Express Production Real-time e-VLBI Service

EXPReS is funded by the European Commission (DG-INFSO),
Sixth Framework Programme, Contract #026642

Protocols Performance Report

- Final Report on Protocols and Network Infrastructure

Title: Protocols Performance Report
Sub-title: Final Report on Protocols and Network Infrastructure
Date: 2009 March 23
Version: 0.9 Final -1
Filename: EuProto-v1-0
Authors: Stephen Kershaw, The University of Manchester
Ralph Spencer, The University of Manchester
Richard Hughes-Jones, The University of Manchester
Paul Burgess, The University of Manchester
Simon Casey, The University of Manchester
Anthony Rushton, The University of Manchester
Paul Boven, JIVE

Summary: We report on investigations on the suitability of protocols for use with high-bandwidth e-VLBI; for the transfer of high bandwidth e-VLBI data across international links and for distributed correlation. The time delay behaviour of TCP has been investigated and TCP variants have been evaluated to ascertain the benefits for real-time data transfer. A new UDP based transfer system (VLBI_UDP) has been developed, tested and portions of the code used to attain correlation at 1024Mbps over 1Gbps light paths by selective packet dropping. A new alternative protocol, DCCP, has also been studied. Measurements of throughput on trans-Atlantic links have been made with the aim of investigating possible global e-VLBI work in future. Tests of multicast techniques on both light and heavily loaded networks have been made showing that it is possible to make more efficient use of lightly loaded networks e.g. a light path.

Delivery Slip

	Name	Partner	Date	Signature
From				
Approved by				

Document Log

Version	Date	Summary of Changes	Authors
1.0	25/03/9	Final formatting	RS
0.9	23/03/09	Edit figure and equation numbering	RS/SC
0.4-2	10/03/09	Revisions to UDP	SC
0.4	16/07/08	Formatting	SK
0.2	18/06/08	Further analysis (internal)	SK
0.1	03/06/08	Initial draft (internal)	SK

Project Information

Project Acronym	EXPReS
Project Full Title	Express Production Real-Time e-VLBI Service
Proposal/Contract number	DG-INFSO #026642

Table of Contents

1 Introduction

This is a report on the testing of protocols for high data rate data transfer in e-VLBI, and forms part of deliverable D150 in the EXPReS project (under the FABRIC JRA1).

e-VLBI data has the feature that the data are continuously streamed at constant bit-rate. The regular and timely arrival of data at the correlator is a major concern and is often of greater importance than reliability and minimising data loss. This has a number of repercussions for protocols: Transmission Control Protocol (TCP) is reliable by design but its congestion control behaviour is in contrast to our desire to maintain transmission at the constant bit-rate generated from the telescope signals. Congestion control can cause delay in individual data streams and skew between data steams from telescopes in the array may impair correlator performance.

VLBI and radio astronomy data are noise-like: the signals are obtained from the statistical properties of the noise, in particular the cross-correlation coefficient. Losing 50% in the number of packets received in an experiment has the same effect on signal to noise as losing 50% of the bandwidth. Selective dropping of packets is therefore as effective a method of congestion control as rate reduction. The adaptation of user datagram protocol (UDP) for e-VLBI is probably the optimum solution, provided selective packet dropping can be implemented without loss of correlation. However, the non-congestion-controlled, connectionless nature of UDP transport may make it undesirable for use over shared, packet switched networks: network providers could interpret high data-rate UDP data streams as a denial of service attack, and switch off such connections to preserve quality of service for other users. For such connections, TCP or one its variants is more suitable, with the unreliable yet congestion-aware DCCP protocol the apparently ideal for e-VLBI, though not yet at the state where it can be used reliably.

To investigate these issues we have run a number of sub-projects:

- TCP_delay – where we measure the effect of TCP rate reduction on the time of arrival of packets, and see if increasing buffer size can compensate. Using TCP Reno as a baseline, we investigate the relative benefits of advanced congestion control algorithms.
- VLBI_UDP – where software for the transfer of e-VLBI data by UDP has been developed to fit in with requirements of VLBI systems. The software enabled tests of the effects of packet loss on the JIVE correlator to be undertaken and the demonstration of high data-rate e-VLBI over UDP.
- A new protocol, DCCP has been recently implemented in the Linux kernel. The suitability of this protocol for e-VLBI is investigated in this report.
- Tests on a trans-Atlantic link undertaken to determine the characteristics of dedicated lightpaths over long distances.
- Tests on the use of multicast techniques on a heavily loaded networks at 2 Mps data rates and at Gbps data rates on the links to JIVE. The latter has proven successful and is used routinely in e-VLBI operations

These workstreams are described in detail in the following sections.

Much of the connectivity for high bandwidth signals in European eVLBI is via dedicated light paths (usually made by combing VLANs) though the academic networks.This means that other users are not troubled by eVLBI traffic, there is no denial of service even when the full capacity is used for eVLBI. We are therefore able to use UDP protocols, which would not be possible in a fully IP switched network.

2 Constant Bit-Rate Data Transfer over TCP

2.1 Introduction

Transmission Control Protocol (TCP) is the most widely used transport protocol on the Internet, largely because it features reliable transfer of data, which is a common requirement. However, it is not ideal to use for constant bit-rate applications because TCP throughput can vary wildly in a lossy environment. Many applications using constant bit-rate data transfer desire timely arrival of data but the rate fluctuations of TCP mean that timely arrival of data is not guaranteed. We examine the effect of packet loss on packet arrival times and investigate whether packet loss and the consequent effect on throughput delays the data irrecoverably. The performance of TCP from the perspective of data arrival time will determine the suitability for real-time applications, such as e-VLBI. Electronic Very Long Baseline Interferometry (e-VLBI) is a technique used for high-resolution observations in radio astronomy which involves the transmission of constant bit-rate data streams which are generated in real-time. Timely arrival of data is a fundamental requirement of e-VLBI and data are often transmitted using TCP, hence tests were conducted using constant bit-rate flows at rates of up to 512 Mbit/s to be representative of e-VLBI observations.

2.2 Transmission Control Protocol

2.2.1 Properties of TCP

TCP is connection-oriented and reliable, ensuring that data sent will be perfectly replicated at the receiver, uncorrupted and in the byte-order sent. From the perspective of the application TCP ensures that the byte stream sent is the same as the byte stream received. Data corruption is detected by checksums and the receipt of all data (reliability) is ensured by using automatic repeat-request (ARQ), whereby the receiving system sends messages (ACKs) back to the sending system to acknowledge the arrival of data and hence indicate the missing data to be retransmitted. TCP assumes lost data packets are due to network congestion and attempts to mitigate congestion by varying the transmit rate - a process known as congestion avoidance, of great importance and described in more detail later.

2.2.2 High performance TCP

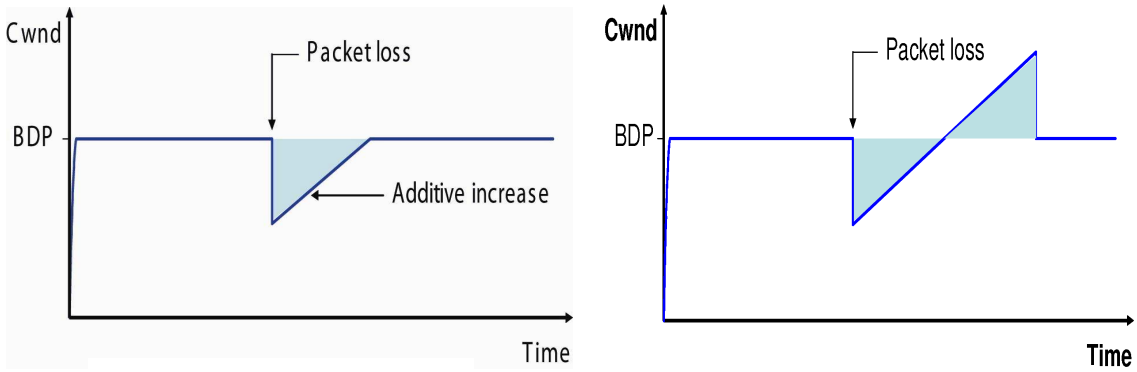
To make effective use of TCP, especially with high-capacity networks, it is often necessary to tune certain parameters. The end-hosts maintain windows over the data and to use the full capacity of a link the windows must be sized to the bandwidth-delay product (BDP) to allow sufficient “inflight” unacknowledged segments [1]. In this investigation, a desired constant bit rate CBR was considered, where bandwidth delay product, BDP, is expressed as:

$$\mathbf{BDP = CBR \cdot RTT} \qquad \text{Equation 2-1}$$

where RTT = round-trip time. With windows sized to the BDP, steady line-rate throughput is achievable if we have no packet losses and so this practice is a generally recommended step to tune a TCP connection. In the event of packet loss the size of one window, the congestion window, on the sending host is adjusted to limit the maximum instantaneous throughput.

2.2.3 Reaction to loss

When TCP detects a lost packet it is assumed that the loss was due to network congestion and TCP enters a congestion avoidance phase, altering the achievable transmit rate dramatically by adjusting the congestion window. This feature of TCP was implemented to prevent congestion collapse of the Internet where competing flows reduce the useful throughput to zero. It is the congestion avoidance behaviour of TCP that creates problems for constant bit-rate flows. The standard NewReno response to congestion is a decrease of the congestion window by a factor of 2, followed by an additive increase of 1 packet per round-trip time. This gives the throughput a characteristic sawtooth shape when a packet loss is detected - a sudden dramatic reduction of the congestion window, followed by a gradual linear increase. Considering this sawtooth congestion avoidance response, as shown in Figure 2-1, the amount of data that is delayed can be calculated.



(a) BDP buffer size

(b) Expanded buffer size

Figure 2-1 Behaviour of the congestion window in TCP

2.2.4 Delayed data

When a packet is lost, the equations of TCP congestion avoidance determine both the decrease of the congestion window and the rate of increase. If we consider a pre-loss throughput of CBR, it can be calculated that the time taken to regain CBR throughput is given by:

$$t_{recovery} = \frac{CBR * RTT^{2}}{2 * MSS} \qquad \text{Equation 2-2}$$

where MSS is the maximum segment size, the maximum amount of data that TCP can encapsulate in one packet [1].

Figure 2-1 Theoretical action of TCP congestion avoidance with CBR data transfer. Comparing (a) and (b) we see the effect of dramatically increased buffer size. The shaded area of delayed data in (a) is compensated for in (b), transmitted faster than the constant bit-rate, having been buffered on the sending host

The shaded triangular area in Figure 2-1(a), whose presence is due to a packet loss, has area proportional to the amount of data that has been delayed. The area is proportional to the recovery time and can be represented simply as:

$$\frac{CBR^{2} * RTT^{2}}{8 * MSS} \qquad \text{Equation 2-3}$$

For applications like e-VLBI, where data are transferred over large distances at high rates it is essential to note from Equation 2-3 that the amount of delayed data scales with the square of the throughput and the square of the round-trip time.

2.3 Constant bit-rate data over TCP

It is often said in the literature that TCP is largely unsuitable for real-time applications and constant bit-rate flows because of the variable rate of TCP over a lossy connection due to congestion avoidance [2],[3],[4]. If CBR data is streamed over TCP, as with some multimedia applications or e-VLBI, the reduced throughput due to packet loss leads to a data arrival rate on the receiver of less than the original constant bit-rate. If the processing or playback at the application level is not to stall then sufficient data must be stored in a play-out buffer to compensate for the lower data-rate at the transport level, allowing CBR data arrival at the application level. This is common practice for

streaming multimedia applications, requiring an initial buffering period and hence a delay between the start of the transfer and the start of the playback.

The situation of bulk data transfer is quite well researched and understood [5],[6], in contrast to the equivalent situation but where the CBR data is generated and transferred in real-time. When a CBR data stream is generated in real-time and cannot be stalled then we must transfer the data at a steady CBR else we have to either discard or buffer the data at the sending end.

2.3.1 Regaining timely arrival

If we temporarily store the delayed data on the sending host and can subsequently transfer it faster than the constant bit-rate then we should be able to regain timely arrival of data at the receiving host. We require the data to be buffered and the maximum window size must permit transfers at a rate higher than the CBR. In the investigation that follows, both functions are performed using the socket buffers in Linux.

Data from a Linux application, destined for a network, is buffered in a socket buffer, the size of which we can specify through kernel and application parameters. The socket buffer in Linux serves two purposes: to retain data for windowing and also as an application buffer, designed to isolate the network from effects of the host system, such as scheduling latency of the Linux kernel. Therefore, on the sending host, the socket buffers, which are an integral part of TCP/IP in the Linux kernel, can be used to buffer the data that is delayed in the event of a loss.

2.4 Experimental configuration

2.4.1 Network setup

The network links used to test constant bit-rate performance over TCP were dedicated fibre optic lightpaths with connections to UKLight in the UK peering with NetherLight in the Netherlands. The links were tested to have a very low bit-error rate, allowing loss-free data transfers with stable delay and jitter characteristics, making for an ideal protocol testing configuration. The lightpaths were used to connect hosts in Manchester, Jodrell Bank Observatory and JIVE[1] with dedicated point-to-point 1 Gbit/s connections. From Manchester the round-trip times (RTT) were 1 ms Jodrell and 15 ms to JIVE. A connection from Manchester to Jodrell, looped back in the Netherlands gave a RTT of 27 ms.

The computers used as end-hosts were server-quality SuperMicro machines, with all configurations tested to give 1 Gbit/s throughput using UDP/IP or TCP/IP over Gigabit Ethernet interfaces. The systems used Intel Xeon CPUs and were running Red Hat or Fedora distributions of Linux. Tests were performed using kernel versions 2.4.20 and 2.6.19 with negligible difference in TCP performance seen between kernel versions. All systems were equipped with onboard Intel e1000 Gigabit Ethernet Interfaces.

2.4.2 Diagnostic software

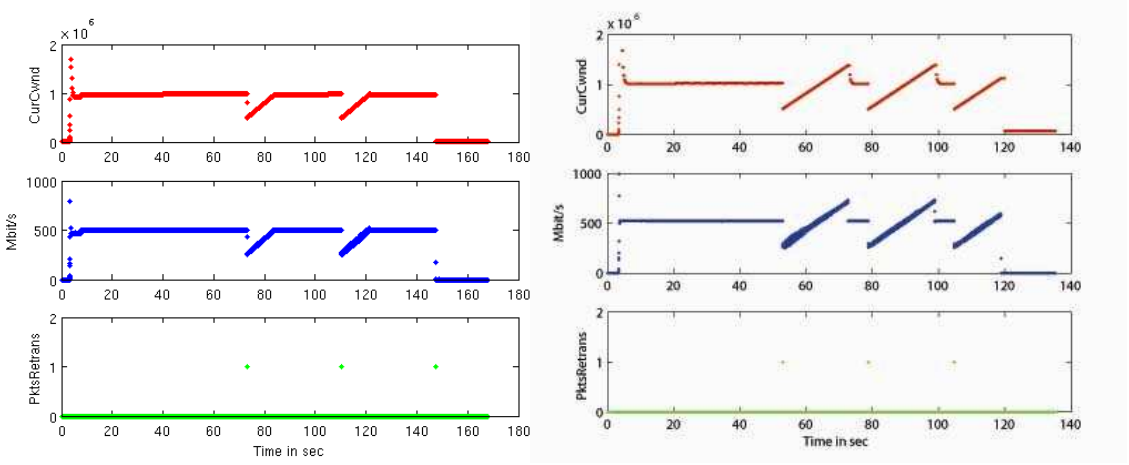
TCPdelay [7] is an application written by Richard Hughes-Jones, used to conduct tests using memory-to-memory TCP streams, sending data to a socket at regular intervals so as to attain a specified average data rate, emulating a CBR data stream. TCPdelay measures the time a packet or rather message is sent and its subsequent arrival at the application level, allowing measurement of whether the data stream is arriving at the receiver in a timely manner.

In order to gain more insight into the behaviour of TCP the web100 kernel patch was used. Web100 [8] is a kernel patch which provides extended TCP instrumentation, allowing access to number of useful TCP related kernel variables, such as the instantaneous value of the congestion window. Packet loss on the test networks is rare, so we simulated packet loss in the receiving hosts using a Linux kernel patch to discard packets at a configurable, regular rate.

[1] JIVE is the Joint Institute for VLBI in Europe, Dwingeloo, Netherlands.

2.5 Results

Using the recommended socket bandwidth-delay product buffer size, the behaviour of a 512 Mbit/s CBR TCP stream over a lossy 15 ms connection was studied with 0.9 Mbyte (BDP) socket buffers. The observed behaviour is shown in Figure 2-1(a). In the event of a lost packet (deliberately dropped on the receiving host) we see the reliable TCP protocol retransmitting a packet (lowest plot) and we see the expected congestion window evolution, as detailed earlier and illustrated in Figure 2-1(a). The rapid decrease and additive increase of the congestion window is apparent, with recovery of the constant bit-rate transfer taking around 10 seconds. We see an amount of delayed data of around 160 Mbyte, in agreement with Equation 2-3 when delayed acknowledgements are accounted for.



(a) 0.9 Mbyte (BDP) buffer

(b) 160 Mbyte (180x BDP) buffer

Figure 2-2 Plots of TCP parameters, logged using web100. Kernel patch used to drop packets. Top: TCP congestion window (bytes), middle: achieved throughput (Mbit/s), bottom: number of packets retransmitted.

Data is further delayed with every subsequent packet lost, the cumulative effect of multiple losses shown in Figure 2-3, which demonstrates the effect of loss rate on message arrival time. The lowest curve in Figure 2-3 shows the observed timely arrival of data, with higher curves showing lossy transfers diverging rapidly away from this ideal. As one may expect, with the throughput dipping below the desired rate many times and never exceeding it, the amount of delayed data increases and the data arrives later as the duration of the transfer increases. Figure 2-1(b) shows the same network configuration of the test in Figure 2-1(a) but with the socket buffers increased to 160 Mbytes, which is the calculated amount of delayed data. As explained previously, the intention was that the delayed data was stored in the TCP socket buffer, to eventually be transmitted at a rate in excess of the constant bit-rate. We see in Figure 2-1(b) that we initially have the same post-loss behaviour as (a) but the buffered data means that we can transmit faster as the transfer from the buffer memory is not limited to the constant bit-rate. Once the buffered data has been exhausted, we transmit new data at the CBR once more, as seen in the Figure 2-1(b). For the duration of the sawtooth the receiver experiences delayed data arrival, but subsequent data arrives in a timely manner once more, until the next loss. In this situation, with a constant bit-rate of 512 Mbit/s and a 15 ms RTT, we can use a 160 Mbyte buffer on the sending side to allow timely delivery to be resumed at the receiver. However, the use of these large buffers introduces temporary delays in the arrival times of the data of many seconds.

Instead of never resuming timely arrival and the departure for timely arrival becoming increasingly worse with time, which is the situation with conventionally sized buffers, we can use larger buffers to instead suffer only a temporary period of delayed data arrival. One must consider however the logistics of providing such large buffers and be able to cope with the temporary period of delay.

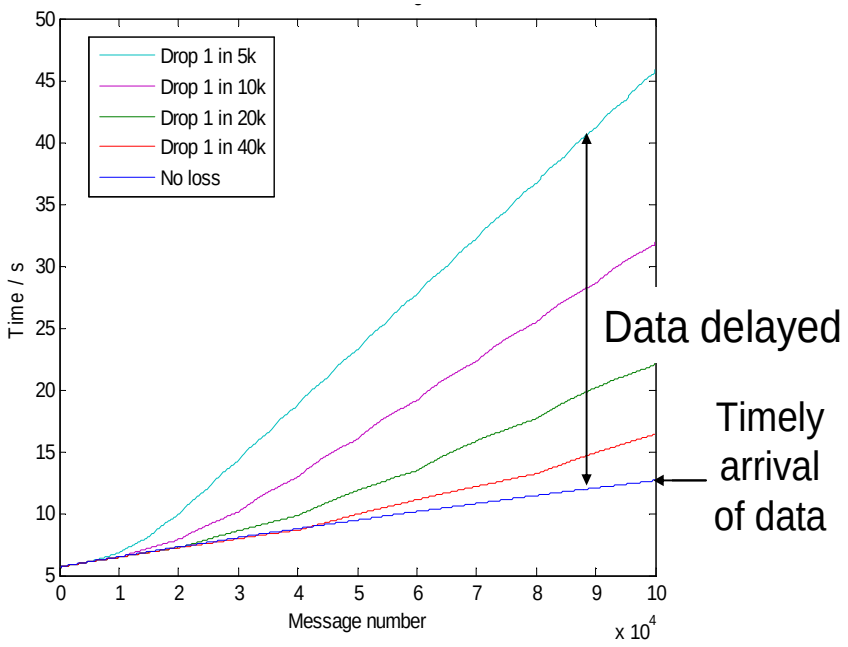
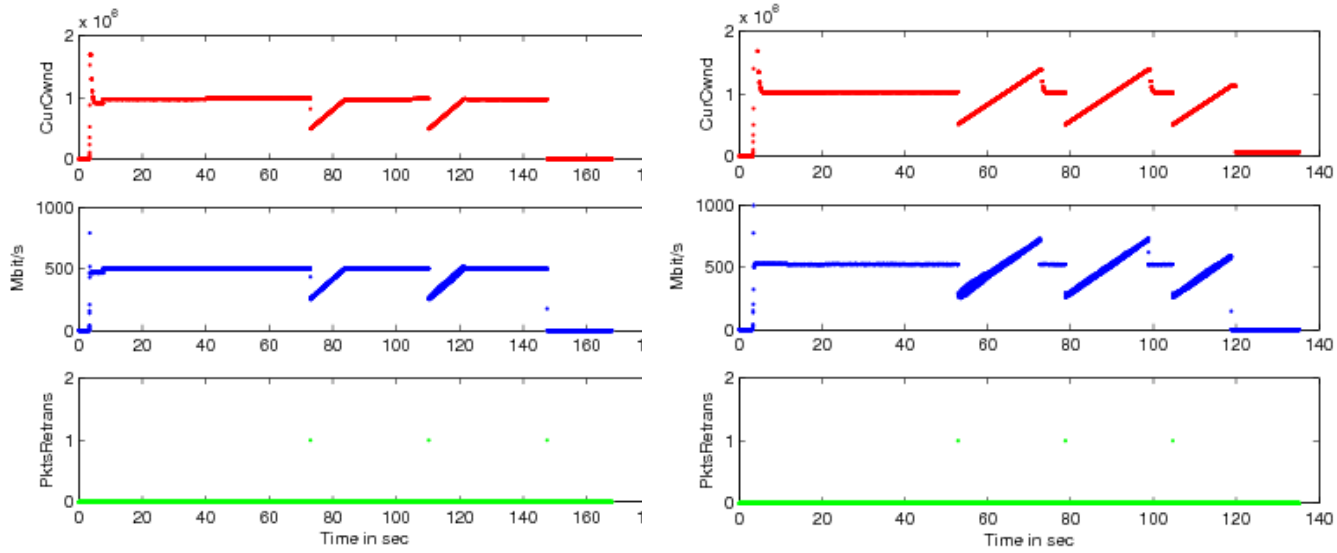


Figure 2-3 The effect of packet loss on message arrival time. Manchester to Jodrell Bank, looped through Amsterdam, 27ms RTT. TCP buffer size 1.8 Mbytes (BDP).



(a) 0.9 Mbyte (BDP) buffer

(b) 160 Mbyte (180x BDP) buffer

Figure 2.4. (a) and (b) above. Plots of TCP parameters, logged using web100, for a 512Mit/s NewReno flow over the 15 ms link, with packet loss found using a kernel patch. The plots in each subfigure are as follows: upper: TCP congestion window (bytes), middle: achieved throughput (Mbit/s), lower: number of packets retransmitted.

2.6 Alternative stacks

Having demonstrated that timely constant bit-rate data delivery is possible over our test network, using 0.9MB buffers for the 1 ms RTT connection and 160MB for the 15 ms RTT connection, it is

desirable to repeat the measurements using the larger RTT links. Calculations show a required buffer of 520MB for 27 ms RTT and 6.4 GB for 95 ms and although we could demonstrate timely delivery on our dedicated, well-specified end-hosts for the 27 ms RTT connection, RAM constraints would not allow us to do so for the 95 ms RTT transatlantic link. Calculations show that it takes 14 minutes to resume 512 Mbit/s CBR throughput after a loss on the transatlantic link when using NewReno, a link having a 6.3 MB BDP. High-throughput, large RTT networks are well known to exhibit this problem with Reno and most of the alternative congestion control algorithms in the Linux kernel are designed to alleviate this problem and reduce the time taken to regain full throughput.

Figure 2.5 shows the reaction of H-TCP when a single packet loss is induced. H-TCP can rapidly regain timely arrival constant bit-rate delivery after a loss, here requiring socket buffering of 40 MB to regain timely delivery in 14 seconds.

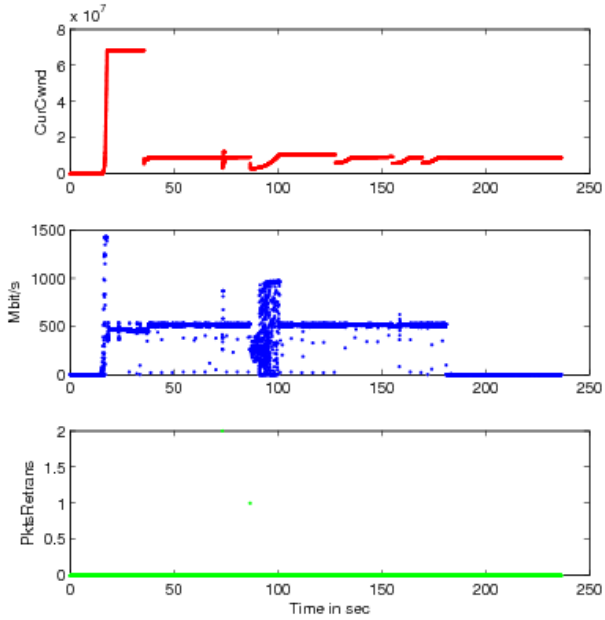
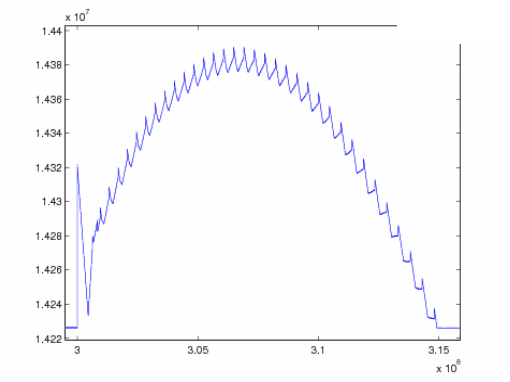


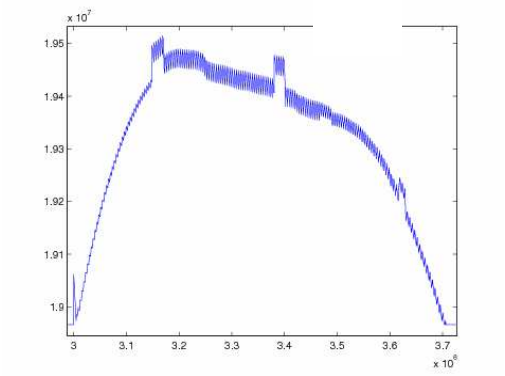
Figure 2.5 Plot of web100 parameters, logged using web100, for a 512Mbit/s H-TCP flow over the 95ms RTT link. There is one induced packet loss at time 87 s.

This is a dramatic improvement over the performance of NewReno, which would require 820 s and buffers if 6.4 GB. The other algorithms tested (figure 2.6) also regained timely delivery of data much faster than NewReno and using buffers of less than 200 MB in every case. The alternative congestion control algorithms are from 14 to 255 times faster to regain timely arrival of data. The relationship between recovery time and amount of data buffered is not the same as with NewReno, as we would expect as we no longer have the largely linear nature of the AIMD algorithm.

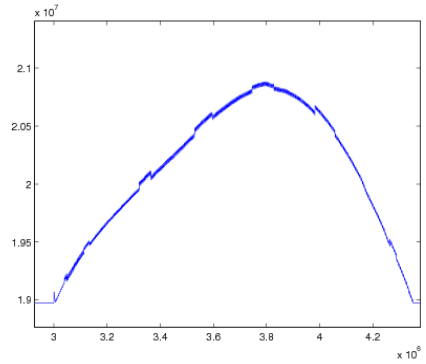
Figure 2.6 shows, from a loss event to the resumption of timely delivery of constant bit-rate data, one-way delay of messages. Comparing the shapes of the plots with that of NewReno in Figure 2.5 illustrates how the buffer dynamics differ between algorithms.



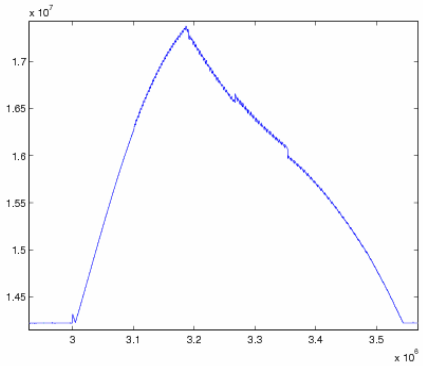
(a) Scaleable



(b) H-TCP



(c) Cubic



(d) BIC

Figure 2.6 Plots of one-way delay vs packet sequence number for different congestion control algorithms ((a) Scaleable, (b) H-TCP, (c) Cubic and (c) BIC. The plots have a single packet loss at the left of the plot, with timely arrival of data attained by the right-hand edge. Note that scales are different for each plot to allow a clear view of the shape.

2.6.1 Bursty behaviour

As Figure 2.5 shows, after a packet loss on the 95 ms link, the Web100 data shows very bursty behaviour, which was not apparent in the Web100 data for the shorter lightpaths. The Web100 data is sampled at 10 ms intervals, which for the transatlantic connection allows details on RTT timescales to be clearly seen. The plots for all algorithms on the 95 ms link show that after a packet loss data is burst from the socket buffer at line-speed upto that allowed by the TCP congestion window, and then we experience a data-free period until the window-increasing acknowledgements return 95 ms later. Examination of the TCPdelay timestamps for individual packets reveals the same behaviour for the 15 ms and 1 ms connections, behaviour which is not apparent from Web100 data due to the sampling time being comparable to the feature size. On our networks where we had uncontended 1 Gbps light-paths, this bursting was not problematic but this unavoidable bursting of data from the sending buffer would be undesirable on a highly contended, shared network.

2.7 Conclusions on TCP

Use of TCP/IP to transfer CBR data with normal TCP buffer settings (BDP) leads to delayed data arrival in a lossy environment. The delay is to the entire stream of data as all data arriving after the first loss will be delayed, with subsequent losses increasing the delay, data arrival times diverging from the expected arrival times. For an application such as e-VLBI this behaviour is not acceptable and can lead to a loss of correlation and lower quality results as streams from different observing stations become desynchronised.

We have used large socket buffers to temporarily store data and demonstrate regaining timely delivery of data following a loss. It was necessary to store a large amount of delayed data and subsequently transmit it at a rate exceeding the constant bit-rate to achieve the average CBR throughput. The practicalities of providing buffers depend strongly on the parameters of the application and network. For a 512Mbit/s flow over a connection with a RTT of 15 ms we are required to buffer 160 Mbytes of data. The scaling apparent in Equation 2 is an important consideration, with transatlantic distances requiring the buffering of upwards of 6 Gbytes of data and temporary departure from timely delivery of tens of minutes. This scale of buffering will often prove impractical. Alternative TCP variants can be used to good effect, reducing the buffer requirement to under 200MB and the delay to under a minute for all algorithms tested (H-TCP, Highspeed, BIC, CUBIC and Scalable TCP).
The success of the demonstrated approach to regain timely arrival of CBR data depends on the networks used. This technique can only be applied for a certain range of loss rates and depends on the connection having available capacity above the constant bit-rate. With large amounts of data stored in the sender socket buffer, data can burst onto the network, often at the maximum achievable line rate, with the potential to cause problems on shared networks and cause further packet loss. Further work with competing flows present could help clarify this.

3 VLBI_UDP

3.1 Introduction

e-VLBI requires vast quantities of data to be sent from several remote telescopes over high-speed computer networks and the Academic Internet to a single correlator. Currently, VLBI data rates of 512 Mbit/s are achievable using the Transmission Control Protocol (TCP) [9]. An alternative to TCP is the User Datagram Protocol (UDP), which is used by VLBI_UDP.

3.2 The case for UDP

Whilst e-VLBI in the EVN can run at 512 Mbit/s with TCP, if longer baselines are used, for example across the Atlantic, TCP may struggle to sustain a constant 512Mbit/s should any packet loss occur. TCP guarantees all data sent will arrive and in the right order, but was designed with congestion control algorithms which reduce the transmission rate by half if any packet loss is detected. There are both software and hardware buffers in the Mark5A systems which can compensate for a reduced transmission rate for short periods of time, but extended slow periods would mean that the buffers would run empty. The higher the round trip time (RTT), proportional to the physical network distance, the longer it takes TCP to recover back to its previous rate after a packet loss event [10]. UDP, on the other hand, does not guarantee delivery of data, and the transmission rate is governed by the user.

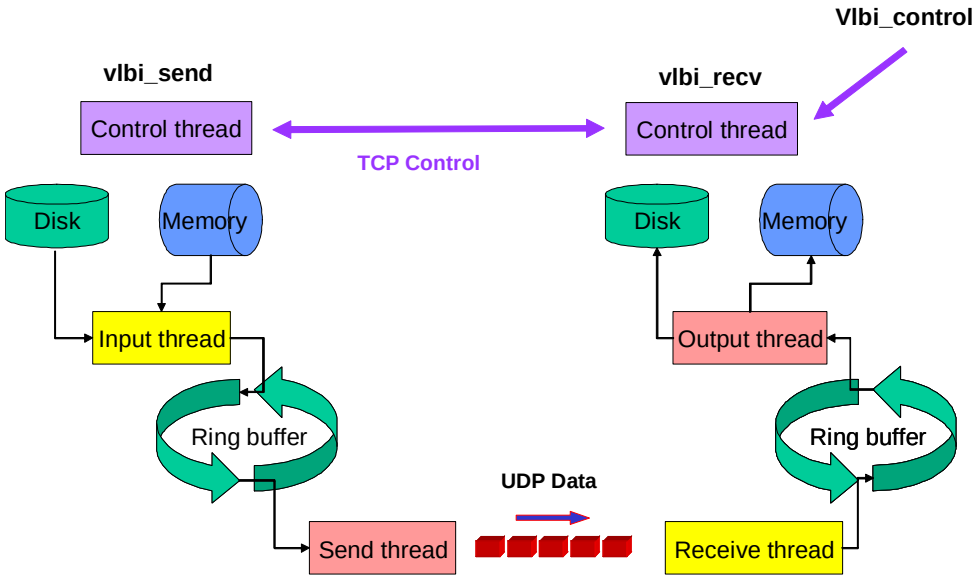


Figure 3-4 The architecture of the VLBI_UDP programs

3.3 VLBI_UDP architecture

VLBI_UDP was originally written as a monolithic application by Richard Hughes-Jones for a proof of concept demonstration at iGrid 2002 emulating of the loads e-VLBI places on the networks. It has

since undergone several revisions with extra features being added, and these are detailed below. The current architecture is represented graphically in

Figure 3-4. There are 3 components to VLBI_UDP, the sending application, receiving application, and the control application. The send & receive components are run as applications with no user input. The control application drives the send & receive components, and may be accessed via a variety of methods. It can either take user input from the console, commands via a webpage through a miniature http interface, or read commands from a file with no user interaction. A single instance of the control application controls multiple send/receive pairs.

3.3.1 Conversion to pthreads

As a monolithic application, VLBI_UDP was consuming almost all available CPU cycles due to constant polling, checking if there is data to be moved around. Clearly this is not an optimal situation, and so the send and receive programs were both split into 3 threads: control, data input and data output. Splitting the application into threads allows each thread to act with a reasonable amount of independence from the other threads, whilst still allowing communication between them.

3.3.2 Ringbuffer

A ring buffer was present in the original iGrid2002 application, but was incomplete so far as it didn't handle out of order packets correctly in all circumstances – not a problem for the demo but needed correcting for use with real data. As each UDP packet is received, it is written directly to the next usable location in the ring buffer. Each packet has a sequence number which allows missing and out of order packets to be detected. If there were one or more packets missing immediately previous to the next received packet, then this packet would be in an incorrect position in the ring buffer. In this case, a function RingMove() is called, which moves the last packet forward the required number of positions within the ring buffer such that it is then correctly placed. The next available location is now set to after the new location of the last packet.

Should the 'missing' packet(s) subsequently arrive, out of order, then they are first written to the next available location as before. The sequence number is checked, and RingMove() is called with a negative offset to place the packet back where it should have been. In this case, the next available location doesn't change and so the next packet will be written to where the last packet originally arrived. This process is illustrated in Figure 3-5.

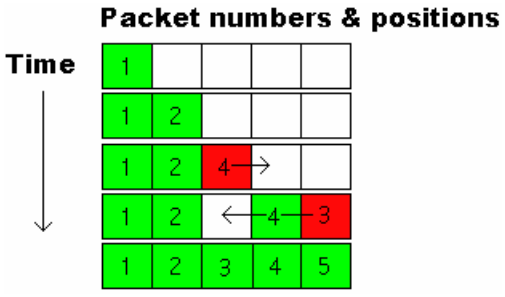


Figure 3-5 The processing performed when packets are placed into ring buffer.

3.3.3 File access

To allow for testing with real data, and as a precursor to interfacing with actual VLBI hardware, file access was implemented. Linux large file support is used, which was a necessity when dealing with VLBI data sets which are almost exclusively >2GB.

3.3.4 Packet dropping

A packet dropping function has been added [11], which, when combined with the file access mode, facilitates the creation of data sets with missing packets under controlled conditions. This function is implemented only in the sender module. The send thread receives a pointer to a packet of data from the ring buffer, passes this to the dropping function as a parameter, along with a choice of dropping algorithm. The return value is a pointer, which will be the same as that passed to the function if the packet was not dropped, else will be a pointer to a portion of memory which will be used to replace the dropped packet. The user can specify what this portion of memory should contain, choosing from two options; either a random series of bytes, or a special fill pattern. If the correlator system observes this fill pattern, then it knows that this portion of data is invalid and therefore should not be correlated. Currently there are two algorithms available. The first drops single packets at a steady rate with no randomisation, the second can drop a bunch of between 1 and 10 consecutive packets, the value chosen randomly. To maintain a fractional loss rate f in the 2nd case, after a bunch of n packets are dropped, the subsequent $n(1/f - 1)$ packets are not dropped

3.4 Results from tests with VLBI_UDP

VLBI_UDP has been used both as a demonstration tool at events such as iGrid2002, and more recently at the GÉANT2 network launch, as well as a tool to probe network conditions over extended periods of time. Figure 3-6 demonstrates a 24 hour flow, transmitting data from a PC based in Jodrell Bank over a dedicated 1 Gigabit fibre connection into a PC based in Manchester University. Each point represents the average received bandwidth in a 30 second period, and it can be seen that rate stability is mostly maintained to 1 part in 1000.

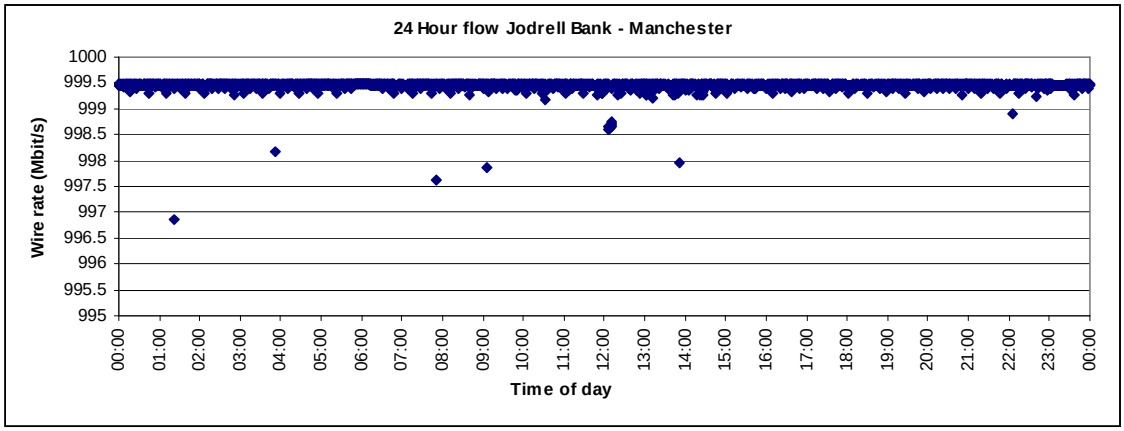


Figure 3-6 The measured UDP achievable throughput for a 24 Hour flow from Jodrell Bank Observatory to Manchester University.

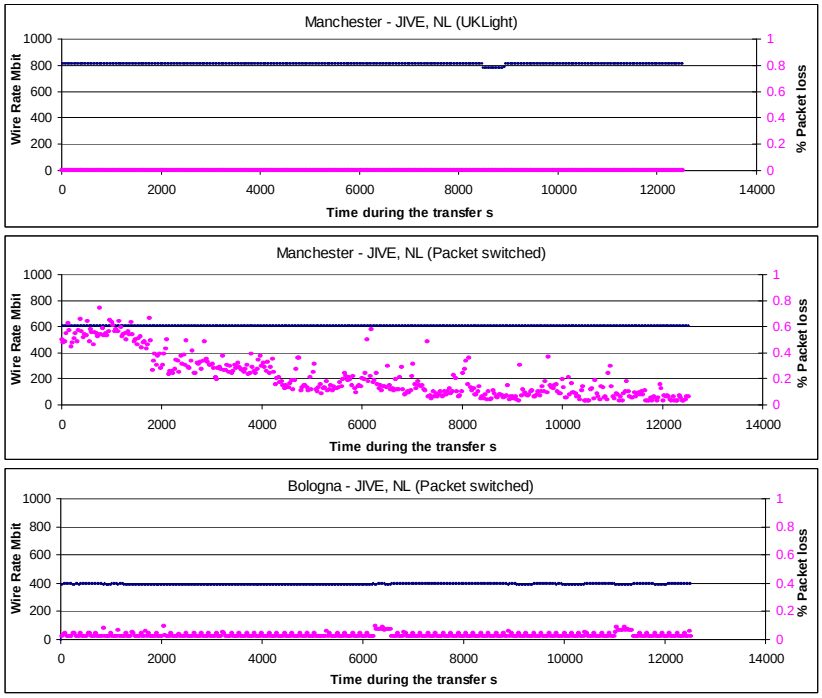


Figure 3-7 Multiple UDP streams into JIVE using VLBI_UDP

Figure 3-7 shows 3 simultaneous transfers into JIVE, one from Manchester travelling over a UKLight dedicated 1 Gigabit lightpath, another from Manchester but crossing the conventional packet switched Academic Internet, and the third from Bologna again over the conventional packet switched Academic Internet. The lightpath performed as expected, with the transmit rate purposely capped at 800 Mbit/s and showing no packet loss. The second flow was capped at 600 Mbit/s, since this was travelling via the Manchester University campus access link and so would have swamped regular campus Internet traffic. Packet loss is present here due to contention, most likely over the campus access links, and can be seen to decrease through the test period, as the campus traffic decreased. The third plot was limited at 400 Mbit/s and gives stable throughput with little packet loss.

3.5 Results from correlation tests of packet-dropped data

The data used for these tests were recorded from three radio telescopes at a data rate of 256 Mbit/s whilst observing a strong astronomical radio source. Two of the sets were left unmodified, whilst the third set was processed several times in VLBI_UDP to simulate the effect of lossy network transmission. Several copies of the original file were created by this method, with each containing packet-dropped data at several loss-rates ranging from 0.5% up to 20%. Correlation was subsequently performed with these files against the other two unmodified sets in order to produce two modified baselines, and a third unmodified baseline.

The output from the correlations consisted of a series of fringe amplitudes, one for each integration period, in this case every second. By comparing the fringe amplitudes obtained from correlating the different data sets, it was possible to observe how the missing data affected the correlation process.
Figure **3-8** shows the effect that lost packets had on the correlation amplitude for the Effelsberg-Jodrell baseline; the Westerbork-Jodrell baseline produced very similar results and so not included for clarity. It can be seen that whilst there is little difference when packets are lost singly or in bunches, there is a considerable difference when the fill pattern is used compared to when it is not.

An effect which can not be seen in

Figure **3-8** is that at loss rates of 20%, there were several integration periods which produced wildly inflated fringe amplitudes. The corresponding correlation weighting for these points was extremely low, thus allowing these points to be weighted out when producing
Figure **3-8**. That this happened, indicates a serious loss of correlation and shows an approximate threshold at which the effects of packet loss become non-linear. It would therefore be safe to recommend that packet loss rates above 15% should be avoided.

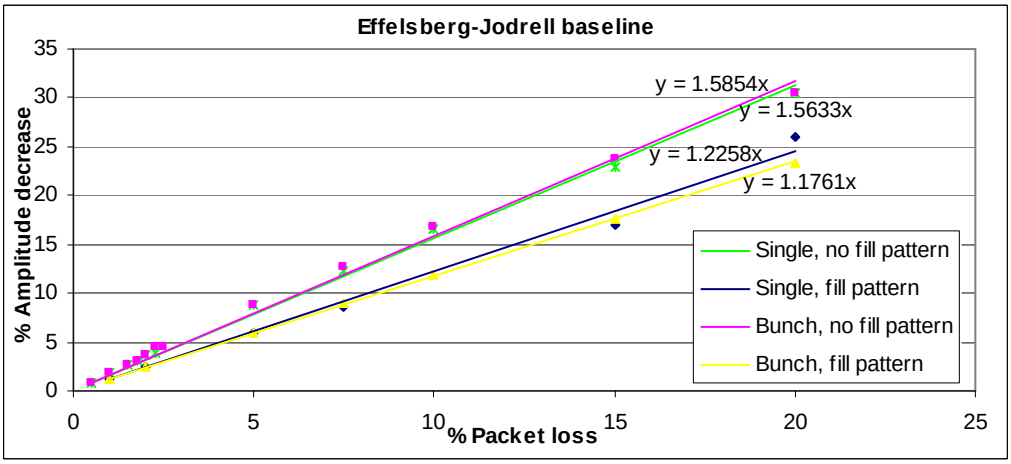


Figure 3-8 Effect of packet loss on correlation amplitude

3.6 1024 Mbit/s e-VLBI experiments performed over 1 Gbit/s Ethernet links

Due to protocol overheads, it is not possible to transmit a full 1024 Mbit/s stream over standard 1 Gbit/s Ethernet links, and so the packet-dropping code was developed within VLBI_UDP. For ease of use, rather than further modifying VLBI_UDP to be compatible with other applications used within e-VLBI, it was decided to add portions of code from VLBI_UDP in to the existing Mark5A software. Live tests with this modified software were performed and data was successfully correlated at 512 Mbit/s. At 1024 Mbit/s, packet-dropped data from one antenna was successfully received and decoded at the correlator, but due to hardware problems the second antenna was unable to capture data at 1024 Mbit/s and so a final correlation was not possible.

Inspired by features within VLBI_UDP, programmers at JIVE have created their own version of the Mark5A software exclusively for e-VLBI use. Through the use of this, correlation of packet-dropped 1024 Mbit/s e-VLBI streams from several antennas is regularly achieved.

4 Testing of DCCP at the Application Level

4.1 Introduction

Datagram Congestion Control Protocol (DCCP) [12] is a recently developed transport protocol, similar in parts to both TCP and UDP with the intention that certain applications and the transport of certain types of data may benefit. Where congestion control is required but reliability is not, DCCP provides a transport level option attractive to many applications such as VoIP and e-VLBI. The congestion control algorithm, CCID, used by DCCP is selectable, allowing DCCP to be tuned more closely to the requirements of a particular application. CCID2 [13] is TCP-like Congestion Control, closely emulating Reno TCP while CCID3 [14] is TCP-friendly rate control, minimising rate fluctuations whilst maintaining long-term TCP friendly behaviour.

DCCP has been in the Linux kernel since 2.6.14, with recent kernel releases such as 2.6.19 and 2.6.20 having an improved implementation, incorporating code developed by ESLEA, that is often considered as fairly stable and high-performance. We report on the porting of a network testing application to DCCP, experiences with creating a stable DCCP testbed and results from initial performance tests.

4.2 Porting of test software

In order to test the performance of DCCP, software tools were required hence DCCPmon [15] is a port of UDPmon [17] by the original author, Richard Hughes-Jones. Guidance was given by Andrea Bittau to help with the port to DCCP and the resulting application is being used and proving to work well. However, the process was not entirely trouble-free – some problems were encountered that were perhaps indicative of a DCCP implementation that was in development, rather than complete and polished. For example, DCCP related #defines were not to be found in the userland include files, an issue mitigated by creating specific include files. Some system calls were missing and the API was in a state of flux with functions changing between kernel releases 2.6.19 and 2.6.20, when the software was initially developed. Further kernel releases, up to and including 2.6.25, have seen further changes, with most DCCP supporting applications failing to compile or run at various points. For this reason, and due to limited testing, DCCPmon is currently still considered by the author as experimental and consideration for the continual revision of software should be made if use id to be made of DCCP at the current stage of its Linux implementation.

During the development of DCCPmon and for corroboration of results, a patched version of iperf [16] was used. In addition to the information from the main test application it is desirable to gather data from as many other sources as possible. One useful window into the kernel networking stack is though the kernel SNMP statistics, however there are currently (as of kernel 2.6.21) no SNMP counters for DCCP variables. These statistics would also have been invaluable when problems became apparent with certain kernel versions and it would certainly be a worthy addition to the implementation at the earliest opportunity.

4.3 End-host setup

The computers used as end-hosts were server-quality SuperMicro machines, with all configurations tested to give 1 Gbit/s throughput using UDP/IP or TCP/IP over Gigabit Ethernet interfaces. The systems used Intel Xeon CPUs and were running Scientific Linux or Fedora distributions of Linux. We had systems using two Dual Core Intel Xeon Woodcrest 5130 CPUs clocked at 2 GHz, dual-booting 32-bit and 64-bit distributions of Fedora Core 5. We also had systems with two Intel Xeon 2.4 GHz Hyper-Threaded CPUs using a 32-bit distribution of Scientific Linux 4.1. All systems were equipped with and DCCP tested with on-board Intel e1000 Gigabit Ethernet ports. Tests with UDP

and TCP gave stable line-rate performance over all tested networks, including 1 Gbit/s over a transatlantic lightpath.

4.4 Experiences with the Linux DCCP implementations

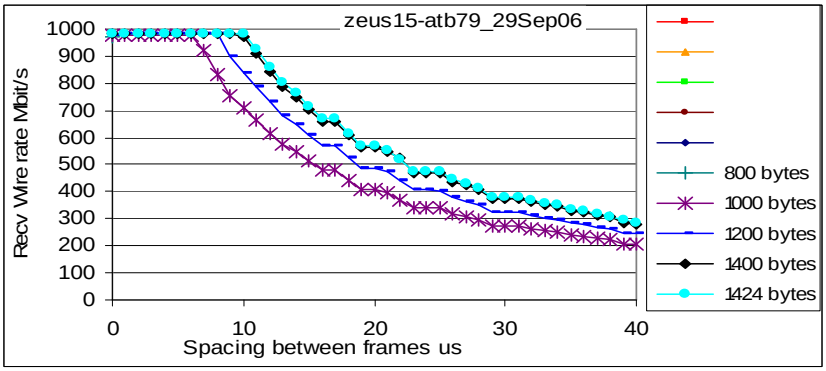
While developing the DCCPmon program and preparing for performance tests, several different Linux kernels have been used, often displaying undesirable effects. With such a new implementation of a new protocol it has often been unclear whether we are seeing problems with the DCCP implementation or something specific to our systems, however we report on our findings and some of the steps taken to achieve a stable DCCP test bed.

4.4.1 Kernel version 2.6.19-rc1 and 2.6.19-rc5

This kernel version is a release candidate for stable kernel version 2.6.19, which was tested before the stable kernel version was released. Using both DCCPmon and iperf it was found that we were not getting a working DCCP connection - tcpdump showed that the connection was successfully made, with packets exchanged both ways but no ACKs were sent in response to data packets received. In the absence of feedback the sender-side DCCP transmit timer progressively fell back until a threshold upon which DCCP terminated the connection.

We conducted many diagnostic tests to establish the cause of the problem. Advanced features of the network interface card were disabled and DCCP data was sent though a tunneled connection to prevent possible discrimination of the new protocol. Eventually, inserting debugging code into the kernel showed that data were incorrectly being discarded due to header checksum errors, a problem that was later fixed in the network development tree and merged into the stable 2.6.19 kernel release.

With a suitably patched 2.6.19-rc5 kernel DCCPmon was able to measure the throughput as a function of the spacing between the packets as shown in Figure 4-1 where a rate of 990 Mbit/s was achieved. However, the system was unstable and there were frequent system crashes



.
Figure 4-1 Achieved DCCP throughput with CCID2 as a function of the packet spacing.

4.4.2 Kernel versions 2.6.19 and 2.6.20

As previously noted, the API calls changed slightly, necessitating further development of the test software code, after which, with the checksum problems resolved it was hoped that interesting tests could be run.

The initial results were promising, with CCID2 showing short-term line-rate throughput - a useful data rate of around 940 Mbit/s after header overheads. CCID3 had an average rate of around 300 Kbit/s but unfortunately DCCP proved to be unstable using either CCID on our 64-bit systems. Transfers would often only last for a few seconds before the receiving system hung with a kernel panic. Some tests would continue for longer, a few minutes with the same throughput performance, but all would trigger a kernel panic within four minutes and repeating tests with larger packet sizes would lead to a quicker crash. The crash dumps associated with the panic generally indicated that the crashes were occurring most regularly in the region of the packet reception code of the network interface card (NIC), where memory is allocated to store incoming packets.

Repeating the tests using a 32-bit distribution and kernel on the same computers yielded the same behaviour, however the older systems running Scientific Linux on Hyper-Threaded Xeon processors proved to be more stable, with extended runs possible, with the majority of transfers persisting until deliberately terminated after many tens of minutes. The system logs, however, showed that everything was not perfect, with many zero order page allocation failures logged, in a similar context to the panics - close to the receive interrupt of the NIC.

4.5 Towards a stable test bed

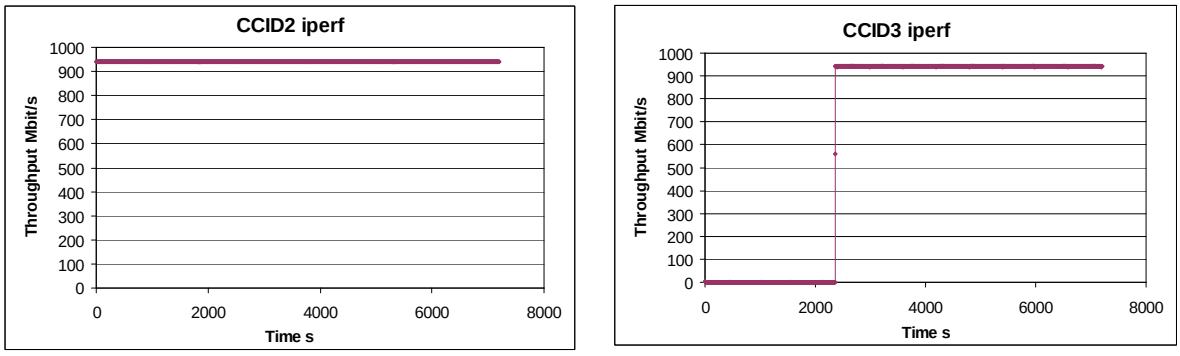
Analysis of crash dumps and kernel messages showed that most error messages were generated when memory was being allocated in NIC RX IRQ handler. To attempt to fix the problem the operation of the NIC driver was analysed together with aspects of the kernel memory management code. In general, when a request is made for memory allocation, the request will either be serviced immediately (if memory is available) or it will be blocked while sufficient memory is reclaimed. However, when memory allocation is requested in an interrupt context, for example memory allocation to store received packets, blocking is forbidden. In order for the memory allocation to have a higher chance of succeeding, the kernel reserves some memory specifically for this situation where the allocation is classed as atomic. The amount of memory reserved for atomic allocations is determined by the value of the min_free_kbytes sysctl variable.

Increasing the min_free_kbytes parameter in the receiving host from the default value of 5741 to 65535 proved to prevent all the previously seen error messages, though it is not entirely clear to us why the memory allocation problems originally occur. It is possible that the default value of min_free_kbytes is not sufficient relative to the time between scheduled runs of the memory management daemon (e.g. kswapd), which are scheduled to keep that minimum amount of memory free. A larger value of min_free_kbytes may mean that the reserved memory is never filled before the memory management routines can be run. As we do not encounter similar problems with UDP and TCP, it is possible that the higher CPU utilisation of DCCP could cause such a situation by using more CPU time. It is strange that on one system the allocation failures prompted error messages while on another the result was a fatal system crash.

The problem is not entirely mitigated though as even with the increased value of min_free_kbytes crashes persist if the packet size is increased sufficiently. More investigation is needed to gain a full understanding of this unwanted feature of our DCCP test.

4.6 Results of recent tests

With an increased value of min_free_kbytes on the receiving hosts, the systems proved to be stable with 1500 Byte packets, with no tests generating error messages of any kind. Every test, with flow durations of up to 2 hours, remained stable and was terminated gracefully at the predetermined time, using all kernel versions of 2.6.19 or later. Having a stable test bed has allowed preliminary tests of DCCP throughput performance, as outlined below.



(a) CCID2 (b) CCID3

Figure 4-2 Comparison of the throughput achieved for CCID 2 and 3 using iperf with 5 s sampling.

4.6.1 Back-to-back tests

With any two systems CCID2 can attain the maximum possible data rate of 940 Mbit/s, which is the line-rate over Gigabit Ethernet and stable for the duration of the test. This result is illustrated in Figure 4-2(a), with Figure 4.2(b) showing the different behaviour of CCID3. With CCID3 there is an initial period with an average rate of only 300 Kbit/s, with the regular rate variation detailed in Figure 4-3. After a number of packets (around 65,500) the rate jumped to line-rate and remained steady, as seen in Figure 4-2(b). This is strange behaviour, with the number of packets being indicative with a 16-bit overflow perhaps, but there have been a lot of patches produced for CCID3 recently which have not yet made it into the stable Linux tree. Using a development tree and patches from numerous authors changes the CCID3 behaviour completely. The most appropriate comment to make is that CCID3 is developing and the performance of current stable kernels is not indicative of what is be achieved by developers.

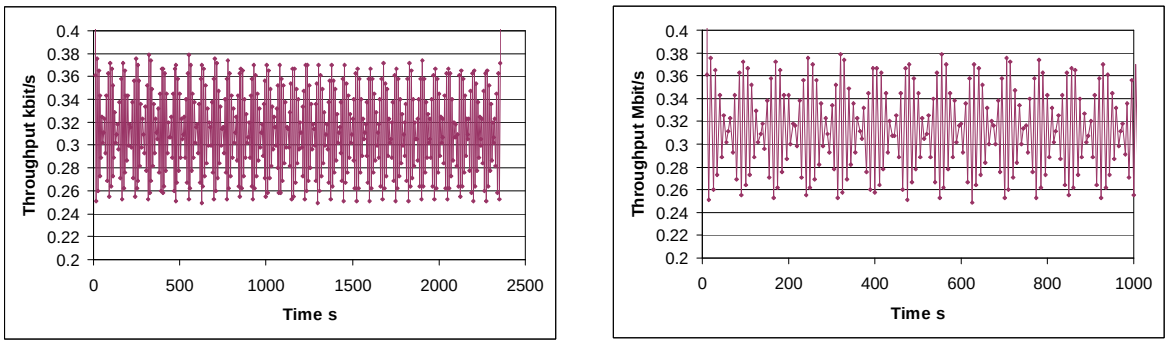


Figure 4-3 Expanded views showing the variation in the CCID3 iperf throughput during the initial part of the test.

4.6.2 Tests over extended networks

Over a transatlantic connection, with end-hosts in Manchester and Chicago, using UDP and TCP we can achieve line-rate throughput (see next section). Although the back-to-back performance of DCCP between identical systems gave line-rate, over the 94 ms transatlantic lightpath only a steady 130 Mbit/s was attained. The performance of DCCP seemed to be CPU limited at the sender, with one CPU showing an average of 98% load, compared to the load at line-rate back-to-back of 82%. Increased CPU load with increasing round-trip time can sometimes be observed with TCP flows but it is not immediately obvious that this should be the case with DCCP and it is curious that the effect seems so dramatic. The performance of DCCP needs to be investigated further over different

distances and with different systems. CPU load profiles can hopefully yield further useful information about the performance of DCCP.

4.7 Developing a new CCID

VLBI has a clear requirement to move constant bit-rate data and can tolerate high-levels of packet loss, making UDP seem like the ideal transport protocol. Other applications have similar requirements, with streaming media and VoIP being examples of applications where constant bit-rate can be advantageous and packet loss is often tolerable. However, there is concern from network providers that UDP traffic could overwhelm other traffic and overload the network. Concerns and opinions have been voiced and mitigating options have been discussed at recent meetings such as the EXPReS & EVN-NREN meeting in Zaandam, NL and PFLDnet 2007 / IRTF workshop in Marina Del Rey, US, with input from Kees Neggers, SURFnet; Glen Turner, AARNET; and Aaron Falk, the IRTF Chair. One option that the authors support is to use DCCP in combination with a new CCID, initially given the name SafeUDP. The proposed CCID aims to address the concerns expressed about using plain UDP by implementing something "UDP like" but with network protection. SafeUDP would use the DCCP ACK mechanism to detect congestion, following which the congestion would be evaluated: to ensure that congestion is not in the end-host and to determine whether the congestion is transient. This evaluation step is useful to remove the assumption that all losses are congestion events, which is a conservative assumption but in some circumstances often unnecessarily detrimental to performance. The application would be notified of the congestion through modified API calls, with sendto() and recvfrom(), etc. having new return codes. The application can then take action. If the application were to take no action within a reasonable time, this CCID would drop input from the application and inform the application that it had done so. This idea is being worked on with the long-term aim of a draft RFC.

4.8 DCCP-TP

DCCP-TP (http://www.phelan-4.com/dccp-tp) is an implementation of DCCP, written by reference to the RFCs rather than borrowing the existing Linux codebase. The author states that it is optimised for portability and he hopes that this feature will make the implementation useful for embedded systems. There currently exists a Linux port and work is in progress on a Windows port. The compilation of the Linux code is straight-forward once the dependencies are resolved and the end result is a userspace DCCP stack and also separate client and server applications to allow testing.

Initial testing gave mixed results, with segmentation faults on some systems but clean running on others. For reference, the segmentation faults occurred on a system running a 2.4.20 kernel and one system running 2.6 series kernels, though this failure case was unexpected as the success cases were on very similar, though not identical, hardware. Attempts to use the supplied client and server applications to test DCCP-TP over network links (back-to-back connection and switched) produced errors from the userland stack which persisted even if the traffic were routed through an end-to-end tunnel to mask the presence of the DCCP protocol. CCID 2 did not work over the local loopback interface but a test was successfully performed over loopback with CCID 3.

Using DCCP-TP with CCID 3 over loopback, a connection was successfully established between the client and server processes, with messages from the userland stack printed to the console to confirm this. Performance over loopback is generally expected to be representative of the maximum performance of a protocol as it is a purely logical, low-latency link, however, the performance observed was low. In one, quite representative, test, with a transfer of 10,000 packets of 1400 bytes, only 9864 packets were received and this took 100 seconds, for a rate of 1.1Mbit/s.

In summary, DCCP-TP is a less mature implementation than the Linux implementation of DCCP but it is worthy of note that it is a younger implementation and the work of one person (Tom Phelan). The code still being developed and is open source and could be worthy of further investigation if the DCCP protocol becomes more pervasive

4.9 Conclusions on DCCP

We have seen that the back-to-back performance of DCCP using CCID2 is good, achieving line-rate for extended (multiple hour) back-to-back, memory-to-memory transfers. The throughput of CCID3 was generally lower though there is much current development with performance changing with every patch. Given the amount of patches being created by developers it is uncertain at what speed the CCID3 implementation in the stable kernel will develop. Tests of CCID2 over extended networks have been quite limited to date, with early results showing that DCCP uses much more CPU time and achieves a lower rate over a transatlantic lightpath. A rate of 130 Mbit/s to compare with 940 Mbit/s back-to-back has been seen, with further work needed to fully assess DCCP performance over long-distances. Achieving a stable test setup has not been trivial and there are some issues still to be resolved. We hope that our investigations of the issues with DCCP on our systems can help improve the implementation and make DCCP work "out-of-the" box on more systems. Working round the issues we encountered revealed a protocol implementation that we look forward to investigating more fully in the near future. Many applications can benefit from DCCP and we hope to extend the utility by considering the concerns of and working with network managers to build a new CCID.

It remains to be seen if DCCP will become a well supported transport protocol that can be used with few issues like TCP and UDP. It is supported as a standard by the IETF but needs supporting by mature implementations of stacks and acceptance as viable by producers of network hardware and software. Many hurdles are apparent in the testing of DCCP and the current recommendation from our investigations is that although DCCP shows promise as a transport protocol for e-VLBI and is supported as a standard by the IETF, the current implementations of DCCP are not suitable for use in a production or even reliable test environment.

5 Trans-Atlantic UDP and TCP network tests

5.1 Testing the trans-Atlantic link

Very long baseline interferometry (VLBI) generates large rates of data from many telescopes simultaneously observing a source in the sky. This can require the information to traverse intercontinental distances from each telescope to a correlator in order to synthesise high resolution images. With the dramatic development of the Internet and high bandwidth networks, it is becoming possible to transmit the data over large area networks. This allows the correlation of radio astronomical data to be done in real-time, whereas this process would take weeks using conventional disk based recording.

Jodrell Bank Observatory has a dark fibre from the MERLIN telescope array to the main campus at the University of Manchester, currently this carries two 1 Gigabit network paths. This local network connects to UKLight at Manchester Computing [18] via a Cisco 7600 switch. UKLight is a network of dedicated optical light paths, operated by UKERNA [19], and provides a guaranteed 1 Gigabit network path between Manchester and StarLight [20] in Chicago. Server quality PCs located at Jodrell Bank, Manchester and Chicago are connected to this infrastructure and enable network tests to be made.

5.1.1 TCP Bandwidth Tests with iperf

The achievable TCP throughput rates were measured between the PCs at Manchester and Chicago with the software package, iperf 0. The results of a test lasting 3.3 hours are plotted in Figure 5-1 and show a user data rate of 940 Mbit/s but multiple drops in throughput were observed during the test. Despite the network utilising a dedicated lightpath, packet losses were observed and as a result TCP goes into the congestion avoidance phase and reduces the transmitted bandwidth. The network application tool, iperf was used to measure the maximum throughput for a period of over 3 hours.

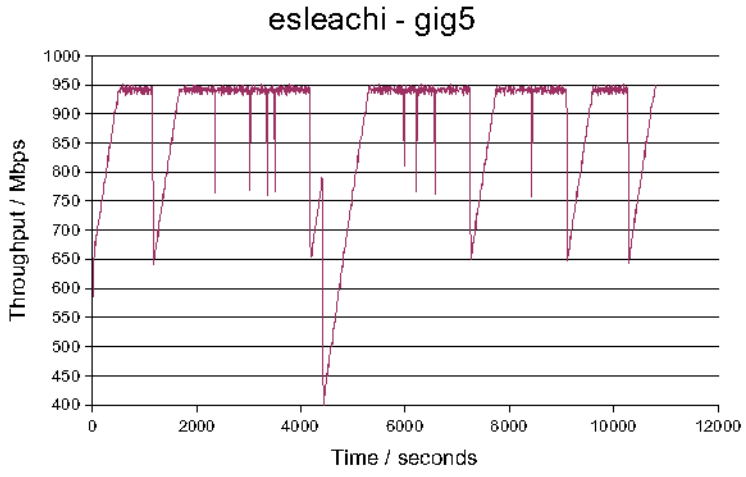


Figure 5-1 A TCP achievable bandwidth test between servers located at Manchester and Chicago.

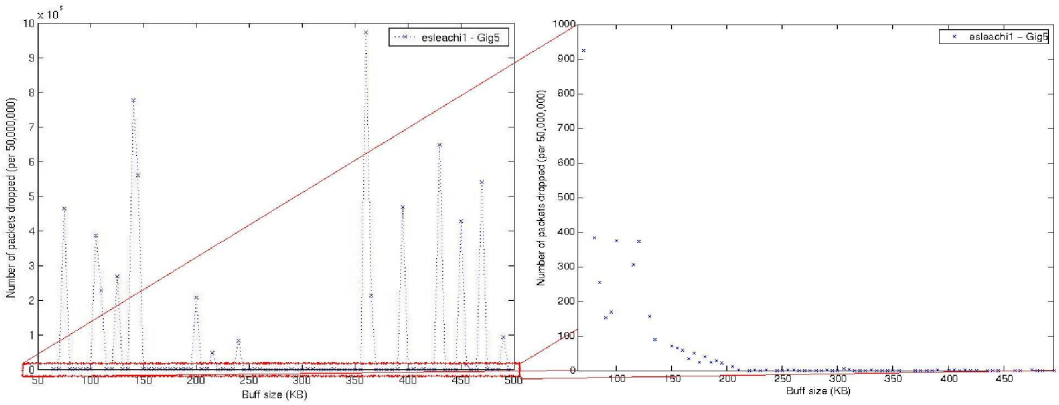


Figure 5-2 (a) (left) The number of UDP packets lost in a throughput test at 940 Mbit/s from Chicago to Manchester as a function of the receive buffer size. (b) (right) The same UDPmon data with the packet loss axis limited to 1000 counts.

5.1.2 UDP Network Tests

In order to better understand the network behaviour, the UDP protocol was used. Unlike TCP, if a UDP packet is lost in the network, the transmission rate is not reduced, nor is the packet resent as discussed earlier. This makes UDP an excellent diagnostic tool for troubleshooting network paths. The network analysis software package, UDPmon 0, was used to investigate the link between Manchester and Chicago by transmitting packets at a constant rate, and reporting packet loss. In order to emulate the transmission of e-VLBI science data, 50 million packets were transferred, which took about 10 minutes for every test. The number of UDP packets lost as a function of the receive buffer size are plotted in Figure 5-2 for a 940 Mbit/s flow from Chicago to Manchester. It shows two effects: large intermittent packet losses of >1 % and a much lower loss rate for small receive buffer sizes.

To understand this result, we tested the UDP performance by linking similar servers directly back-to-back in the lab. (i.e. without the transatlantic network). The application transmitted UDP at line rate (940 Mbit/s) and the buffer size of the receive host was varied from 65 KByte to 300 KByte. Figure 5-3 shows that packet loss was observed when the receive buffer was less than ~ 230 KByte and at low receive buffer sizes there is a linear relationship between packets lost per test and buffer size. These losses occurred in the end host when the queue between the UDP stack and the application became full. It seems probable that this may be due to effects of the scheduler in the Linux 2.6.20 kernel de-selecting the UDPmon application. Changing the priority of the UDPmon application using *nice* does not eliminate the loss with small buffers and produced little effect overall when the priority was varied across the entire range. Further work would involve changing the priority of kernel processes or changing the scheduler algorithm, with the investigation of real-time fully preemptible kernels of particular interest.

Increasing the receive buffer size at the application layer did not stop nor reduce the large > 1 %. intermittent packet losses shown in Figure 5-2.

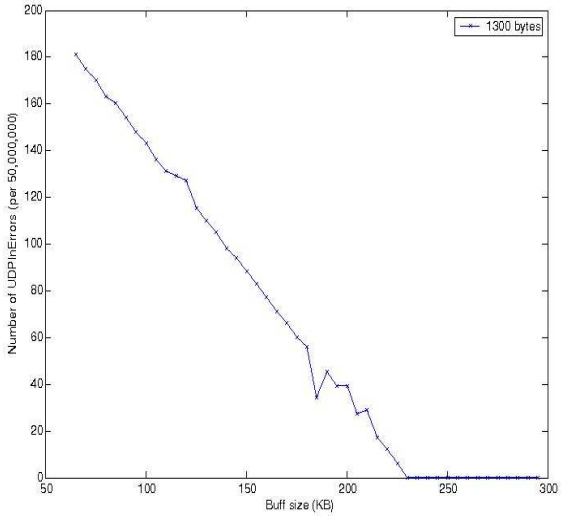


Figure 5-3 The number of packets lost between two computers directly connected between network interface cards (i.e. no transatlantic network).

5.1.3 Constant packet loss when running at line rate

The network tests at 940 Mbit/s with UDPmon were repeated in the opposite direction (from Manchester to Chicago). The receive buffer was set to 300 KB and the test was performed for a longer period of time (~ 30 hours). The number of packets lost as a function of time is shown in Figure 5.4
.

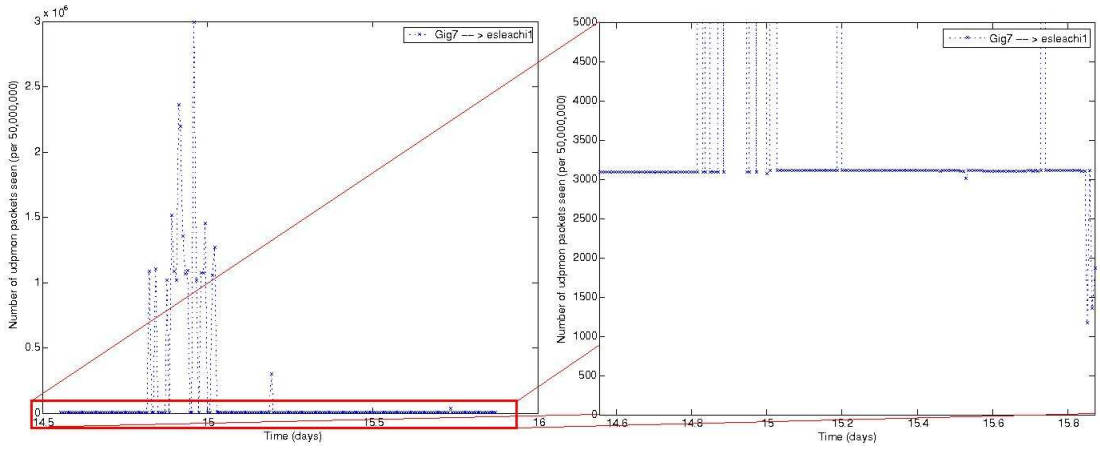


Figure 5-4 (left) The number of packets lost in a UDPmon test at 940 Mbit/s from Manchester to Chicago as a function of time. Once again large intermittent packet losses of >1 % were observed. (right) The same UDPmon data with the packet loss axis limited to 5000 counts. A constant loss of at least 3000 packets per test is observed.

Once again the application reported that occasionally large fractions of the packets (>1%) were lost in the network. However this time, as seen in the right plot of Figure 5-4, a constant loss of at least 3000 packets per 50,000,000 sent (0.001%) occurred in every test.

5.2 Network isolation

In order to have characterised this network link, it was important to examine where the data packets were dropped at the lowest point of the OSI model, i.e. layer 2. To do this we examined the SNMP (simple network management protocol) counters of the network interface cards and the Cisco 7600 switch connecting to UKLight. The results showed the constant packet loss discussed in section 5.1.3 were within the Cisco 7600. All 50,000,000 packets were received by the switch throughout each test. However if the transmission rate was reduced to 800 Mbit/s, the switch could transmit all 50,000,000 packets without loss.

We tested the switch's interface from Jodrell Bank to the Cisco 7600 using a different connection to the computer in the University of Manchester campus. Both the switch's SNMP counts and UDPmon's reports showed the switch transmitted every packet at line rate without packet loss in this configuration. The ports used on the 7600 for these tests and those showing packet loss were on different types of switch line cards, and this could have an effect.

5.2.1 UKERNA's maintenance

We concluded through a process of elimination that the large intermittent packet loss of > 1% was therefore within UKLight. After private communication with UKERNA, now known as JANET, it is believed the network issues were due to a broken fibre on the Manchester Computing to StarLight connection. After multiple maintenance tickets were issued by UKLight, we repeated the UDPmon tests.

The configuration of the local network from Jodrell Bank into UKLight did not give the constant packet loss seen in section 5.1.3 even at line rate. The results in Figure 5.5 show that when running at line rate (940 Mbit/s) from Jodrell Bank to Chicago after the network maintenance, very few packets were lost over a period of 2.5 days. Over 20 billion packets were transmitted (~ 200 TB) with the loss of only four packets in the network.

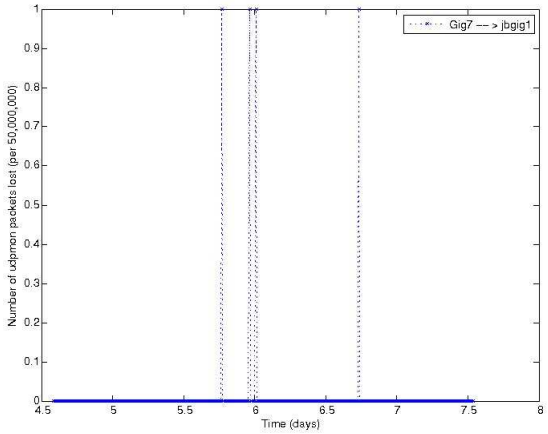


Figure 5-5 UDPmon tests from Jodrell Bank to Chicago sending UDP packets at 940 Mbit/s. Packet loss is shown against time. Over 200 TB of data was transferred over a period of over 60 hours. Only 4 packets were lost throughout the experiment.

5.3 Conclusion on the Trans-Atlantic Tests

This work demonstrates some of the challenges encountered when using high bandwidth networks. Software application tools have simulated the date rates required by e-VLBI science by continually sending large numbers of packets for many hours. This has shown the need for the receive buffers of the applications to be capable enough to collect data at these rates for long periods of time. Issues have arisen with the Cisco 7600 switch showing, that under certain circumstances the instrument does not perform as expected. This highlights the requirement to identify and test equipment to maximum abilities. Problems with the link were isolated by eliminating the client, end-host servers and local network by inspecting level 2 SNMP packet counts.

This led us to confidently conclude that large packet losses initially observed were within UKERNA's UKLight network. After maintenance on UKLight, our tests were repeated for a large time period (~ 3 days). This successfully showed it was possible to transmit packets between Manchester and Chicago at 940 Mbit/s without losing a significant number of packets.

6 Multicast

The appropriate protocol for the transfer of e-VLBI data when using distributed processing is an area for investigation. VLBI depends on cross correlation, a situation where an array of n telescopes has a much larger number of potential telescope pairs whose signals must be correlated. The relationship here can be expressed as n-choose-2, as discussed later. Individual telescope streams or blocks of data need to be used for more than one correlation operation. In a distributed correlator, this one-to-many relationship means that IP multicast is worthy of investigation as efficient data replication is a key feature of the technology.

6.1 Terminology and introduction

Unicast refers to the transmission of data from one host to another, using a one-to-one address to host relationship. Broadcast is a transmission to all the hosts on a subnet, for example an Ethernet ARP request to the broadcast MAC address FF:FF:FF:FF:FF:FF. Multicast is a transfer from one host to many where the destination hosts are well defined and represented by one 'group' address. Multicast can also be many-to-many hosts but we will not consider this in discussion.

A naive summary of multicast operation is as follows: clients join a multicast group using IGMP messages. Group membership is propagated from the client to the local gateway router and then between routers on a path between the clients and the server. The server sends data through the nearest multicast enabled router, which will propagate through all down-stream interfaces that have membership of the same group, as established by the IGMP messages. Multicast can span large areas and also limit the number of links used to the minimum required, in contrast to broadcast behaviour, which is not selective in its use of interfaces. The operation of multicast is entirely dependant on router support for successful operation. Academic networks, such as SuperJANET, support multicast, in contrast to most commercial ISPs who do not enable multicast. It is shown later how switches can impact the successful application of multicast.

The key feature of IP multicast is its potential for low impact data distribution with replication and pruning of data packets as required, using a standards track mechanism which is well supported by network hardware (though not necessarily widely deployed, as previously mentioned). Due to the one-to-many relationship, reliability becomes difficult to implement and there is no widely implemented, practical, mature solution to reliable multicast. The number of potential back channels is the key consideration for both reliability and congestion control. We are effectively restricted to using UDP or other unreliable, unidirectional protocols, which is acceptable in many situations as UDP has been shown to give acceptable e-VLBI operation. Issues with network owners restricting the use of high-bandwidth UDP, as discussed earlier, may be easier to resolve given that multicast is more bandwidth efficient than unicast on routed networks. Note that although multicast is technically distinct from unicast and broadcast, switched networks may elect to employ methods to treat multicast traffic as either unicast or broadcast. This is discussed later

6.2 Thought experiment

Imagine 4 telescope e-VLBI session with telescopes A, B, C, D. Baselines are formed by pairs AB, AC, AD, BC, BD, CD - 6 pairs in total. If we assume 1 node processes each baseline, we need 6 PCs and we will assume that they are found in one cluster, connected to the internet through a router. Without multicast each PC requires 2 streams, meaning that traffic at the ingress link to this 6 PC cluster scales as double the number of baselines., or twice 4-choose-2 (generally n-choose-2). With multicast, the ingress link only carries 4 (generally, n)

This is only one possible topology, for which we see a benefit with multicast, but many other situations will too. For instance, the membership-request nature of multicast allows widely distributed

clusters to request and accept only what is required by them and permissible in terms of bandwidth and processing constraints.

No. of 'scopes	Baselines	Incoming unicast	Incoming multicast	Delta
2	2	2	2	0
3	6	6	3	3
4	12	12	4	8
5	20	20	5	15
6	30	30	6	24
7	42	42	7	35
8	56	56	8	48
9	71	71	9	63
10	90	90	10	80
11	110	110	11	99
12	132	132	12	120

6.3 Relevant network technologies

6.3.1 Grid computing

Grid computing for distributed correlation is discussed in more detail in other EXPReS documents. Grid farms commonly comprise of worker nodes that operate through a head node, obtaining files from storage elements. This scheme lends itself more to offline processing but even in this situation multicast may provide benefits.

6.3.2 IGMP snooping

Membership of multicast groups is achieved by the exchange of IGMP messages. A simple layer 2 Ethernet switch will simply pass these messages as it would any other Ethernet frame and hence have no record of the hosts and hence ports that have requested membership of a given multicast stream. The default behaviour, to ensure that all interested ports are satisfied, is often to broadcast the frames to all ports. This can cause problems if there are high-bandwidth multicast streams or memberships of many separate multicast groups. For example, if take a switch with Gigabit Ethernet access ports and a 10 Gigabit uplink. If five 256Mbps streams were used, each GE port requiring just two, everything should operate fine, but if all ports were broadcasting all five streams then the system would not work well. IGMP snooping aims to alleviate this problem by interpreting IGMP messages and building a table used to intelligently switch traffic to only subcribers. Multicast will only operate optimally using routers or IGMP aware switches – this is an important consideration and individual clusters must be assessed for multicast suitability. Again, the membership nature of multicast means that every cluster could assess its own requirements and request only the appropriate number of streams to the appropriate destinations.

6.4 Experiments

6.4.1 Testing routed 2Mbps UDP multicast

The tests comprised a number of desktop PCs on the Manchester University campus joined to multicast streams generated by a host located in London. Each stream was 2Mbps. An application was written to measure the inter-packet arrival time of the multicast UDP packets, in order to evaluate the

suitability of multicast for real-time data. Ideally, we hope for a regular packet spacing, as witnessed with unicast UDP.

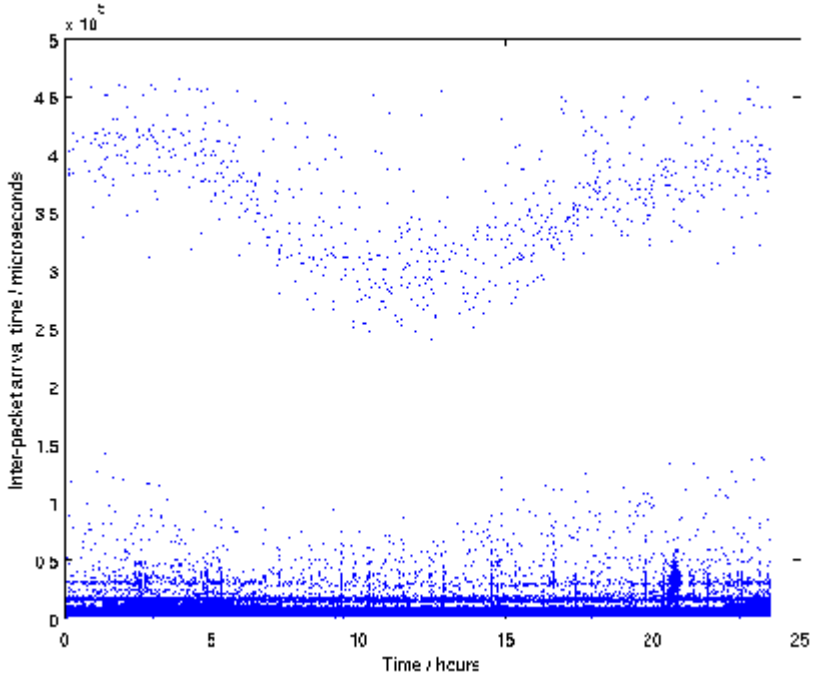


Figure 6-1 Inter-packet arrival time for a 24-hr run over the London – Manchester University network.

The lower dense area in Figure 6-1 shows the usual inter-packet arrival time of approximately 6ms, with some variation apparent. The detail at the top of the graph comprises latency spikes to values of 300-400ms, varying over the course of 24 hours; there appears to be a reduction in latency overnight (test started 7:37pm, drop therefore is approx 0030 to midday). This trend is probably representative of variation of performance with router load. The reduction in latency bears a great resemblance to the reduction in traffic forwarded by routers that is seen overnight, as in figure 6-2.

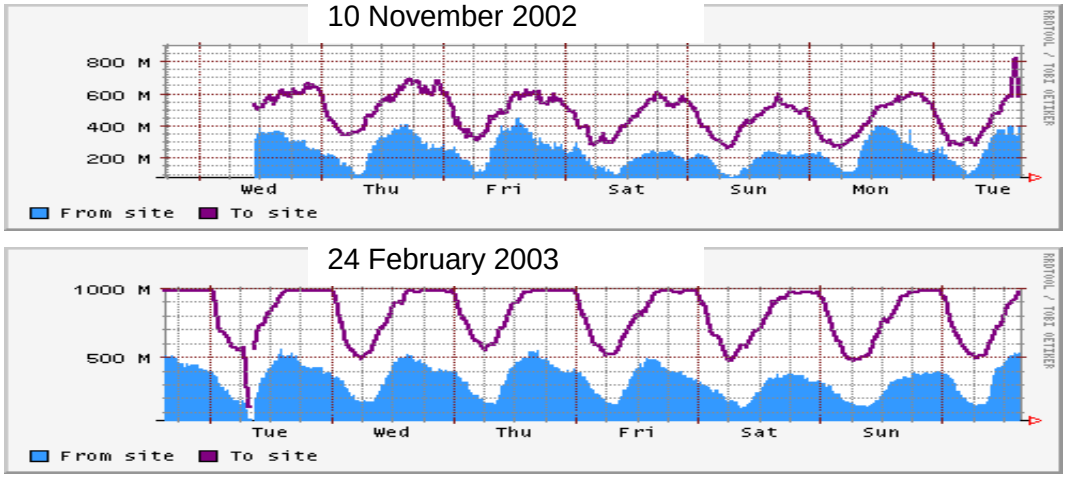


Figure 6-2 Network Traffic between NNW and SuperJanet4 with the purple line giving the traffic out of NNW. This shows the importance for forward looking capacity planning – on 26 Feb the access link was upgraded to 2.5Gbit/s. (Figure from 'High Bandwidth High Throughput Data Transfers in the MB-NG and EU DataTAG Projects', Richard Hughes-Jones et al., http://www.hep.manchester.ac.uk/u/rich/mb-ng/allhands03_v3.doc)

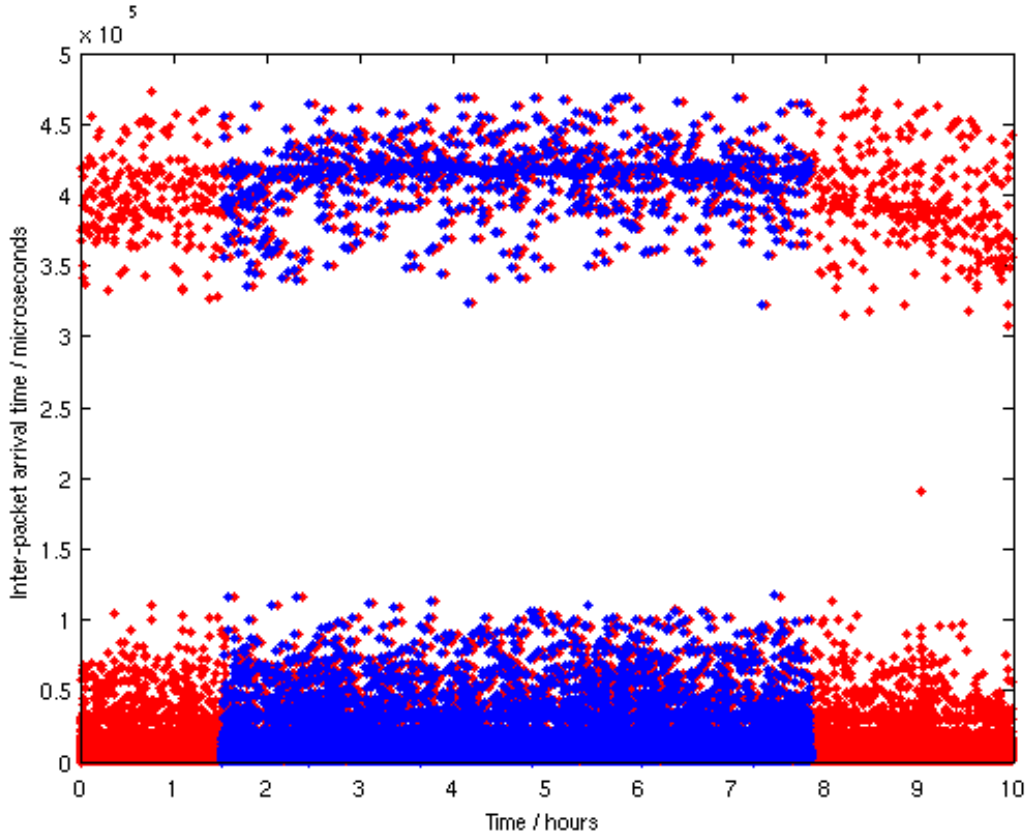


Figure 6-3 Further tests on the London-Manchester multicast link

The correlation of results at locations separated across a campus network indicates an issue arising at a common point. The inter-packet spacing at the sending host was verified as correct, so we have isolated the issue slightly but there is still a sizable amount of network infrastructure in this subset. The variation of maximum latency with time of day, and hence network load, may be indicative of a system operating at close to capacity, as one would hope and expect that the operation of network hardware is transparent from the user perspective. The latency spikes associated with packet loss is often indicative of buffer overflows and the regular periodic nature may indicate a scheduled task increasing, for example, router CPU usage. Further investigation showed that the same latency spikes could be observed using the same systems with unicast UDP. No spikes were present for UDP (multicast not possible to use) if we take measurements of UDP traffic which bypass the campus network.

Tests have been repeated after a period of 4 months and many network changes and upgrades in many places. The results are now as shown in Figure 6-3.

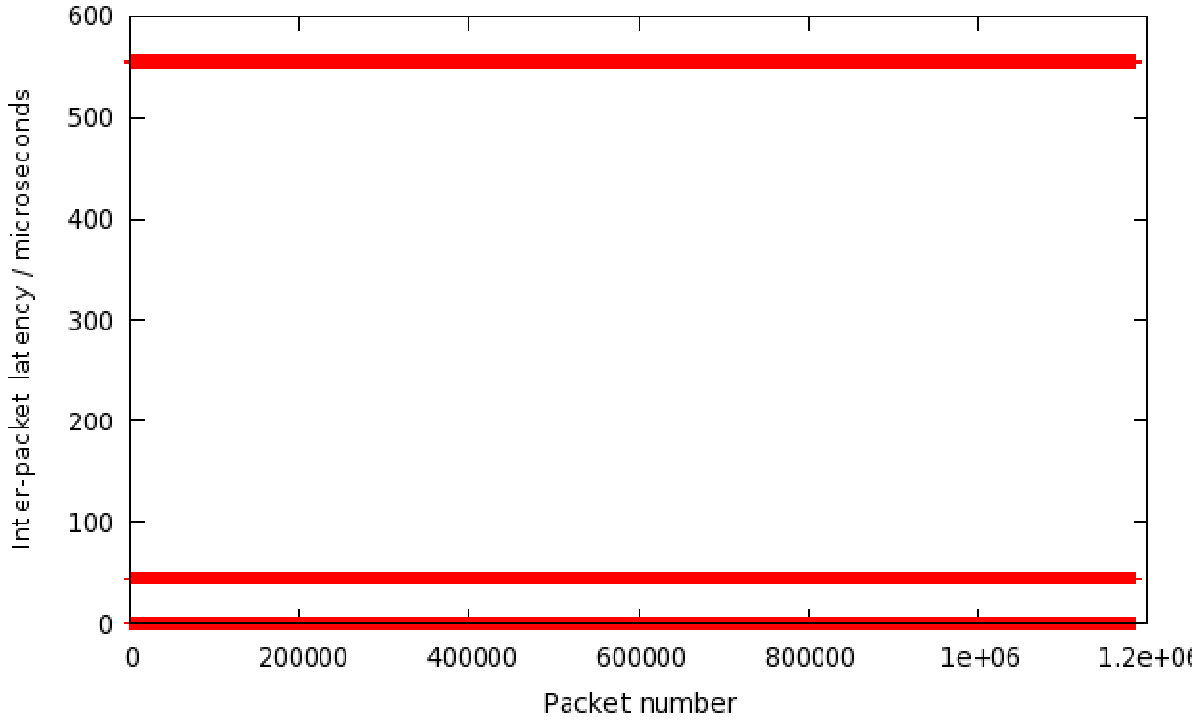


Figure 6-4 Local tests of 1Gbps multicast traffic traversing an unloaded 7609,

Tests on an unloaded Ciso 7609 Router showed line rate performance, (Figure 6-4) with no difference between multicast UDP and unicast UDP. The router CPU load was low and the same for multicast and unicast, regardless of whether the ports were set to be switchports or routed interfaces. Note the much decreased time scale for inter packet delay compared to earlier. Here the results show a regular inter-packet latency of 0.56ms, with the points at lower values assumed to be Linux scheduling artefacts. This proved to be stable for a test of 24 hours with no packet loss observed, showing that multicast could be used for real-time delivery of data, given a proven network.

6.4.2 CPU usage

We were able to compare CPU loads running unicast and multicast by simply sending 1Gbps of either unicast UDP or multicast data using iperf between two hosts connected using on-board Intel E1000 NICs through Cisco 7609 switching ports (Figure 6-5).

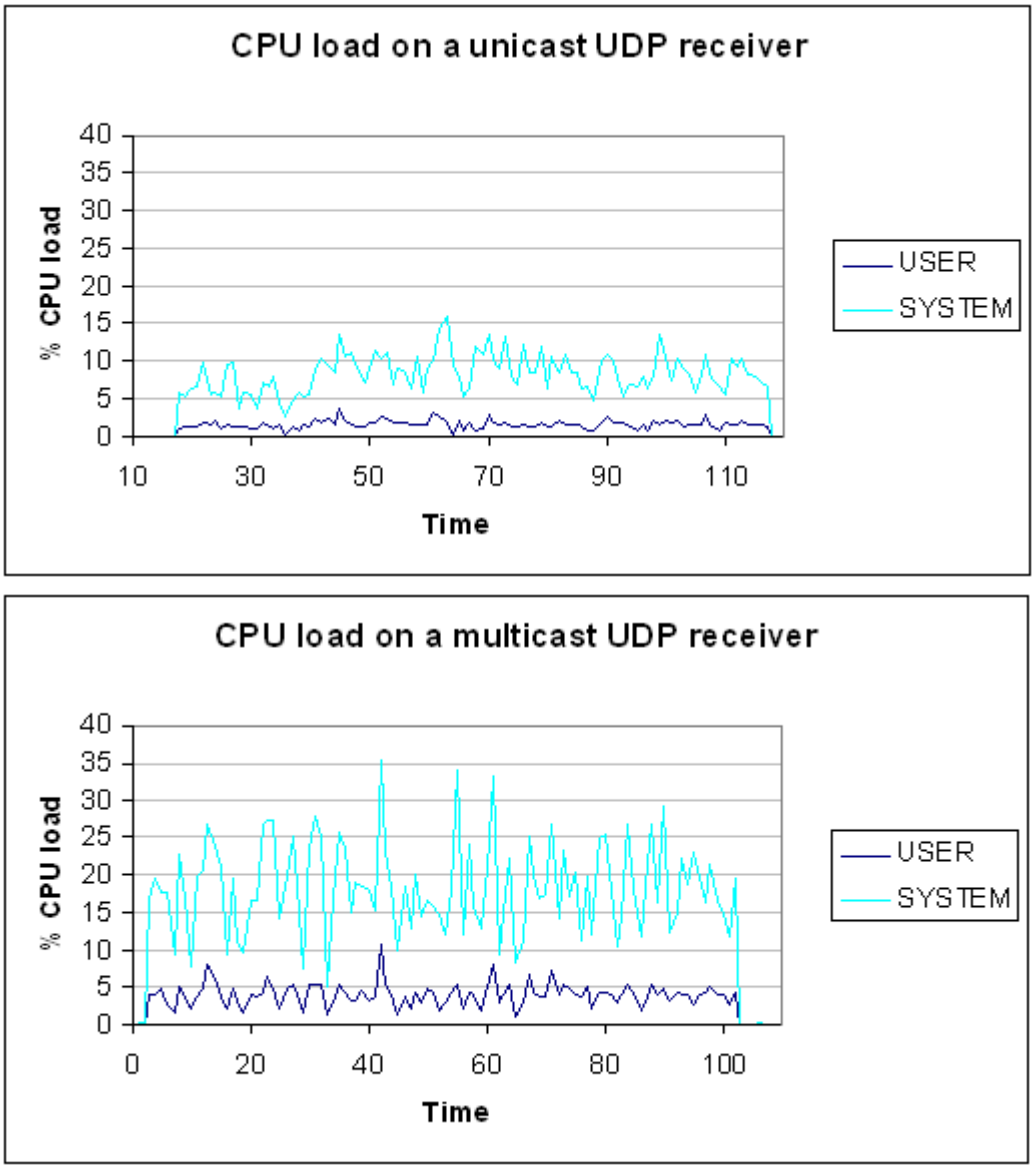


Figure 6-5 CPU load vs time for Unicast (upper plot) and Multicast (lower plot)data flows,

Line rate was achieved and there was no significant difference between CPU load with multicast and unicast flows on the sending machine. Figure 6-5 shows that on the receiving machine the Linux kernel and the iperf application both used more CPU cycles for the multicast transfer than for the unicast transfer. The difference is, however, not huge and it is observed that the CPU requirements for multicast UDP are close to those of unicast UDP and lower than unicast TCP.

Tests over lightpaths were unfortunately not possible over the links available. The IGMP traffic did not transit the links successful and it is assumed that this is due to configuration of switches along the path. Further investigation is required.

6.5 Use of Multicast in production e-VLBI

(Contributed by P. Boven)

In VLBI, the resolving power of an array increases with the longest projected baseline between telescopes. But to properly image both small scale and large scale image structure, it is important to have a goodmix of short and long baselines in a VLBI array. The shortest baselines in the array determine the sensitivity to large scale structures in the image. In regular e-VLBI the baselines (distances between telescopes) span the range of 198 km (Jodrell Bank - Cambridge) to 8419 km (Jodrell Bank - Shanghai). On several occasions, we have been able to increase our longest e-VLBI baselines by connecting telescopes on other continents. Due to the geographic distribution of radio telescopes it has been much harder to include shorter spacings in the e-VLBI array.

MERLIN (Multi Element Radio Linked Interferometer Network) is a VLBI array located in the southern half of the UK, operated by Jodrell Bank Observatory. It consists of 7 radio telescopes with baselines ranging from 11.2 km (Jodrell Bank - Tabley) to 217 km (Cambridge - Knockin).. The data from each 'outstation' is sent to Jodrell Bank via a dedicated microwave link which has a throughput of approximately 128Mb/s.

Two of the Merlin telescopes (the MkII at Jodrell Bank and the 32-m at Cambridge) take part regularly in EVN and e-VLBI observations, so there are two Mark5 recorders available at Jodrell Bank. But Merlin and the EVN can also perform observations together, which greatly increases the number of baselines and the ranges they cover. In these instances, the 'home' telescope (Lovell or MkII) is connected to one Mark5 recorder, and the 128Mb/s data from each of up to 4 outstations are recorded together on the other Mark5. For disk-based observations, the disk-packs with data from multiple Merlin outstations will then be shipped to JIVE, where they are duplicated so that the same dataset can be mounted on multiple play-back units at JIVE. Each play-back unit applies the appropriate delay for its assigned telescope and ignores the data from the other telescopes on the same disk-pack, and the data from all telescopes is then correlated together and processed.

To include more MERLIN telescopes in e-VLBI, the same duplication of data needs to happen on the fly. For our first experiments, we used the 'Port Mirroring' (aka SPAN) facility of the central e-VLBI switch/router at JIVE. By designating the network port for one Mark5 to receive copies for all the packets that are leaving the switch destined for another Mark5, the copying itself is fairly easy to accomplish. But the networking stack on the 'slave' Mark5 will ignore the incoming stream of copied packets because they are addressed to the ethernet MAC address of the 'master' Mark5. And it turns out that the ethernet driver for the Marvell Gigabit ethernet adapter in the current Mark5 systems cannot change the hardware address for the adapter. At JIVE we also have Chelsio 10Gb/s Ethernet interfaces on some Mark5s, and those can easily have their hardware address changed. By setting up a 'slave' Mark5 with identical Ethernet Mac address and IP number, it will then consider all the 'mirrored' packets it receives as valid and deliver them to the application layer. This can of course only work because the e-VLBI data is sent as a uni-directional stream of UDP packets from telescope to JIVE: there is no back traffic at all and we have in fact carried out observations on miss-configured lightpaths that only allowed traffic in one direction. Use of the port mirroring technique for combined Merlin/e-EVN observations was demonstrated successfully on the 22nd of July 2008 when we recorded simultaneous fringes from Cambridge, MkII and Darnhall (see Figure 6-6 below), where the data from Cambridge and Darnhall was sent using the same Mark5 at Jodrell Bank.

The port mirroring setup, although working perfectly, is not really suited to production use. It does not work on all network adapters and requires manual configuration of slave and mirror port on the switch/router, and having duplicate ethernet and IP addresses in the network. IP Multicast seems a

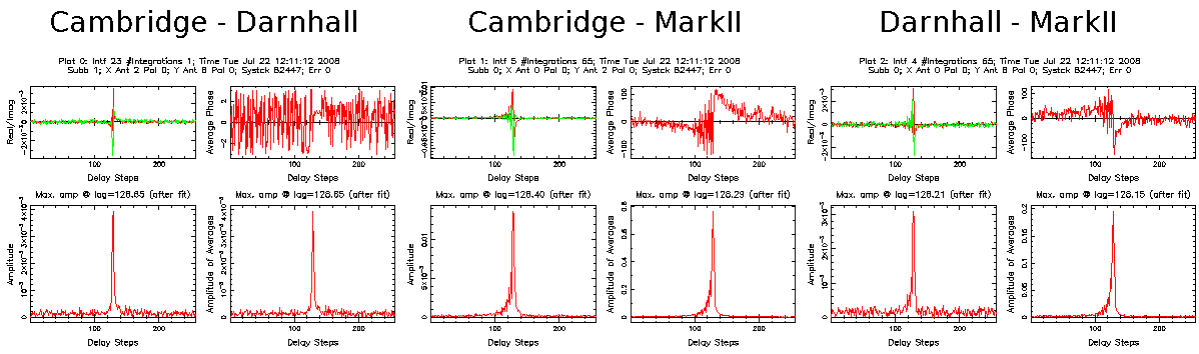


Figure 6-6 Correlator monitor plots showing fringe amplitude vs delay.

perfect match to the requirements of packet duplication for e-VLBI: it is intended for the efficient distribution of uni-directional UDP streams. IP Multicast is a well established and supported internet standard, but its actual deployment on the Internet is still fairly limited. And the data rates for e-VLBI (512Mb/s in this case) are much higher than the Multicast streams that are encountered on the Internet. Jodrell Bank and JIVE are directly connected by two 1Gb/s lightpaths, with only a single 'router hop' separating the UK based telescopes from the receiving systems at JIVE. This router (the JIVE e-VLBI switch/router, a HP Procurve 5412zl) supports Multicast on layer 2 out of the box, but for routing Multicast traffic between VLANs, the firmware of the switch had to be upgraded with the 'Premium Edge' license. This switch is able to handle the packet forwarding and duplication for Multicast streams directly in its switching fabric. We have tested that it can easily handle multiple simultaneous Multicast streams, each 512Mb/s, without any drop in performance or increase in CPU load.

Nowadays all e-VLBI with JIVE is carried out with the 'jive5a' application that was developed at JIVE. We use UDP over dedicated network connections (such as lightpaths). The use of UDP is to completely sidestep all the problems associated with TCP, such as poor behaviour on long-distance paths and implementation specific bugs we've encountered. Adding Multicast capability to this software was a fairly simple exercise and we are now regularly using three Merlin telescopes for e-VLBI, where previously we were limited to only two. Expanding to a bigger subset of the MERLIN telescopes is currently under investigation, but the challenges are now mostly in the areas of RF technology and the intricacies of the VLBI recording terminals.

6.6 Conclusions

As with most results reported here, the network topology is key as to the suitability of the protocol. Multicast is perhaps the optimum method for duplicating data streams, but to ensure that one gains the advantages and none of the potential drawbacks, topology and hardware support is paramount.

Low data-rate multicast has been shown to exhibit no undesirable features in a widely distributed routed environment, once network hardware is functioning correctly. The interpacket spacing is no different to that of unicast UDP, even when scaled to 1Gbps (tested in a more local environment). Receiving multicast UDP incurs greater SPU usage than unicast, but not more than TCP, leading to the recommendation that multicast can be used in place of unicast UDP wherever the benefits are desired. This test showed that contrary to our experience with a heavily loaded network, a lightly loaded network or a light path may be able to cope well with multicast, as has been shown to be the case.
The use of multiple telescopes in the UK requires multiple data streams on the two light paths available from Jodrell Bank to JIVE, and this has successfully been achieved using multicast techniques.

7 Overall Conclusions and Future Work

Our studies over the last two years have shown the following:

1) The concern in VLBI with the timeliness of data arrival means that TCP is not ideal for the transfer of e-VLBI data. However increasing the buffer size by perhaps and order of magnitude allows TCP rates to catch up, and produces timely arrival of data with only temporary but significant delays of the order of seconds. Other TCP variants have improved preformance in this regard but still need large buffers to work effectively.

2) VLBI_UDP has been shown to be able to transport e-VLBI data efficiently. The software is capable of being modified to drop packets selectively. Tests on the JIVE correlator have shown that the correlation system can stand high packet loss before losing synchronisation. Recent developments in selective packet dropping have enabled a closer match between VLBI data rates and those available on networks, with negligible effects on correlator performance., and have now become routine.

3) DCCP is potentially of great use in e-VLBI, however further work is required, both on the implementation, due to stability problems, and in the application. The correct form of congestion control for e-VLBI has yet to be implemented. The controlled flow datagram protocol of choice is not yet apparent for e-VLBI. At the moment UDP (on designated lightpaths) is the most commonly used.

4) Tests on the trans-Atlantic links show that high (>500 Mbit/s) data rate transmission is easily obtainable once the link has been set up correctly. The tests point out the necessity for full testing prior to operations, as the links may not work to full capacity when initially provided. The tests have also shown that all the other network components such as Ethernet switches, routers and Network Interface Cards also need to be tested at the desired data rates.

5) IP multicast appears to be an appropriate protocol for the transfer of e-VLBI data when using distributed correlation, incurring negligible penalties for a vast reduction in traffic, when used with appropriate topologies. However heavily loaded networks are likely to give high latency in unpredictable ways. Local multicast techniques have enable data from multiple telescopes of the MERLIN array in the UK to be used for e-VLBI.

Acknowledgements
We thank Arpad Szomoru, Paul Boven and the staff at JIVE for their invaluable help in this project. We also thank JANET(UK) (formerly named UKERNA) for their flexible approach in provisioning lightpaths in the UK, and Profs. Phil Diamond and Simon Garrington for their support at Jodrell Bank. Anthony Rushton was supported by a research studentship from PPARC (now STFC), and Simon Casey was supported by an e-Science studentship from EPSRC.

8 References

[1] W. R. Stevens, “TCP/IP Illustrated: The Protocols”, Addison-Wesley Publishing Company, Reading, Mass., 1994.

[2] K. Li et. al, “The Minimal Buffering Requirements of Congestion Controlled Interactive Multimedia Applications”, Lecture Notes in Computer Science, 2158, 2001.

[3] C. Krasic, K. Li, J.Walpole, “The Case for Streaming Multimedia with TCP”, Lecture Notes in Computer Science, 2158, 2001.

[4] B. Wang et. al., “Multimedia streaming via TCP: An analytic performance study”, Performance Evaluation Review, 32, 2004.

[5] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, “The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm”, in Computer Communications Review, 27(3), July 1997.

[6] J. Padhye et. al., Modeling TCP Throughput: A Simple Model and its Empirical Validation, in proceedings of ACM SIGCOMM, September 1998.

[7] R. Hughes-Jones, TCPdelay home page, Available at http://www.hep.man.ac.uk/u/rich/Tools_Software/tcpdelay.html

[8] Web100 Project home page, Available at http://www.web100.org/

[9] M. Strong et al., “Investigating the e-VLBI Mark 5 end systems in order to optimise data transfer rates as part of the ESLEA Project” in “Lighting the Blue Touch Paper for UK e-Science - – Closing Conference of ESLEA Project”, PoS(ESLEA)007, 2007. See http://pos.sissa.it

[10] R. Hughes-Jones et al., “Bringing High-Performance Networking to HEP users”, in proceedings of CHEP04 Interlaken, 27 Sep - 1Oct 2004.

[11] S. Casey et al., “Investigating the effects of missing data upon VLBI correlation using the VLBI_UDP application”, in “Lighting the Blue Touch Paper for UK e-Science - – Closing Conference of ESLEA Project”, PoS(ESLEA)007, 2007. See http:/pos.sissa.it

[12] E. Kohler, M. Handley, S. Floyd, “Datagram Congestion Control Protocol”, RFC 4340, Mar 2006, http://tools.ietf.org/html/rfc4340

[13] S. Floyd, E. Kohler, “Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control”, RFC 4341, Mar 2006, http://tools.ietf.org/html/rfc43401

[14] S. Floyd, E. Kohler, J. Padhye, “Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)”, RFC 4342, Mar 2006, http://tools.ietf.org/html/rfc43402

[15] R. Hughes-Jones, DCCPmon Home Page. Available at : http://www.hep.man.ac.uk/u/rich/Tools_Software/dccpmon.html

[16] National Laboratory for Applied Network Research, NLANR/DAST : Iperf. Available at : http://dast.nlanr.net/Projects/Iperf/

[17] R. Hughes-Jones, UDPmon Home Page. Available at : http://www.hep.man.ac.uk/u/rich/net

[18] Manchester Computing, http://www.mc.manchester.ac.uk

[19] UKERNA, United Kingdom Education and Research Networking Association, http://www.ukerna.ac.uk and now also know as JANET at http://www.ja.net

[20] StarLight, http://www.startap.net/starlight