DS3100 and DS5400,

a performance survey

Steinar Kjaernsroed
Dpt. of Informatics, University of Oslo

November 1989

Institutt for Informatikk

Contents

1 Abstract **2**

2 Background **2**

3 Configuration **3**

4 Booting a DS3100 from a Sun 3/280 **3**

5 Benchmarks and "real-life" experiences **7**
5.1 DS3100 CPU performance 7
5.2 Process management 7
5.3 Memory management 8
5.4 Local disk I/O 9
5.4.1 Mixed read/write, single drive, single file-system 10
5.4.2 Read only, multiple drives 10
5.4.3 Conclusion on local disk i/o 15
5.5 Disk I/O through the network 15
5.6 TCP/UDP/IP implementation 17
5.6.1 Original Ultrix implementation 18
5.6.2 TCP implementation based on Van Jacobson's algorithm 18
5.6.3 Conclusion on TCP/UDP 19

6 Summary **19**

1 Abstract

This report explores some *possible* malfunction of software (Ultrix 3.1) and hardware components of the DecStation 3100 and DecSystem 5400. Some of the results in this report are readily reproducible, others are not. The conclusions and problem diagnosis herein are solely our own, and based on simulations and benchmarks as well "real-life" use.

The report is first of all meant as an extensive bug-report. Since some of the problems pin-pointed are fairly serious from a user's point of view, we would like to make this report available to the public. This will eventually be done with Digital's blessing after they've had the chance to investigate some of the things in more detail. We feel that it is most important for the DS user-community to get around these problems to really be able to benefit from the strength and nice things about the DS family of workstations and file servers.

This survey and bench-marking was done at Dpt. of Informatics, University of Oslo by members of the system staff. All of us can be reached by email if anyone should be interested in getting in touch with us.

Anders Ellefsrud	Rein Tollevik	Leif Arne Neset
`anders@ifi.uio.no`	`rein@ifi.uio.no`	`leifa@ifi.uio.no`
Jens Thomassen	Steinar Kjaernsroed	Tore Solvar Karlsen
`jens@ifi.uio.no`	`steinar@ifi.uio.no`	`tore@ifi.uio.no`

2 Background

Dpt. of Informatics, University of Oslo is operating a large [1]heterogeneous network of Unix workstations and file servers. Until recently the network was dominated by the Sun family of workstations. At the beginning of the summer we did extensive testing of both the new products from Sun (Sun 3/80 and SparcStation 1) and Dec, and chose to buy 35 DS3100 because of good price/CPU-performance figures and a really good implementation of the X11 server. We already knew Ultrix as one of the "closest-to-SunOS" Unix'es in the market, and expected the integration of DS3100s in the network to be straight forward and simple.

However, we had one major obstacle to get around before we would do the purchase:

[1] more than 200 network nodes

All our Sun workstations are running disk-less, booting from and swapping to their nearby file server. We had no intention of buying local SCSI disks to the DSes either, so they had to be able run disk-less too. Moreover, we wanted the DSes to be able to use a Sun 3/280 as a file server! We also considered buying a DS5400 as a combined compute server and file server, but if we could boot from a Sun we would get the freedom to place DecStations *anywhere* around campus, not only on the DS5400 client ethernet segment.

The problem was solved and we are now booting DecStations from Sun 3/280s — probably as the one and only place in the world! The rest of this report explores what had to be done to be able to do this and what kind of bugs and misfeatures we discovered on our way and afterwards.

3 Configuration

Our departmental LAN is set up quite traditionally — a backbone ethernet extended with fiber optics and with some ethernet devices (bridges, routers and a couple of terminal servers) connected to it. All the file servers are connected to the backbone and act as ethernet gateways and IP routers (see figure 1), serving disk-less clients on one ethernet interface and routing cross ethernet traffic through the other.

We have 12 Sun 3/280, each with 8Mb RAM and one Xylogics XY753 disk-controller and two Fujitsu disks. The two Fujitsu disks are different and have slightly different specs, one MK2344 (SMD) and one MK2382 (ESMD). These machines serve more than 150 disk-less Sun 3/50 and Sun 3/60, all upgraded to 8Mb RAM. We also have one VS3200 with an Emulex QD32 disk controller and 2 Fujitsu MK2344 disks. This machine provides boot and swap services for 5 VS3100. At last, and not to forget(!), we have 35 *disk-less* DecStation 3100s. They mingle with the Sun workstations on the local ethernet segments.

All users are running X10 or X11. Prior to our memory upgrading, the use of these windowing systems (X11 in special) caused fairly high swapping activity on the client ethernet segments. Today the swapping is of no major burden to the network throughput.

4 Booting a DS3100 from a Sun 3/280

When powering up a disk-less workstation, the only thing the workstation knows about itself is its ethernet address. This is the ticket the machine can use to get remote boot services from a boot server in the network.

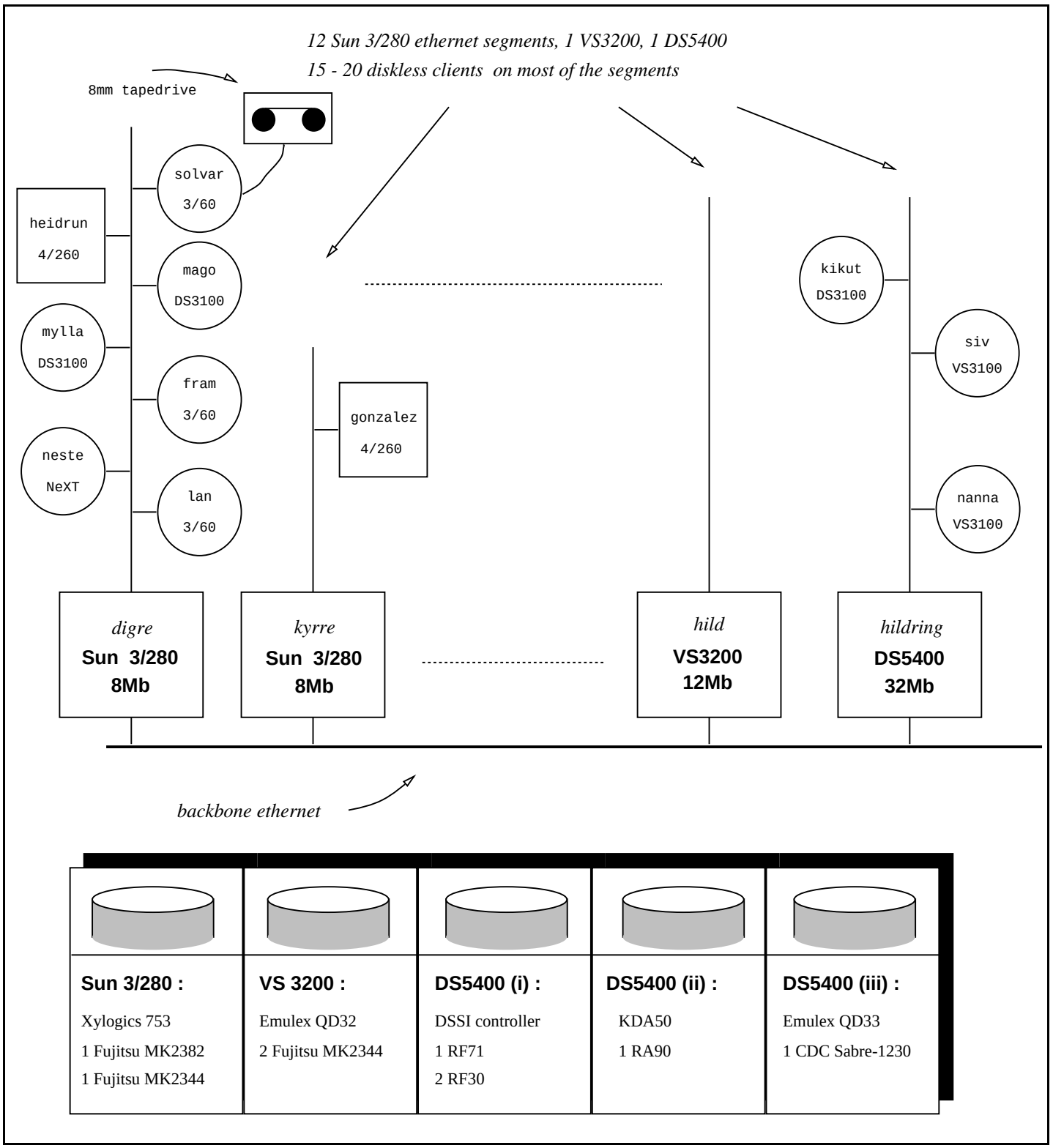

Figure 1: *File Servers and workstations involved in the benchmarks*

The booting procedure consists of three phases:

1. the primary loader

   The ws has to know its Internet address, its hostname, the name and address of its boot server and the name of the bootstrap program. A request for booting services is broadcasted on the local ethernet by the disk-less workstation and caught by the appropriate boot server. A reply packet containing the information is sent back to the workstation.

2. the secondary loader

   This phase downloads the bootstrap program from the boot server.

3. the tertiary loader

   This is the execution of the bootstrap. The bootstrap gets vmunix and some other pertinent information [2]. After the bootstrap, vmunix is loaded and the root filesystem and swap area is mounted by RPC/NFS calls.

Our disk-less Sun workstations use the *bootparam* protocol as the primary loader. Since the DS3100 boot PROM doesn't support this protocol, our approach was to try to use the *bootp* protocol which was supported by the DS boot PROM.

The most common protocol used for the secondary loader is the TFTP[3] protocol. This was also supported by the DS boot PROM.

The bootstrap program itself also uses TFTP to get vmunix from the remote host. In addition, the bootstrap retrieves a file called netblk.o from the server. This file contains information such as the name of the server, the name of the disk-less client, the IP netmask and broadcast, swap size, where to find the swapping area on the remote host and where to find the root file system.

Before actually trying to boot a disk-less DS from a Sun 3/280, a couple of things has to be done on the Sun:

- Configuring swap space and root file-system for DS clients

  This is rather trivial. Disk-Less Sun clients have their root file system located on /export/root/name-of-client on the server and the swap file on /export/swap/name-of-client. For DS clients we choose to put the root filesystem in /export/root/mips/name-of-DSclient. The DS swap files resides on the same area as the Sun swap files.

[2] for the DS3100 this is the file netblk.o
[3] TFTP: Trivial File Transfer Protocol

- Configuring the *bootp* daemon

  As mentioned, disk-less Sun clients use the *bootparam* protocol. The bootparam daemon uses three Yellow Pages maps to get the information needed in the initial phase of the boot process. These are the maps *'ethers'*, *'hosts'* and *'bootparams'*. To be able to maintain all this information in only one place, we patched the bootp daemon to use these maps as well. An entry in the bootparam map may look like this

  ```
  brenna root=/export/root/mips/brenna boot=/tftpboot/bootpload.mips
  ```

  bootpload.mips is the name of the bootstrap.

- Create the file netblk.o

  Digital supplies a program called netblk.c for creating the netblk.o file on a DS. This program was undocumented, but *Jens Thomassen* was able to use this program as a starting point for creating a program called gennetblk.c which can be run on the Sun boot servers.

It was exciting to try out the booting for the first time! The two first phases of the load passed smoothly, but when the DS client tried to mount the root and swap area, everything crashed. The information on where to find the root and swap area on the server was apparently wrong, so we went through all the configuration once more but couldn't find anything. It was rather unclear what actually happened, so we put up an ethernet "sniffer" between the server and DS client and started to unwrap all the packets to see what was going on. After unwrapping tons of ethernet packets, *Jens Thomassen* spotted the problem. Here is the scenario:

When the DS issues NFS mount requests for its root file system and swap file, it gets a so called *file handle* back from the NFS server. According to the NFS protocol, this file handle is an opaque 32 bytes handle which should be used in all succeeding calls. However, some Ultrix programmer decided to extract only the Ultrix-significant part of this handle and reconstruct the NFS file handle from these bytes every time it was going to be used. As a result, when sent back to the SUN the next time, this handle contained null bytes everywhere except from the 10 bytes of "Ultrix relevant" data. What Ultrix does here, is an obvious violation of the protocol. It is the server's responsibility to construct the handle — clients should not do anything with this but use it! Even if the handle was propagated properly through the network layers by RPC and XDR serializing/deserializing, the server wasn't able to recognize this handle since the SUN used another interpretation of the 32 bytes of opaque data.

However, it was rather trivial for Digital to fix this. This was done in Valbonne in France by the Norwegian Digital engineer *Jan Erik Pedersen* who also helped and advised us a lot before we started with the work.

The next version of Ultrix will probably have this fix installed, so go on mix disk-less DSes and SUNs in your network ..

5 Benchmarks and "real-life" experiences

Dpt. of Informatics has benchmarked several types of machines during the last few years, for instance many of the workstations and file servers from Sun, Digital, Apollo, HP and IBM. We have developed our own suite of simple benchmark programs, covering areas such as local disk i/o, floating point performance, process management and the implementation of the IP protcol suite. Not all of these benchmarks were run on DS5400 and DS3100 at this time, mainly because we already knew the pertinent performance figures but also just because this was not an issue this time.

In the discussion to follow, sometimes DS3100 is the machine being tested, sometimes DS5400 is. This is indicated in the text. However, sometimes *Ultrix* is the real candidate in the test. Hopefully, this becomes clear from the context.

5.1 DS3100 CPU performance

All our CPU and floating point benchmarks on DS3100 were done early in the summer of 1989. The results were promising, and the comparison with for instance Sun 4/260 was more or less as announced by Digital. These benchmarks are not included in this report.

5.2 Process management

Process management involves many things, but *context switching* is probably one of the most important ones.

We have made a very simple program for testing context switching: Two processes are communicating through a pipe and passing a bulk of data between them. Exactly one context switch has to be performed for each transfer. The real time spent on performing 2000 context switches is measured.

When running this test on different machines, one should expect the results to correspond to the relative CPU performance index for each machine. However, some RISC

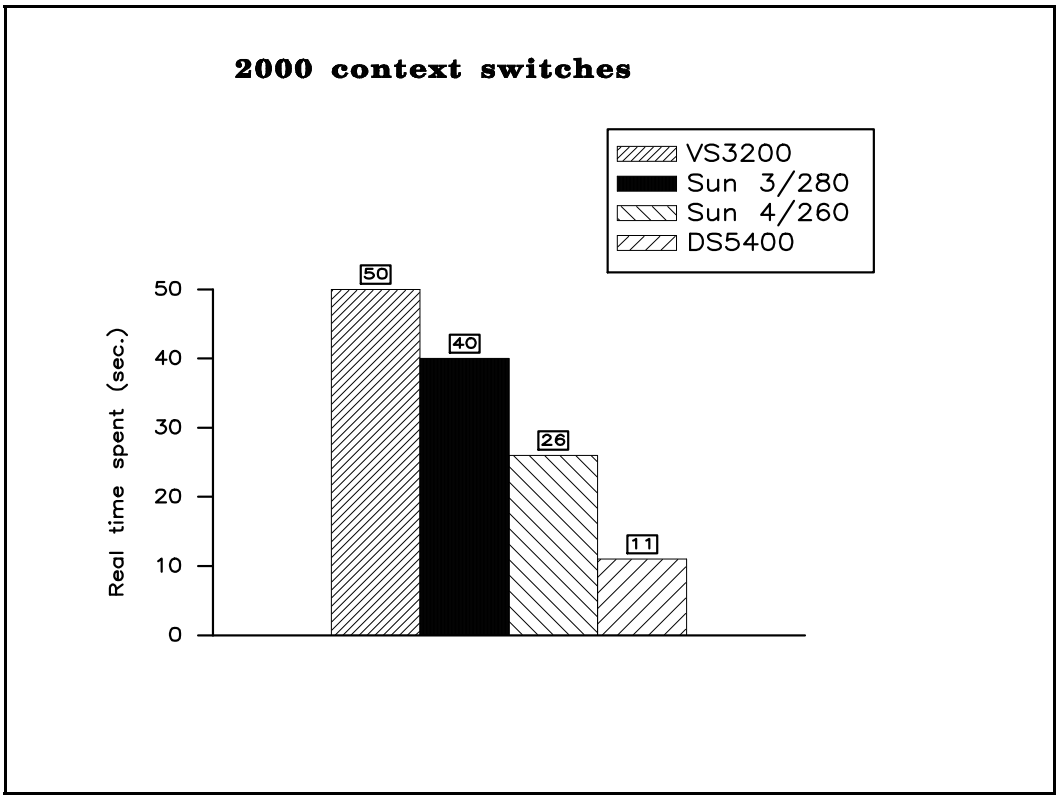

Figure 2: *Real time spent on 2000 context switches*

architectures with a lot of fast registers will probably perform this test even better than their relative performance indexes indicate.

The results can be seen in figure 2. There is nothing much to say about this test, results are as expected.

5.3 Memory management

A lot has been said about the DS3100 and memory usage. On the net, you will occasionally see things like "DS3100 is a memory pig", "Don't run DS3100 disk-less", "If you're running DECWindows, you need at least 12Mb RAM" etc. People also often experience strange crashes, hung situations and incredible slow-downs when the DS runs out of physical memory and starts to swap or page.

As already mentioned, we're violating two of the advises given above: We are running all our DSes disk-less and with 8Mb memory only. But indeed we have experienced and still experience strange things with respect to memory management! The problem is that not all these things are reproducible in a deterministic way. However, *Anders Ellefsrud* has

experimented a lot with his DecStation and has been able to isolate part of the problem:

> *When active virtual memory*[4] *reaches the size of physical memory, the currently running process is deferred for a looooooong time (30 seconds to 5 minutes!). Often the threshold is reached as a consequence of a* fork() *call. In this case the new process is created and reaches the scheduler state "START" before the system dives into the mud!*

The scenario above is easily reconstructed and the observations always hold true.

There are more problems, for instance regarding Ultrix's administration of swap space. Fairly often we are seeing messages like

> pid *nnnn* was killed in xalloc: no swap space

This seems totally un-motivated and happens most often at times when there *should be* more swap space available, i.e. the process activity on the DS is low. Our impression is that Ultrix over time "looses" part of its swap space in one way or another. It's not possible to reproduce this in a *deterministic* way, however, this problem is experienced several times a day here.

Of course it's most important to configure enough swap space for the DSes out on the file server. Our experience is that an 8Mb DS should have at least 32Mb swap space available. Depending on what the users are doing, this may vary a bit from site to site.

> *NB! Please note that everything which has been said about swapping and paging here has to do with DecStations running disk-less. Swapping and paging to local disks has not been tested by us.*

5.4 Local disk I/O

These benchmarks will try to give indications on how well a file server operates. A file server serves disk read/write requests from the network by the NFS protocol. These benchmarks

[4] the parameter 'avm' when running vmstat

does not involve the network part of the i/o transfer, the benchmarks are run locally on the file server and the disks are accessed by local file system calls only.

5.4.1 Mixed read/write, single drive, single file-system

This test is quite simple: A process is creating 200 temporary files on /tmp. For each file the file size is 20K and the block size is 1K so the process is doing 20 write() calls per file. After all files have been created, the files are read back the same way they were written, i.e. 20 read() calls per file with a block size of 1 K. The program was run a number of times on a stand alone machine with minimal load and the results were averaged.

The results can be seen in figure 3. The Y axis shows the real time in seconds spent on all the read/write operations. As can be seen, the Sun 3/280 with the Fujitsu MK2382 and the Xylogics XY753 disk controller beats all the other configurations. The ranking shown is only partially as what could be expected. Both the Sun configuration and the CDC Sabre-1230/Emulex QD33 configuration on the DS5400 has ESMD[5] characteristics. One might probably expect that the powerful R3000 CPU would bring the DS5400 ahead of Sun 3/280, but both machines are probably driving their disks at full speed already. The figures for the RA90/KDA50 configuration come close to that of the CDC/Emulex configuration, as would be expected — the QD33 emulates the MSCP protocol in the KDA50 controller and presents the drive to Ultrix as an RA81 drive. The disappointing and confusing results are those from the DSSI configuration on the DS5400. These results make no sense at all, and it's fair to guess that the DSSI configuration suffers from non-optimized and/or buggy software (device drivers) or hardware. At the time of the benchmark, the /tmp directory was located on a partition on the RF71.

The rest of the disk benchmarks done on the RFxx disk will probably be biased the same way as this benchmark. Read on, please!

5.4.2 Read only, multiple drives

This program access the disk in the "raw" way, that is — the disk partition itself, /dev/..., is opened for read, thereby bypassing some file system mechanics in the kernel. However, this was not the main purpose of the test, only a side effect. The main reason we made this test is that we wanted to see if the DSSI concept and the RF disk technology really give a linear increase in performance when you add new drives to your system. This is the assertion given by Digital. The idea is that since the disk-controller logics reside on the disk itself, you will benefit from having several small disks, each with the disk-controller embedded, instead of having a a small number of large disks, all served by one single disk-controller.

[5] ESMD: Extended SMD

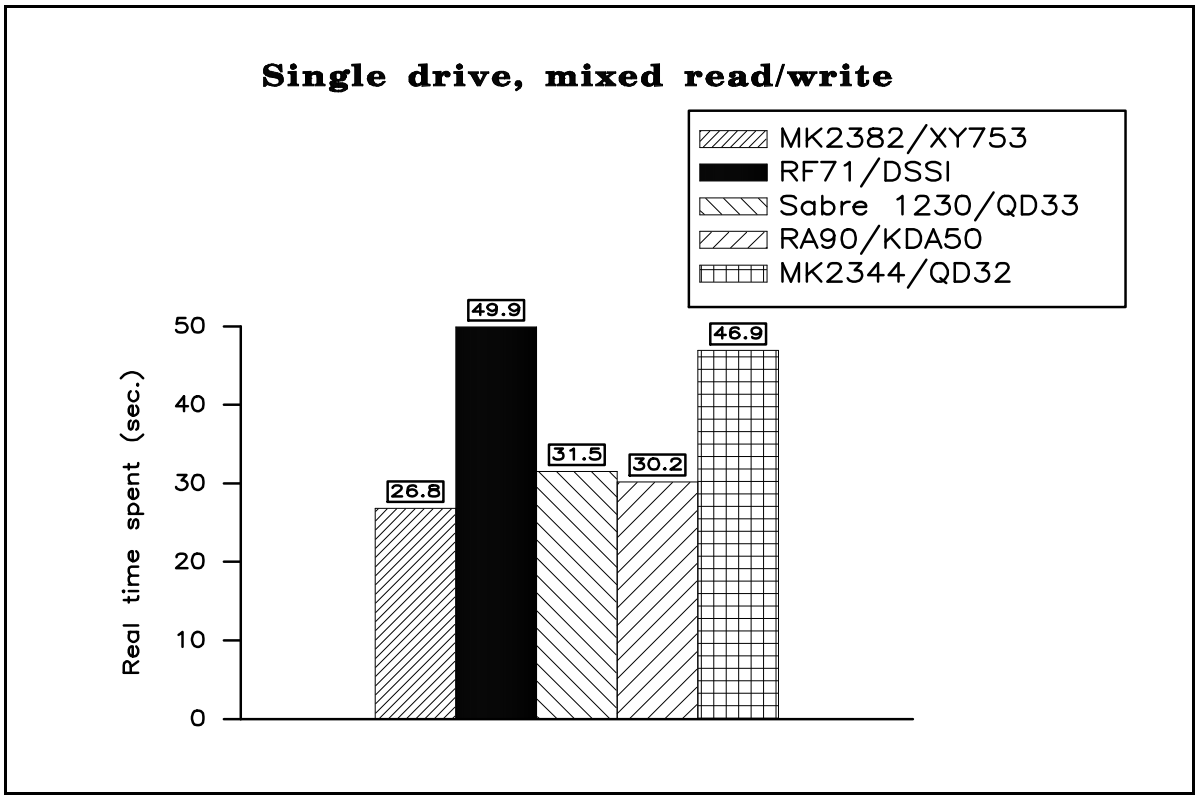

Figure 3: *Mixed read/write, single filesystem*

The test is run a number of times for different combinations of two parameters — the disk partitions to be accessed and the block size for the read requests. When the test is run with more than one partition, the partitions are accessed in a "round-robin" manner. For each read request, the spot on the disk surface to position to is determined by a random lseek() within the span of the partition in question.

The results can be seen in the bar diagram figure 4. The five groups of bars represent the figures for block sizes of $1, 10, 100, 1000$ and 10000 bytes respectively. The Y axis shows the number of served read requests/minute.

No matter how many times we run this benchmark, we were not able to see the linear increase in performance. There are some "discontinuities" and abnormalities to say the least. As can be seen, when accessing the first RF30 the throughput actually drops. When adding the second RF30, the throughput increases to a level slightly above the situation with only one RF71. The average performance increase here is about 3%.

The throughput is independent of block size as long as the block size is less than the optimal (maximal) block size. This maximal block size is 8192 (8K) bytes for the RF disks used in the tests. When using a block size of 10000 bytes, the drive actually has to do two i/o transfers to serve the request.

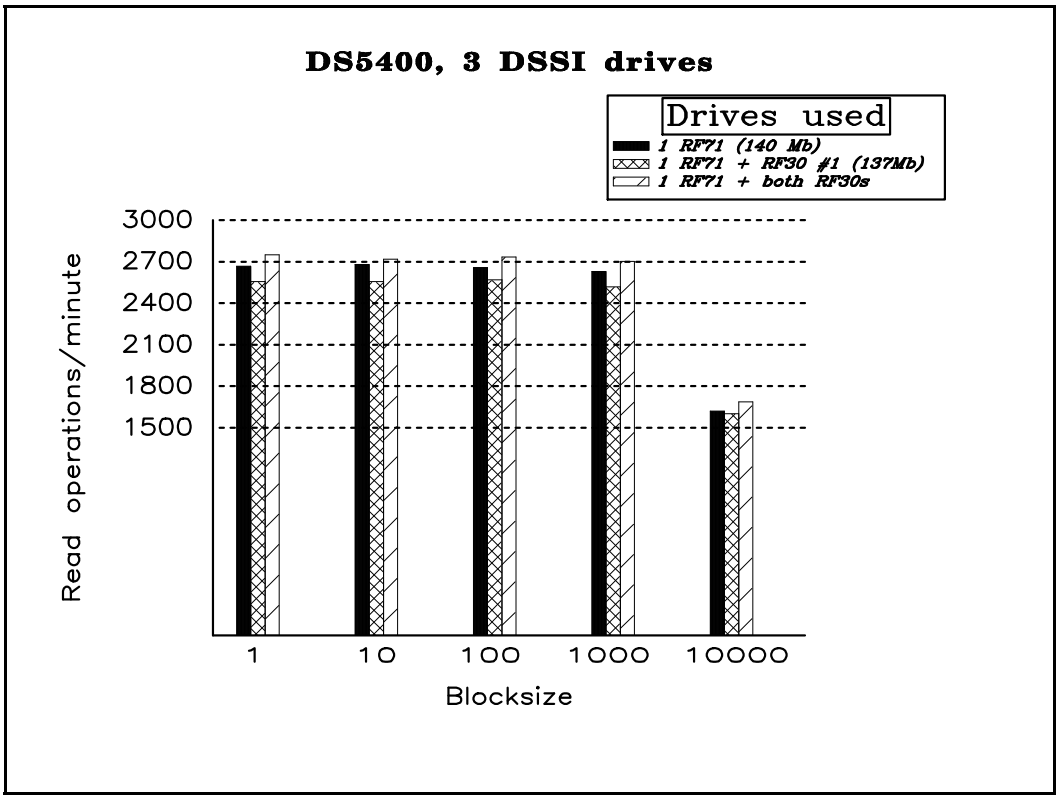

Figure 4: *Multidrive DSSI performance, RF71 + RF30 + RF30*

When trying to explain the abnormalities mentioned above, we changed the order in which the disks were accessed, i.e. we let the program access the RF30 #1 before the RF71. The throughput figures can be seen in figure 5. The behavior is now more "normal", and stems apparently from the fact that only accessing the RF71 partition is faster than only accessing the RF30 partition. The 300Mb RF71 has a bigger surface than the 150Mb RF30, but the RF71 partition in question, `/dev/ra0d`, is of approximately the same size as the RF30 partitions which occupy the whole disk. One might guess that depending on where the /dev/ra0d partition is located physically on the disk surface, the heads may have to traverse a shorter distance on average when this partition is accessed compared to accessing the RF30 partitions where accesses are spread out over the whole disk surface.

Anyway, the conclusion must be that Digital's announcement is "true". But to really be able to benefit from this in a predictable way, you have to use disks of the same size and/or partition the disks and design the filesystem in such a way that all i/o requests spread out evenly on the drives.

To see the impact of different drive characteristics and one controller serving more than one drive, we also ran this test on the Sun 3/280 with it's two different Fujitsu disks and on the VS3200. The results are shown in figure 6 and figure 7. The MK2382 with ESMD characteristics gives excellent performance when accessed alone. Also, the MK2344 delivers good performance alone, the performance decrease is only about $7-8$ % compared to the

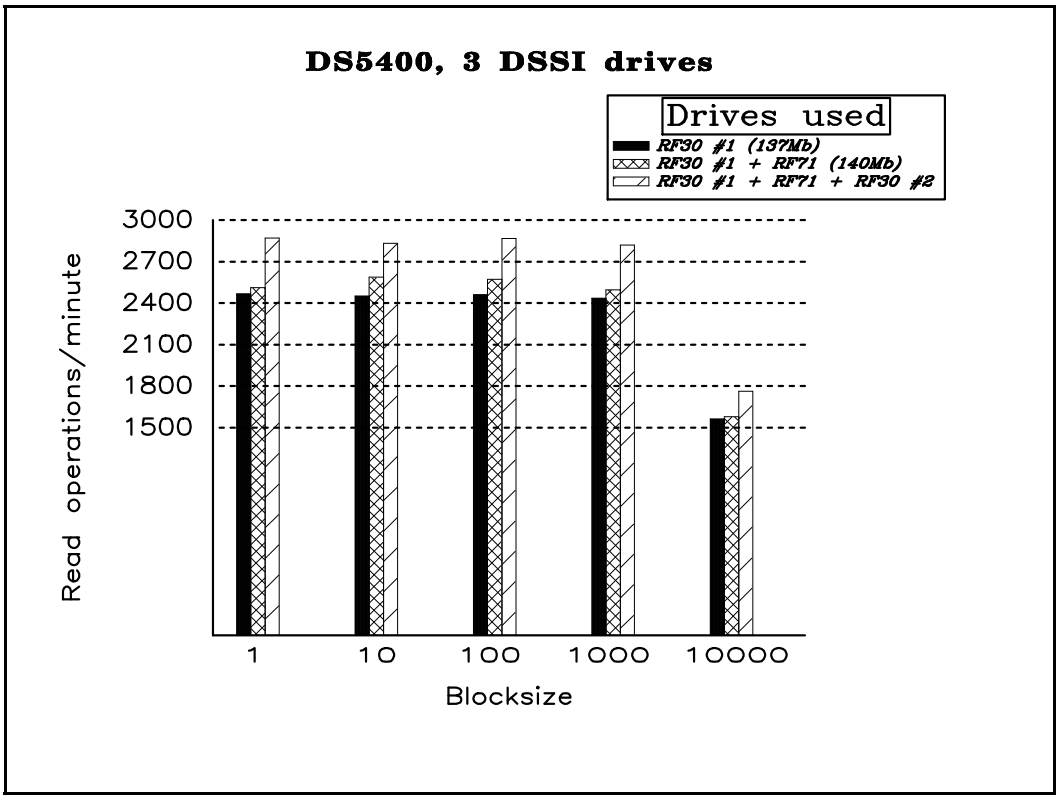

Figure 5: *Multidrive DSSI performance, RF330 + RF71 + RF30*

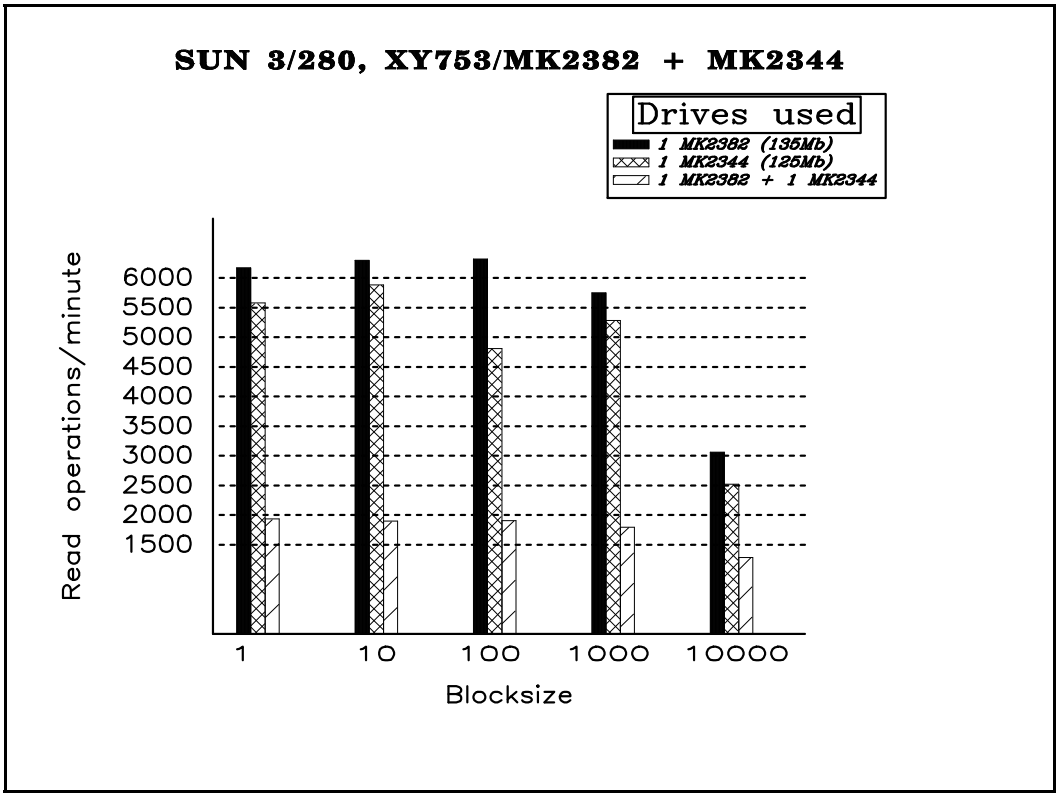

Figure 6: *Multidrive Fujitsu performance on Sun 3/280*

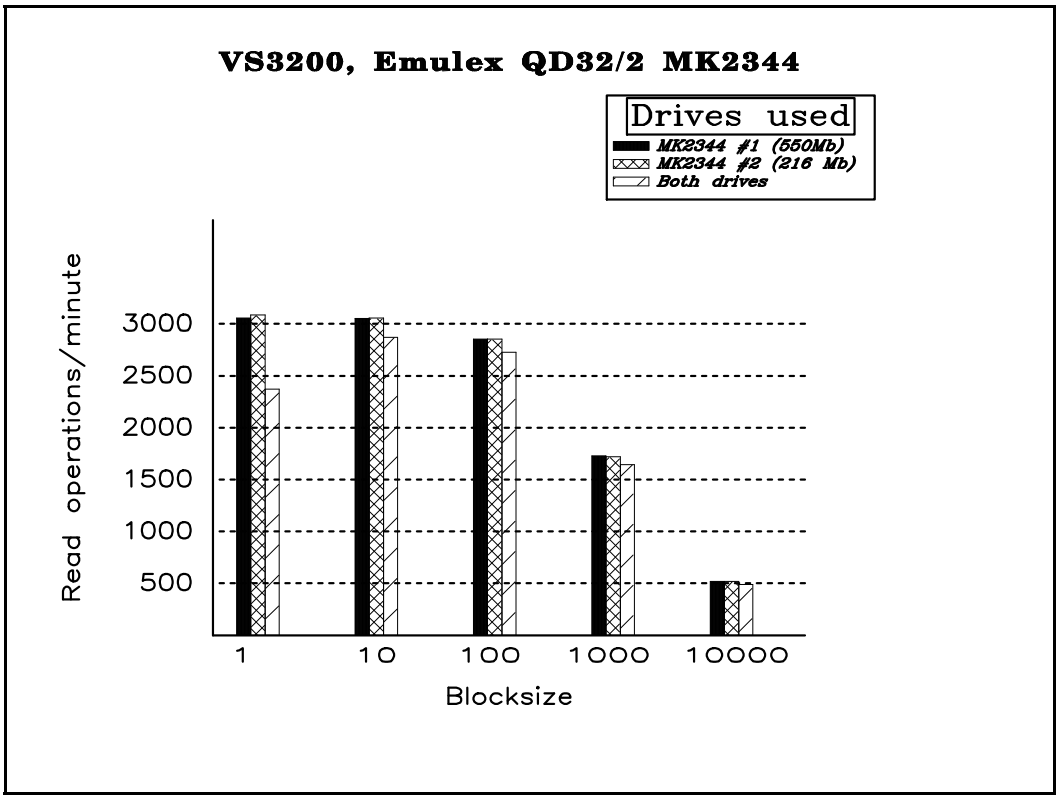

Figure 7: *Multidrive Fujitsu MK2344 performance on VS3200*

MK2382. When accessing the MK2382 and the MK2344 in a "round-robin" manner, the performance seems to drop about 60%! The only explanation to this is that the Xylogics controller uses much time to handle the switching from drive to drive. The XY753 is also known to use a caching mechanism and a "read-ahead" technique on a *per drive basis*. Depending on how effective the randomization algorithm in the test is, one might expect that the performance increase because of this caching is minimal. If so, one might expect an even better single-drive performance in real life because the requests then tend to access more contiguous areas of the disk than this test does. However, when switching between drives the cache is wiped out and new data read in from the new drive. Apparently this is an "expensive" operation for the XY753.

The results for the VS3200 with an Emulex QD32 controller (see figure 7) are also readily explained. The QD32 emulates the KDA50 controller which maintains queues of i/o requests. Requests are queued to the drives in such a way that head movements and drive switchings are minimized. The peak single-drive performance suffers from this, but the decrease in throughput when more than one drive is used is minimal.

5.4.3 Conclusion on local disk i/o

The tests show that Digital's DSSI concept is "nice" in the way that you may get a near linear performance increase when you add new drives to your system. However, you must take care when designing the file system and try to spread out disk accesses evenly to all the drives. It's a rather naive assumption to believe that you're able to design your system this way. In practice, i/o requests tend to arrive in bulks to the controller where most of the requests are to the same drive. A typical example will be when you're dumping a single disk partition to tape or when disk-less workstations are swapping. With the increased program size in a RISC architecture, this becomes even more typical.

Consequently, it's therefore also very important to have good peak performance on single-drive accesses.

Also note that this test was designed just to see if Digital's assertion held true. The test scenario is therefore optimal for the DSSI concept, but to say the least – very non-optimal for the XY753 controller serving multiple drives.

Time will show if Digital will improve the peak performance on one single DSSI drive. If they do, the DSSI configuration on the DS5400 will probably be a good basis for the DS5400 as a file server.

5.5 Disk I/O through the network

In our distributed and disk-less environment, most of the disk i/o requests at any time stems from network requests from a workstation client process. Moreover, most of these requests are issued and served by means of the NFS protocol. It would therefore seem reasonable to develop an NFS benchmark program since running such a benchmark could be a close approximation to the real life situation.

We didn't do this at this time. The NFS protocol is a higher layer protocol and involves several lower layers protocols. Instead we ran a "non-artificial" benchmark for testing network based disk i/o – the rdump program. This program is used for dumping remote filesystems to any backup media on the local host. We are running this program every night for backing up one or more more filesystems out in the network. The dumps are written to high capacity Exabyte 8mm tape drives connected to the SCSI port on a Sun workstation.

The rdump program uses it's own simple protocol[6] on top of TCP/IP and requires

[6] this protocol is called rmt, Remote Magnetic Tape protocol

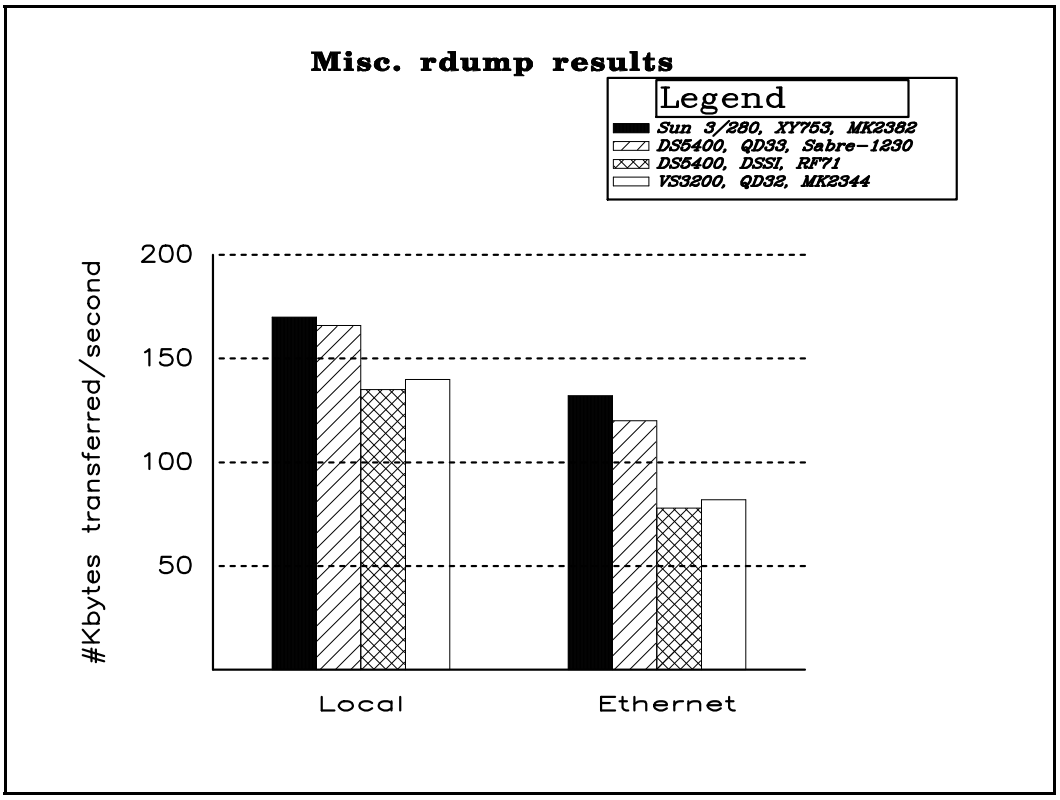

Figure 8: *Disk tests using* rdump

remote login access and thereafter access to all the remote filesystems to be read. There are several versions of rdump around. We are using a public domain version with triple buffering of data, i.e. the process is forking itself into three cooperating processes each with an open network connection. rdump as distributed by Digital does not use this method, but the public domain version was readily compiled on the DSes. The following tests show results from the public domain version only – the /bin/rdump program as distributed by Digital gives a throughput approx. $\frac{1}{3}$ to that of the pd version. The results can be seen in figure 8.

All tests are run at night-time with minimal traffic on the ethernets involved. The leftmost cluster of bars represent rdumps of file systems of comparable sizes on the local hosts to device localhost:/dev/null, that is – files are only read off the local disk and not written to a network tape device, but to /dev/null on the local host. The i/o is done through a socket.

The rightmost cluster of bars involves the same hosts and disk configurations. Here the file systems are written to Exabyte tape drives connected to the Sun 3/60 `solvar`, see figure 1.

In both situations, the Sun 3/280 and the XY753/Fujitsu drive configuration gives best performance. Since the relative difference between the configurations is bigger when file

systems are dumped to the network tape drive, one might be tempted to guess that SunOS has the best and most efficient implementations of the network protocols involved. This is the issue for the next test.

5.6 TCP/UDP/IP implementation

The efficiency of the implementation of the TCP/UDP/IP protocol suite together with higher level application protocols, is probably one of the most crucial points when we're speaking of the overall performance of a networked UNIX workstation or file server. SunOS and Ultrix are usually both known to have good implementations, both of them being derived from the original Berkeley 4.2 and the Berkeley 4.3 implementations.

Today, the most significant difference between the two (SunOS 4.0.3 and Ultrix 3.1) is probably that Sun incorporated the so called *Van Jacobson* TCP in version 4.0.3 of SunOS while Digital still uses the old Berkeley implementation. The Van Jacobson implementation has the ability to dynamically adapt to network load and big round trip times due to long-distance connections. The net effect is that the chances for network congestion to occur are reduced and the overall effective throughput is maximized. On a moderately loaded local ethernet it's reasonable to expect the algorithm to behave well too — however, no performance increase is usually seen.

Note that the Van Jacobson modifications only affect network connections using TCP as transport protocol, *not* UDP based connections.

The program we used for testing TCP and UDP is a public domain program named ttcp written by *Mike Muuss* at the Ballistic Research Laboratory. The program is started in "receive mode" in the receiving end of the connection and in "transmit mode" in the transmitting end. Options and arguments controlling the transport protocol to use, buffer size, name of the remote host, port number etc. can be passed to the program. Statistics are reported in both ends of the connection. We designed a simulation where all hosts mentioned in figure 1 would set up a connection between itself and one by one of the other hosts, including itself, and then run the test. This was done sequentially so that only one connection was active at any time. As can be seen from figure 1, many different parameters are involved:

- source- and destination host type
- # hops between source and destination
- for servers and gateways, type of the ethernet interfaces involved in the connection

We also wanted to see if it was possible to implement Van Jacobson TCP under Ultrix

based on the sources which can be found on uunet.uu.net. *Jens Thomassen* started to work on this and succeeded to build an Ultrix kernel with VJ TCP. The issue for the test then became two-folded — to test TCP and UDP between the hosts AND to see if the VJ implementation under Ultrix gave performance improvements compared to the original Ultrix implementation.

Among the statistics reported by ttcp, we chose the number of Kb transferred per second (*real time*) as the comparision figure in the benchmark. No graphical representations have been created for this test since the matrices with data values are so large. However, there are a number of observations to discuss:

5.6.1 Original Ultrix implementation

- UDP connections from a DS to a Sun seem to overflow the kernel buffers on the Sun. We measured a packet loss of about 90%(!), and the average throughput was about $\frac{30Kb}{sec.}$.
- UDP connections from a Sun to a DS have good throughput when the two hosts are on the same ethernet segment. That is, ttcp reports tansfer rates about $\frac{600Kb}{sec.}$. When one more hop is involved on the net, the performance drops but is still acceptable. The average throughput seems to be roughly $\frac{350Kb}{sec.}$.
- UDP connections between any two DSes and/or VSes give reasonable throughput ($\approx \frac{600Kb}{sec.}$).
- TCP connections from *a Sun or VS or DS* on one of the leftmost client segments in figure 1 to a DS or VS on the DS5400 client segment suffer from extremely bad throughput, i.e. the transfer rate is about $\frac{20Kb}{sec.}$ as opposed to about $\frac{400Kb}{sec.}$ to a DS or VS on the same segment.
- TCP connections from a DS or VS on the DS5400 client segment to *a Sun or DS or VS on any other segment* suffer from the same bad throughput as mentioned above.

5.6.2 TCP implementation based on Van Jacobson's algorithm

We installed a new Ultrix kernel with the Van Jacobson TCP patches on the DS5400, on one of its DS clients and on one DS client on one of the Sun ethernet segments. If we were going to see any changes in performance, we expected only the pertinent TCP connections to be affected. Here is what happened:

- The UDP results are basically the same as before.

- All TCP connections between any two hosts on any segments have reasonable performance ...
- ...to begin with! After some time the performance suddenly drops to about what it was before, that is — less than $\frac{1}{10}$ of what you might expect.
- Also, when the performance drops on the TCP connection between the DS5400 and one of the Suns, this seems to trigger all the other drops.

5.6.3 Conclusion on TCP/UDP

- The UDP overrun by the DS5400 and buffer thrash on the Sun side is probably to be expected. The ttcp benchmark generates UDP packages at higher rates than any application program one might think of. As NFS uses UDP at the transport layer, it would be interesting to design an NFS test to try to see effects of this overrun in a real application. We think that any effects will be minimal, probably limited to some retransmissions on the NFS client side.
- The TCP problems are more serious. The problems have probably nothing to do with "VJ TCP or not VJ TCP". It looks like the problems are more fundamental, and maybe related to some kind of buffer thrashing. There are also strong indications that the problems are related to connections "going through" the DS5400. Hence, *maybe* the problem is limited to only one of the ethernet interfaces on the DS5400. Thus, a good starting point will be two switch ethernet interfaces on the client segment and backbone segment and run the test again. Due to time constraints, this was not done by us. However the rdump tests mentioned in section 5.5 were done using both interfaces. No big difference in i/o throughput was observed.
- The reason that everything looked so nice to begin with after the installing of VJ TCP, is probably the fact that the three machines were rebooted, *not* VJ TCP!

6 Summary

As this report indicates, we believe there are bugs in Ultrix which are more serious than "just another OS bug". It's our hope that Digital will take this seriously and that they will be able to use this report as a guide and reference during the problem solving.

When these problems are solved, the DS user community will get a major benefit as the brute power and nice software environment of the DS family of workstations no longer are obscured by buggy software and hardware.

On behalf of the system staff,

Steinar Kjaernsroed,
February 1, 1990.