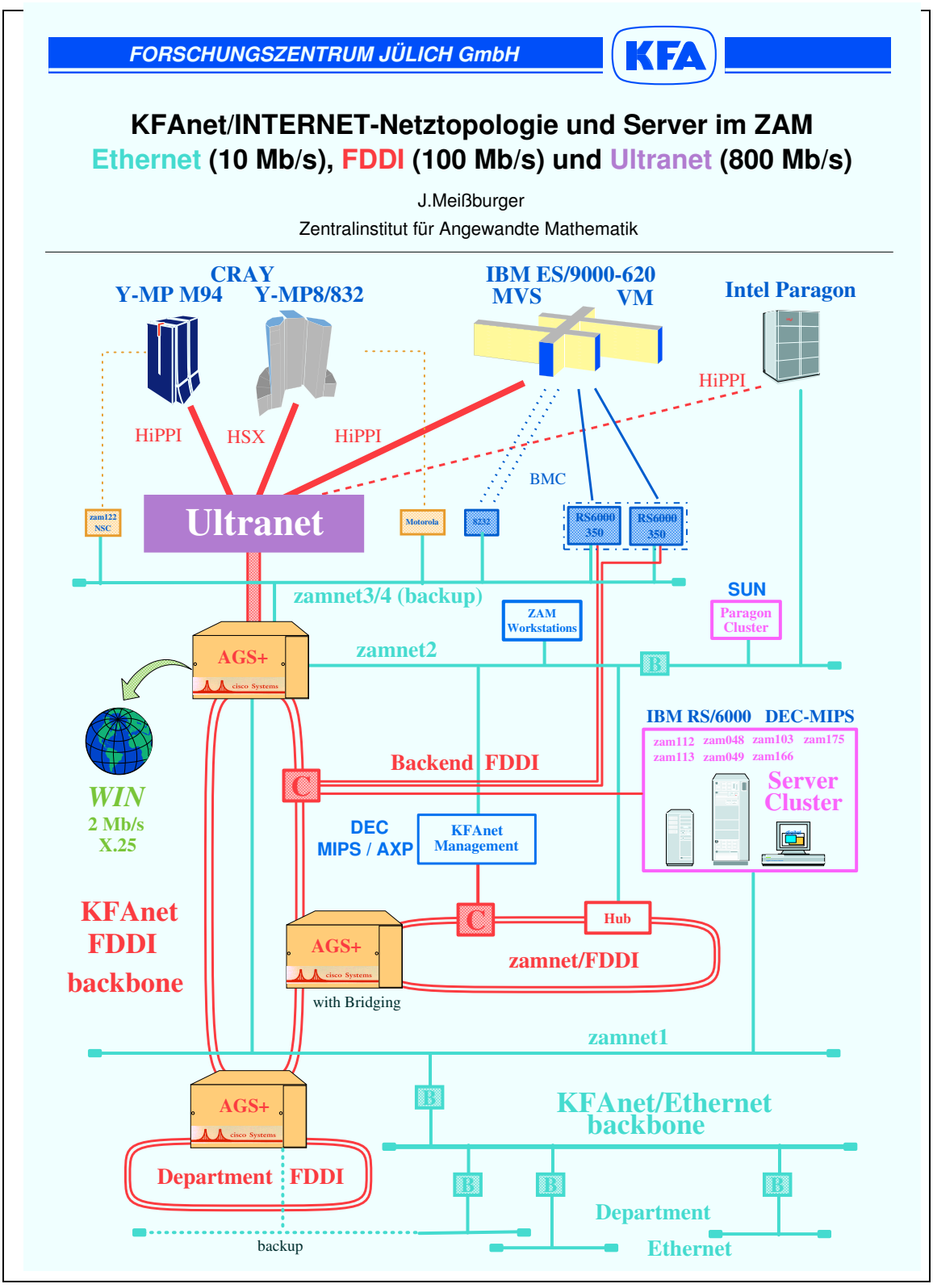


Figure 1 KFAnet/Internet configuration

## 2.3 UltraNet

The UltraNet network is a hub-oriented system providing a high-performance network, that links a variety of supercomputers, minisupercomputers, mainframes and workstations. The UltraHub operates at a data transfer rate of 1 gigabit (125 Mbyte) per second. The hub provides high speed interconnection between the Ultra network adapters which are located within the hub. The network adapters attach to host computers. They provide high-performance virtual circuit and datagram services and can route network traffic to a locally attached cisco router. One possible host connection is via HiPPI, another for example a HSX-Interface to the Cray-supercomputers. A software module, the Ultra Network Manager, initializes, monitors and controls the UltraNet network. Connections within the UltraNet can be done by an ISO-TP4–stack, whereas connections to the KFAnet/INTERNET have to be done via the Ultra-IP-stack [ULT-01], [ULT-02].

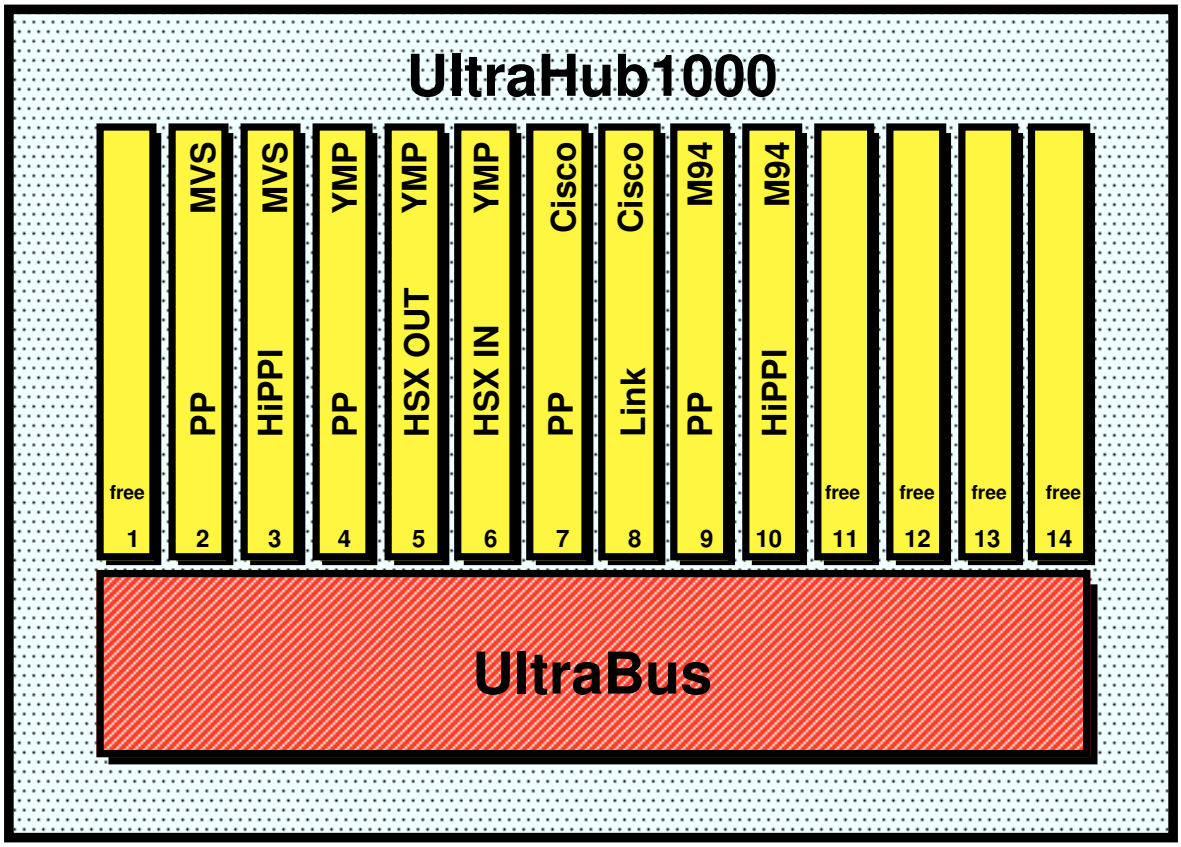


Figure 2 UltraHub1000 internals at KFA Juelich

## 2.4 HiPPI

The High Performance Parallel Interface, HiPPI, is an efficient simplex high-performance point-to-point interface, designed for transmitting digital data at peak data rates of 800 or 1600 Mbit/s. The IBM HiPPI interface is an implementation of the American National Standard Institute's (ANSI) X3.183–1991 High Performance Parallel Interface (HiPPI-PH) standard [ANS-05].

The transmission between data-processing equipment is designed using multiple twisted-pair copper cabling at distances up to 25 meters. The HiPPI-PH (physical layer) was implemented to cover the industry market needs, expressed both by users and manufacturers, to standardize the interconnection of data processing equipments at these data rates. The signalling protocol is designed to be distance independent, allowing the average data rate to approach the peak data rate, even over distances longer than 25 meters.

The HiPPI architecture is designed in an ISO-OSI like fashion. The two lower layers are organized as follows [ANS-01], [ANS-02], [ANS-03], ANS-04], [ANS-05]:

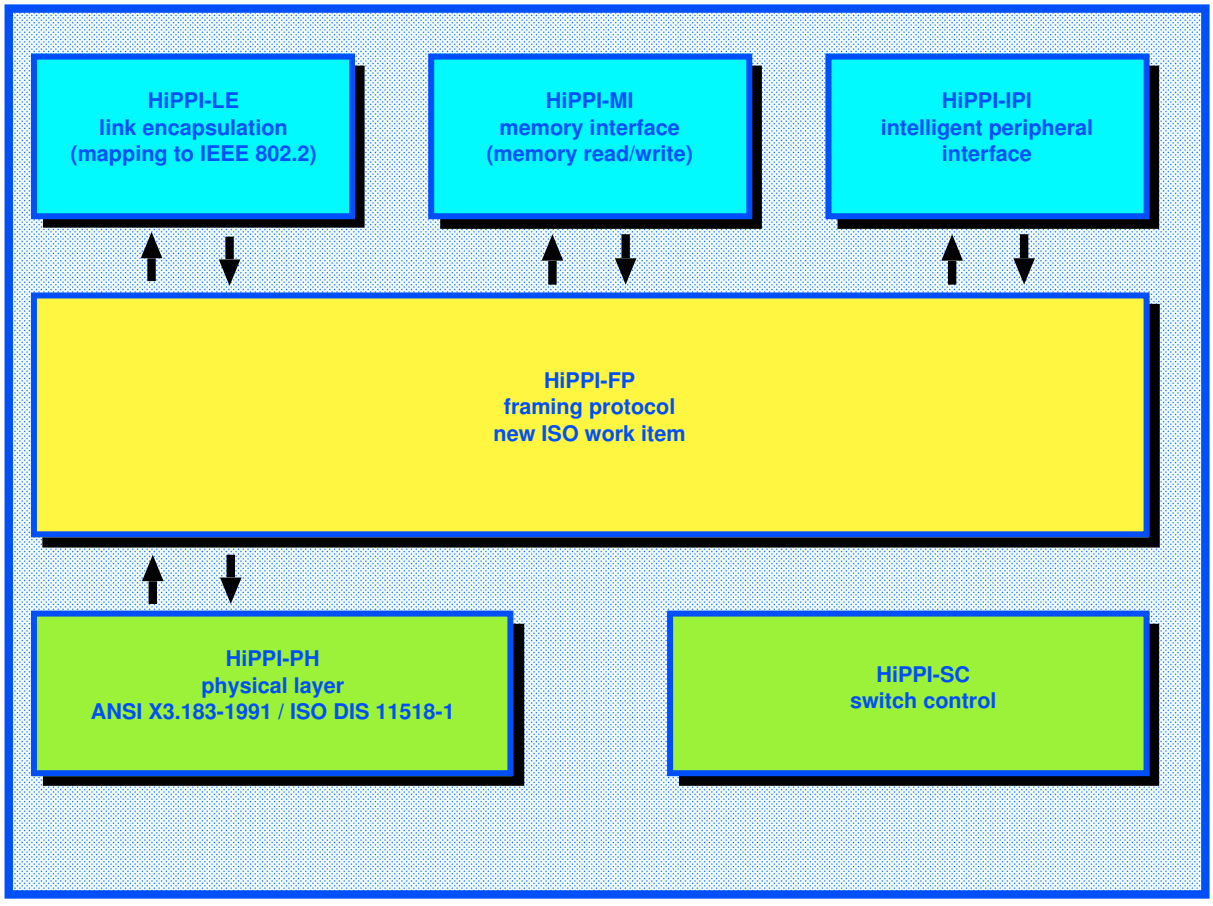


Figure 3 HiPPI Architecture

*Data transmitted across the IBM HiPPI may be defined in terms of packets, pages and bursts. Typically, a single transmission makes up a packet, although the HiPPI software interface provides the ability to break the transmission of a packet into multiple partial packets. A packet is made up of one or more pages (4096 bytes). This is the minimum resolution of the ES/3090 or ES/9000, and the smallest amount of data that may be transmitted over the HiPPI. Pages, once they have been moved on to the HiPPI device itself, are organized into bursts. A burst is a fixed-length sequence of 256 four-byte words.* [IBM-02]

## 2.5 UltraNet HiPPI network processor

The UltraNet HiPPI network processor is the Ultra high performance implementation of the ANSI standards Committee X3T9.3 High Performance Parallel Interface. It is a simple but very high performance channel interface for use by both computers and intelligent devices, such as routers. Ultra's HiPPI implementation includes a network protocol processor solving the integration issues faced by system administrators. The HiPPI consists of two independent HiPPI channels controlled by a network processor (NP). Ultra's standard HiPPI NP acts as a host computer connection, where a computer system has an HiPPI interface available for connection to the network. The HiPPI is tightly coupled with Ultra supplied host software for seamless integration into existing network-based applications, further details see below.

## 2.6 HiPPI for MVS at KFA

The purpose for a HiPPI attachment of an ES/9000 model 620 to the KFA network was to provide maximum aggregate bandwidth for bulk data transfers, as it is typical for applications like workstation backup, archival of workstation data or transmission of large files between fast external systems and a central file server. In this context it was not intended to attach disk arrays, nor visualize mainframe generated data on a special purpose graphical device. Special issue for the HiPPI-Project at KFA was to analyse the opportunities available with the IBM HiPPI interface in the context of the existing local KFAnet infrastructure.

The HiPPI interface should be connected to an IBM ES/9000 model 620 running MVS/ESA 4.1. The IBM is running MVS/ESA and VM/ESA in logical partitions under PR/SM. The installed communication software is IBM TCP/IP Version 2.2.1 on both systems.

The HiPPi test scenario implemented at KFA is displayed in the following picture:

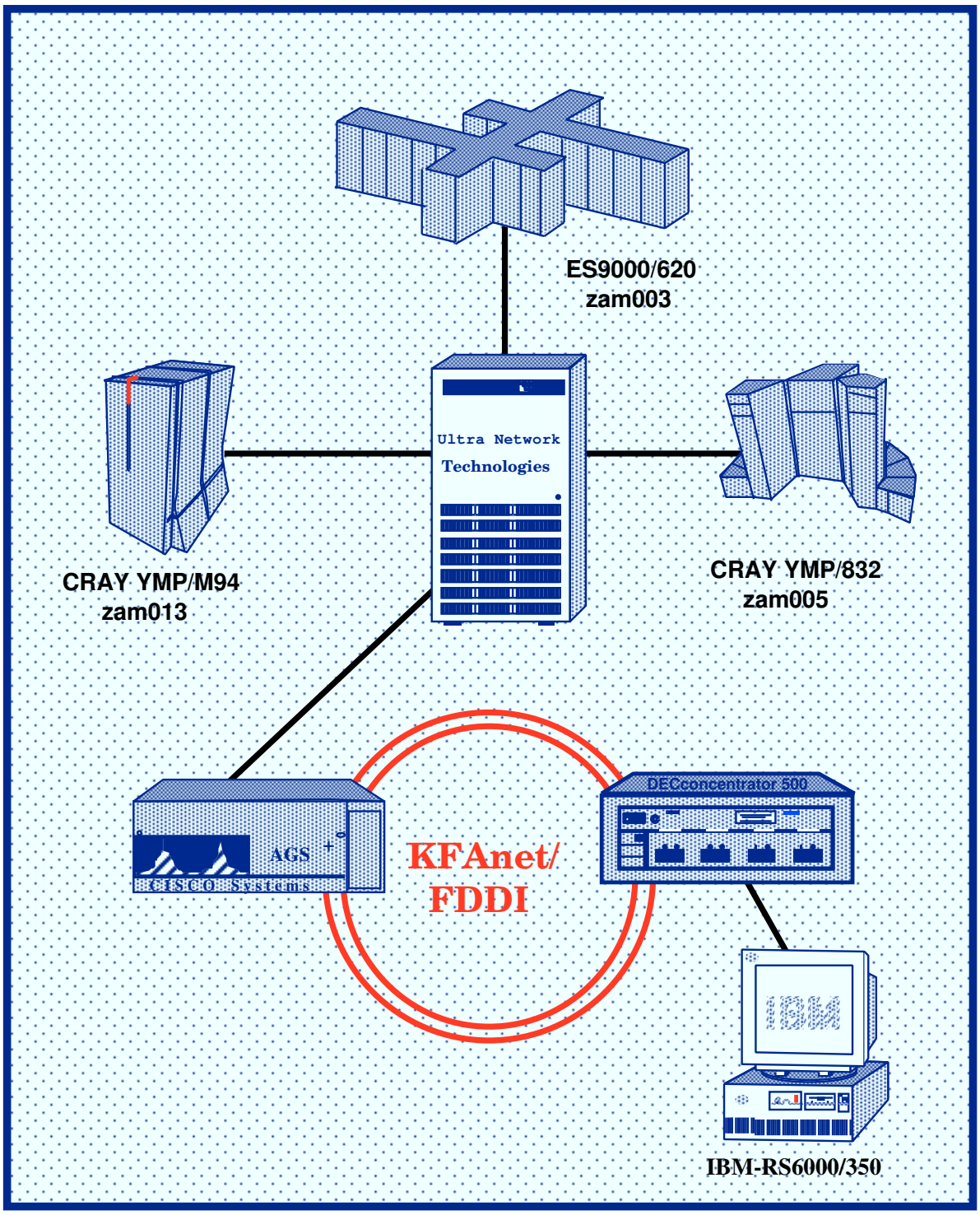


Figure 4 The HiPPI test scenario

# 3 Hardware Installation of the HiPPI-Interface

## 3.1 The IBM-HiPPI-interface

The IBM HiPPI interface has been installed at KFA in the middle of December 1992. Two employees of IBM needed about 8 hours for the HiPPI interface installation. No problems occured during the interface hardware installation. The HiPPI interface had to be installed in expanded storage, so that half of the possible expanded storage could not be further used. The old expanded storage had to be removed against a new smaller one, because of space problems. The expanded storage size has not been reduced. Additional power supply had to be installed. The HiPPI cabeling has been attached to CP2 internaly. New micro code for HiPPI support had to be installed. No other definitions have been required.

Hardware and Software tests, regarding the IBM-HiPPI-Interface have been made by IBM during the installation phase. No problems have been found. Cabeling has been prepared to attach to the HiPPI Ultra board. Since all hardware components needed for Ultra HiPPI communication are located in the same room and the distances between the components are within the allowable range, no problems in this area had been found and had to be solved.

## 3.2 The Ultra-HiPPI-board

The Ultra HiPPI board has been installed into the UltraHub at the 5th of January 1993. The time had to be fixed to this date, since the Hub had to be halted (SHUTDOWN) for the installation. So no connection to the CRAY supercomputers had been possible for 30 minutes. After installing the Ultra HiPPI board the IBM-HiPPI-cable could be connected. So the hardware installations had been closed.

## 4.1 The Software incompatibilities

To use the IBM-HiPPI adapter with UltraNet and the TCP/IP protocol there are three software components, which have to be installed (the software preloaded in the UltraHub environment is out of scope). These are the IBM-HiPPI software, the Ultra Host software and the IBM TCP/IP software.

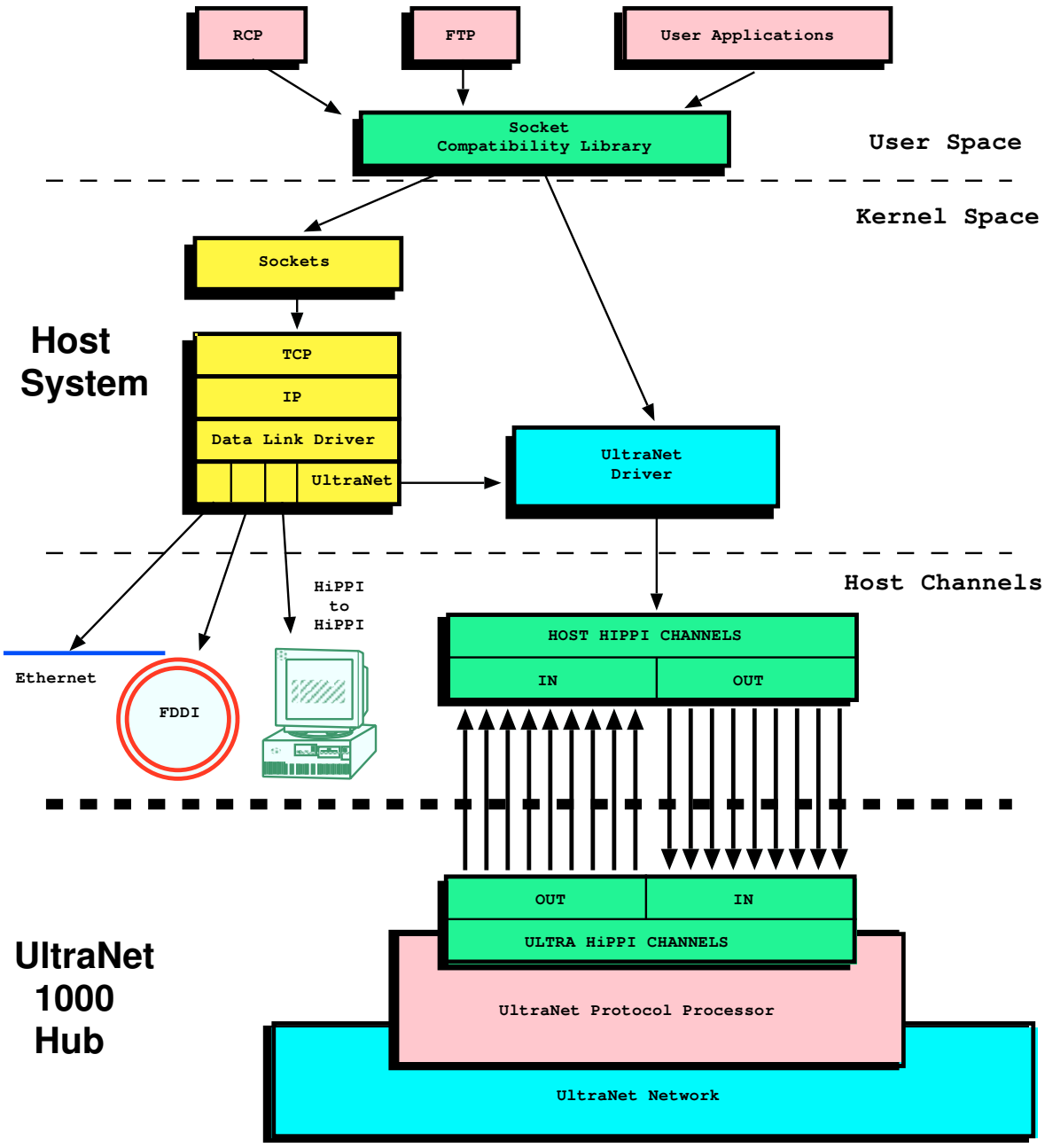


Figure 5 HiPPI, Ultra and TCP/IP software components

When installing the HiPPI interface IBM provided a *Beta test* HiPPI-software.The IBM European Center for Advanced Networking, ECAN, in Heidelberg/Germany, cooperated with KFA-Jülich in this project. The *Beta test* HiPPI-software was not compatible with the currently released Ultra Host software. Furthermore some tests regarding the MVS connection to the local FDDI network had been nessecary. For this tests IBM TCP/IP software version 2.2.1 had to be used. But this software was not compatible to the Ultra host software. Until this date Ultra only shipped software compatible to IBM TCP/IP 1.2. Taking into account the many mails and telephone calls, we got the three software components working together, when Ultra shipped the Ultra host software H390 in March 1993.

Configuring the software was no problem using the IBM and Ultra provided manuals. The default parameters as documented in the manual could be used. Some parameter settings, required by Ultra, are different to the settings we would prefer and recommend.

## 4.2 Software Bugs

Using the term *'NULLFILE'* as destination file with binary transmission leads to crash of IBM FTP server software on the MVS system. This crash does not occure, if we use the CLAW communication via an IBM-RS6000/350–gateway or an IBM 3172 model 1 or IBM 8232 interconnect controller.

Using buffer length of more than 1024 byte and communicate via Ultra's tsock program to MVS system causes a crash of the Ultra software. This two problems may coincident.

Calling *tsock* on MVS as source with a destination host on Ultra-TCP-stack seems to produce TP4 communication. Using *tsock* with a TP4 hostname as destination does not work. (Network unreachable message). See also chapter *Test environment and results* section *TSOCK*. Using *tsock* in the other direction you have to specify the TP4 hostname of the MVS system.

Ultra software on the CRAY systems had a bug. Communicating via FTP between the two CRAY systems using the host-stack leads to transmission speeds of 50 KByte/s. This seems to occur because of overrunning the protocol processors (PP's) at the UltraHub 1000. Some Ultra software patches had been in sight and are now available and working.

The maximum transmission unit term (MTU) within the MVS TCP/IP software can be defined to 8400 Byte at a maximum. Taking into account the throughput degradation, using low packet sizes, this seems to be a design failure. The ANSI HiPPI protocol allows packet sizes of up to 1 Mbyte.

The MTU sizes defineable, defaulted and recommended by Ultra on the different systems do not correlate (MVS 8400 Byte, CRAY 32 Kbyte, Cisco-router 1500 Byte).

# 5 Test environment and results

## 5.1 Introduction to test series

The IBM HiPPI tests have been made on an MVS test system using PR/SM. The four systems running on the ES9000/620 are VM, MVS, Test VM and Test MVS. They are weightend in the following manner:

| Operating System | Weighting |
|---|---|
| Production VM | 280 |
| Production MVS | 100 |
| Test VM | 100 |
| Test MVS | 100 |

Table 1 LPAR weighting at ES9000/620

This 580 entities have to be spawned over the available 4 cpus of the IBM mainframe, which has two operating systems running in a production environment, as seen in the table above.

The FDDI attached IBM RS6000/350 has been out of production and has only been used for this test purposes. The CRAY supercomputers and the Cisco-AGS+ router have been in real production. The test scenario has been shown above.

The tested programs have been TCPSPRAY, FTP and TSOCK (TCP and OSI/TP4 communication). Some UDP tests have been made too.

The test constellations have been (if possible, always in both directions and to all destinations):

- Direct Ultra connection MVS to CRAY/YMP
- Direct Ultra connection MVS to CRAY/M94
- Direct Ultra connection CRAY/YMP to CRAY/M94
- FDDI to Ultra connection IBM-RS6000/350 to MVS
- FDDI to Ultra connection IBM-RS6000/350 to YMP
- FDDI to Ultra connection IBM-RS6000/350 to M94

In general the listed values in the following tables have been produced, taking the average of five tests in succession.

Oscilating values have been rounded, ignoring values, which correspond to high network or cpu load. If there have been great oscilations, control test series have been made. The tables use column headings **6000, MVS, YMP, M94** and **Cisco** with arrows showing the direction of the data traffic. In this context the headings have the following meanings:

| Table entry / Abbreviation | System description |
|---|---|
| **6000** | IBM Risc System 6000 model 350 with AIX 3.2.1 |
| **MVS** | IBM ES9000 model 620 with MVS/ESA 1.1 and TCP/IP 2.2.1 |
| **YMP** | CRAY YMP/832 with UNICOS 7.0 |
| **M94** | CRAY YMP/M94 with UNICOS 7.0 |
| **Cisco** | Cisco AGS+ router |

Table 2 Abbreviations used in throughput tables

## 5.2 TCPSPRAY

The TCPSPRAY program is a socket based C-program. A source code version of tcpspray can be obtained by *anonymous ftp* from *nic.funet.fi*. It was compiled on IBM-ES9000/620 for MVS/ESA, CRAY for Unicos and IBM/RS6000 for AIX. The source program is *public domain* (see copyright).

For the Unix systems the program could be compiled without any changes. For MVS we had to do some minor changes.

**usage: TCPSPRAY [-v] [-e] [-h] [-b blksize] [-n nblks] [-f file] host**

| | |
|---|---|
| —v | verbose |
| —e | use echo service for full duplex transfer |
| —h | print this message |
| —b | blocksize in bytes (default 1024) |
| —n | number of blocks (default 100) |
| —f | file to preload buffer ( zero buffer by default) |

Figure 6 Usage of TCPSPRAY command

The paramters and options possible with the TCPSPRAY program are displayed above. The test series have been done always using —b and —n. Always the discard daemons was used.

Example:

**tcpspray -b64 -n50000 zam003.zam.kfa-juelich.de**

All listed speeds are shown in KByte/sec.

### 5.2.1 TCPSPRAY (Ultra+FDDI)

The following table lists the speeds with the tcpspray program measured from or to the IBM-RS6000/350 connected to the FDDI network. The datablocks had to cross the Ultra Hub and the FDDI network.

The test series could not be done to MVS, since there is no echo or discard daemon on MVS available.

| Kon-stellation / Block-width | 6000 ↓ (FDDI) ↓ Cisco ↓ (Ultra) ↓ MVS | 6000 ↓ (FDDI) ↓ Cisco ↓ (Ultra) ↓ YMP | 6000 ↓ (FDDI) ↓ Cisco ↓ (Ultra) ↓ M94 | MVS ↓ (Ultra) ↓ Cisco ↓ (FDDI) ↓ 6000 | YMP ↓ (Ultra) ↓ Cisco ↓ (FDDI) ↓ 6000 | M94 ↓ (Ultra) ↓ Cisco ↓ (FDDI) ↓ 6000 |
|---|---|---|---|---|---|---|
| **64** | - | 390 | 400 | 40 | 580 | 430 |
| **128** | - | 460 | 460 | 80 | 590 | 500 |
| **1024** | - | 1950 | 1950 | 400 | 1870 | 1650 |
| **1460** | - | 2230 | 2200 | 420 | 1970 | 1800 |
| **4096** | - | 2800 | 2600 | 460 | 2030 | 1800 |
| **8192** | - | 2500 | 2400 | 460 | 2100 | 1850 |
| **32768** | - | 2400 | 2300 | - | 1000 | 1900 |

Table 3 Throughput TCPSPRAY (Ultra + FDDI)

Annotation: All transmission speeds are listed in KBytes/sec.

### 5.2.2 TCPSPRAY( Ultra)

The following table lists the transmission speeds with the tcpspray program measured between the systems connected directly to the Ultra HUB. The datablocks had to cross only the Ultra Hub. No other network has been involved.

The test series could not be done to MVS, since there is no echo or discard daemon on MVS available. The tests with blockwidth 32K from MVS systems to the CRAY supercomputers have not been made, since buffers in MVS tcpspray program are limited to 16Kbyte and the MVS MTU size is limited to 8400 Byte.

| Kon-stellation / Block-width | YMP ↓ (Ultra) ↓ MVS | M94 ↓ (Ultra) ↓ MVS | MVS ↓ (Ultra) ↓ YMP | MVS ↓ (Ultra) ↓ M94 | YMP ↓ (Ultra) ↓ M94 | M94 ↓ (Ultra) ↓ YMP |
|---|---|---|---|---|---|---|
| **64** | - | - | 30 | 30 | 440 | 350 |
| **128** | - | - | 60 | 70 | 630 | 520 |
| **1024** | - | - | 330 | 560 | 1660 | 1080 |
| **1460** | - | - | 410 | 740 | 1830 | 1270 |
| **4096** | - | - | 540 | 1340 | 2020 | 1660 |
| **8192** | - | - | 550 | 1420 | 3800 | 1820 |
| **32768** | - | - | - | - | 3700 | 2430 |

Table 4 Throughput TCPSPRAY (Ultra)

Annotation: All transmission speeds are listed in KBytes/sec.

## 5.3 FTP

The File Transfer Protocol (FTP) is designed to transfer files between two systems. Establishing the session the user has to be authorized at the remote system. This is done by prompting for destination userid and password and checking this values at the remote system.

The general format of the ftp command is:

**usage: ftp [-d] [-g] [-i] [-n] [-v] [host]**

| | |
|---|---|
| —d | enable debugging |
| —g | disable expansing of metachars in filenames |
| —i | turns off interactive prompting during multiple file transfers |
| —n | prevents an automatic login on initial connection |
| —v | verbose |
| —host | name of the host machine to which files will be transfered |

Figure 7 Usage of FTP command

We always used the *ftp put* subcommand, which implies, that the disk read operation is done by the client process. The client has been an unprivileged account at the local system.

The test series could have been done using the *ftp get* subcommand too, but because all tests have been started from all systems, all directions and servers have been tested.

### 5.3.1 FTP (Ultra+FDDI)

The following table lists the ftp transfer rates measured from or to the IBM-RS6000/350 connected to the FDDI network. The datablocks had to cross the Ultra Hub and the FDDI network.

As recommended by Ultra the test series to */dev/null* to the MVS system could be done by specifying the destination file as file *nullfile*, after changing the directory to root. This can be achieved by typing *"cd .."*. Transferring the file this way causes an FTP software crash.

| **Kon-stellation / Block-width** | **6000 ↓ (FDDI) ↓ Cisco ↓ (Ultra) ↓ MVS** | **6000 ↓ (FDDI) ↓ Cisco ↓ (Ultra) ↓ YMP** | **6000 ↓ (FDDI) ↓ Cisco ↓ (Ultra) ↓ M94** | **MVS ↓ (Ultra) ↓ Cisco ↓ (FDDI) ↓ 6000** | **YMP ↓ (Ultra) ↓ Cisco ↓ (FDDI) ↓ 6000** | **M94 ↓ (Ultra) ↓ Cisco ↓ (FDDI) ↓ 6000** |
|---|---|---|---|---|---|---|
| **6MByte auf Disk Ascii** | 50 | 500 | 500 | 210 | 620 | 600 |
| **6MByte auf Disk Binary** | 50 | 700 | 740 | 210 | 640 | 610 |
| **6MByte /dev/null Ascii** | - | 530 | 530 | 210 | 650 | 610 |
| **6MByte /dev/null Binary** | - | 720 | 740 | 210 | 650 | 620 |

Table 5 Throughput FTP (Ultra + FDDI)

Annotation: All transmission speeds are listed in KBytes/sec.

### 5.3.2 FTP (Ultra)

The following table lists the transmission speeds with the ftp program measured between the systems connected directly to the Ultra HUB. The datablocks had to cross only the Ultra Hub. No other network has been involved.

Regarding the test series to */dev/null* to the MVS system see the annotations in the section above.

| Kon-stellation / Block-width | YMP ↓ (Ultra) ↓ MVS | M94 ↓ (Ultra) ↓ MVS | MVS ↓ (Ultra) ↓ YMP | MVS ↓ (Ultra) ↓ M94 | YMP ↓ (Ultra) ↓ M94 | M94 ↓ (Ultra) ↓ YMP |
|---|---|---|---|---|---|---|
| **6MByte auf Disk Ascii** | 440 | 600 | 950 | 980 | 4300 | 3800 |
| **6MByte auf Disk Binary** | 430 | 700 | 970 | 960 | 7700 | 7000 |
| **6MByte /dev/null Ascii** | - | - | 600 | 1050 | 4400 | 4000 |
| **6MByte /dev/null Binary** | - | - | 640 | 1070 | 7500 | 7500 |

Table 6 Throughput FTP (Ultra)

Annotation: All transmission speeds are listed in KBytes/sec.

## 5.4 TSOCK

The tsock program is a Ultra test program, based on the *ttcp* program originally developed by T.C.Slattery, USNA, and improved by Mike Muuss and Terry Slattery. Due to the large number of changes, however (asynchio, data checking, random length and offset, fake-data option, ...), it has effectively been completely rewritten. The processing logic is as follows:

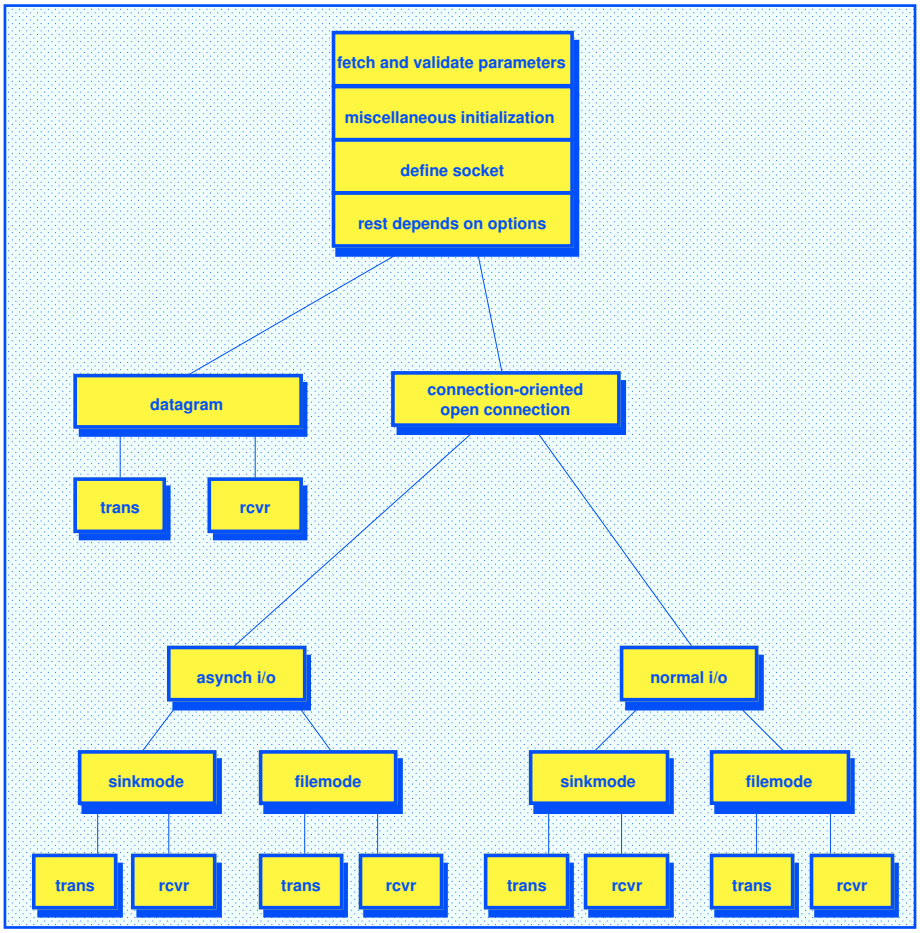


Figure 8 TSOCK processing logic

A general description of the Tsock command can be found in the appendix. A short description is shown in the following usage figure.

| | | |
|---|---|---|
| **usage:** | **tsock** | **[-options] [host] > out** |
| | -r | operate as receiver (passive initiation, X receive dara) |
| | -t | operate as transmitter (active initiation, X transmit data) |
| | -s | source/sink mode (generate and discard a data pattern) |
| | -l## | length of network buffer (default 1024) |
| | -n## | number of messages to transmit (with -s only, default 1024) |
| | -m## | mark progress with 'X' for every ## bytes transferred |
| | -?l | use random length messages <= -l## value (without -d only) |
| | -?s | use random starting location (i.e., random byte boundary) |
| | -?r | do NOT seed random number generator (for repeatability) |
| | -p## | port number for connection or datagram path (default 2000) |
| | -B | only transmit full blocks, as specified in -l## (for TAR) |
| | -d | use datagrams ( with -s only) |
| | -a## | use asynch network I/O (eg. UNICOS reada) with ## buffers |
| | -f | set 'fake data' option |
| | -c | check received data (receiver with -s only) |
| | -u## | set TPDU size to 2^##bytes (eg. 12=4K; default 15) |
| | -x## | run ##processes in parallel (via fork) |
| | -z## | same as -x## but fork both receivers and transmitters |
| | -Pxpat | use alternate pattern hsx 'xpat' ( default 0x00-0xff) |
| | -Ffile | use alternate pattern from file |
| | host | host name (optional local for -r, required remote for -t) |

Figure 9 Usage of TSOCK command

The calling syntax at the source system has been:

tsock —t —s —l*buflen* —n*numberofbufs host*

The calling syntax at the destination system has been:

tsock —r —s —l*buflen*

The number of buffers has been choosen, so that the overall communication time for a request does not need more than 15 seconds.

Communicating to the MVS system with a bufferlength greater than 1024 Byte caused an Ultra software crash.

The communication path, the TSOCK program uses, depends on the destination host. Ultra defines two types of communication paths. First the Ultra TCP path (host stack or uh-path) and second the Ultra TP4 path (native path) can be used. Since the Ultra Hub internally uses the TP4 path, and many of the communication parts are fully implemented

in hardware, this communication path should be the faster. The way of communication, TCP or TP4, will be choosen by the Ultra socket library.

### 5.4.1 TSOCK (Ultra TCP-communication)

The following table shows the transmission speeds measured transfering data with different bufferlength via the Ultra host stack ( uh-path) to directly connected Ultra hosts. The communication pathes from and to the MVS system could not be tested, since the MVS-software had not been linked with the required Ultra software components. This tests have not been done later on, since the overall communication throughput showed nonacceptable values.

| **Kon-stellation / Buffer-length** | **YMP ↓ (Ultra) ↓ MVS** | **M94 ↓ (Ultra) ↓ MVS** | **MVS ↓ (Ultra) ↓ YMP** | **MVS ↓ (Ultra) ↓ M94** | **YMP ↓ (Ultra) ↓ M94** | **M94 ↓ (Ultra) ↓ YMP** |
|---|---|---|---|---|---|---|
| **64** | ? | ? | ? | ? | 560 | 370 |
| **128** | ? | ? | ? | ? | 850 | 850 |
| **1024** | ? | ? | ? | ? | 2900 | 2800 |
| **1460** | ? | ? | ? | ? | 3200 | 3700 |
| **4096** | ? | ? | ? | ? | 4700 | 4400 |
| **8192** | ? | ? | ? | ? | 5500 | 5000 |
| **32768** | ? | ? | ? | ? | 5700 | 5800 |
| **1MByte** | ? | ? | ? | ? | 5700 | 5700 |

Table 7 Throughput TSOCK (Ultra TCP-communication)

Annotation: All transmission speeds are listed in KBytes/sec.

### 5.4.2 TSOCK (Ultra TP4-communication)

The following table shows the transmission speeds measured transfering data with different bufferlength via the Ultra native path (TP4-path) to directly connected Ultra hosts.

Communicating to the MVS system with a bufferlength greater than 1024 Byte caused an Ultra software crash. We did not try to fix the problem, because of limited time.

The transmission speeds we see here between the two CRAY systems show, which data throughput can be achieved using aggregate bufferlength. The Ultra Hub and IBM HiPPI implementation seems really not to be designed for common TCP/IP communication protocols like Telnet and FTP.

| **Kon-stellation / Buffer-length** | **YMP ↓ (Ultra) ↓ MVS** | **M94 ↓ (Ultra) ↓ MVS** | **MVS ↓ (Ultra) ↓ YMP** | **MVS ↓ (Ultra) ↓ M94** | **YMP ↓ (Ultra) ↓ M94** | **M94 ↓ (Ultra) ↓ YMP** |
|---|---|---|---|---|---|---|
| **64** | 10 | 10 | 10 | 10 | 33 | 40 |
| **128** | 26 | 24 | 26 | 20 | 66 | 83 |
| **1024** | 260 | 190 | 170 | 150 | 570 | 3500 |
| **1460** | Error | Error | 245 | 290 | 750 | 3700 |
| **4096** | Error | Error | 680 | 680 | 2130 | 4300 |
| **8192** | Error | Error | 1400 | 1300 | 3850 | 5000 |
| **32768** | Error | Error | 6800 | 4300 | 11200 | 11000 |
| **1MByte** | Error | Error | 19800 | 18300 | 30500 | 26300 |

Table 8 Throughput TSOCK (Ultra TP4-communication)

Annotation: All transmission speeds are listed in KBytes/sec.

# 6 Conclusion

To achieve consistent HiPPI performance, the cpu running the HiPPI has to be dedicated. Running under LPAR will lead to degraded data transfer rates on the inbound HiPPI channel than in a non-LPAR environment. The cpu's in the KFA production environment cannot be dedicated, since the free cpu capacities of the VM system at night times will be used for MVS batch jobs running, and since only the cpu capacities at day times, which will not be used at VM, may be used by MVS jobs.

*It may be possible to achieve acceptable HiPPI performance without dedicated CPs by weighting the HiPPI partition so that its weight is 4 times the sum of the weights of other partitons. However, this type of shared environment is not recommended.* [IBM-02]

At KFA this weighting would lead to an unacceptable favour of the MVS batch system, if the HiPPI is located to this system. The cpu performance of the CMS interactive system running in another logical partition will be degraded extremely, since there are MVS jobs queued, which have to consume the rest of the available cpu power. If no HiPPI activity is done, this batch jobs would use 80% of the 4 cpu's.

*HiPPI interrupts are directed to a fixed CP (usually CP2 for a HiPPI installed on side 0 and CP4 for a HiPPI installed on side 1). When LPAR assigns a HiPPI partition to run on a CP other than the one receiving the HiPPI interrupt, the interrupt must be redirected. This interrupt redirection takes time and can cause a reduction in the sustained data transfer rates on the inbound HiPPI channel.*[IBM-02] Since the HiPPI adapter at KFA is directly attached to one cpu, using PR/SM leads to the reduction in transfer rates.

The most significant fact effecting the data throughtput is the HiPPI packet size. Since each packet causes an interrupt, small packet sizes will cause a greater drop in performance than larger packet sizes.

Performance estimates indicate, that there is a performance degradation on the HiPPI channel, if a small packet size is used of up to 5% loss in throughput, if the packet size is 32 KB. If there is a 4 KB packet size the loss in throughput is 32% [IBM-07]. Other IBM sources mention a loss of throughput for 32 KB packet sizes of up to 42%. So the loss for 8KB packet sizes will be up to or more than 50%. Since Ultra limits the TCP/IP maximum transmission unit size (MTU) to 8400 Bytes, 8192 Bytes data plus headers etc., this loss of throughput related to the packet size can be assumed.

Another performance degradation will be the data moves in main storage. *The HiPPI performance for a 3090/H0 HiPPI interface can be estimated to 98 MBytes/s, if no data moves in main storage take place. If one data move appears, the performance degrades to 50–60 MBytes/s. If data moves are greater equal two, the transmission speed can be estimated to 15–30 MBytes/s* [IBM-08].

The measured throughputs for the experienced applications TCPSPRAY, FTP and TSOCK lead to the result, that only an application using the TP4 protocol, direct Ultra connection, can use the bandwidth associated with the UltraHub and the HiPPI MVS

interface. Though the data rates using the HiPPI at MVS with TP4 are much better than using TCP-protocol, nevertheless the data rates are not satisfactory.

The first assumption the HiPPI MVS interface in connection with the UltraHub could be used as a data collector of low bandwidth ethernets and therefore maximize the throughput to the MVS system has been an error. The experiences we made show no better data rates than using an gateway solution with IBM-RS6000 workstations [KFA01, KFA02] or an IBM 3172 model 1, sometimes lower than an IBM 8232.

The applications using the HiPPI performance between the MVS system and the Cray supercomputers are not in sight. Backup of Cray disks will go to the storage tek robot at KFA and do not need Ultranet. Archiving, backing up or restoring files via ADSM can only be done, if a ADSM client is available on the Cray supercomputers, which is not implemented until now. The interaction of MVS and Cray users is currently limited. Some datasets will be fetch from MVS and stored to the MVS system, but the usage degrades. If the MVS system will not become a raised value in the near future and will not home an important server with high bandwidths requirements, the installation and usage of the MVS HiPPI interface can not be recommended.

The software errors we found will be solved in the next time or have been solved until now, but we have to consider, that every new error we will find, has to be solved by Ultra Networks Technologies in conjunction with IBM. Taking into account the low number of installations using a MVS Hippi connection to an UltraHub in a production environment, getting solutions for software errors will become more difficult. Here we have to consider too, that IBM stopped the development of HiPPI. Support will only be done on an *"as is"* base. Further development will be done on Fiber Channel Standard (FCS).

Furthermore tests using an Ultra-BMC (Block-Multiplexer-Channel) connection for the IBM ES9000 MVS-Operating system have shown the same data throughput, than using an Ultra HiPPI interface. The price/performance ratio leads to the use of the Ultra-BMC solution.

Taking into account the various limitating factors like LPAR, PR/SM, IBM TCP/IP software implementation, many software components (TCP/IP, HiPPI, Ultra), many different connected systems with various required and recommended parameter settings, Disk I/O, Batch Operating System and so on, it does not surprise , that the IBM ES9000 MVS connection via the Ultra HiPPI interface does not provide an aggregate data communication throughput.

As a consequence the KFA decided to renounce the use of the IBM HiPPI interface and instead of this implement the Ultra-BMC solution. Because of the persuading performance of the IBM RS6000 gateway solution, we have seen in 1993, regarding maintenance and throughput, in future the Ultra-BMC communication path will only be used for backup purposes or for special CRAY to MVS communications.

# 7 Appendix

## 7.1 MVS/ESA Definitions

### 7.1.1 MVS-Ultra Definitions

#### 7.1.1.1 Ultra HiPPI Profile (UPROFILH)

```
# ---------------------------------------------------
# KFA Ultra startup parms for the single HiPPI adapter
# on the IBM host.
# ---------------------------------------------------
# command_name  sub_cmd_name  function
# --------  --------  --------------------------------
# unetdb        set debug options
# hippicf       define config info for HiPPI driver
# unetcf    madd      add     mapping info
# unetcf    mdis      display mapping info
# unetcf    radd      add     route   info
# unetcf    rdis      display route   info
# unetcf    aadd      add     adapter into
# unetcf    adis      display adapter into
# ---------------------------------------------------
unetdb trace off
unetdb error uerror err_rb
# ---------------------------------------------------
# en/disable IP broadcast: Format: broadcast on|off
broadcast on
# ---------------------------------------------------
unetcf madd -f 'KFA1.UH390M20.PARMLIB(KFASTAT)'
# ---------------------------------------------------
# cmd  input_driver_devid  HiPPI_id
hippicf 01       0
# ---------------------------------------------------
unetcf aadd hippi0 zam003-b 0101 0101
# ---------------------------------------------------
hippidb +trace
# ---------------------------------------------------
unetcf adis
# ---------------------------------------------------
unetcf radd zam003-b hippi0 0xffffff00 zam003-b
unetcf radd zam003-uh hippi0 0xffffff00 zam003-uip -i
# ---------------------------------------------------
unetcf rdis
# ---------------------------------------------------
```

#### 7.1.1.2 Ultra Stations file (STATIONH)

```
#=========================================
# KFA Stations file for the single HiPPI
# adapter on the IBM host.
#
#========================== MVS ===========
zam003-ultra    5/33
zam003-b        5/33
zam003-uip      5/33
zam003-uh       5/33
zam003-u        5/33
mvs-ultra       5/33
#========================== M94 ===========
zam013-a        5/35
zam013          5/35
zam013-uip      5/35
zam013-uh       5/35
zam013-u        5/35
zam013-usl      5/35
zam013-uap      5/35
sn225           5/35
m94             5/35
#========================== YMP ===========
zam005-a        5/40
zam005-b        5/40
zam005          5/40
zam005-uip      5/40
zam005-uh       5/40
zam005-u        5/40
zam005-usl      5/40
zam005-uap      5/40
ymp             5/40
sn1011          5/40
#========================== Cisco =========
zam047-uap      6/33
zam047-usl      6/33
zam047-u        6/33
zam047-uh       6/33
zam047-uip      6/33
zam047          6/33
#=========================================
```

### 7.1.2 TCP/IP Version 2.2 (Definitions)

#### 7.1.2.1 PROFILE TCPIP

```
;-------------------------------------------------------------
; KFA PROFILE TCPIP file for the single HiPPI
; adapter on the IBM host.
;-------------------------------------------------------------
LARGEENVELOPEPOOLSIZE  900 16384
DATABUFFERPOOLSIZE 160 16384
SMALLDATABUFFERPOOLSIZE 256
;-------------------------------------------------------------
INFORM  OPERATOR ENDINFORM
OBEY    OPERATOR SNMPQE SNMPD ENDOBEY
;-------------------------------------------------------------
; Set Telnet timeout to 10 minutes
INTERNALCLIENTPARMS TIMEMARK 600 ENDINTERNALCLIENTPARMS
;-------------------------------------------------------------

 DEVICE UHIP HIPPI 1
 LINK   ULT1 HIPPI      1 UHIP
 HOME 134.94.72.4   ULT1
 GATEWAY
 134.94     =             ULT1   8400   0.0.255.0    0.0.72.0
 134.94     134.94.72.10  ULT1   4470   0.0.255.0    0.0.76.0
 134.94     134.94.72.10  ULT1   4470   0.0.255.0    0.0.100.0
 134.94     134.94.72.10  ULT1   4096   0.0.255.0    0.0.96.0
 134.94     134.94.72.10  ULT1   4096   0.0.255.0    0.0.112.0
 134.94     134.94.72.5   ULT1   8400   0.0.255.0    0.0.68.0
DEFAULTNET 134.94.72.10  ULT1   1500   0
 START UHIP
;-------------------------------------------------------------
AUTOLOG
   FTPHIP                        ; FTP server
ENDAUTOLOG
;-------------------------------------------------------------
PORT
 ; Values from RFC 1010, "Assigned numbers"
   21  TCP FTPHIP                ; FTP server
   20  TCP FTPHIP   NOAUTOLOG    ; FTP default data port
   23  TCP INTCLIEN              ; Telnet server
   111 UDP PORTMAP
   111 TCP PORTMAP
;-------------------------------------------------------------
```

```
; Define the VTAM parameters required for the TELNET server
BEGINVTAM
    ; Define logon mode tables to be the defaults shipped
    ; with the latest level of VTAM
  3278-3-E NSX32703 ; 32 line screen - default of NSX32702 is 24
  3279-3-E NSX32703 ; 32 line screen - default of NSX32702 is 24
  3278-4-E NSX32704 ; 48 line screen - default of NSX32702 is 24
  3279-4-E NSX32704 ; 48 line screen - default of NSX32702 is 24
  3278-5-E NSX32705 ; 132 col screen - default of NSX32702 is 80
  3279-5-E NSX32705 ; 132 col screen - default of NSX32702 is 80
    ; Define the LUs to be used for general users
  DEFAULTLUS
      TCP00001 TCP00002 TCP00003 TCP00004 TCP00005
      TCP00006 TCP00007 TCP00008 TCP00009 TCP00010
      TCP00011 TCP00012 TCP00013 TCP00014 TCP00015
      TCP00016 TCP00017 TCP00018 TCP00019 TCP00020
      TCP00021 TCP00022 TCP00023 TCP00024 TCP00025
      TCP00026 TCP00027 TCP00028 TCP00029 TCP00030
  ENDDEFAULTLUS
  ;-------------------------------------------------------------
 ALLOWAPPL *
;-------------------------------------------------------------
ENDVTAM
SYSCONTACT F.J. Schoenebeck ENDSYSCONTACT
SYSLOCATION KFA-ZAM 02461-61/6432 ENDSYSLOCATION
;-------------------------------------------------------------
```

## 7.2 Participating Systems

**CRAY Y-MP8/832**

| | |
|---|---|
| Central processors | 8 |
| CPU cycles | 6 ns |
| Performance | 2666 MFLOPS |
| Main storage | 256 MB |
| I/O-Processors | 4 |
| Schnellspeicher (SSD) | 1024 MB |
| Operating System | UNICOS 7.0 |
| Number of UserId's | ca. 390 |

**CRAY Y-MP M94**

| | |
|---|---|
| Central processors | 4 |
| CPU cycles | 6 ns |
| Performance | 1333 MFLOPS |
| Main storage | 2048 MB |
| I/O-Processors | 2 |
| Operating System | UNICOS 7.0 |
| Number of UserId's | ca. 420 |

**IBM ES/9000 Model 620**

| | |
|---|---|
| CPU's | 4 |
| Performance | 84 MIPS |
| Main storage | 512 MB |
| Expanded storage | 512 MB |
| Parallel channels | 80 |
| Fiber channels | 32 |
| Vector engines | 2 |
| Performance | 138 MFLOPS/Vector engine |
| Operating System | VM/ESA with CMS and<br>MVS/ESA in logical partitions |
| Number of UserId's | VM: ca. 2250<br>MVS: ca. 2000 |

**IBM RISC System/6000 Model 350**

| | |
|---|---|
| Central processors | 1 |
| CPU Cycles | 24 ns |
| Performance (Speckfp92) | 65 MFLOPS |
| Main storage | 32 MB |
| Data cache | 32 KB |
| Instruction cache | 8 KB |
| Memory bus | 64 bit |
| DASD | 400 MB |
| Operating System | AIX V3.2 |
| Number of UserId's | ca. 10 (System users) |

**Cisco AGS+ Router**

| | |
|---|---|
| Central processors | 1 csc/4 |
| cBus | cbus I |
| Main storage | 16 KB |
| Multibus Memory | 64 KB |
| Non Volatile Config Memory | 64 KB |
| Software Version | 9.03 |
| Interfaces | 4 Serial |
| | 4 Ethernet |
| | 1 FDDI |
| | 1 HSCI (High Speed Channel Interface) |

## 7.3 Man page TCPSPRAY

```
TCPSPRAY(1)              (23 October 1991)              TCPSPRAY(1)


NAME
     tcpspray - print average throughput for a tcp connection

SYNOPSIS
     tcpspray [ -v ] [ -e ] [ -h ] [ -b blksize ] [ -n nblks ]
          [ -f filename ] hostname

DESCRIPTION
     tcpspray sends data to either the discard or echo TCP
     service on the specified host and prints the average
     throughput.

OPTIONS
     -v             Prints a dot for each block sent. Will also
                    print a backspace for each block received in
                    echo mode.  Note: the I/O required for this
                    option will affect the throughput rates.

     -e             Use the TCP echo service instead of discard
                    (the default) and print throughput rates for
                    both transmission and reception.

     -h             Print a usage description.

     -b blksize     Sets the size of a block (the internal
                    buffer) in bytes.  Defaults to 1024.

     -n nblks       Sets the number of blocks to transfer.
                    Defaults to 100.

     -f filename    Copy the contents of the specified file into
                    the internal buffer (sized by -b option).
                    The buffer is zeroed by default.  If the file
                    is larger than the buffer, only the first
                    blksize bytes will be used.  If the file is
```

smaller than the buffer, the remaining bytes are zeroed.

This option is useful in determining the relationship of the data transferred to throughput. E.g., if data compression is used on any of the intermediate links comprising the TCP connection, preloading the buffer with a text file will produce greater throughput than with a file that has already been compressed.

ping(8), spray(8)

AUTHOR

Greg Christy (gmc@quotron.com)

## 7.4 Man page FTP

ftp Command

Purpose

Transfers files between a local and a remote host.

Syntax

ftp [ -d] [ -g] [ -i] [ -n] [ -v] [ HostName]

Description

The ftp command is the interface to the File Transfer Protocol (FTP). This command uses FTP to transfer files between the local host and a remote host or between two remote hosts.

The FTP protocol allows data transfer between hosts that use dissimilar file systems. Although the protocol provides a high degree of flexibility in transferring data, it does not attempt to preserve file attributes (such as the protection mode or modification times of a file) that are specific to a particular file system. Moreover, the FTP protocol makes few assumptions about the overall structure of a file system and does not provide or allow such functions as recursively copying subdirectories.

Note: If you are transferring files between AIX systems and need to preserve file attributes or to recursively copy subdirectories, use the rcp command.

At the ftp> prompt, you can enter subcommands to perform tasks such as listing remote directories, changing the current local and remote directory, transferring multiple files in a single request, creating and removing directories, and escaping to the local shell to perform shell commands.

Security and Automatic Login

The ftp command also provides for security by sending passwords to the remote host and permits automatic login, file transfers, and logoff.

If you execute the ftp command and specify the host name (HostName) of a remote host, the ftp command tries to immediately establish a connection to the specified host. If the ftp command connects successfully, the ftp command searches for a local $HOME/.netrc file in your current directory or home directory. If the file exists, the ftp command searches the file for an entry that initiates the login process and command macro definitions for the remote host. If the $HOME/.netrc file or autologin entry does not exist or if your system has been secured with the securetcpip command, the ftp command prompts the user for a user name and password. This occurs regardless of whether the HostName parameter is specified on the command line.

Note: The queuing system does not support multibyte host names.

If the ftp command finds a $HOME/.netrc autologin entry for the specified host, the ftp command attempts to use the information in that entry to automatically log in to the remote host. The ftp command also loads any command macros defined in the entry. In some cases (for example, when the required password is not listed in an autologin entry), the ftp command prompts for the password before displaying the ftp> prompt.

Once the ftp command completes the autologin process, the ftp command executes the init macro if the macro is defined in the autologin entry. If the init macro does not exist or does not contain a quit or bye subcommand, the ftp command then displays the ftp> prompt and waits for a subcommand.

Note: The remote user name that you specify either at the prompt or in a $HOME/.netrc file must exist and have a password defined at the remote host. Otherwise, the ftp command fails.

Flags

The following flags can be entered on the shell command

line:

-d Enables debugging by turning on the logging feature. See the debug subcommand.

-g Disables the expansion of metacharacters in file names. Interpreting metacharacters can be referred to as expanding (sometimes called globbing) a file name. See the glob subcommand.

-i Turns off interactive prompting during multiple file transfers. See the prompt, mget, mput, and mdelete subcommands for descriptions of prompting during multiple file transfers.

-n Prevents an automatic login on the initial connection. Otherwise, the ftp command searches for a $HOME/.netrc entry that describes the login and initialization process for the remote host. See the user subcommand.

-v Displays all the responses from the remote server and provides data transfer statistics. This is the default display mode when the output of the ftp command is to a device, such as the console or a display. However, if output is redirected, such as to a file, or if the ftp command is started by a daemon, such as the cron daemon, verbose mode is not in effect unless the -v flag or the verbose subcommand is used.

Parameter

HostName The name of the host machine to which files will be transferred.

Subcommands

If you execute the ftp command and do not specify the Host-Name of a remote host, the ftp command immediately displays the ftp> prompt and waits for an ftp subcommand. To connect to a remote host, you then execute the open subcommand. When the ftp command connects to the remote host, the ftp command then prompts for the login name and password before displaying the ftp> prompt again. The ftp command fails if no password is defined at the remote host for the login name.

The ftp command interpreter, which handles all subcommands

entered at the ftp> prompt, provides facilities that are not available with most file-transfer programs, such as:

* Handling file-name parameters to ftp subcommands.

* Collecting a group of subcommands into a single subcommand macro.

* Loading macros from a $HOME/.netrc file.

These facilities help simplify repetitive tasks and allow you to use the ftp command in unattended mode.

The command interpreter handles file-name parameters according to the following rules:

* If a - (hyphen) is specified for the parameter, standard input is used for read operations and standard output is used for write operations.

* If the preceding check does not apply and file-name expansion is enabled (see the -g command or the glob subcommand), the interpreter expands the file name according to the rules of the C shell. When globbing is enabled and a pattern-matching character is used in a subcommand that expects a single file name, results may be different than expected.

For example, the append and put subcommands perform file-name expansion and then use only the first file name generated. Other ftp subcommands, such as cd, delete, get, mkdir, rename, and rmdir, do not perform file-name expansion and take the pattern-matching characters literally.

* For the get, put, mget, and mput subcommands, the interpreter has the ability to translate and map between different local and remote file-name syntax styles (see the case, ntrans, and nmap subcommands) and the ability to modify a local file name if it is not unique (see the runique subcommand). Additionally, the ftp command can send instructions to a remote ftpd server to modify a remote file name if it is not unique (see the sunique subcommand).

Note: The ftp command interpreter does not support pipes.

To end an ftp session when you are running interactively, use the quit or bye subcommand or the End of File (Ctrl-D) key sequence at the ftp> prompt. To end a file transfer before it has completed, press the Interrupt key sequence. The default Interrupt key sequence is Ctrl-C. The stty command can be used to redefine this key sequence.

Transfers being sent (from the local host to the remote host) are normally halted immediately. Transfers being received (from the remote host to the local host) are halted by sending an FTP ABOR instruction to the remote FTP server and discarding all incoming file transfer packets until the remote server stops sending them. If the remote server does not support the ABOR instruction, the ftp> prompt will not be displayed until the remote server has sent all of the requested file. Additionally, if the remote server does something unexpected, the local ftp process may need to be ended manually.

List of Subcommands

.....

Examples

.....

Implementation Specifics

This command is part of the TCP/IP Facility in Network Facilities of AIX for RISC System/6000

Files

$HOME/.netrc Specifies automatic login information for the ftp command and the rexec command.

/usr/bin/ftp Command executable file.

/usr/lpp/tcpip/samples/.netrc Sample .netrc file.

Suggested Reading

Prerequisite Information

Network Overview .

Related Information

How to Copy Files Using the ftp Command.

The csh command, rcp command, tftp command, stty command.
How to Copy Files Using the ftp Command

.....

Suggested Reading

Prerequisite Information

Glossary Terms: host, local host, permissions, remote host.

The cd command, dir command.

Related Information

The ftp command, rcp command.

The ftpd daemon.

## 7.5 Man page TSOCK

```
TSOCK(8)

NAME
     tsock - UltraNet socket-based network exerciser

SYNOPSIS
     tsock     [-r] [-t] [-s] [-lbuflen] [-nnbufs] [-mnby]
          [-?l] [-?s] [-?r] [-pport] [-B] [-d] [-cfreq] [-Pxpat]
          [-Ffile] [-aabufs] [-f] [-utpdu] [-xtimes] [-ztimes]
          [host]

DESCRIPTION
     tsock is a program which uses the UltraNet socket emulation
     library to perform numerous network exercises.  It may be
     used for a variety of reasons, including timings, connec-
     tivity testing, fault diagnosis, software checkout, and so
     forth.
     The program is a derivative of the public-domain ttcp
     program developed by T.C. Slattery, USNA, as improved by
     Mike Muuss, BRL.

OPTIONS
     -r   Operate as receiver.

     -t   Operate as transmitter.

     -s   Operate in source/sink mode, in which the transmitter
          sends a standard pattern of internally-generated data
          and the receiver discards all data.  This avoids disk
          I/O (useful for timings) and allows validation of
          incoming data (see the -c option below). Alternate
          patterns can also be specified (see the -P and -F
          options below).

     -lbuflen
          Use a transmit or receive buffer of buflen bytes
          [default: 1024].  If a 'K' (or 'k') is appended to
          buflen the value is multiplied by 1024; an 'M' (or
          'm') multiplies by 1048576; a 'G' (or 'g') multiplies
          by 1073741824.

     -nnbufs
```

Transmit a total of nbufs messages [default: 1024]. Applicable for transmitter only; ignored for receiver.

-mnby
Mark the progress of the transfer by printing an "X" for every nby bytes sent or received [default: 10240]. A multiplication character may be appended as for buflen. Does not work well in conjunction with the -x or -z options.

-?l Use a random-length buffer ranging from one byte to buflen. Applicable only in source/sink mode and, if receiver, for nondatagrams only; ignored otherwise.

-?s Use a random starting byte location within the buffer (that is, a random byte boundary). Applicable only in source/sink mode; ignored otherwise.

-?r Do not seed the random number generator. Used to produce repeatable "random" runs.

-pport
Use port number port for the connection or datagram path [default: 2000]. Port numbers are 16 bits, and there is a BSD convention that port numbers under 1024 are usable only by privileged programs; therefore, port must be between 1024 and 65535.

-B Output only full buflen-sized blocks (for TAR). Applicable for receiver only; ignored for transmitter.

-d Use datagrams (ISO Connectionless Transport Service).

-cfreq
Check every freq incoming message for correctness; if freq is omitted every message is checked. Applicable only in source/sink mode and for the receiver side only; ignored otherwise. This option will significantly slow down the transfer, depending upon the value of freq.

-Pxpat
Use an alternate pattern as specified by the hexadecimal string xpat. If no xpat is specified, an alternate cyclical pattern is used (the default pattern runs from 0x20 through 0x7e repetitively;

the alternate pattern runs from 0x00 through 0xff). Requires -s, and must be specified on both the receiver and the transmitter if -c is also specified. If xpat is specified, data checking (-c) will work only if the transmitter and receiver have identical buffer sizes and no -? options are used. The xpat string must contain an even number of hexadecimal digits (0-9, a-f, or A-F) only; it does not start with 0x.

-Ffile
Use an alternate pattern from the specified file. Requires -s, and must be specified on both the receiver and the transmitter if -c is also specified. Data checking (-c) will work only if the transmitter and receiver have identical buffer sizes and no -? options are used.

-aabufs
Use asynchronous I/O with abufs simultaneous operations [no default; abufs must be specified]. Applicable only on systems which support asynchronous I/O, such as Cray UNICOS; ignored otherwise.

-f Use fake data mode, in which the adapter or adapter interface generates or discards the data instead of accessing host memory.
Applicable only in source/sink mode and with appropriate hardware installed.

-utpdu
Sets maximum TPDU size to 2^tpdu bytes [default is configuration-dependent, usually 15 (32Kbyte TPDUs)]. May be specified for either the transmitter or the receiver; if specified for both, the minimum of the two sizes will be used.

-xtimes
Run the command times times in parallel by forking. For tsock -t, this will multiply the number of bytes transmitted by times; for tsock -r, each receiver process will receive approximately 1/times of the bytes transmitted. (Approximate because there is no coordination among the receivers; they are all in a big free-for-all fight for messages.) This is not the same as using multiple tsock commands, because

all participants share the same port numbers (that is, share the same connection or datagram path). This option is obviously most useful in source/sink mode (due to the free-for-all nature of both transmitting and receiving).

-ztimes
Run the command times times in parallel by forking (same as -x) except that the forked processes will alternate transmitter and receiver. For example, -z4 results in two transmitters and two receivers. This allows operating a connection or a datagram path in true full duplex (data flowing simultaneously in both directions). Several unique considerations must be kept in mind when using -z; see USAGE below.

host The local (receiver) or remote (transmitter) host name to use for the connection or datagram path; required for transmitter, optional for receiver.

USAGE

The receiver command (tsock -r) should be issued first, on the receiving system, followed by the transmitter command (tsock -t) on the sending system. The receiver takes the passive role, and either listens for an incoming connection or issues a recvfrom() for an incoming datagram; the transmitter takes the active role, and either initiates a connection or starts sending datagrams. Once the transmitter has sent the specified number of messages (nbufs) it either closes the connection or sends a special "end of transmission" datagram. Both the transmitter and receiver then generate detailed timing information on stderr (the use of stderr allows the receiver to redirect stdout to a file).

Since message boundaries are significant for ISO Connection-Oriented Transport Service (unlike sockets), each message which is larger than the receive buffer will satisfy more than one read(), but each message that is smaller than the receive buffer will satisfy a separate read(). For example, assume a receive buffer of 1024 bytes. If the transmitter were to send a 2000-byte message (either because buflen were 2000 or by random choice), then the receiver's first read() would return 1024 bytes and its second read() would return 976 bytes. If the transmitter were to send a 5-byte message, the

1024-byte read() would return only the five bytes.

The maximum size for an UltraNet datagram is one TPDU (normally 32768 bytes), and datagrams are truncated rather than continued. In the first example above (receiver buflen of 1024, transmitter sends 2000 bytes), only the first 1024 bytes of the message (datagram) would be delivered; the remaining 976 bytes would be discarded.

An alternate pattern file (specified with the -F option) is an ASCII file containing a hexadecimal representation of the pattern to be repeated througout each message; the file must be less than 1000 characters in length. Each valid hexadecimal character in the file (0-9, a-f, A-F) is converted into binary and included in the pattern, and all other characters are ignored (to allow inclusion of tabs, spaces, carriage returns, and so forth). There must also be an even number of hexadecimal characters (the pattern is a string of bytes, not nibbles). For example, each of the following files would result in messages consisting of the 16 hexadecimal digits repeating throughout:

```
0123456789abcdef

01 23 45 67 89 AB CD EF

01/23  45/67    89/ab  cd/ef

"01234567"
"89abcdef"
```

When using the -z option, values for nbufs must be specified on both commands, since otherwise the transmitters and receivers will not know when to terminate; these may be specified either with the -n option or by using the default. Also, care must be taken to balance the total number of transmissions and receptions. Note that this does not mean simply that the two values of nbufs must be the same. For one thing, nbufs is essentially multiplied by the number of processes of each type; this is particularly bothersome when times is odd or is different on the two commands. Also, a single transmission may satisfy several receives (e.g., one 3000-byte transmit will satisfy four 800-byte receives); this is significant when buflen is different on the two

commands. Because of this, -z used with random-length buffers (-?l) will generally not work correctly (it will transfer data, but will generally not terminate). Finally, note that use of -z does makes it meaningful to specify -c along with -t.

EXAMPLES

The following pair of commands will send 1024 canned messages of 1024 bytes each (the default values) from host mulberry to host juniper.

```
on juniper:  tsock -r -s juniper

on mulberry: tsock -t -s juniper
```

The following pair of commands will send 100000 random-sized messages (ranging from 1 byte to 50000 bytes) from host mulberry to host juniper, where the messages will be received in a randomly-located, random-sized buffer (ranging from 1 byte to 10000 bytes) and will be checked for correctness. It will also mark every 100000 bytes received, and will use a maximum TPDU size of 8192 bytes. Note the use of backslash to bypass shell interpretation of the `?'.

```
on juniper:  tsock -r -s -l10000 -\?l -\?s -c i
                 -m100000 -u13 juniper

on mulberry: tsock -t -s -l50000 -\?l -n100000 juniper
```

The following pair of commands will send the file testdata from host mulberry to host juniper, using 256-byte messages.

```
on juniper:  tsock -r -l256 juniper >testdata

on mulberry: tsock -t -l256 juniper <testdata
```

The following pair of commands will send 1024 datagrams of 20K (20480) bytes each from host mulberry to port 5555 on host juniper, where they will be checked for correctness. In addition, an "X" will be printed on mulberry for every ten datagrams sent, and on juniper for every ten datagrams received.

```
on juniper:  tsock -r -s -d -p5555 -l20k -m200k
```

```
-c juniper
```

```
on mulberry: tsock -t -s -d -p5555 -l20K -m200K
juniper
```

The following pair of commands will send 1024 random-length datagrams of up to 20000 bytes each from host mulberry to host juniper. Note that since the receive buffer is only 10000 bytes long, any datagrams longer than 10000 bytes will be truncated (this should be about half the datagrams if the random number generator is reasonable).

```
on juniper:  tsock -r -s -d -l10000 juniper
```

```
on mulberry: tsock -t -s -d -l20000 -\?l juniper
```

The following pair of commands will send 100 1024-byte messages from each of five parallel transmitters to four parallel receivers, using a single connection. There will be a total of 512000 bytes transmitted (default length of 1024 times -n100 times -x5), although each of the four receivers will probably not receive exactly one-fourth of them; the total bytes received will be 512000.

```
on juniper:  tsock -r -s -x4 juniper
```

```
on mulberry: tsock -t -s -n100 -x5 juniper
```

The following pair of commands will result in 200 1024-byte messages being sent in each direction between juniper and mulberry (400 messages total), using a single connection. The data will use an alternate pattern and will be checked on both ends.

```
on juniper:  tsock -r -s -c -P0000ffff -n100 -z4 juniper
```

```
on mulberry: tsock -t -s -c -P0000ffff -n100 -z4 juniper
```

The following commands are identical to the previous pair, but use the alternate pattern specified in the file alt-pat4.

```
on juniper:  tsock -r -s -c -Falt-pat4 -n100 -z4 juniper
```

```
on mulberry: tsock -t -s -c -Falt-pat4 -n100 -z4 juniper
```

DIAGNOSTICS

Error messages are self explanatory. Status and timing output is directed to stderr to allow the receiver to redirect stdout into a
file. If invoked with no parameters, tsock prints the following usage summary:

Usage:

```
  tsock [-options] [host] >out
     -r      operate as receiver (passive initiation,
             receive data)
     -t      operate as transmitter (active initiation,
             transmit data)
     -s      source/sink mode (generate and discard a
             data pattern)
     -l##    length of network buffer (default 1024)
     -n##    number of messages to transmit (with -s
             only, default 1024)
     -m##    mark progress with 'X' for every ## bytes
             transferred
     -?l     use random length messages <= -l## value
             (without -d only)
     -?s     use random starting location (i.e., random
             byte boundary)
     -?r     do NOT seed random number generator (for
             repeatability)
     -p##    port number for connection or datagram path
             (default 2000)
     -B      only transmit full blocks, as specified in
             -l## (for TAR)
     -d      use datagrams (with -s only)
     -a##    use asynch network I/O (eg, UNICOS reada)
             with ## buffers
     -f      set 'fake data' option
     -c      check received data (receiver with -s only)
     -u##    set TPDU size to 2^## bytes (eg, 12=4K;
             default 15)
     -x##    run ## processes in parallel (via fork)
     -z##    same as -x## but fork both receivers and
             transmitters
     -Pxpat use alternate pattern hsx 'xpat'
             (default 0x00-0xff)
     -Ffile use alternate pattern from file
     host    host name (optional local for -r, required
```

remote for -t)

BUGS

Until multiple-adapter support is implemented, whenever the current UltraNet configuration has more than one adapter defined, a host parameter must be specified for the receiver. (A host parameter is always required for the transmitter.)
See unetcf(8) for information on configuring an UltraNet.

In source/sink mode using datagrams, the sender produces datagrams as fast as possible, and in general will overrun the ability of the receiver to receive; some datagrams will usually be discarded. This situation is exacerbated by the use of the -c option. Occasionally the special "start of transmission" or "end of transmission" datagrams will be missed, resulting in a hung receiver. This situation is exacerbated by using the -x option on a datagram receiver, which is really not recommended.

The option parser is relatively stupid, and requires that there be no spaces between an option letter and its argument (e.g., -n1000 rather than -n 1000 and -?s rather than -? s).

USMID @(#)man/man8/tsock.8 301.2 09/14/92 15:04:48

## 7.6 Literature


- [ANS-01] HiPPI — Framing Protocol (HiPPI-FP), preliminary draft proposed, ANSI, X3.210–199x, X3T9/89–146, X3T9.3/89–013, Rev 4.3, 24 February 1992
- [ANS-02] HiPPI — Encapsulation of ISO 8802–2 (IEEE Std 802.2) Logical Link Control data Units — (802.2 Link Encapsulation) (HiPPI-LE), working draft proposed, ANSI, X3T9.3/90–119, 29 July 1991
- [ANS-03] HiPPI — Memory Interface (HiPPI-MI), preliminary draft proposed, ANSI, Rev 2.6, 13 March 1992
- [ANS-04] HiPPI — Physical Switch Control (HiPPI-SC), working draft proposed, ANSI, X3.222–199x, X3T9/91–139, X3T9.3/91–023, Rev 2.4, 16 December 1991
- [ANS-05] HiPPI — Mechanical, Electrical and Signalling Protocol Specification (HiPPI-PH), preliminary draft proposed, ANSI, X3T9/88–127, X3T9.3/88–023, Rev 8.1, 24 June 1991
- [COM-01] D.E.Comer, —Internetworking with TCP/IP, Vol. I, Second Edition, Prentice-Hall International Editions, 0–13–470188–7
- [COM-02] D.E.Comer, —Internetworking with TCP/IP, Vol. II, First Edition, Prentice-Hall International Editions, 0–13–465378–5
- [CRA-01] Cray's Perspective on the HiPPI Standard, John K. Renwick, CRAY Research Inc. Eagan, Minnesota
- [CRA-02] IP and ARP on HiPPI, Internet-Draft, Internet Engineering Task Force, Network Working Group, J.Renwick, A.Nicholson, Cray Research Inc., February 1992
- [IBM-01] Program Directory for use with IBM HiPPI SCSE for MVS/ESA, Version 1, Release 2, Modification Level 0, Program 5799–DKW, Feature 5417/5418/5419, 27 March 1991
- [IBM-02] HiPPI — User's Guide and Programmer's Reference, X3.183–1991, SA23–0369–03, IBM, February 1992
- [IBM-03] IBM, February 1992, HiPPI — HiPPI and Fiber Channel Standard — FCS, Reiner H. Philipp, IBM-ECAN Heidelberg, EMEA HiPPI Class Heidelberg, October 1992
- [IBM-04] IBM Visualization Solutions for scientific and Engineering Problems, Dr. David Watson, IBM UK Scientific Centre, Winchester, UK, 1992
- [IBM-05] IBM — Power Visualization System, Dr. David Watson, IBM UK Scientific Centre, Winchester, UK, December 1991
- [IBM-06] HiPPI — The 100 Megabyte/sec ANSI Standard, Henry R. Brandt, IBM ES Kingston, New York, May 1992
- [IBM-07] HiPPI Hardware for the 3090J and ES/9000 Processors, Ronald A. Linton, High Parallel Supercomputing Systems Laboratory, IBM Kingston, New York, May 1992
- [IBM-08] S/390 HiPPI Performance, Robert Seidel, IBM ES Kingston, New York, May 1992

- [KFA-01] KFAnet/INTERNET — Performance-Test der IBM-RS6000–32H als FDDI-BMPX-Router, W.Anrath, R.Niederberger, First Edition, July 1992, KFA-ZAM-IB-9208,
- [KFA-02] KFAnet/INTERNET — Experiences with IBM-RS6000 as FDDI gateway to IBM mainframes, R.Niederberger, First Edition, November 1993, KFA-ZAM-IB-9311,
- [MVS-01] IBM TCP/IP Version 2 Release 2.1 for MVS: Planning and Customization, Third Edition, September 1992, SC31–6085–2
- [MVS-02] IBM TCP/IP Version 2 Release 2.1 for MVS: Installation and Maintenance, First Edition, March 1991, SC31–6085–0
- [MVS-03] IBM TCP/IP Version 2 Release 2.1 for MVS: Programmer's Reference, Third Edition, September 1992, SC31–6087–2
- [MVS-04] IBM TCP/IP Version 2 Release 2.1 for MVS: User's Guide, Third Edition, September 1992, SC31–6088–2
- [MVS-05] IBM TCP/IP Version 2 Release 2.1 for MVS: Messages and Codes, Third Edition, September 1992, SC31–6142–02
- [ULT-01] Network Addressing Manual, Part Number 06–0020–001, Revision B, Ultra Network Technologies, 18 November 1991
- [ULT-02] Overview of Products for IBM Computers, Part Number 06–0009–001, Revision B, Ultra Network Technologies, 24 April 1990
- [ULT-03] UtraNet IBM Host Software Release Notes, Release 3.80.05 MVS Versions, Ultra Network Technologies, 15 November 1991
- [ULT-04] Host Software Installation Guide for MVS Systems, Part Number 06–0037–001, Revision A, Ultra Network Technologies, 4 November 1991