GridNM – Grid Network Monitoring Infrastructure

1st Year Report

Yee-Ting Li

14th June 2002

Abstract

Grid Computing is necessary to aid the High Energy and Nuclear Physics research of today and in the future. Combining globally distributed computers and information sources into a universal source of computing power and information, Grid applications typically need to share the resources available in the Grid. The most predominant shared resource is the network - connecting the various sites that build up the grid. With a predicted 5 terabytes of data being shipped across the internet every day, the challenge now is meeting the ability to transfer data across the internet at ever faster speeds.

We focus on the network characteristics, or metrics, and their relevance to both Grid and Managed Bandwidth Services: We argue that the information about the network topology and characteristic between these computers has to be provided to middleware too to ensure efficiency of network utilization. In order to offer such as service, we describe the design and implementation of a monitoring framework: GridNM, describing it's flexibility in gathering, interpreting and disseminating results on a live WAN environment consisting of various geographically distributed sites across the Europe. Finally we identify further directions of research.

Table of Contents

Abstract....1
Table of Contents....1
Background....2
Network Metrics....3
Network Measurements....6
- Active and Passive Monitoring....6
- Statistical Sampling....7
- Network Measurement Applications....7

GridNM....8
- Overview of Monitoring Architectures....8
- GridNM Context and Description....9
- Modularity and Scalability....10
- Architecture....11
- Dynamic....12
- Security....12
- Tools....13
- Configuration and Data Storage....13
- Presentation....13

Network Analysis....14
- Theory....14
- Case Studies....15
- Case 1: RTT measurements....15
- Future Directions and Conclusion....31

Transport Protocols NG....32
- GridFTP....33
- HighSpeed TCP....34
- IBP....35
- Multicast and PGM....35

Future Directions....35
Conclusion....36
References....36

Background

Currently, there are two types of transport protocol widely in popular use: User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). The former offers a unreliable way of transferring information, i.e., one does not know explicitly that a send packet is received; whilst the latter offers a reliable service; for every packet of information that is sent, some kind of 'acknowledgement' is received by the sending host. UDP can offer a greater raw performance as it does not require the extra overhead of acknowledging information. However, as UDP offers no kind of signalling fromthe network to discover it's state, it can be potentially dangerous if used maliciously (denial of service attacks), and unless used within certain thresholds, could yield potentially worse performance than TCP flows.

Currently, the Internet is based mainly on the connectionless communication model of the IP protocol, in which UDP and TCP is encapsulated and used to get packets across the internet. IP has no inherent mechanisms to delivery guarantees according to traffic contracts and hence mechanisms to reserve network resources have to be done via other means. Because of this, IP routers on a given data path from source to destination may suffer from congestion when the aggregated input rate exceeds the output capacity. Consequently, flows tend to experience varying network conditions that affect the original traffic profile from the source. In order to provision the network such that a guarantee the predictability of network state can be established, we must differentiate traffic flow. This idea is called Quality of Service (QoS) and requires certain high level actions such as classification, metering, policing, marking, scheduling and shaping on the IP traffic.

This is a departure from the standard 'best-effort' model of the current internet. As a result, the internet community have defined the Differentiated Services [DiffServ] (diffserv) model to provide QoS. The implementation is simple; each individual flow of data is marked with a certain 'class of service' from which differentiation is achieved. The definition of a flow can be as simple as a single host conducting a video conference to another host, or as an aggregation of file transfers out of an Autonomous System (AS) who have their own network administration policies. In terms of real world implementation, each router inside of an AS would have the same QoS 'policy' which would guarantee certain characterises of the network performance. The difficulty in implementation is mainly through how different AS's should treat each other's classes of traffic. This is defined in formal management documents called Service Level Agreements [SLA] (SLAs) that define precisely how each AS should treat certain classes of traffic from neighbouring AS's. In technical terms, packets from one AS would be marked or remarked on routers at the edge of an AS (edge router) to a type that has been defined in the SLA's. This (re)marking simply defines each packet to a class of flow defined in an agreed class of service from which it is forwarded to the next router.

The benefit to this approach is that AS's without SLA's could easily be incorporated into the QoS model with traffic marked as the standard best-effort traffic class. In principle, diffserv could then be slowly implemented throughout the internet without interoperability issues.

Diffserv works on the IP layer. Through the idea of encapsulation, putting this at this level has the advantage that higher layers need not have any knowledge of the QoS policies installed and hence could be backwards compatible to all existing internet applications. However, it also imp lies that any transport protocols that rely on the IP layer should automatically take advantage of the increased, or indeed, decreased performance of the network. This is most evident with adaptive protocols such as TCP that tries to adjust to the network. Whilst TCP has adapted well to the improvements in network speed since it's incarnation with link speeds of kilobits/sec, with networks becoming standard at 100mbits/sec, a user is often lucky to get over 40mbits/sec when transferring a file without knowledge of 'tuning' or tweaking. Research Groups such as Web100 [Web100] are putting effort to achieve 100% utilisation of network resources by placing more informative logging information about a TCP connection. However, these efforts seem to be more of a patch to the problems of TCP rather than a real fix, and as such some research groups believe that TCP have any place in the future of High Speed Grid Networks.

The introduction of the Web100 group is a good example of the lack of understanding of the interaction of complex transport protocols and the aggregation of 'self-similar' [self-similar] traffic. Even though TCP has been around for over 25 years, little is understood about its macroscopic effects. This was most probably due, in part, to the fact that there has never been a need to define a framework for performance measurements with best effort networks. By representing the performance of protocols

such as TCP under a methodology that can be easily repeated and compared, we can easily see the benefits and disadvantages of minor or major changes in transport protocol algorithms. Work is currently being conducted by the IETF IPPM [IPPM] group in this respect to define frameworks for network performance measuring and comparisons, and it is hoped that this will lead to a fuller understanding of network protocol interactions and router level implementations and their affect on network performance.

This paper is organised as follows; a brief overview of network metrics are presented in Network Metrics followed by measurement frameworks and techniques through which metrics can be easily obtained and analysed. A description of a measurement infrastructure is presented in GridNM followed with case studies of real WAN network traffic and their implications on user level perception of performance. Lying down a baseline from which performance can be measured, we present a survey of alternative transport protocols and conclude with possible research areas within the field of Grid Transport.

Network Metrics

As the number of internet users raise and the implementation of Grid Computing is successfully implemented, internet traffic volume is likely to grow exponentially. As such, it is essential that a set of network performance metrics and measurements be introduced to both the user and network service providers at all levels to provide accurate and common understanding of the performance and reliability of the connections between nodes and how each component of the internet path affects what performance.

Without this understanding, it would difficult to define how QoS policies should be implemented - it would certainly be impossible to police and provision. By standardising and implementing quantitative network performance metrics, intelligent networking decisions and 'smart' Grid applications can be developed, this could include:

- *Benchmarking*: A user can determine if some network equipment/service is really delivering the claimed bandwidth and or latency. This would be especially important in QoS provisioned networks.
- *Policing*: With a quantitative method of defining performance, and the bounds allowed by the service provider, one may check whether traffic from a domain is within the bounds of the SLA.
- *Cache selection*: Given several sites with replicated data, a client could use bandwidth measurements in selecting the cache that would give it the best performance to retrieve the data.
- *Protocol selection*: By providing a measure of how well different transport protocols perform under certain conditions, a user (either end user or middleware) could select the one that suits their needs more.
- *Protocol tuning*: By adapting a transport protocol within its design parameters to provide performance suitable for at particular application, a quick way of obtaining performance from the network can be achieved. Different degrees of tuning can be easily compared and developed.
- *Protocol development*: By providing a framework from which metrics are universal and well defined, one may quantify how better or worse a new transport protocol performs compared to other transport protocols.
- *Application-level network adaptation*: An application could transform its data based on the current network conditions. Examples of this are real time video applications that reduce their frame rates when bandwidth drops. Understanding of how the network performs is paramount in this case.

There have been several research efforts on network performance measurement and analysis, of which the IETF IPPM is a major contributor. In this section, we outline some of the formal metrics that have been devised or are thought to be useful in determining network performance.

In [NMWG], a metric is defined as follows:

A metric is a quantity related to the performance and reliability of the Internet. More specifically, a metric is a primary characteristic of the Internet, or of the traffic on it. A metric is the characteristic itself, not an observation of that characteristic. An example of a metric is link capacity.

Metrics can be further refined in terms of how they are gathered as follows:

- *Direct measurement of a performance metric using injected test traffic.* One may actually find out the instantaneous value of a metric by directly performing a test to see what it is. This could be trying to contact a node to determine whether it is connected on the internet.
- *Projection of a metric from lower-level measurements.* One could possibly use the aggregation of propagation delays from each router to give an indication of the metric representing the end-to-end propagation delay of the link.
- *Estimation of a constituent metric from a set of more aggregated and identical measurements.* If we were to monitor a single link continuously with different sizes of packets, we may give an estimate of the metric given a certain degree of uncertainty.
- *Estimation of a given metric at one time from a set of related metrics at other time.* If we were to measure the flow capacity of a link at some past time, and given a model of flow dynamics, the current flow capacity could be estimated.

Through the measurement of metrics, we often find that certain metrics can be treated statistically. As such [RFC2330] defines a framework from which we should apply to metrics that have been measured. It defines the following definitions of metrics:

- *Singleton metrics* are metrics that are essentially atomic. For example, a single ping would create a single round trip time metric. We can also define more complex singleton metrics such as a single instance of "bulk throughput capacity" from one host to another. This could be defined as a singleton metric, even though the instance involves measuring the timing of a number of Internet packets.
- A *sample metric* refers to a metric that have been derived from a given singleton metric by taking a number of distinct instances together. For example, we might define a sample metric of one-way delays from one host to another as an hour's worth of measurements. We may also define that for each measurement, we use a sample of 10 pings to determine statistical metrics for each measurement.
- A *statistical metric* refers to metrics derived from a given sample metric by computing some statistic of the values defined by the singleton metric on the sample. For example, the mean of all the one-way delay values on the sample given above might be defined as a statistical metric.

We use the term *node* to define an element on the internet that does something. As such, a router is an internet node that forwards packets, a Compute Element (CE) is one that processes data and a Storage Element (SE) is one that stores data. This distinction is important as each node may have different performance metrics associated with it that is related to its function.

Given the ways in which we can define a metric, we present the types of metrics which are important in defining the characteristic of a network path. One point to note is that each type of metric should be conducted and defined with Type-P packets. Type-P packets are packets that have a certain bit pattern in both the IP level and transport level headers. As such, possible changes in QoS would be presented in differences in the IP header and hence would constitute a different type of packet. However, one still needs to know a priori what policies are involved.

Connectivity

Connectivity [rfc2678] is the basic metric of the internet. It defines whether a node is reachable. This could be as high level as whether the host is physically connected onto the internet, or whether a specific transport protocol port is open for communication. Represented as a Boolean, should a node have false connectivity, then it would not be possible to anything with that specific node.

One Way Delay

The One Way Delay [rfc2679] (OWD) is the aggregation of delays caused by propagation and router delays along a specific path (and possibly host delays). Delay gives an indication of the response of the network which is important to real-time applications; for example, a high value of OWD may result in un-useable tele-immersion applications due to the poor response of the network.

Jitter

Also known as Inter Packet Delay Variation [rfc-ipdv] (IPDV), it defines the factor by which the OWD varies over a certain amount of time. Current draft papers by IPPM suggest that the IPDV should be measured in pairs and the IPDV computed as the difference of the variation of OWD or RTT from the first and second packet given the same source and destination.

Jitter gives an indication of the predictability of inter packet arrival times. This is especially important in applications such as video conferencing that need to adapt to the rate of data flow.

Round Trip Time

The Round-Trip Time [rfc2681] (RTT) is the sum of the OWD of one node to another and then back again. Whilst this metric is basically two OWD's, it is important to note that the internet does not guarantee symmetric paths. As such, this asymmetry may cause the two values of OWD to differ.

Some applications are sensitive to variance in transit time, such as an FTP transfer that relies on TCP to sustain high throughput. Should the RTT be large, one may need to tune TCP to achieve acceptable throughput.

Loss

Loss [rfc2680] gives an indication of the quality of the link between one host and another. Loss can be caused either by router buffer overflow or packet corruption at the physical link layer. It may also occur at kernel and user spaces. Should the loss be high, data transmission is degraded and as a result may lead to poor performance.

Loss is said to be *bursty* when several consecutive packets are lost within a single stream. There is a set of packet-loss based metrics that help with the characterization of the packet loss profile:

- *Loss Distance*: the interval between two consecutive lost packets; it can be expressed in time or in sequence numbers.
- *Loss Period*: represents a sequence of consecutive lost packets.
- *Loss Length*: the number of packets in a Loss Period.

Link Path

Whilst not a quantifiable value, the path link from one node to another defines the path a packet may take to get from the source to the sink. As each node along the link may be changed or unavailable, network traffic along a link may be reconfigured to follow an alternative route. Usually, major changes in other metrics may be attributed to a change in link path.

Path Bandwidth and Available Bandwidth

The path bandwidth is the amount of data that can be carried from one end node to another that is possible. It is a fundamental limit that data can be transferred by the underlying technologies of the network path topology and is the minimal physical bandwidth along a link should there be no competing or cross traffic. It also defines the maximal rate in which data carried in packets can be carried from one end to another.

The available bandwidth can be thought of as the instantaneous achievable bandwidth as a result of cross queuing in routers along a network path and may be lower than the path bandwidth as a consequence of competing traffic. Both these definitions give an indication of the optimal performance bandwidth that the link can deliver.

Throughput and Goodput

The throughput is the speed at which the link can carry data at the transport layer, constrained by the bandwidth. This value may be lower than the available bandwidth of the link due to overheads in transport level headers. It is also known as link utilisation.

It is important to add that we are often more interested in the bandwidth available to an application, this is called *goodput*. The difference between the two values is that *goodput*, under a congestion aware transport protocol such as TCP, would not include data that has been resent due to lost packets, and as such is the data rate of useful information transferred between two nodes (or more if multicasting).

Examples of network metrics for end users and Applications follow.

- *Video Conferencing*: whilst the technology is already in place to offer good quality video conferencing facilities over the internet, there are times during heavy internet use that the

video conference quality deteriorates to such a degree that it is no longer usable. This is often due to a sudden change in OWD in the network as Video Conferencing is often a relatively low bandwidth Application, i.e. it only needs a relatively low goodput. Newer video and audio conferencing Applications can often adapt to changes in OWD and hence a predictable variance (IPDV) is required.

- *File Transfer*: Whilst most users are quite happy to wait for a large file to download, and as such can be given a 'lower' quality of service, there may be times when downloading a large file may need priority. An example would be an imaging application from which a user interacts with. The longer the time that the file takes to download, the longer (and more impatient) the operator has to wait. The metric that we are concerned with in transferring a file is goodput. As stated above, goodput depends on many different metrics of the internet and as such IPPM have defined a metric called Bulk Transfer Capacity [rfc3148] (BTC) that define that goodput should be measured such that the transport protocol is in steady state (should it incorporate it), and that explicit descriptions of the transport protocols should be given.
- *Web Traffic*: The current explosion of the internet has been due mainly to web traffic. While similar to transferring a file, web traffic involves transferring many small files instead of one large one. This has a major consequence in goodput as all web traffic is currently based on TCP (through HTTP), which is highly correlated to the performance of TCP during its slow start period. As a result, goodput for web based traffic is often lower than that for other Applications such as ftp.

Network Measurements

In order to obtain metrics, we must define a measurement methodology in which we will use to acquire metrics. The idea is that by defining and stating a measurement methodology, the metric should be repeatable with comparable results under different circumstances. It is important to note that repeatable does not necessarily mean identical as two consecutive and identical measurements of goodput may yield different results.

Active and Passive Monitoring

In order to understand how a network performs, we must actually conduct tests to probe the network. Tests can be defined under two different approaches, active and passive.

The passive approach

The passive network monitoring approach does not increase the traffic on the network when conducting measurements. It also measures real traffic through measuring the network behaviour by observing the packet arrival rate at nodes and make some deduction on the state of the network, and therefore deducing the network performance. We may use devices to watch the traffic as it passes by. These devices can be special purpose devices such as a packet sniffer, or they can be built into other devices such as routers, switches or end node hosts. Examples of such built in techniques include Remote Monitoring [StallingSNMP] (RMON), Simple Network Monitoring Protocol [StallingSNMP] (SNMP) and netflow [netflow] capable devices. Passive monitoring devices are polled periodically and information is collected to assess network performance and status.

However, the polling required to collect the data and the traps and alarms all generate network traffic, which can be substantial. Also, the amount of data gathered can be substantial especially if one is doing flow analysis or trying to capture information on all packets.

The passive approach is extremely valuable in network trouble-shooting, however they are limited in their ability emulate error scenarios or isolating the exact fault location.

Since the passive approach may require viewing all packets on the network, there may also be privacy or security issues about how to access/protect the data gathered.

The active network monitoring approach relies on the controlled injection of test packets into the network. This may involve setting up Application servers at the receiver or sink end and sending packets to it. As such, active measurements create extra traffic, and the traffic or its parameters are artificial. The volume and other parameters of the introduced traffic should be altered as to not affect the existing network traffic as much as possible, yet active monitoring traffic volumes should be large enough to obtain meaningful measurements.

Active measurements are often used to give an indication of the network performance as observed by end nodes. And as such can be use to check if QoS or SLA 's are met as testing under certain policies should guarantee certain network performances (hence metrics) of which can be compared against the metrics obtained through active testing.

Given the complementarily of the two mechanisms, we need to explore ways to get the best of both worlds. A possibility is for the active measurement probe to schedule passive measurements of appropriate metrics at appropriate points along the path, while the active measurements are being made. When the active measurement is completed then the appropriate passive measurements can be paused thus reducing the gathering of unnecessary data. By comparing and contrasting the active and passive measurements, the co-validity of the different measurements can be verified, and much more detailed information on carefully specified/scheduled phenomena is made available.

Statistical Sampling

An important issue in obtaining measurements is the sampling pattern used to collect the individual observations of the metrics. Some techniques, such as packet traces from `tcpdump` [tcpdump], capture all details during the trace; it may not be feasible to gather and analyse and those traces continuously due to the volume of data collected. Therefore, a technique in which the interval at which measurements are made should be specified.

Whilst sampling techniques are varied, there are primarily two ways of sampling network metrics:

- *Periodic intervals*: beginning each observation at a consistent interval
- *Aperiodic intervals*: typically distributed according to a Poisson or geometric distribution.

Periodic sampling is a method by which measurement are taken with a constant and consistent interval between measurements. For example, one might send ten ICMP packets every hour on the hour to determine the minimum, maximum and average round trip statistical metrics time of a particular link. Having defined these statistical metrics we can obtain a good view of the condition of the link over a period of time. However, this method of sampling could miss periodic patterns in traffic flow that could occur between our measurements; for example, if there is consistent router buffer overflow at half past every hour, our ping measurements would never see this periodic pattern as we are only taking measurements on the hour, every hour. We could account for this by increasing the sampling frequency (i.e. decreasing the sampling interval) against the interest of reducing the intrusiveness of our tests. Therefore, we may always end up sampling network metrics at low points in network activity and hence reduce the effectiveness of our measurements.

Another factor with periodic sampling is that it may also cause extra traffic load should the sampling occur (either incidentally or accidentally) at network peak periods, possibly overloading the router and hence interfering with real network traffic.

Another method of taking network measurements involves conducting tests under with a random yet predictable aperiodic interval. By varying the interval between samples, we hope to increase the possibility of obtaining a true measure of the network state. By varying the interval between measurements under a random but statistically predictable distribution, effects of noise induced through temporary increases or decreases of network performance are reduced.

It is also important that if we are to conduct many measurements at the same time, we do not interfere with our own measurements. For example, if we were to run two goodput measurements at the same time from the same source-sink pair, the contention between the two measurements could yield a result that is less than it would be for just one measurement.

Network Measurement Applications

Whilst passive monitoring has the advantage of not disturbing the network as we do not inject any additional traffic to conduct the measurement, passive monitoring often requires access to MIB's in routers and switches which are often not available due to administration security. Also, monitoring SNMP traffic at a node in a link also requires the isolation of a particular flow for analysis, which may not be possible.

Active network monitoring often involves executing a program on a host which generates traffic for statistical analysis. These Network Measurement Applications (NMA) include common programs such as ping and even ftp (although ftp could be argued as passive monitoring if we use it to transfer 'useful'

files). Often running a tool will present network metric(s) that can be used for analysis. For example, running ping will give the metric of rtt for a series of Type ICMP packets. This sample set of metrics can then applied to give a statistical metric such as the mean rtt of the measurement. We could also derive other metrics such as the standard deviation and percentile distributions of the measurement. Most tools, on the other hand, often only provide one metric as a result of running the measurement. NMAs such as iperf [iperf] will generate real tcp traffic and will after a set period of time terminate and report the average throughput or bulk transport capacity of the link.

Other NMAs use statistical methods to determine metrics such as the bandwidth of a link. Tool such as pathrate [pathrate] and pipechar [pipechar] send packet trains of udp/tcp or icmp packets to work out the dispersion of packet arrival times. This can then be used to work out the path bandwidth of the link. Other NMAs such as sprobe [sprobe] and pathload [pathrate] can also be used to estimate the available bandwidth of a link. These tools also use packet pair/trains to statically calculate the available bandwidth. However, these tools do not generate the traffic of a real flow like iperf and are hence less perturbing to the network but may result in less accurate metrics.

With such variation in NMA's and interpretation of network metrics, it is important to understand and analyse the performance of these tools. This performance criterion can be based as follows,

- The accuracy of the NMA compared to other active and passive techniques.
- The frequency of measurement to make the NMA metrics meaningful and useful.
- The amount of time the NMA requires to run and present its metric(s).
- How intrusive the NMA is for existing network traffic.
- How applicable the NMA is to user level performance.

Whilst NMAs provide user level access to network metrics, it also presents a measurement methodology from which we can conduct network measurements. As each NMA has a well defined way of generating and analysing packet information.

GridNM

Overview of Monitoring Architectures

With the unprecedented volumes of data being gathered by current and future High Energy and Nuclear Physics experiments, it is anticipated that Grid technologies will be deployed at hundreds of sites worldwide for storage, searching and processing capabilities. Projects coordinating these technologies include the Particle Physics Data Grid, GriPhyn and GridPP. Interconnected worldwide, these storage centres form the backbone of tens of thousands of computers and storage devices. This infrastructure will require the ability to sustain long periods of transferring large amounts of data between collaborating sites with possible real time interaction required.

With increasingly complex structures that defy traditional modelling techniques developed for early simple networks, the performance of today's networks cannot be accurately predicted with conventional modelling techniques and tools, thus making modern traffic engineering, network management, and planning a difficult and challenging task. Network measurement and analysis provide alternative and comparative techniques to complex statistical techniques that are often based on unrealistic approximations.

Through implementing an architecture for network monitoring we can provide the basis for diagnosing and predicting the end-to-end performance of high-performance network through passive and active measurements.

The impetus behind any Network Monitoring Infrastructure (NMI) is the ability to baseline the performance of the network. A baseline is a value or profile of a performance metric against which changes in the performance metric can be usefully compared. For example, if we know that the average utilization of a particular link over the last week, this average could be used as a baseline against which to compare future changes in the utilization of that link.

Through inference from a baseline metric, network performance reports may help users figure out the quality of the network service and the available bandwidth, as well as identifying the traffic pattern on the network in order to avoid heavy traffic and also to characterize network links between sites.

Network measuring has three basic functions: to collect, store, and analyze metrics and traffic traces. The collection of network metrics often involve the use of a specifically designed NMA which can be run to either collect or infer network metrics. NMIs often wrap up these NMAs as to store and present the collected metric data into a useful format. For example, one may wrap up the ping program to conduct tests to a particular host every hour in order to obtain a metric of rtt and loss. As such they mainly differ in terms of their complexity and presentation of metrics.

AMP

Similar to the PingER project [PingER], the Active Measurement Project [AMP] (AMP) is an infrastructure is based around two fundamental NMAs: ping (rtt) and traceroute (network path). Each node in the AMP infrastructure conducts network measurements using ping and traceroute periodically to every other node in the designated configuration. These mesh measurements allows a fuller picture the entire network state of intertwined links and can give information such as performance asymmetry and whether network anomalies are associated with a particular link or with a particular network site. Under AMP, data collection and processing of metrics are stored on the remote node and polled every day to a central repository for presentation. Presentation of data is through graphs of rtt as a function of time. Text lists of traceroutes information are also available.

NIMI

The National Internet Measurement Infrastructure [NIMI] (NIMI) is one that is similar to other very large scale Internet services such as Domain Name Service (DNS), the Network Time Protocol (NTP) and the USENET system. The approach of these architectures have been successful due to the hierarchical, distributed architecture allowing their infrastructure to grow without significant central administration. NIMI consists of especially configured probe machines situated through the Internet that will be used to monitor end-to-end (e2e) performance. The information provide by the probes can also be used to localise faults to particular pieces of the infrastructure.

Of most interest in this proposal, is the framework of a national Internet profile that is constructed from measurements produced by NIMI probes. Using the network performance measurements, they plan to assess the health of the national Internet as a whole over time.

WP7

Working Package 7 is the group of the Euro DataGrid [EDG] (EDG) that deals with Network Services. As part of their infrastructure, they have released a set of monitoring tools that wrap up NMAs to present network state information. The current toolset include ping (rtt), iperf (goodput) and udpmon (udp throughput). Based around the PingER infrastructure, each node is responsible for configuration and testing to any other defined. In the framework of the EDG, there have been a set of specific nodes that have been nominated for testing.

The full set of network monitoring tools of WP7 has to be installed on each node that wishes to take part in the network measurements. Each nodes network metrics are then available through a web front end located on that node; so even though the network monitoring infrastructure is a mesh, presentation is segregated over the entire measurement infrastructure. Graphs are available as a time series of each separate metric or as a frequency graph.

IEPM-BW

IEPM-BW [IEPM-BW] performs network measurements from a single selected node to a series of other nodes. NMAs that are currently being used to monitor nodes include ping (rtt), traceroute (network path), iperf (goodput), bbcp (goodput) and pipechar (network capacity).

Presentation is through a central web interface located at the source node that shows a summary table of all monitored hosts with all monitored tools. As well as time series graphs, frequency and NMA comparison graphs are available. They also provide a facility to compare the performance of different end nodes based on operating system.

GridNM Context and Description

Whilst all of the above measurement infrastructures have been well defined and tested, each NMI has certain disadvantages. APM is currently limited to tests involving only ping and traceroute, WP7 tools are difficult to use to analyse data from many sources and difficult to centrally administer. IEPM-BW - whilst the most informative, is limited to unidirectional tests.

As a result, we decided to build our own NMI based on existing technologies with the emphasis on being able to disseminate and collect useful metrics for network diagnosing and performance base-lining.

In order to analyse data efficiently and effectively, we must understand the performance of applications that make measurements. As NMAs also represent a measurement methodology, NMIs should be flexible to incorporate new NMAs that are developed. As little research has been conducted to compare the performance of different NMAs, NMIs should also allow comparison of the different NMA metrics against each other. From this information it should be possible to determine and manage a final set of NMAs and parameters that are known to be reliable and useful for network diagnosis and base-lining.

It would also be beneficial if we could gain control over how the NMAs function so that we could leverage the usefulness from the tool. A specific example would be to perform a series of iperf transfers with varying window sizes to compare the performance over time.

Due to asymmetry of internet paths, it would be beneficial to obtain bi-directional link information between a mesh of nodes rather than from a single source. This would imply a mesh infrastructure for our NMI. For certain types of network traffic, such as BTC, that depend on both forward and return paths for optimal performance, such measurements should give useful insights into both network and protocol performance. Mesh measurements would also give insight into whether performance of a node is related to a particular link or the site in general.

NMIs should be easy to manage and configure and adaptable to the needs of the nodes under test. Specific examples include measuring from different OS's and firewall configurations in a nodes' domain. This may be as simple as limiting the number of measurements to the domain, to performing only certain measurements.

Another useful feature of an NMI would be the ability to conduct tests on demand rather than at preset intervals. This, for example, could be used to determine the current state of the network should a Network Cost [NetworkCost] function be applied to determine how long a transfer might take.

In summary, we can define the requirements of a NMI with the following needs:

- *Scalability*: It should be easy to add nodes onto the test bed. Should the architecture conduct tests between nodes as in a mesh system, we should make sure that we do not conduct tests between nodes at the same time as another test. This would guarantee that we do not interfere with out test own tests.
- *Dynamic:* Monitoring of network nodes should be modifiable 'on the fly'. The same should apply for the addition and removal of NMAs to monitor nodes.
- *Robust:* Different nodes may require different settings from each MNA. Examples include firewall blocked ports which may require relocation.
- *Flexible:* We should ensure that tests between nodes can be easily turned on or off. An example may be that a certain AS may not want iperf traffic coming in or out of their domain.
- *Secure:* The nodes participating in the test bed should be at least as secure as they were before taking part in the infrastructure. As such the network monitoring infrastructure should not impose any additional security risk.

Whilst IEPM-BW does meet all of these criteria, it was unknown us that the project was underway. As such many of the features of GridNM are also available in IEPM-BW. We are currently seeking collaboration with SLAC to coordinate in a joint infrastructure program.

Modularity and Scalability

The diversity of remote nodes needs to be considered. Different hardware, operating-systems, network interfaces, file locations, user ids and firewall settings need to be taken account before conduct a test. Also diversity of NMAs and their corresponding options and specific node parameters need to be taken into account. Similarly, some NMA require a server to be initiated on the remote sink before a test may commence.

We define end nodes participating in the test bed as *hosts*. We define the period in which we perform a network measurement with a tool as a *test*. We define conducting all tests in a time frame as a *test phase*. Hosts may be also defined as source or sink nodes corresponding to whether tests are performed at the node or is the target of a test. We also define a specific configuration of an NMA as a *tool*. An

example is an iperf NMA with specific tcp window size setting; similarly, we may have two ping *tools*, each with different size packets used for measurement of rtt.

In order to create a flexible way of storing and configuring all this information we used XML. Two xml files are defined; `hosts.xml` and `tools.xml`. The `tools.xml` stores configuration information such as the name of a tool, corresponding generic command lines and specific defaults of each configuration of a NMA. As such, it defines an instance of a tool rather than the NMA itself. Similarly, the `hosts.xml` file gives configuration information on each instance of a node or host. It stores information such as the username for login, and contact information for that node. It also contains specific information on NMA file location and parameters for each NMA. The latter may be the port number of a NMA that we should use to conduct tests to.

We defined a mesh infrastructure in which each host can act as both a source and sink for network tests. We can also define offer the option to configure tools and hosts to participate in the test bed. This is possible through a simple xml element in each xml file defining whether the tool or host is 'online'. We have also implemented filtering mechanisms that can be used to preventing the testing of a particular tool to a host, or a host to another host. This is defined under the `</exclusions>` element defined in the xml files. Is possible also possible to define whether it is necessary to run a server for a tool if it needs one, or to use an existing server available on the sink host.

Architecture

A simple and controllable method for initiating and collecting job information is used. This is done through automated ssh access to relevant nodes in the test bed. Currently, NMA executables are distributed by hand to end nodes, however, it is recognised that an automated method of file distribution will be necessary in the future. This may be to use tools such as `rsync` or even simple `scp`'s on the executables stored on a central server.

The use of this central server is also used for metric logging and sending commands through ssh to hosts to conduct tests. Processing of raw logging information from NMAs is also conducted at the central server. As such, the requirements of the nodes in the test bed are minimal – requiring only an ssh account with storage space to hold the NMA's executables.

A set of perl scripts complements what is essentially the backend of the GridNM infrastructure to collect, disseminate and store the data gathered.

- *host.pl*: deals with all matters to do with host information processing such as obtaining username information, obtaining specific tool information like server port numbers and settings.
- *hool.pl*: similar to host.pl but specifically to do with tools and the command line parameters. Note that values supplied in tools.xml may be overridden by settings in `hosts.xml`.
- *data.pl*: processes and stores log data from a measurement.
- *ssh.pl*: a simple interface to allow execution of a specific command on a host.
- *test.pl*: deals with overall test bed control and command line generation.
- *linear.pl*: interface to run a test with a tool between a source and sink.
- *mesh.pl*: interface to run all tests on all sources and sinks.
- *server.pl*: command line to start and stop remote tool servers.
- *conduct_test.pl*: command line executable to run one or all tests between some or all nodes with interactive display.
- *GridNM*: the main GridNM server that runs in the background to conduct tests.

The interaction of the scripts is shown in Figure 1.

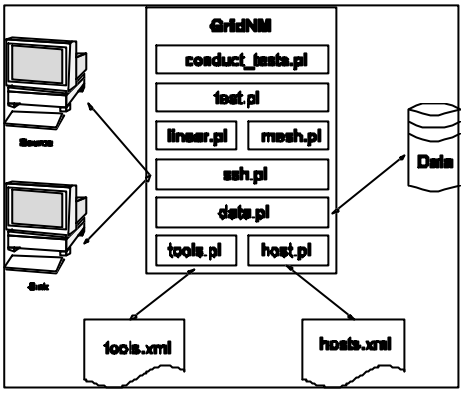

Figure 1. GridNM architecture.

Dynamic

As we are dealing with mesh measurements between hosts, it was important to prevent performing tests to hosts which are currently under test. As such, a locking mechanism was implemented in which each configured tool would poll for contention to conduct a measurement between a specific source-sink pair. Should a pair of hosts be in use by another tool at any particular instance, another tool wishing to perform its test between the hosts would gracefully wait a random period and before trying again.

Different tools may also take varying lengths of time to complete a measurement. An example is that ping will only take one rtt to complete for one measurement. Tools such as pipechar, on the other hand, can require up to 5 minutes to complete a measurement. As such, a mechanism to timeout a test was introduced. This is especially important as the locking mechanism mentioned above could prevent future measurements from taking place. Instead of predefining a test length from which all tools must finish by, we decided that each tool should have a user definable timeout implemented as part of its definition. The xml element `<test></timeout></test>` located in `tools.xml` defines this value. Should the test take longer than this value, it terminates gracefully and removes any lock files associated with the test.

It was also recognized that it may not be best practice to conduct the tests in the same order for a particular test phase. As such we decided to randomize both host and tool test orders according to a flat, unbiased distribution. The randomization of the tools would also ensure that long measurements do not necessarily hold up other shorter tests. In order to factor in contention, we decided to bias the frequency of polling of tools to conduct a test relative to the value of the timeout as defined in each tool. This would bias the tools such that those with a lower timeout would have a higher contention probability and hence complete faster.

The currently version of GridNM incorporates periodic sampling of all tools. However, the order of tests performed by each tool to each set of hosts is randomised prior to each test phase. By doing this, we can prevent traffic patterns from tests affecting other tests. An example may be that sending large pings prior to performing an iperf transfer may prime router queues hence possibly improving performance for the subsequent test. We hope to incorporate aperiodic sampling under well defined statistical distributions for comparison of sampling techniques. Contention of tests between tools also introduces a further factor of randomisation in the interval of measurements.

As some NMAs may require lower measurement frequencies, we implemented phased testing. This allows each test phase to only conduct tests with certain tools. This allows flexibility in reducing network traffic induced by active measurements by lowering the frequency measuring with certain tools.

Security

In order conduct tests in a mesh, remote access was required. This was achieved through standard ssh keys. The nominated central server is used to communicate with all hosts automatically. As such it is required that its dsa public key is distributed to all hosts in the test bed. It is also important to ensure that the public key does not require a pass phrase to allow automatic login from the server to the test bed. The use of ssh keys has the benefit of using a well established and secure infrastructure to conduct tests from. Future incarnations of GridNM will hopefully also use globus keys.

One of the current threats of the internet is the denial of service attack. As we are dealing with potentially intrusive measurements, it may be unwise to leave a NMA server running on a remote host. As such, a facility to initiate and shutdown a remote server before and after each measurement was implemented into the architecture. This also removes the need of remote `cron` jobs to ensure that remote servers were running. This functionality is also extended such that if a test timeouts, the remote server, if running, will terminate.

Tools

NMAs currently used to monitor hosts are ping, iperf, udpmon [udpmon], traceroute [traceroute] and pipechar. As these tools conduct a measurement from a specific source node to a specific sink node, we define them as *link tools*. In addition to NMAs monitoring the network link status, *host tools* were also implemented. A host tool is defined as tools that can monitor more general variables other than network metrics; such tools include cpu utilisation, memory usage and tcp socket buffer sizes. Under this definition SNMP agents would also be defined as a host tool. Whilst the metrics obtained by host tools may not necessary relate directly to the performance of a network, they do contribute to network performance. The typical example is that a host with high cpu utilisation will see a degradation in network performance as a result of scheduling. The type of tool is defined in the `tools.xml` file under the `<\type>` element.

Configuration and Data Storage

Information on metrics gathered from each NMA are not processed on the remote host. Instead, the raw log from the NMA output is sent back to the central server where it is processed and stored.

The collected data is stored in a user defined directory and is structured such that all measurements from a specific source host are stored under the same directory. Similarly, measurements from link tools to a specific sink from that source are stored in a directory underneath that source directory. Host tools are stored, logically, under the source node only.

The standard format of storage is NMA centred rather than network metric centred. Logging of each measurement is stored in a file beginning with the name of the NMA followed by a date stamp. Each file holds a months worth of data stored in standard ASCII tab delimited files for the storage of measurements. The format of these logs are defined in the `tools.xml` under the `</logging>` element.

Presentation

Whilst many existing network monitoring architectures supply web based presentation tools, it can be sometimes difficult to find specific information or be able to compare specific metrics against each other for a particular situation.

By defining the storage of GridNM data, and maintaining a xml reference system for log information, it is possible to 'plug-in' any front-end that recognises the xml `</logging>` element and the simple structure of the file storage system.

Currently, a java based applet is available for interactive analysis of network metrics stored in GridNM. It allows metrics to be added or removed from graphs, separate graphs to be compared to each other and filtering of the data to be conducted. It also allows plotting of frequency graphs and the comparison of different NMAs metrics.

The work on the GridNM java front-end is currently under development and is shown in

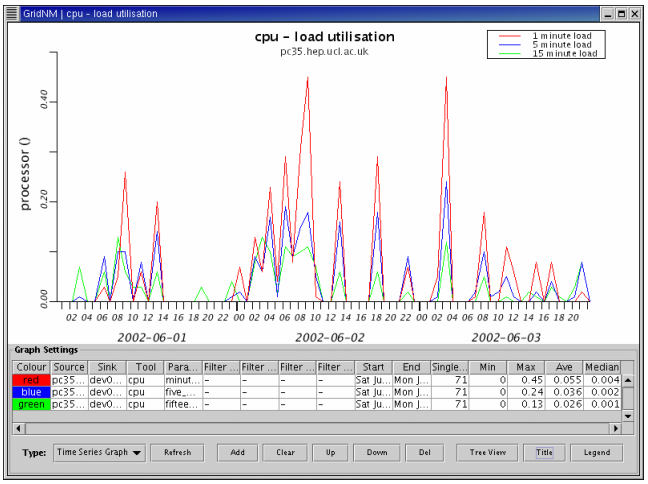

Figure 2. GridNM 'plot' front-end.

Network Analysis

Theory

By utilising the GridNM architecture, we were able to obtain numerous network metrics to analyse network characteristics between 4 nodes around the UK and a couple of nodes in the Netherlands (NL) and at CERN. We were fortunate enough to monitor hosts with Gigabit Ethernet (GigE) connectivity. This should allow us to monitor high speed links. However, the links of these hosts are shared with other network monitoring project and thus prone to a degree of noise. However, through monitoring this noise can be characterised as the state or condition of the network and is the normal obtainable performance from the network.

The need of network monitoring over the WAN needs to be compared against lab tests. However, due to the intrusiveness of most NMAs, we are limited by the number of tests we can perform as to not decrease the performance for other network users. As such, we plan to collect a lot of data over a period of time to compare to lab tests which are controllable and repeatable.

It is hoped from comparisons between WAN and lab test that a framework and mathematical model can be created to characterise traffic and relationships between network metrics. From this framework, it is hoped that it will be possible to improve the performance of transport layer protocols to make the most of available network bandwidths and characteristics.

Network monitoring can also be used to baseline the performance of links between hosts. For example, by analysing the performance of links between CERN and the rest of the world, one may be able to provision the network around CERN to improve file transfer times, for example. One may also be able to find patterns in network state over time and be able to time dataset distribution as to minimise the effect on users.

We present case studies that show the applicability and usefulness of the GridNM infrastructure in providing a tool for network analysis.

Case Studies

Case 1: RTT measurements

High Performance Link

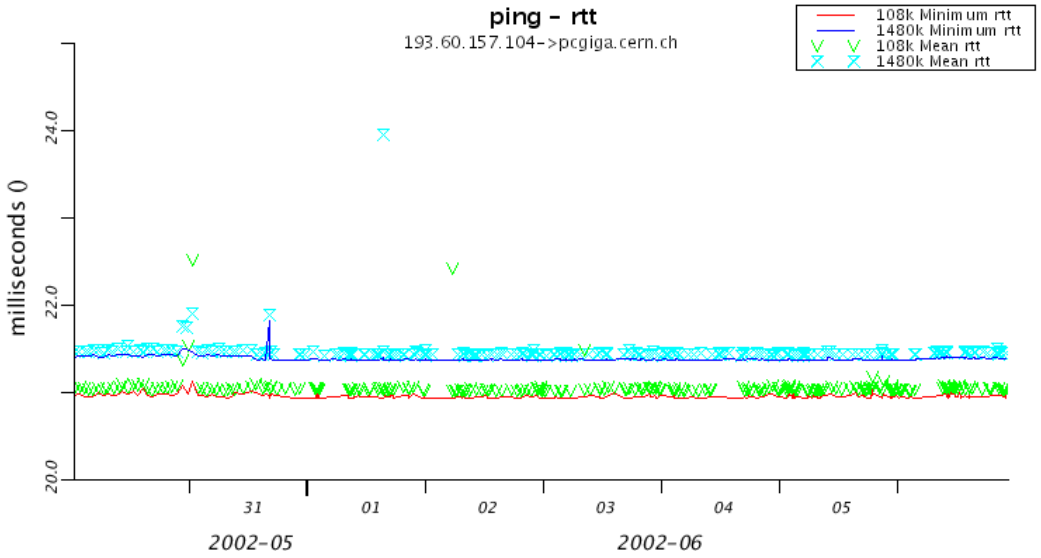

Figure 3. Example of high performance network connection.

Figure 3 shows a connection of a good link between Manchester and CERN. It is characterised by the lack of rtt variation. The similarity of the rtts for different packet sizes also implies a fast link as the slowest node in the path can process and send different sizes of packets with little time difference. The high correlation between the minimum and mean rtts for each packet size also implies that there is little congestion along the link that would cause the packets to have to queue for long before being serviced.

Asymmetry of Paths

Figure 4 shows a relative good path between UCL and RAL. Whilst the minimum rtt is quite constant, the degree of variation of rtts for each measurement is shown by the Mean rtt.

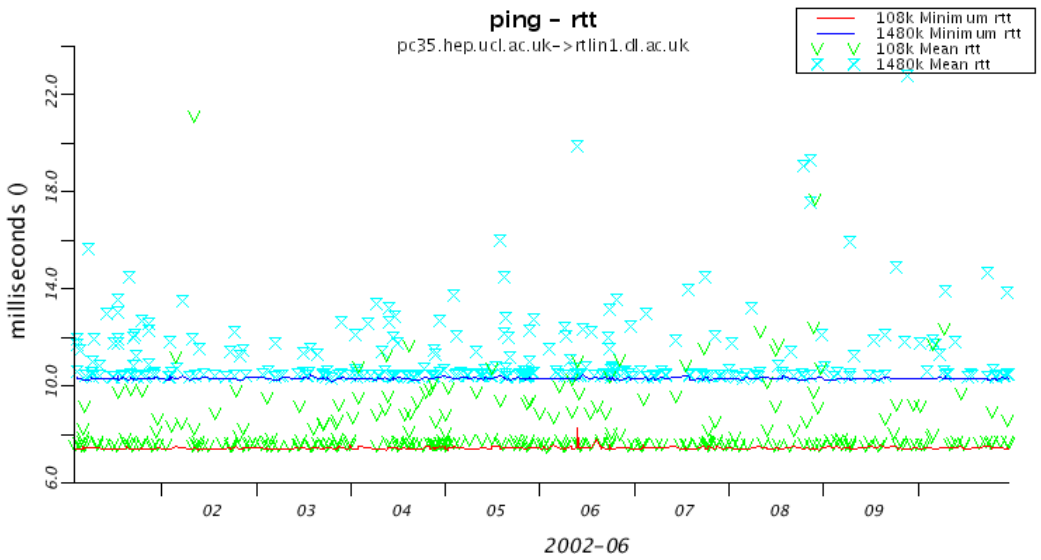

Figure 4. rtt from UCL to DL.

However, looking at the reverse path for the link, i.e. from RAL to UCL shows a different story. Figure 5 shows that the variation in the minimum rtt is substantial. The clustering of rtts for 1480k packets at 10ms also imply that the performance of the route is hampered by long transient queues in routers

along the path from which our packets consistently arrive at the end of the queue – whilst few packets manage to arrive in periods with small queues and hence are able to obtain a lower rtt.

The clustering of small packet sizes at the bottom of the distribution also imply smaller packets are possibly serviced first. Hence it is possible that some form of policy is in place to serve smaller packets in priority. This is shown in Figure 6. This would be related to the scheduling mechanisms in place in the routers along the path.

As such, it would be interesting to monitor this path with OWD measurements to determine whether the poor performance of the UCL to RAL path for the mean rtt is caused by the even worse performance of the RAL to UCL route.

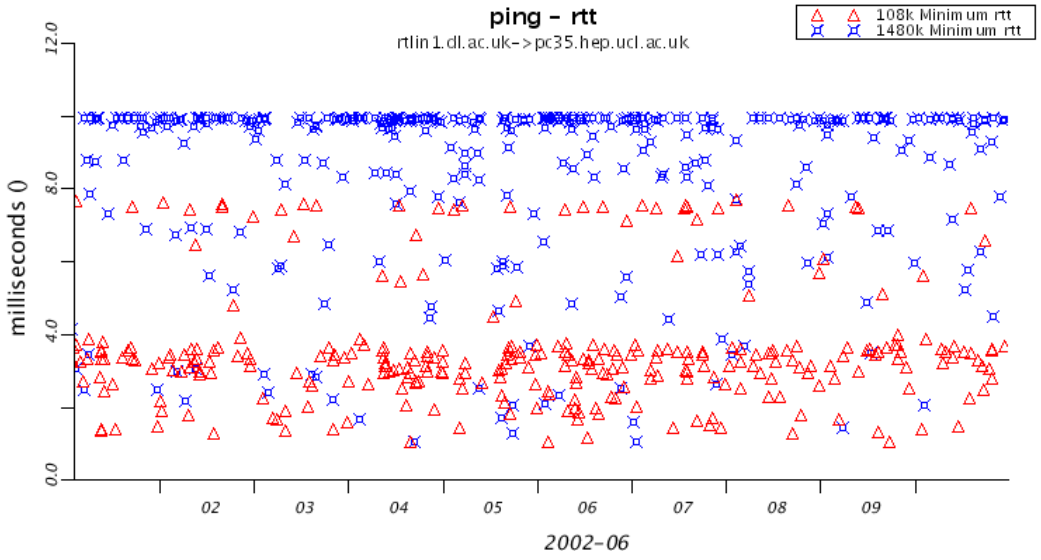

Figure 5. rtt from DL to UCL.

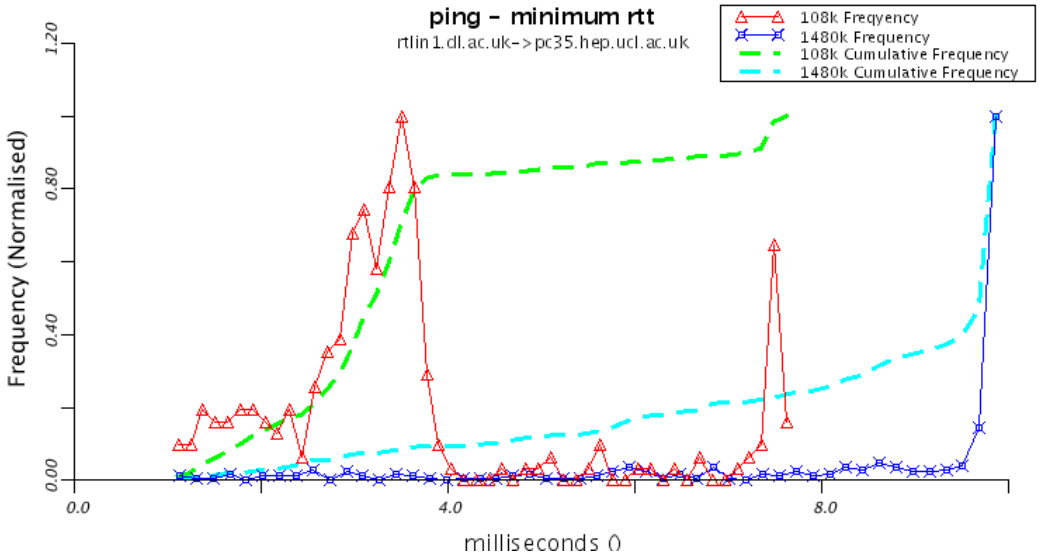

Figure 6. Frequency distribution of pings from DL to UCL.

As rtt requires a path from the source to sink and back again, the traceroute of the path between DL and UCL is shown in Table 1. It shows a slight change in path on the route back. The set of unnamed routers prior to the arrival at `pc35` from `rtlin1` is constant and relatively lightly loaded. Assuming this is the case throughout the measurements, and that we have symmetric queuing, we can say that the poor performance of the link is caused most likely at DL. The addition of `alan3.dl.ac.uk` in the reverse path could suggest it as being the bottleneck in the link – causing the high variance in the distribution of rtt, and would warrant further investigation.

UCL to DL	DL to UCL
pc35.hep.ucl.ac.uk:128.40.4.35	rtlin1.dl.ac.uk:193.62.119.20
128.40.4.245:128.40.4.245	alan3.dl.ac.uk:193.62.119.1
128.40.20.149:128.40.20.149	gw-fw.dl.ac.uk:193.63.74.131
128.40.20.190:128.40.20.190	nnw-gw.dl.ac.uk:193.63.74.232
ulcc-gsr.lmn.net.uk:194.83.101.5	gw-nnw.core.netnw.net.uk:194.66.25.29
london-bar1.ja.net:146.97.40.33	manchester-bar.ja.net:146.97.40.177
po9-0.lond-scr.ja.net:146.97.35.1	po10-0.warr-scr.ja.net:146.97.35.45
po2-0.read-scr.ja.net:146.97.33.74	po1-0.read-scr.ja.net:146.97.33.53
po0-0.warr-scr.ja.net:146.97.33.54	po1-0.lond-scr.ja.net:146.97.33.73
manchester-bar.ja.net:146.97.35.46	london-bar1.ja.net:146.97.35.2
gw-nnw.core.netnw.net.uk:146.97.40.178	ulcc-gsr.lmn.net.uk:146.97.40.34
gw-dl.netnw.net.uk:194.66.25.30	ucl.lmn.net.uk:194.83.101.6
gw-fw.dl.ac.uk:193.63.74.233	128.40.20.189:128.40.20.189
rtlin1.dl.ac.uk:193.62.119.20	128.40.20.150:128.40.20.150
rtlin1.dl.ac.uk:193.62.119.20	pc35.hep.ucl.ac.uk:128.40.4.35

Table 1. traceroute between UCL and DL and vice versa.

Periodic Behaviour

With network traffic often occurring during peak times, it can sometimes be interesting to note the periodicity of network characteristics. This is shown in Figure 7. 1st, 2nd, 8th and 9th are weekend days. The distribution of rtt is relatively constant, however, the minimum rtt peaks at the end of each weekday (and a little on Sunday). This could be attributed to staging of transfers to RAL to backup or gather data during off peak times to minimise the effect felt by users. It would be beneficial to have a longer trace of results to fully analyse this periodicity.

The relatively poor performance indicated by the large difference in rtts for different packet sizes is attributed to the known use of an encryption router between the links. The performance of the link for udp and tcp traffic is shown in Figure 8. Whilst correlation is difficult to judge between rtt and throughput, there is a noticeable low throughput obtained with udpmon and iperf. We are currently investigating the dynamics of this link.

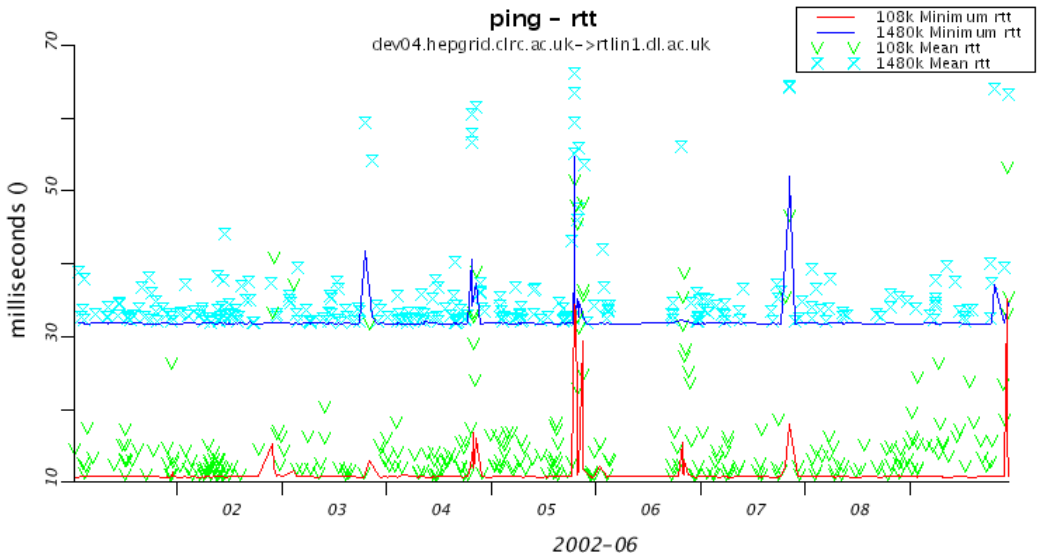

Figure 7. Periodic behaviour of rtt.

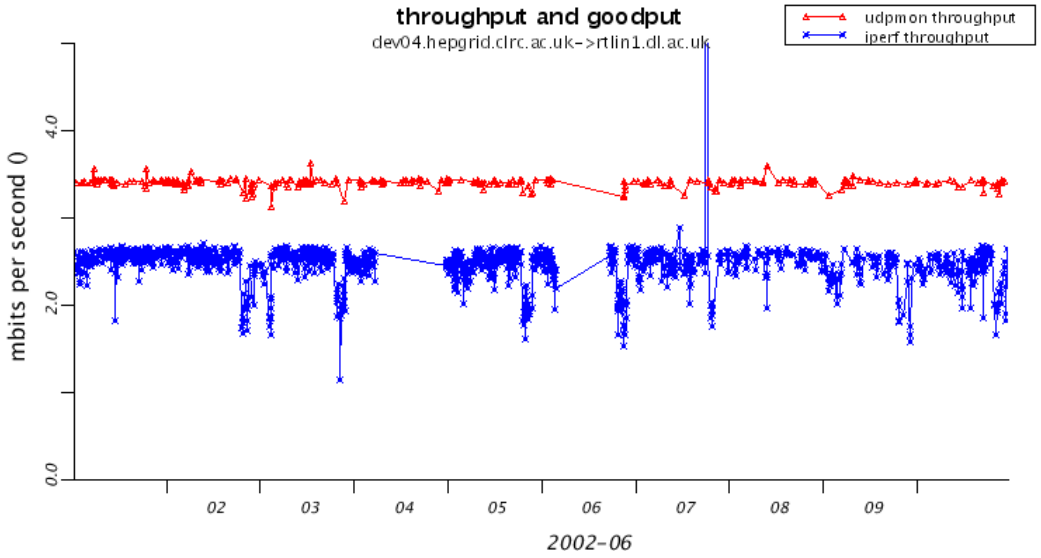

Figure 8. Throughput from RAL to DL.

Case 2: TCP goodput

In order to send tcp traffic across the network, we must supply data to tcp protocol. This data is stored in a buffer for each individual tcp connection which is called the socket buffer size. Should this buffer be small, we in effect limit the amount of tcp traffic into the network.

TCP is ACK clocked. This means that it responds the network only when it receives acknowledgements from the tcp receiver. As such, a connection to a receiver with a high rtt value will mean that the TCP would take longer to adapt to the network and hence will be limited to how much data it can send. This idea is encapsulated in a theory called bandwidth-delay product. This is shown in Table 2.

		Manchester	Amsterda	CERN	Italy	Chicago	SLAC
				rtt (millisecs)			
		5	***10***	***25***	***50***	***100***	***200***
Socket Buffer Size (kbytes)	***64***	105	52	21	10	5	3
	128	210	105	42	21	10	5
	256	419	210	84	42	21	10
	512	839	419	168	84	42	21
	102	1678	839	336	168	84	42
	204	3355	1678	671	336	168	84
	304	4994	2497	999	499	250	125
	409	6711	3355	1342	671	336	168
	512	8389	4194	1678	839	419	210
	614	10066	5033	2013	1007	503	252
	716	11744	5872	2349	1174	587	294
	819	13422	6711	2684	1342	671	336

Table 2. Bandwidth delay product showing the maximum bandwidth in mbits/sec depending on the socket buffer size and the round trip time of a tcp connection from UCL.

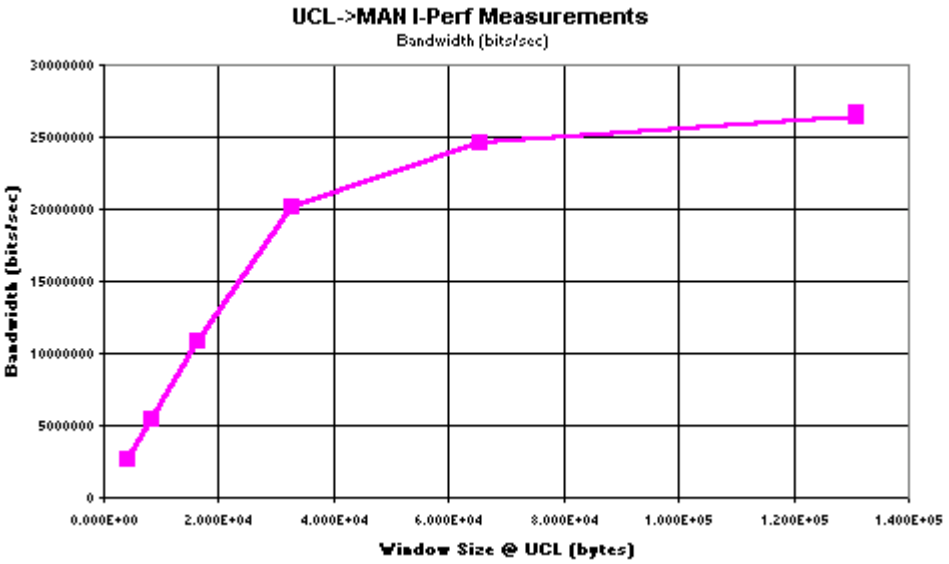

Figure 9. Graph to show variation of socket buffer size to tcp goodput.

Real life tcp transfer tests for a relatively low speed (100mbits/sec) link is shown in Figure 8. The increase in goodput as we increase the socket buffer size is shown for tcp throughput tests between Manchester and UCL. A region of linearity is followed by a plateau of goodput. The former is a consequence of the limitation of data supplied to tcp, whilst the latter is caused by the network performance.

We conducted iperf tcp measurements between Manchester and CERN over a 7 day period continuously monitoring the network (approximately) every hour. We wanted to investigate the effect of different socket buffer sizes upon the achieve link utilisation. This is shown in Figure 11. It shows the frequency distribution of approximately 400 10 second iperfs for socket buffer settings of {128k, 256k, 512k, 1024k, 2048k}.

Notice that for buffer sizes less than 1mb, there is a proportional improvement in link utilisation if we increase the socket buffer size. However, once we use larger buffer sizes, the additional increase in throughput is much reduced and also more prone to variation.

With many standard distributions of Linux set to a default of 64k socket buffer size for tcp transfers, this demonstration gives an example of how 'tcp tuning' can be used to get better throughput.

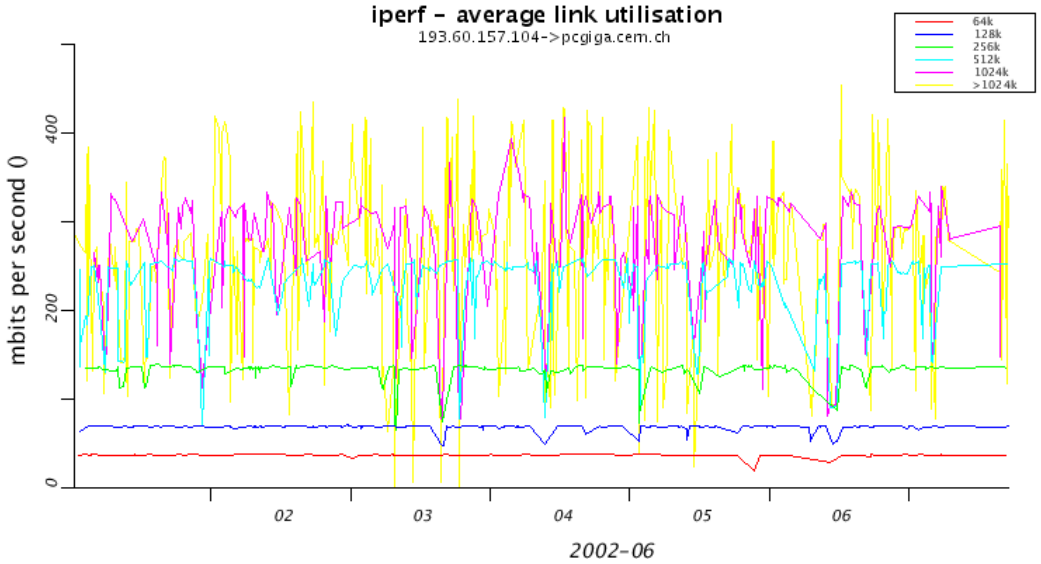

Figure 10. Effect of varying TCP socket buffer size over time.

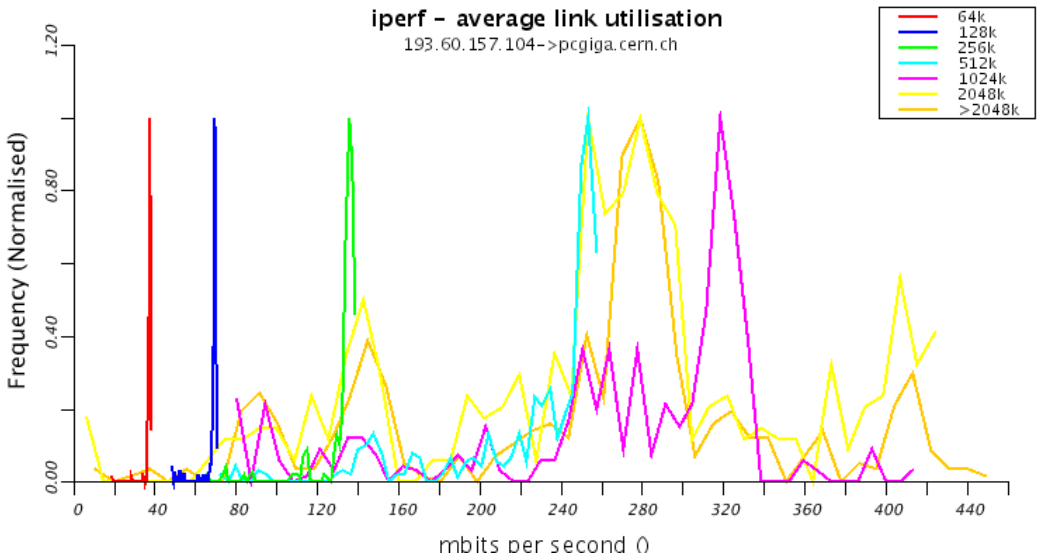

Figure 11. Frequency distribution of achieved goodput with various socket buffer sizes.

An unexplained anomaly was discovered that results in the deterioration of goodput as a result of large buffer sizes. This is shown in Figure 11 and also Figure 12 of the link between Manchester and CERN, and NL and CERN. Each frequency distribution has about 500 individual singleton measurements. Interestingly, past 1024kbyte socket buffer sizes, there is a noticeable drop in throughput achieved. On the NL to CERN link, a 4096k socket buffer sizes actually achieving a much overall mean throughput that is less than that of the typical 64k connection. In order to check this anomaly, the reverse link of the path is also shown in Figure 13 with approximately 400 singleton measurements per distribution. Again the optimal socket buffer size appears to be at 1024kbyte. However, the degradation in performance with larger buffer sizes is not as pronounced with similar goodput of 2mb yte and 4mb yte buffer sizes similar to that of 512kbyte. Even though the median distribution for the larger buffer sizes is centred at about 240mbits/sec, there is a pronounced peak of goodput at 400mbits/sec. This suggests that even though the large buffer sizes can and sometimes does obtain a larger goodput, network conditions limit the goodput to just over 240mbits/sec.

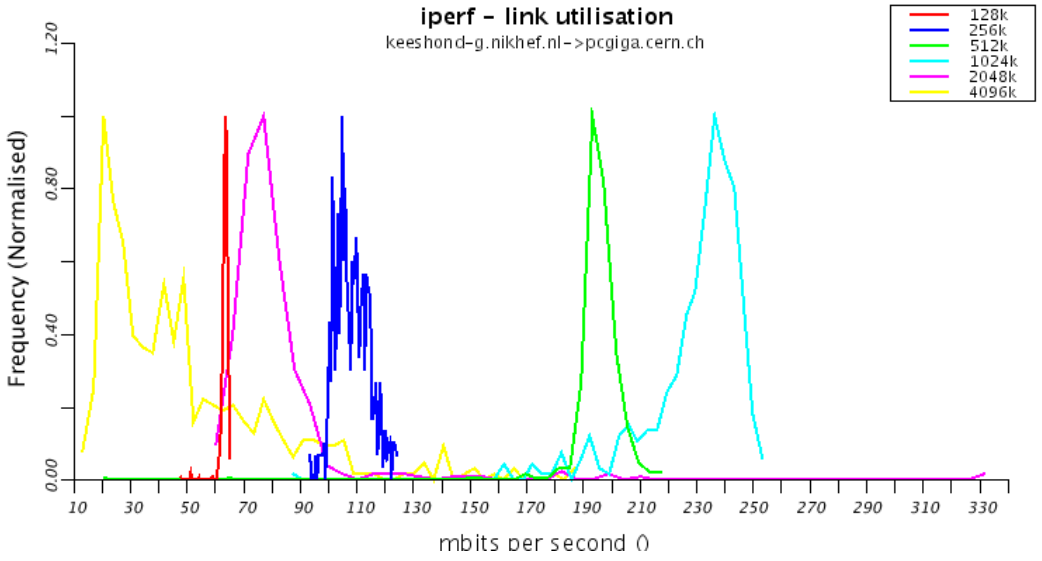

Figure 12. Link utilisation with various socket buffer sizes between NL and CERN.

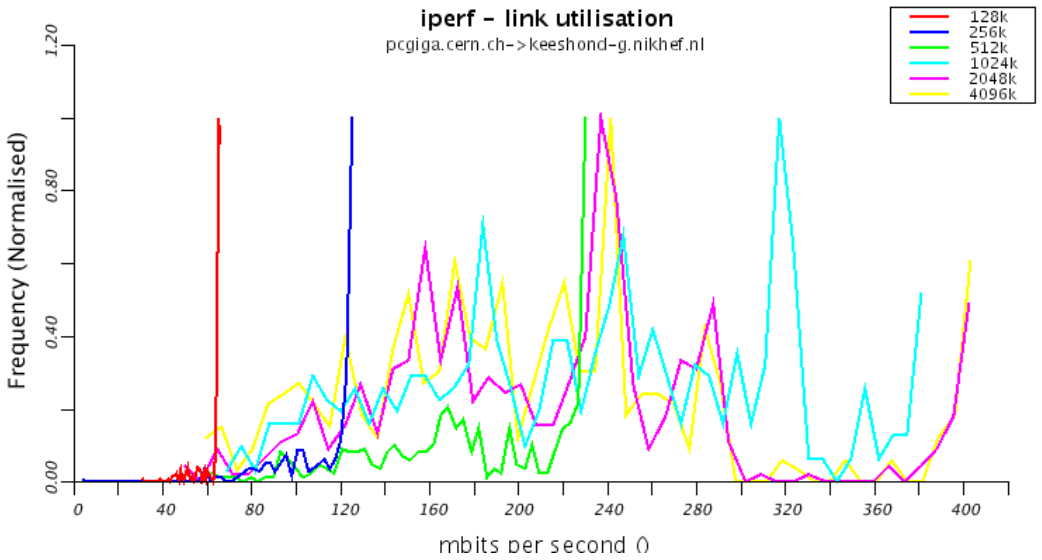

Figure 13. Link utilisatin with various socket buffer sizes between CERN and NL.

If we start a tcp transfer with a socket buffer size that is less than a specific value, the bottleneck, assuming constant network state, would be at the socket buffer size as tcp has no data to send into the network. However, there is a limit, shown in

Figure 9 that demonstrates that increasing the socket buffer size past an optimal value would not give any benefits to the goodput – the plateau. This is due to the fact that TCP is clocked by ACKs which determines the amount of data that can be sent into the network - even if we have a larger socket buffer size. This limit is imposed through the way that TCP's requires ACKs from the receiver to govern how to adapt its sending data rate and there independent of the socket buffer size. Of course, this assumes that there is sufficient data in the socket buffer to create tcp packets.

What we discovered was that this threshold actually gives the peak rate in which goodput is achieved. Larger socket buffer sizes actually reduce the performance of the tcp protocol. As TCP merely acts as a way of determine how much data the network can handle, has no regard for the size of the socket buffer. It also only determines how much data can be send into the network and tries to do so. If we assume that there is sufficient data in the socket buffer at all times and that the socket buffer is large enough to supply tcp to the rate in which it wants to send data, we would be limited only by the rate in which tcp can send data into the network – not its socket buffer size.

In order to account for this, we present our understanding of the TCP algorithms. TCP uses a windowing system from which data is transmitted into the network. The variable that controls this value is called the congestion window (`cwnd`). It acts as a way of regulating the amount of data that a source is allowed to send into the network. `cwnd` grows according to certain principles called Additive Increase Multiplicative Decrease (AIMD). It may also be affected by TCP options such as SACK [SACK] and Fast Retransmits [FastRetranmits]. Under certain circumstances `cwnd` may prevent TCP reaching an optimal goodput. One such circumstance is if the receiver advertises a small receiver window (`rwin`) which tells the sender how much data it can receive. Should `rwin` be smaller than `cwnd`, the sender always limits the sending rate to `rwin`, regardless of the current value of `cwnd`.

It is important to set the value of the receiver buffer to a value greater than the iperf sender socket buffer size as this would limit the receive rate and may lead to overflow at the receiving end rather than in the network and may lead to excessive loss resulting in reduced goodput.

In order to check this, we can look at the tcp kernel parameters in the `/proc/sys/net/core/` directory. Investigation showed that NL was set with a default tcp buffer write size of 256kbytes and a maximum of 32mbytes in `wmem_default` and `wmem_max`. The kernel read buffers at CERN were set to 64kbytes default and 8mbyte maximum; set in `rmem_default` and `rmem_max` respectively. As we used 8mbyte receiver windows on the receiver and only up to 4mbytes socket buffer on the sender, there didn't seem to be a problem with kernel parameters.

Another feature of tcp is that it maintains a value called the slow start threshold (`ssthresh`) that maintains a snapshot of the network condition at every detected congestion event. A congestion event is either the detection of a duplicate ACK due to reordering of packets along the network, or actual packet loss which is indicated through a timeout. `ssthresh` determines the current detected condition of the network from which tcp can adapt is data rate. In relation to the value of `cwnd`, `ssthresh` is used to deterine which regime of AIMD it should enter. If `cwnd` is less than `ssthresh`, tcp is known to be is in slow start. Else it is in congestion avoidance. The difference between the two regimes is the way in which `cwnd` is grown. Should it be in slow start, `cwnd` grows exponentially with each ACK. Else it grows linearly.

When tcp experience congestion, `ssthresh` is set to half the value of `cwnd`. If a timeout is experienced on a packet, `ssthresh` is set to half the value of `cwnd` – however the difference is that `cwnd` is now set to 1 segment size. A segment size is basically the size of data that can fit in one tcp packet. TCP then goes into slow start until it reaches `ssthresh` and then congestion avoidance to lower the injection rate of packets into the network. This cyclic mechanism is used until the tcp connection is over.

As such, `ssthresh` is an important parameter to consider. Should `ssthresh` be calculated to a value that is not indicative of the network, tcp will under perform in the face of congestion.

As congestion is the only way that tcp can update its value of `ssthresh` and therefore adapt to the network, constant congestion experienced by tcp is good as it can adapt more readily to the network. However, a congestion event caused through a timeout reduces the amount of data sent into the network and hence reduces goodput. Also the duration of the timeout, although dynamically calculated, can be several orders of magnitude of the rtt and as such can cause TCP to dwindle during the timeout period and hence also reduce the goodput rate.

We therefore looked into the loss rates from the path from CERN to NL. This is shown in Figure 14 in which we sent 300 udp packets using udpmon for each measurement. Unfortunately, loss information was not available for the NL to CERN path. One can note from this graph that loss rates are relatively high for small UDP packet sizes. This results in a decrease of udp link utilisation as shown in Figure 15. Figure 15 also shows the iperf link utilisation using just one socket buffer size (the optimal value of 1024k) – showing that it consistently underperforms to the available bandwidth of the link obtaining an average of only about 235mbits/sec.

It is known the link between CERN and NL is physically limited 622mbits/sec. This loss from the UDP packets causes the udpmon receiver wire rate to just over 500mbits/sec. The relative loss rate compared to other links is actually quite high. We assume that udp packets are treated in the same way as tcp along this link.

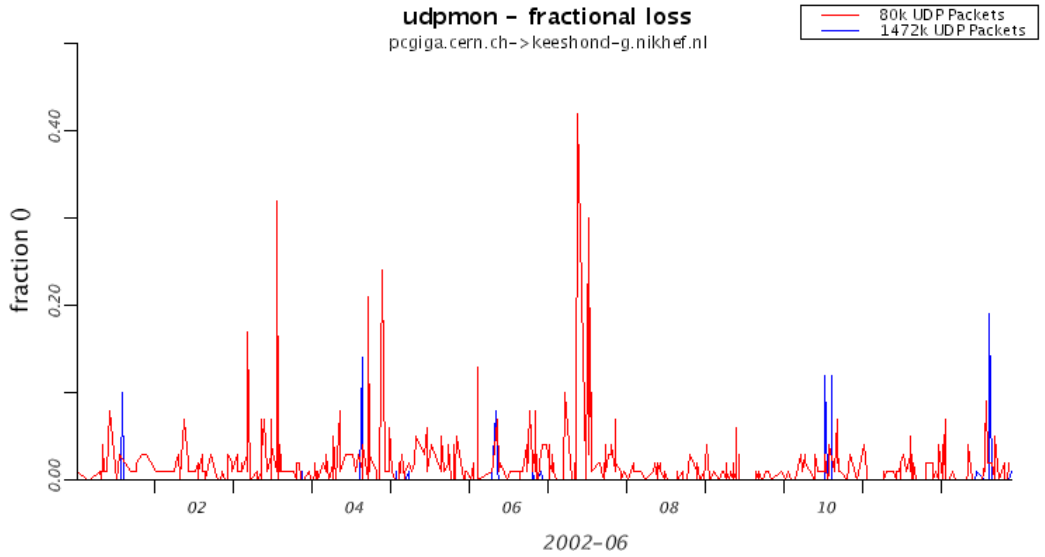

Figure 14. Loss rates of UDP packets from CERN to NL.

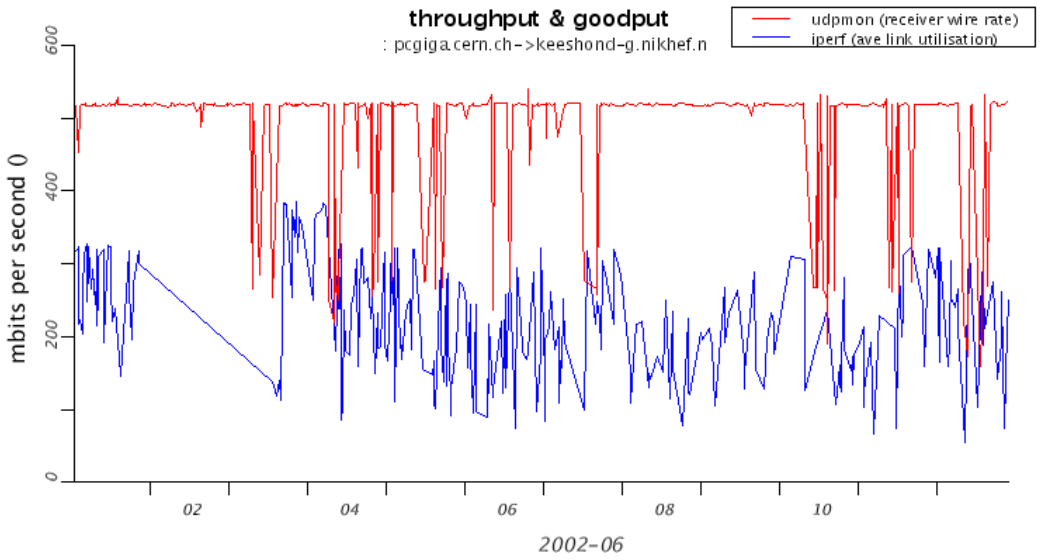

Figure 15. updmon and iperf link utilisation from CERN to NL.

When we start a tcp connection, we have filled the socket buffer with data. A packet of data is sent into the network and ACKs are required for us to grow `cwnd` exponentially. This allows us to send even more packets into the network until some form of congestion is noticed. If the congestion was only due to packet reordering, there is no change in `cwnd` and thus keeps on growing exponentially. Should the congestion be in the form of packet loss, then tcp would timeout and set its `cwnd` to only one segment size – hence reducing its data rate. It also updates its ssthresh to a value half of what the `cwnd` was before the timeout.

TCP would then re-initiate in slow start again, putting packets back into the network at an exponential rate with each incoming ACK until its `cwnd` gets above `ssthresh`. When `cwnd` does get above `ssthresh`, tcp will limit the number of packets it is sending out to a more conservative rate until it experiences a packet loss, forcing the cycle to restart.

An important point to note is that if the value of `ssthresh` is actually higher than its optimal value, then TCP will keep on sending out packets at an exponential rate. Should the network not be able to handle the extra packets sent out during slow start, many packets will be lost. As each loss will halve the `ssthresh` value, an accumulation of losses imposed through sending too many packets at once will decrease `sshthresh` to a very low value. When TCP starts sending data again, it will get to this value of `ssthresh` quickly and then enter into congestion avoidance. Congestion avoidance limits the sending of data to one segment per rtt which is much less than the sending rate in slow start. Should the value of `ssthresh` be much lower than what the network is capable of maintaining, the tcp connection will take a very long time to grow the value of `cwnd` and hence limit the achieved goodput.

As such, the growth of the cwnd should be similar if the network state is the same for each measurement. An accumulation of measurements should give an accurate indication of the network state and hence the protocol performance over that time.

A possibility with the phenomenon of decreasing performance with larger socket buffer sizes is that the application (iperf) cannot actually fill the socket buffer properly. As such, it stumbles and causes the socket buffer to empty. This then results in a decrease in data supplied to the tcp protocol and hence reduced throughput. Investigation into the iperf algorithms to generate data need to be investigated. It is unlikely it is to do with the efficiency of the data generation algorithms as faster machines should therefore be able to generate data at a faster rate. The experimental results from Figure 11 , Figure 12 and Figure 13 show that the optimal buffer size is approximately 1mbyte. Should the efficiency of the data generation algorithms of iperf be the bottleneck, then this value should change for different machine speeds. An investigation into this is currently under way.

Although tuning is possible with tcp connections, the result may actually be worse performance as a result of certain conditions such as relatively high loss. We are currently investigating into this phenomenon with more WAN measurements and hope to find a solution by comparing these against lab tests.

Case 3: Bandwidth and goodput tools

We want to investigate into the performance of NMAs compared to each other. We are particular interested in tools that report bandwidth and throughput related metrics.

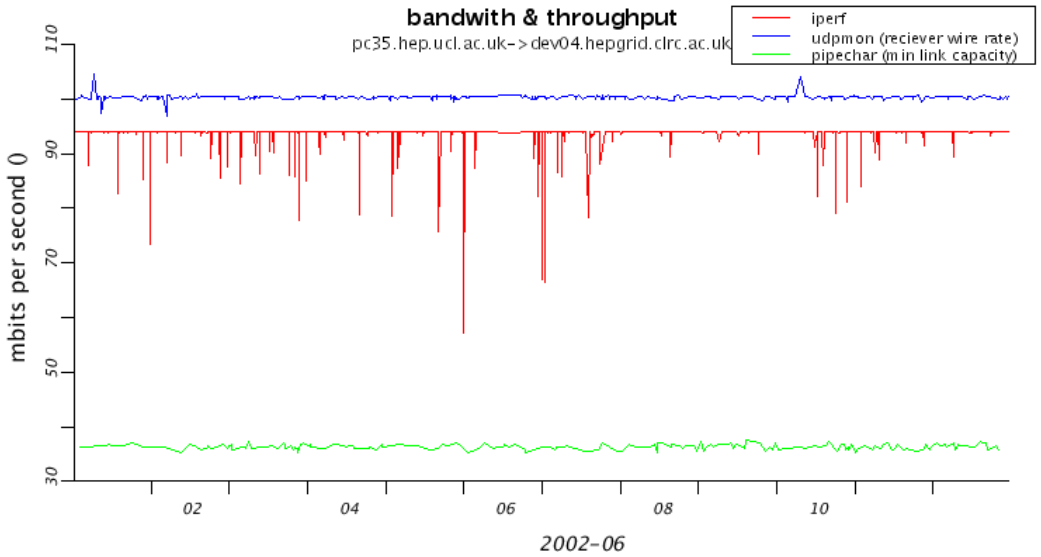

Figure 16. Comparison of bandwidth and throughput tools from UCL to RAL.

Figure 16 shows the comparison of metrics obtained through different NMA's. It shows that UDP based transport protocols should achieve maximum throughput as the link is limited physically by 100mbit Ethernet at UCL. It also shows that iperf TCP performance, whilst not quite a 100mbits/sec, achieves about 93mbits/sec. There is a slight deviation of goodput, but there are no long periods that experience bad goodput and is most likely due to contention of other tcp flows that would halve our throughput achieved.

The green line, which shows the predicted minimum available bandwidth reported by pipechar says that the link should saturate at 36mbits/sec. This is most obvious incorrect as we were able to achieve much higher bandwidth utilisation with the other two tools. We used the value of minimum available bandwidth of pipechar as it should give an indication of the bottleneck of the link which would limit our throughput.

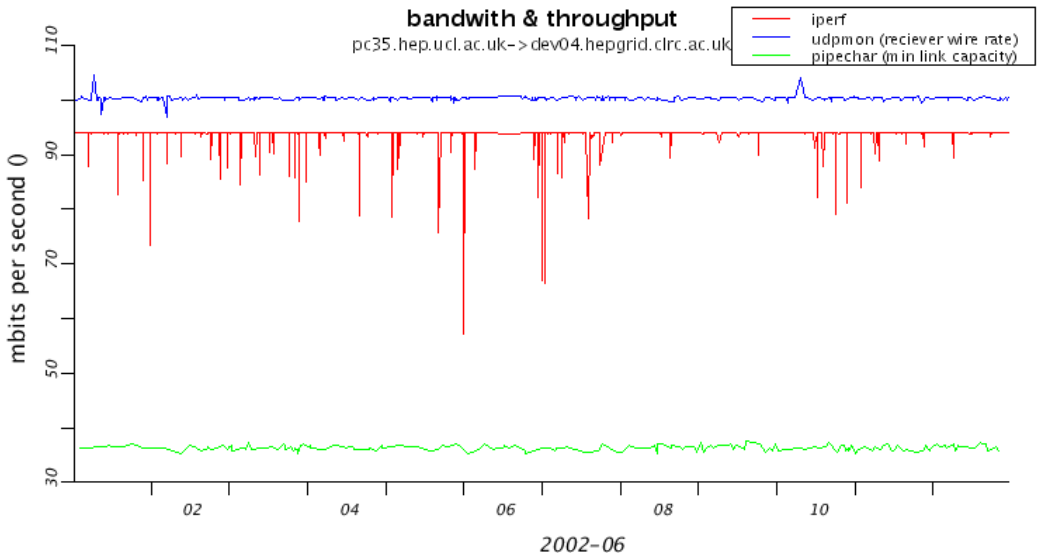

Figure 17. Comparison of bandwidth and throughput tools from UCL to RAL.

Similar results can be seen for the link between UCL and RAL.

One can infer from these results that pipechar is not very good at estimating available bandwidth.

Case Study: Rerouting of Trans-Atlantic Traffic from UK via GEANT

TeleGlobe, who supply UKERNA (the UK academic network carrier) with a transatlantic link, went into receivership recently. As such, UKERNA were forced to investigate alternate methods of transporting America bound traffic. Whilst we were not currently monitoring network performance to the States, we were monitoring traffic to the Netherlands – where UKERNA planned to reroute their traffic. This offered us an opportunity to investigate the effect of changing network patterns and performance.

The planned routing change was to occur at 6am on the 31st May 2002. We provide a UK centric view of the affect of the change as a result of the rerouting.

UCL and Manchester

We began to investigate the path between UCL and Manchester. Figure 18 shows a noticeable change in rtt before the planned rerouting. At approximately 29/05/02 15:30, an increase of 1ms was added to the rtt. This was followed by a small subsequent decrease at the planned routing change. This then went back to a normal baseline value about 31/05/02 19:30.

Unfortunately, we were unable to obtain traceroutes for this path due to a router along the path rejecting traceroute packets. As such we are unable to investigate possible changes in network path that may lead to this marked change in rtt over the period. We are currently trying to find a way to implement partial traceroute logging within the GridNM architecture.

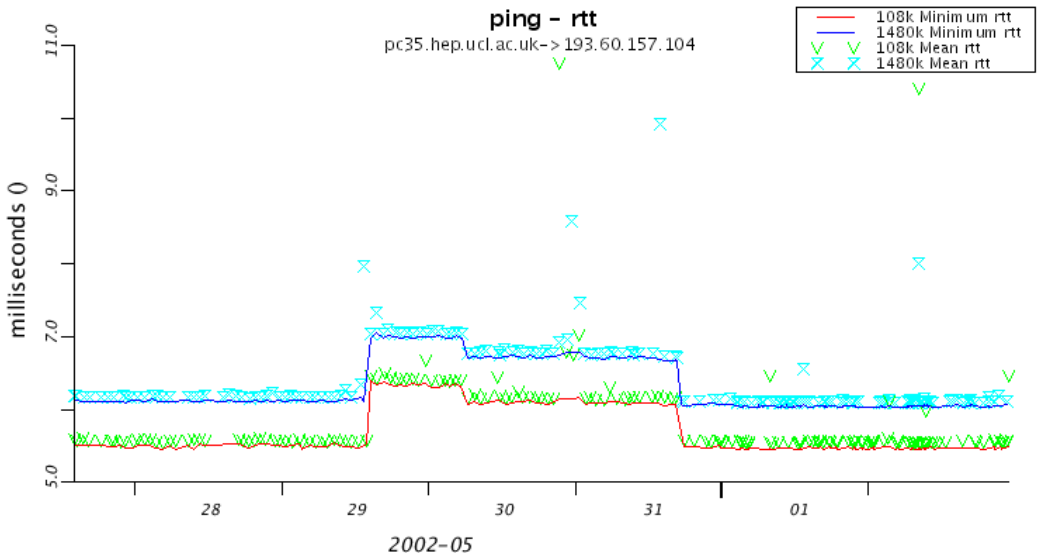

Figure 18. rtt from UCL to Manchester.

Figure 19 shows the reverse path rtt of Figure 18. We were unable to obtain a full trace due to technical difficulties. However, what we did managed to record shows a similar pattern that that shown in Figure 18.

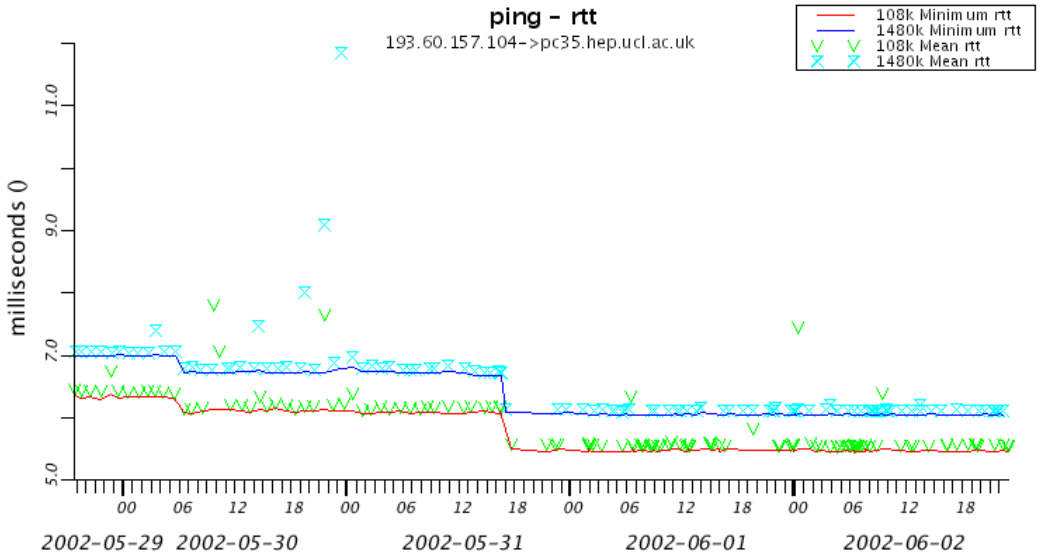

Figure 19. rtt from Manchester to UCL.

Unfortunately, we were only able to obtain traceroute information from Manchester to UCL after 31/5/200214:26. Interestingly, it was found that this link path from Manchester to UCL during measurement shows a path similar to that from DL to UCL. This is shown in Table 3. One can see that once traffic gets to `gw-nnw.core.netnw.net.uk`, the path is identical.

Manchester to UCL	DL to UCL
193.60.157.104:193.60.157.104	rtlin1.dl.ac.uk:193.62.119.20
gw-man.netnw.net.uk:193.60.157.102	alan3.dl.ac.uk:193.62.119.1
gw-nnw.core.netnw.net.uk:194.66.25.97	gw-fw.dl.ac.uk:193.63.74.131
manchester-bar.ja.net:146.97.40.177	nnw-gw.dl.ac.uk:193.63.74.232
po10-0.warr-scr.ja.net:146.97.35.45	gw-nnw.core.netnw.net.uk:194.66.25.29
po1-0.read-scr.ja.net:146.97.33.53	manchester-bar.ja.net:146.97.40.177
po1-0.lond-scr.ja.net:146.97.33.73	po10-0.warr-scr.ja.net:146.97.35.45
london-bar1.ja.net:146.97.35.2	po1-0.read-scr.ja.net:146.97.33.53
ulcc-gsr.lmn.net.uk:146.97.40.34	po1-0.lond-scr.ja.net:146.97.33.73
ucl.lmn.net.uk:194.83.101.6	london-bar1.ja.net:146.97.35.2
128.40.20.189:128.40.20.189	ulcc-gsr.lmn.net.uk:146.97.40.34
128.40.20.150:128.40.20.150	ucl.lmn.net.uk:194.83.101.6
pc35.hep.ucl.ac.uk:128.40.4.35	128.40.20.189:128.40.20.189
	128.40.20.150:128.40.20.150
	Pc35.hep.ucl.ac.uk:128.40.4.35

Table 3. traceroute from Manchester to UCL and DL to UCL.

DL and UCL

We therefore started to look at traceroute information from DL to UCL. Unfortunately, the rtt data from DL to UCL was quite noisy and hence rtt patterns were difficult to obtain. Table 4 shows the change in link path between DL and UCL for the same period as that presented for Figure 18. Note that we were unable to obtain traceroute information between 31/05/2002 14:46 and 31/05/2002 23:54.

29/05/02 – 14:30	29/05/02 – 15:40	31/05/02 – 07:31	31/05/02 – 23:54
rtlin1.dl.ac.uk:193.62.119.20	rtlin1.dl.ac.uk:193.62.119.20	rtlin1.dl.ac.uk:193.62.119.20	rtlin1.dl.ac.uk:193.62.119.20
alan3.dl.ac.uk:193.62.119.1	alan3.dl.ac.uk:193.62.119.1	alan3.dl.ac.uk:193.62.119.1	alan3.dl.ac.uk:193.62.119.1
gw-fw.dl.ac.uk: 193.63.74.131	gw-fw.dl.ac.uk: 193.63.74.131	gw-fw.dl.ac.uk: 193.63.74.131	gw-fw.dl.ac.uk: 193.63.74.131
nnw-gw.dl.ac.uk: 193.63.74.232	nnw-gw.dl.ac.uk: 193.63.74.232	nnw-gw.dl.ac.uk: 193.63.74.232	nnw-gw.dl.ac.uk: 193.63.74.232
gw-nnw.core.netnw.net.uk: 194.66.25.29	gw-nnw.core.netnw.net.uk: 194.66.25.29	gw-nnw.core.netnw.net.uk: 194.66.25.29	gw-nnw.core.netnw.net.uk: 194.66.25.29
manchester-bar.ja.net: 146.97.40.177	manchester-bar.ja.net :146.97.40.177	manchester-bar.ja.net: 146.97.40.177	manchester-bar.ja.net: 146.97.40.177
po10-0.warr-scr.ja.net: 146.97.35.45	po10-0.warr-scr.ja.net:146.97.35.45	po10-0.warr-scr.ja.net: 146.97.35.45	po10-0.warr-scr.ja.net: 146.97.35.45
po1-0.read-scr.ja.net: 146.97.33.53	po2-0.leed-scr.ja.net:146.97.33.77	po2-0.leed-scr.ja.net: 146.97.33.77	po1-0.read-scr.ja.net: 146.97. 33.53
po1-0.lond-scr.ja.net: 146.97.33.73	po2-0.lond-scr.ja.net:146.97.33.30	po2-0.lond-scr.ja.net: 146.97.33.70	po1-0.lond-scr.ja.net: 146.97.33.73
london-bar1.ja.net: 146.97.35.2	london-bar1.ja.net:146.97.35.2	london-bar1.ja.net: 146.97.35.2	london-bar1.ja.net: 146.97.35.2
ulcc-gsr.lmn.net.uk: 146.97.40.34	ulcc-gsr.lmn.net.uk: 146.97.40.34	ulcc-gsr.lmn.net.uk: 146.97.40.34	ulcc-gsr.lmn.net.uk: 146.97.40.34
ucl.lmn.net.uk:194.83.101.6	ucl.lmn.net.uk:194.83.101.6	ucl.lmn.net.uk:194.83.101.6	ucl.lmn.net.uk:194.83.101.6
128.40.20.189:128.40.20.189	128.40.20.189:128.40.20.189	128.40.20.189:128.40.20.189	128.40.20.189:128.40.20.189
128.40.20.150:128.40.20.150	128.40.20.150:128.40.20.150	128.40.20.150:128.40.20.150	128.40.20.150:128.40.20.150
pc35.hep.ucl.ac.uk: 128.40.4.35	pc35.hep.ucl.ac.uk: 128.40.4.35	pc35.hep.ucl.ac.uk: 128.40.4.35	pc35.hep.ucl.ac.uk: 128.40.4.35

Table 4. traceroute information from DL to UCL.

From the traceroute information from DL to UCL, the change in rtt appears to be caused through a routing change through Leeds. This would account for the first raise in rtt at 29/05/2002 15:40. The rtt from DL to Leeds is approximately 10ms. This link path then goes back to the 'normal' path sometime between 14:46 and 23:54 on the 31/05/2002.

As the Manchester to UCL path is similar to the DL to UCL path, we hypothesise that the changes in rtt observed in Figure 19 of Manchester to UCL traffic were also caused by the rerouting of packets through Leeds as shown in Figure 7. However the only way to check this analytically is via traceroute information which is not available.

Investigating into the reverse path, i.e. from UCL to DL, shows a similar jump in rtt at the same times. The change in rtt is also about 1ms for both packet sizes measured. This is shown in Figure 20. Traceroute information is also shown in Table 5.

One can see that traffic is diverted from Reading to Leeds at 29/5/2002 15:38. This corresponds to the rerouting shown on all other paths within the sites presented thus far. A change back to the original path via Reading was recorded at 31/5/2002 18:19.

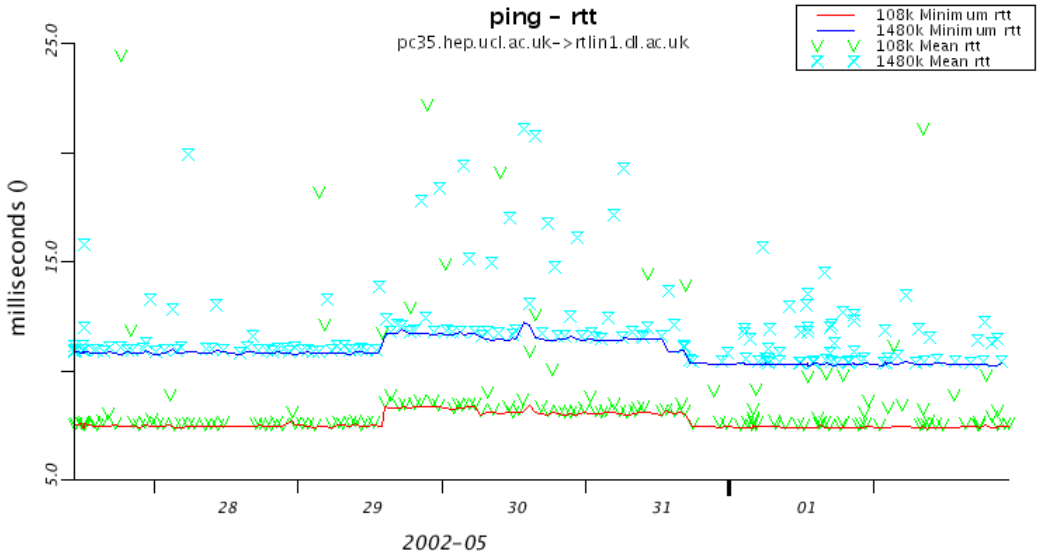

Figure 20. rtt from UCL to DL.

29/5/2002 - 14:38	29/5/2002 - 15:38	31/5/2002 – 18:19
pc35.hep.ucl.ac.uk:128.40.4.35	pc35.hep.ucl.ac.uk:128.40.4.35	pc35.hep.ucl.ac.uk:128.40.4.35
128.40.4.245:128.40.4.245	128.40.4.245:128.40.4.245	128.40.4.245:128.40.4.245
128.40.20.149:128.40.20.149	128.40.20.149:128.40.20.149	128.40.20.149:128.40.20.149
128.40.20.190:128.40.20.190	128.40.20.190:128.40.20.190	128.40.20.190:128.40.20.190
ulcc-gsr.lmn.net.uk:194.83.101.5	ulcc-gsr.lmn.net.uk:194.83.101.5	ulcc-gsr.lmn.net.uk:194.83.101.5
london-bar1.ja.net:146.97.40.33	London-bar1.ja.net:146.97.40.33	london-bar1.ja.net:146.97.40.33
po9-0.lond-scr.ja.net:146.97.35.1	po9-0.lond-scr.ja.net:146.97.35.1	po9-0.lond-scr.ja.net:146.97.35.1
po2-0.read-scr.ja.net: 146.97.33.74	po1-0.leed-scr.ja.net: 146.97.33.29	po2-0.read-scr.ja.net: 146.97.33.74
po0-0.warr-scr.ja.net: 146.97.33.54	po2-0.warr-scr.ja.net: 146.97.33.78	po0-0.warr-scr.ja.net: 146.97.33.54
manchester-bar.ja.net: 146.97.35.46	manchester-bar.ja.net: 146.97.35.46	manchester-bar.ja.net: 146.97.35.46
gw-nnw.core.netnw.net.uk: 146.97.40.178	gw-nnw.core.netnw.net.uk: 146.97.40.178	gw-nnw.core.netnw.net.uk: 146.97.40.178
gw-dl.netnw.net.uk:194.66.25.30	gw-dl.netnw.net.uk:194.66.25.30	gw-dl.netnw.net.uk:194.66.25.30
gw-fw.dl.ac.uk:193.63.74.233	gw-fw.dl.ac.uk:193.63.74.233	gw-fw.dl.ac.uk:193.63.74.233
rtlin1.dl.ac.uk:193.62.119.20	rtlin1.dl.ac.uk:193.62.119.20	rtlin1.dl.ac.uk:193.62.119.20
rtlin1.dl.ac.uk:193.62.119.20	rtlin1.dl.ac.uk:193.62.119.20	rtlin1.dl.ac.uk:193.62.119.20

Table 5. traceroute for UCL to DL.

One can say from this information that all monitored traffic between UCL and DL (in both directions) was diverted to Leeds on the 29th. On the 31st, traffic was then redirected back to its original path going through Reading. The routing change through Leeds resulted in an increase in rtt of approximately 1ms in both directions.

DL to NL

As all UK transatlantic traffic should go through NL for a time after 30/05/02 06:00, traffic from DL to NL should show a marked increase in queuing delay should the volume of traffic be substantial enough to affect router queues along the path. However, this was not the case - with no real correlation between a change of neither minimum rtt nor mean rtt and the times of the routing changes described above. This suggests that the amount of traffic from DL to NL, including all cross traffic, did not change much over this time period.

There was also no change in path route from DL to NL during this period. This means that all traffic from DL to NL (and presumably America) remains on same path.

The path route from UCL to NL also did not change during this period. One can note from Table 6 that traffic from DL, and assumingly Manchester as discussed above, goes through `london-bar4.ja.net` – which UCL does also. As such, this gives a central node which could need

provisioning or policy control should it be the hub from which all UK to US traffic would flow out of the UK.

DL to NL	RAL to NL	UCL to NL
rtlin1.dl.ac.uk:193.62.119.20	dev04.hepgrid.clrc.ac.uk:130.246.	Pc35.hep.ucl.ac.uk:128.40.4.35
alan3.dl.ac.uk:193.62.119.1	130.246.200.254:130.246.200.254	128.40.4.245:128.40.4.245
gw-fw.dl.ac.uk:193.63.74.131	fw1.routers.net.rl.ac.uk:130.246.8	128.40.20.21:128.40.20.21
nnw-gw.dl.ac.uk:193.63.74.232	192.100.78.1:192.100.78.1	128.40.20.62:128.40.20.62
gw-nnw.core.netnw.net.uk: 194.66.25.29	ral-bar.ja.net: 146.97.40.73	ulcc-gsr.lmn.net.uk: 194.83.101.5
manchester-bar.ja.net: 146.97.40.177	Po13-0.read-scr.ja.net: 146.97.35.81	london-bar1.ja.net: 146.97.40.33
po10-0.warr-scr.ja.net: 146.97.35.45	london-bar4.ja.net: 146.97.35.134	po9-0.lond-scr.ja.net: 146.97.35.1
po1-0.read-scr.ja.net: 146.97.33.53	geant-gw.ja.net: 146.97.37.82	london-bar4.ja.net: 146.97.35.130
london-bar4.ja.net: 146.97.35.134	janet.uk1.uk.geant.net: 62.40.103.149	geant-gw.ja.net:146.97.37.82
geant-gw.ja.net: 146.97.37.82	uk.nl1.nl.geant.net: 62.40.96.113	janet.uk1.uk.geant.net: 62.40.103.149
janet.uk1.uk.geant.net: 62.40.103.149	PO6-0.BR1.Amsterdam1.surf.net: 62.40.103.98	uk.nl1.nl.geant.net: 62.40.96.113
uk.nl1.nl.geant.net: 62.40.96.113	PO12-0.CR1.Amsterdam1.surf.net:	PO6-0.BR1.Amsterdam1.surf.net: 62.40.103.98
PO6-0.BR1.Amsterdam1.surf.net: 62.40.103.98	PO0-0.AR5.Amsterdam1.surf.net: 145.145.162.2	PO12-0.CR1.Amsterdam1.surf.net:
PO12-0.CR1.Amsterdam1.surf.net:	hef-router.nikhef.nl:145.145.18.14	PO0-0.AR5.Amsterdam1.surf.net: 145.145.162.2
PO0-0.AR5.Amsterdam1.surf.net: 145.145.162.2	deel.nikhef.nl:192.16.186.162	hef-router.nikhef.nl:145.145.18.14
hef-router.nikhef.nl:145.145.18.14	keeshond-gsk.nikhef.nl: 192.16.186.134	deel.nikhef.nl:192.16.186.162
deel.nikhef.nl:192.16.186.162		keeshond-gsk.nikhef.nl: 192.16.186.134
keeshond-gsk.nikhef.nl: 192.16.186.134		

Table 6. traceroute of DL to NL and UCL to NL.

As such once can say that the rerouting of traffic to Leeds was a result of trying to reduce the load at Reading `{po1-0|po13-0}.read-scr.ja.net`. However, traffic bound to NL from DL remained on the same link path as before the rerouting – i.e. still going though Reading.

RAL and UCL

Due to a crash at `dev04`, we were unable to obtain any traceroute information from the 30/5/2002 10:30:45 and 31/5/200211:58:31. The traceroute changes before and after during this period are shown in Table 7. It appears that a routing change occurred after 29/05/2002 14:30 to divert traffic through Bristol before getting to the London PoP. `po1-0.bris-scr.ja.net` could not be pinged, but a host in the Physics department of Bristol University shows approximately 5ms latency from RALusing 1480k packets. The subsequent ping from the host at Bristol to UCL takes about 6ms. The addition of these two values equate well to the approximate 12 seconds shown at the plateau in Figure 21.

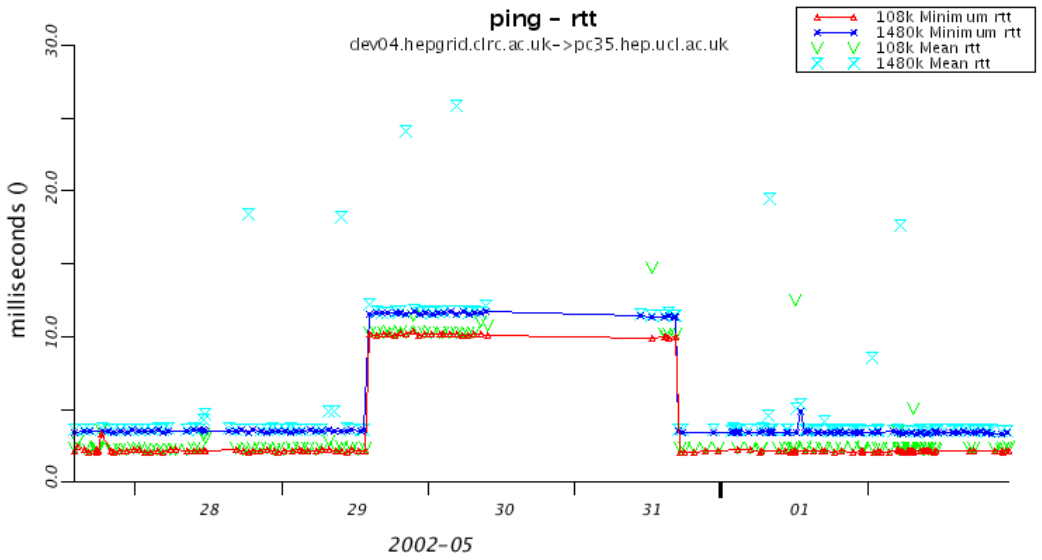

Figure 21. rtt from RAL to UCL.

There was no change in this route during the time UKERNA were supposed to change the path taken for transatlantic traffic at 30/05/2002 06:00. There was also no obvious change in network performance at this time as can be seen in Figure 21.

29/05/02 – 14:30	**29/05/02 – 15:25**	**31/05/02 – 11:58**	**31/05/02 – 18:06**
dev04.hepgrid.clrc.ac.uk: 130.246.200.1	dev04.hepgrid.clrc.ac.uk: 130.246.200.1	dev04.hepgrid.clrc.ac.uk: 130.246.200.1	dev04.hepgrid.clrc.ac.uk: 130.246.200.1
130.246.200.254: 130.246.200.254	130.246.200.254: 130.246.200.254	130.246.200.254: 130.246.200.254	130.246.200.254: 130.246.200.254
fw1.routers.net.rl.ac.uk: 130.246.80.5	fw1.routers.net.rl.ac.uk: 130.246.80.5	fw1.routers.net.rl.ac.uk: 130.246.80.5	fw1.routers.net.rl.ac.uk: 130.246.80.5
192.100.78.1:192.100.78.1	192.100.78.1:192.100.78.1	192.100.78.1:192.100.78.1	192.100.78.1:192.100.78.1
ral-bar.ja.net:146.97.40.73	ral-bar.ja.net:146.97.40.73	ral-bar.ja.net:146.97.40.73	ral-bar.ja.net:146.97.40.73
po13-0.read-scr.ja.net: 146.97.35.81	po13-0.read-scr.ja.net: 146.97.35.81	po13-0.read-scr.ja.net: 146.97.35.81	po13-0.read-scr.ja.net: 146.97.35.81
po1-0.lond-scr.ja.net: 146.97.33.73	po1-0.bris-scr.ja.net: 146.97.33.9	po1-0.bris-scr.ja.net: 146.97.33.9	po1-0.lond-scr.ja.net: 146.97.33.73
london-bar1.ja.net: 146.97.35.2	po1-0.cosh-scr.ja.net: 146.97.33.5	po1-0.cosh-scr.ja.net: 146.97.33.45	london-bar1.ja.net: 146.97.35.2
ulcc-gsr.lmn.net.uk: 146.97.40.34	po0-0.lond-scr.ja.net: 146.97.33.1	po0-0.lond-scr.ja.net: 146.97.33.1	ulcc-gsr.lmn.net.uk: 146.97.40.34
ucl.lmn.net.uk:194.83.101.6	London-bar1.ja.net: 146.97.35.2	london-bar1.ja.net : 146.97.35.2	ucl.lmn.net.uk:194.83.101.6
128.40.20.189:128.40.20.189	Ulcc-gsr.lmn.net.uk: 146.97.40.34	ulcc-gsr.lmn.net.uk: 146.97.40.34	128.40.20.189:128.40.20.189
128.40.20.150:128.40.20.150	ucl.lmn.net.uk:194.83.101.6	ucl.lmn.net.uk:194.83.101.6	128.40.20.150:128.40.20.150
pc35.hep.ucl.ac.uk: 128.40.4.35	128.40.20.189:128.40.20.189	128.40.20.189:128.40.20.189	pc35.hep.ucl.ac.uk: 128.40.4.35
	128.40.20.150:128.40.20.150	128.40.20.150:128.40.20.150	
	pc35.hep.ucl.ac.uk: 128.40.4.35	pc35.hep.ucl.ac.uk: 128.40.4.35	

Table 7. traceroute from RAL to UCL.

It was found that the path from UCL to RAL was being rerouted through Bristol during the same period as that of DL and UCL to Leeds. This resulted in a similar graph to that shown in Figure 21. This also implies that there was some provisioning in the network to prevent traffic from going through Reading (`po13-0.read-scr.ja.net`).

RAL to UCL	RAL to NL
dev04.hepgrid.clrc.ac.uk:130.246.200.1	dev04.hepgrid.clrc.ac.uk:130.246.200.1
130.246.200.254:130.246.200.254	130.246.200.254:130.246.200.254
fw1.routers.net.rl.ac.uk:130.246.80.5	fw1.routers.net.rl.ac.uk:130.246.80.5
192.100.78.1:192.100.78.1	192.100.78.1:192.100.78.1
ral-bar.ja.net:146.97.40.73	ral-bar.ja.net:146.97.40.73
po13-0.read-scr.ja.net:146.97.35.81	po13-0.read-scr.ja.net:146.97.35.81
po1-0.lond-scr.ja.net:146.97.33.73	london-bar4.ja.net:146.97.35.134
london-bar1.ja.net:146.97.35.2	geant-gw.ja.net:146.97.37.82
ulcc-gsr.lmn.net.uk:146.97.40.34	Janet.uk1.uk.geant.net:62.40.103.149
ucl.lmn.net.uk:194.83.101.6	uk.nl1.nl.geant.net:62.40.96.113
128.40.20.189:128.40.20.189	PO6-0.BR1.Amsterdam1.surf.net: 62.40.103.98
128.40.20.150:128.40.20.150	PO12-0.CR1.Amsterdam1.surf.net: 145.145.166.1
pc35.hep.ucl.ac.uk:128.40.4.35	PO0-0.AR5.Amsterdam1.surf.net: 145.145.162.2
	hef-router.nikhef.nl:145.145.18.14
	deel.nikhef.nl:192.16.186.162
	keeshond-gsk.nikhef.nl:192.16.186.134

Table 8. traceroute the original path from RAL to UCL and RAL to NL.

Table 8 shows the routing of traffic from RAL. One can see that all NL bound traffic from RAL goes through the Reading – the same as the DL link. However, sometime before 29/05/02 15:25, rerouting via Bristol occurred. It was also found that traffic from UCL to RAL was also being rerouted via Bristol during this period. As such, there was also provision by UKERNA to decrease load via Reading by diverting traffic between UCL and RAL via Bristol.

Summary

Prior to 29th June 2002, traffic between UCL and DL, and RAL and UCL, was routed through the Reading PoP. Soon after 29/05//2002 14:30, a routing change to divert traffic between DL and UCL via Leeds, and RAL and UCL via Bristol was monitored. This was done most likely to provide the extra capacity for the transatlantic that was to be rerouted via NL through the London PoP `london-bar4.ja.net`. This rerouting of transatlantic traffic was reported to happen at 30/05/02 06:00.

The result of this routing around the UK was an increase in rtt for traffic between these three nodes with the exception of traffic between DL and RAL. Traffic bound for NL from sites in the UK did not notice any benefits or disadvantages from the change.

Through rerouting all UK to UK bound traffic between the monitored hosts such that it avoid the routers at Reading, we believe that UKERNA was hoping to lower the activity at Reading to maximise the performance of transatlantic traffic.

At the time of re routing of all UK and US traffic through GEANT at 30/05/2002 06:00, the affect to UK traffic was that a slight dip which was noticed on some traces. This was most likely mainly due to slight changes in routing tables and thus different link paths. It is also possible that at this time, the rerouting of transatlantic traffic through Reading reduced the overall load on certain routers around the UK and therefore increased network performance (a reduction in rtt) due to reduced queue lengths.

The network was then re-established to the route found before rerouting away from Reading sometime after 31/05/2002 18:00. This resulted in network performance that was similar to that before the change in routes.

Future Directions and Conclusion

GridNM Architecture

Even though the interactive java front-end adds flexibility to analysis of network metrics, the benefits of summary tables and graphs used for network 'browsing' would be beneficial to get an indication of networks state. A web front-end is especially useful for distributed network analysis and quick summary reports for administrators of the monitored hosts and casual network users.

Whilst udpmon and ping provides ample statistics from which metrics can be analysed and compared, tcp tools are often lacking in information regarding the operation and efficiency of a tcp connection – this would have been very useful in diagnosing some of the performance results of iperf tests. One passive NMA that allows more in-depth analysis of tcp metrics is Web100. Integrating Web100 monitoring based tools into GridNM should be relatively straight forward and could involve either adding a new NMA that uses Web100, such as pathprobe [pathprobe], or alternatively, to implement a generic wrapper to monitor any tcp connection from a host. This is currently work in progress.

One question that arises is how to monitor network tests for different QoS policies. Without prior knowledge of the policies in use in a domain, it may be difficult to accurately judge the performance of the network around a certain host. In terms of the technical applicability of creating QoS marked flows, many NMAs allow measurement traffic leaving a source to be marked automatically with a user defined DiffServe CodePoints (DSCP). This would allow us to monitor according to different QoS policies. However, we need to also give a method of measuring QoS tools that do not have the ability to self mark the DSCP of packets. Another potential problem is that certain network configurations may automatically remark the DSCP at the edge of the network, and so our interpretation of the format of the Type-P packet may be wrong as it changes across many domains.

The current tool set used in GridNM focuses on raw network performance using active measurement techniques. It has been found that at GigE speeds, the bottleneck of transferring a file is often not the network itself or the limitations imposed by transport protocols, but by the speed of the hard disk on hosts. As such, investigations into remote file replication will be introduced into the GridNM architecture. This may impose a security risk should mesh measurements be introduced due to the requirement of shh key distribution. Tools such as bbftp [bbftp] and GridFTP are planned for future testing. It should also be possible to plug-in hard disk performance tools.

We also intent to implement the monitoring of passive measurements such as the polling of remote SNMP nodes to correlate results and metrics obtained through active measurement.

Network Analysis Techniques

In order to judge the effects of certain metrics against other network metrics, we need a way of comparing data sets. The comparison of metrics via scatter plots is already planned in the development of the presentation front-end. However, we need more concrete statistical methods of correlating metrics. It is planned to implement correlation coefficients to enhance analysis.

It was demonstrated that traceroute is vital in the analysis of changes in network metrics. Currently, there is no presentation front-end in GridNM for traceroute information but will be implemented. As changes in link paths are more useful than tables of the same data, a method of automated path change discovery is planned.

It would also be very interesting to compare WAN results to back to back tests between machines. By abstracting data from controllable links, we should be able to see the difference caused through WAN measurements. The type of tests should replicate the NMAs that are in use in the GridNM infrastructure. We also hope to be able to perform high speed tests direct from UCL with projects such as MB-NG [MB-NG] and DataTAG [DataTAG] in order to understand the interaction and measurement of network metrics.

In order to better define baseline measurements, it may be beneficial to enable a method of removing the noise of metrics caused by introducing smoothing functions and fuzzy logic. Also more statistical methods of metric analysis need to be developed. This may be as simple as introducing percentile statistical metrics to aid means and medians. Also, metric correlations need to be established. As well as visual representations for such statistics, mathematical models need to be established to aid network characterisation.

Transport Protocols NG

The optimisation of high performance, high throughput network transport protocols is very important. There is a clear need to maximize the network transport performance and efficiency in order to maximise the network resource utilisation. These considerations apply to both the throughput and latency of network connections. This was demonstrated in the case studies shown.

TCP is one of the most widely used transport protocols with friendly congestion control mechanisms. It has been extensively studied, extended, and refined in the last two decades and has proven to be reliable under a variety of network conditions and applications requirements. Increasingly, however, the standard TCP implementations are becoming inadequate to support high-performance distributed science applications emerging in the science community. These science applications are expected to generate petrabits/sec traffic to be distributed to scientists in different geographical location. With the practical throughput of enhanced TCP implementations still far below Gbit/sec speeds, further research and development efforts are required to enhance the existing TCP stack or to develop a new flexible universal transport protocols that will deliver and sustain multi-Gbits/sec to scientific applications.

The development of new transport protocols must be scalable, programmable, self-configurable, and adaptable to different types of transmission medium and different types of applications.

New transport protocols such as Tsunami [tsunami] that relies on UDP data channels to achieve consistent high throughput. However, the trade off is that they often do not include network friendly techniques to obtain this performance. This often involves sending as much data into the network as possible without concern for the resulting degradation in performance experienced by other internet users. As such, it is not scalable in the face of many competing streams and may saturate the network and cause congestion collapse.

As such, transport protocols should have some indication of what is possible on a particular internet path. TCP does this by probing the network and using implicit or explicit congestion signals to alter the amount of data forwarded into the network, and hence preventing congestion collapse. As such TCP can be thought of as an *adaptive* protocol. On the other hand, technologies such as Explicit Congestion Notification [ECN] (ECN) work in the network to provide possible congestion information to traffic flows – in essence giving explicit information about network state. As such, any ECN capable flow should be programmed to adapt to this notification – usually by backing off or decreasing its transfer rate. For TCP the current consensus would be to react to the signal as a form of congestion. Even though still in its infancy, ECN holds a promising method of improving network performance for all traffic rather than a single flow.

New and existing transport protocols that are to be widely on the internet should include an in depth analysis of its *fairness*. This parameter gives an indication of how well a network protocol will share a network path with other flows – possibly with other, different transport protocols. One technique in popular use today is to take advantage of this fairness by pretending to be more than one person; Parallel TCP [ParallelTCP] streams can be used to achieve a higher link utilisation by pretending to be more than one connection – aggregating the throughput. Another protocol which is currently still under research is MulTCP [MulTCP]. It also pretends to be more than one stream by changing its behaviour to congestion control and slow start, while still operating with one single data stream.

However, while it is possible for end users to use and implement these protocols to achieve higher performance, the question of scaling arises: if everyone uses the same techniques to obtain higher performance, will all users actually experience a resultant degradation in performance?

One way of making sure people do not over use their performance allocation is through the introduction of QoS. However, network protocols are currently blind to different QoS policies due to the use of encapsulation. As such, there is a clear need for transport protocols to be as adaptive as possible to improved or restricted performance of networks set about through different Quality of Service policies also.

The applicability of transport protocols is also important. Some applications perform poorly on the Internet because of a mismatch between the expectations of the application and the services offered by the network. For instance, web browsing is often restricted by the algorithms inherent to TCP.

GridFTP

There is much movement in networking to develop innovative, secure and scalable network services and tools for network-aware scientific applications. These tools, services, and APIs, commonly referred to as middleware, form the basis of the emerging computational data gird infrastructures. They provide the framework for enabling scientific collaboration and for sharing computational resources over wide area networks.

GridFTP [GridFTP] is one such piece of middleware. As such it is not a transport protocol, but a convenient way for grid users to transfer a file from one place to another. The idea is to use a set of protocols which end users can use to do something on the network. In fact GridFTP is nothing more

than an extension of the standard FTP protocol in popular use today. As such it still relies on the use of TCP as a transport protocol – with all its inherent advantages and disadvantages.

In order for GridFTP to be truly useful for grid technologies, we must not only investigate into potential alternate transport protocols for use on the grid, but also facilities and functions that help grid users make the most out of the network. One example is to integrate QoS into the GridFTP framework to automatically segregate and prioritise traffic according to the type of traffic and or end user requirements. Although provided as a separate set of APIs, GARA [GARA] allows end user to define the type of QoS a particular traffic stream should use. It may be beneficial to incorporate both GARA and GridFTP API's to allow QoS'ify all grid traffic.

Another extension to GridFTP could be the possibility of utilising the benefits of different transport protocols – in effect making GridFTP an interface for network transport. This would involve research into quantifying the performance of different transport protocols under different network conditions and with different user Applications. This would in essence transform GridFTP into an intelligent wrapper in which transport protocols can be used in a 'plug and play' fashion.

HighSpeed TCP

TCP is limited primarily in the way it controls its window for low bandwidth paths. However, in the face of increasing high speed networks, TCP increasing the window size is not enough. TCP works under Additive Increase Multiplicative Decrease (AIMD) mechanisms which define how the window show grow and shrink when packet losses occur. The primary reason for the window is congestion control. It has been proven that the current congestion avoidance and congestion control algorithms supported by the protocol limit the efficiency in network resource utilization especially on high-speed data paths given the conservative approach adopted in case of congestion.

The varying types of TCP implementation have changed little in terms of how it changes it window size, but instead have relied on extra feedback from the network to achieve better performance. Even though useful, tcp options such as SACKs and ECN are limited by there functionality when dealing with multi Gigabit networks.

Sally Floyd's stack

High Speed TCP [HSTCP] is a modification to TCP's current congestion control mechanisms for high speed links. The current implementations of TCP are limited by the way it controls its congestion windows in the face of losses which result major disruptions in goodput. Based around the inference that a TCP throughput is inversely proportional to the square root of the loss, Sally Floyd proposes changes to the following 'response function' [HSTCP] for tcp;

$$w = \frac{1.2}{\sqrt{p}}$$

Equation 1. Response function for tcp.

Where w is the congestion window size in MSS sized segments and p is the steady state packet drop rate.

Through implementing three parameters `Low_Window`, `High_Window` and `High_P`, she defines a threshold by which TCP will perform differently to standard TCP. When the congestion window of a TCP connection is at most `Low_Window`, High Speed TCP proposes to use the normal AIMD algorithms; when it is greater than `Low_Window`, it proposes the use of the HighSpeed response function with alternate values of AIMD. As such, when performing high throughput transfer with HighSpeed TCP, it should be less prone to losses and hence able to maintain a high speed connection. This is directly related to the reduced number of rtts required to the size of the congestion window constant. The direct constants involved with the change in AIMD are related to the values of `High_Window` and `High_P`.

One issue with the change to the AIMD algorithms is that HighSpeed TCP may impose a certain degree with unfairness as it does not reduce its transfer rate at much as Standard TCP. Similarly, under congestion control, its slow start can be more aggressive. This is especially pronounced given higher loss rates as HighSpeed TCP will have a larger congestion window and hence able to push more data into the network.

It is our intention to analyse the parameter space of HighSpeed TCP and to perform real life WAN tests over the GridNM infrastructure to judge the advantages of HighSpeed TCP over Standard TCP.

Web100

Web100 is a project to enable automatic and transparent high throughput transport of TCP/IP. The current implementation (version 1.2 alpha) involves a kernel patch (currently only for version 2.4.16) to allow the 'live' kernel level monitoring of TCP variables. Important TCP variables such as the number and type of packets sent and retransmitted, ECN and SACK availability, timeouts and RTO's, the slow-start threshold (`ssthresh`) and the congestion window size (`cwnd`) are all provided on a time-basis, unlike `tcpdump`/`tcptrace` which can supply information on a packet-basis. Web100 can also provide an indication of the TCP sliding window like `tcpdump`/`tcptrace`, but does not require root access to the system.

At a high level, one could use web100 to simply monitor the instantaneous throughput of a TCP stream and debug simple TCP stack metrics such as sender and receiver window sizes. These values may 'cap' transfer rates due to bottlenecks at the sending and receiving hosts rather than any network related bottleneck.

Whilst much of the variable list available through web100 (known as the TCP-MIB) is read-only, a few variables allow write access to allow manipulation of the TCP stack in situ of a TCP connection. These variables currently include adjusting the sender buffer sizes such that a TCP connection can 'auto-tune' given the appropriate parameters.

Benefits to the of using such a NMA would include a instantaneous logging facility to any TCP stream, providing a way of characterising and diagnosing all TCP traffic with more than just a throughput/goodput value. Advantages include providing a visual representation of, for example, the effect of different QoS parameters at the host level on a TCP stream.

IBP

The Internet Backplane Protocol [IBP] (IBP) is a service that allows users to store data in the network. IBP allows allocations up to 4 GB in size. When you request an allocation, the depot (IBP server) returns to you a capability (or key). It is safer than ftp or http for file distribution since you must have the allocation key to access the file and, unlike ftp and http, it does not reveal anything about your file-system. Often termed as Logistical Networking, the impetus is to ship data around the internet such that it is place conveniently near the receiver(s), thereby allowing an increase network performance perceived by the user.

Although technically not a network layer protocol that defines how internet traffic flows are shaped at endpoints, the technology could allow the better use of internet resources through provision.

Multicast and PGM

Multicasting is another mechanism to improve link utilization. The basic idea is to avoid sending the same packet more than once on each link of the network by having router duplicate the packets.

Whilst the areas of research in multicast traffic has had a lot of support, there have been little commercial and hence real life implementation of multicast-enabled technologies, and certainly multicast data replication is still research in progress.

Standard Multicast replies primarily on the use of UDP as the form of transport. However, Pragmatic General Multicast [PGM] (PGM) introduces a new transport protocol that extends UDP for multicast traffic. It also introduces the possibly of reliable transport through existing infrastructures. Another interesting feature of PGM is the ability to nominate Designated Local Repairer (DLR) or alternate sources from data can be received. This facility also allows state to be stored in the network in a similar fashion that that offered by IBP.

Future Directions

The analysis of network metrics can be used to readily compare the performance of not only network paths, but also the transport level protocols. It is our intent to use the GridNM infrastructure to analysis the real life performance of existing protocols to character important links for future the future grid infrastructure. Once an understanding of the factors limiting performance is understood, a framework

towards tuning and possibly new network transport protocol development can be established. Care and attention need to be taken in understanding the accuracy and usefulness of NMAs to determine network performance and a complete survey of NMA will be investigated, tested and compared.

Also more advanced statistical methods of correlation and analysis need to be developed. For WAN test, it may be important and useful to better determine baseline metrics to analyse deviations from this baseline. This may be through smoothing functions and fuzzy logic to remove temporary noise generated by cross traffic.

The results collated through WAN tests need to be compared to lab results and possibly computer based network simulations. Experiments such as back to back goodput tests will give an indication of host bottlenecks. It would also be interesting to see the distribution of metrics for different tools on such a clean and controllable link. Possible characterisations include the effect of rtt, loss and goodput with various background traffic load. An investigation into host bottlenecks will also be conducted. Once this information is obtained in the lab, our findings can be compared our real life WAN measurements.

It is important to find either new protocols that can be used for grid transport or to adapt existing protocols for high speed use. A very good example is an investigation into HighSpeed TCP. We plan to first investigate into the parameter space affecting the performance of HighSpeed tcp. We then plan to implement the new algorithms in a version of Linux and perform both Lab and WAN tests using the modified transport protocol. A theoretical and possible practical analysis of its fairness will also be conducted.

A study of GridFTP and its use in the grid should also be conducted. As it is defined as an interface by which all grid applications to use to communicate with other machines, it is important that GridFTP should evolve into a flexible and adaptable method through which grid applications can make optimal use of the internet. Potential areas of integration are implementation of various transport protocols under a to-be-defined framework and a user level access method to QoS policy control and managed bandwidth services.

Conclusion

We have successfully developed a lightweight ssh based active end-to-end high performance network and application measurement infrastructure that is measuring the performance to various sites in and around the UK and Europe. We hope to expand this mesh to include important grid related sites around the world. The infrastructure includes measurements, archiving, and a flexible method of tool and host addition, configuration and removal. We have also demonstrated the flexibility of the infrastructure by presenting a set of case studies to deduce network status and the performance of various NMAs.

We are working on validating other tools such as web100, raw udp traffic using rude/crude [rude], pathrate, pathload, bbcp, GridFTP, snmp and various other tools and hope to select a recommended set of base measurement tools.

Network monitoring with GridNM will be developed to sustain a systematic, analytical and theoretical method of network diagnosing and performance base-lining. A model of performance under certain network criteria will be created in which potential network characteristics will be determined and applied to other networks. We have also defined how applicability of these tools relate to Grid applications through GridFTP and transport protocols for high performance networks and an overview of network transport protocols and issues relating not only to performance but also usability in the real world. We have identified a potential successor to TCP for high speed applications that is relatively fair to other network users and will implement and test on real life production WANs over the GridNM infrastructure.

References

[DiffServ] http://www.ietf.org/html.charters/diffserv-charter.html

[SLA] Differentiated Services for the Internet, Kalevi Kilkki

[Web100] http://www.web100.org/

[self-similar] Self-Similar Network Traffic and Performance Evaluation, K, Park and W.

Willinger.

[IPPM] http://www.ietf.org/html.charters/ippm-charter.html

[NMWG] http://www-didc.lbl.gov/NMWG/

[RFC2330] Framework for IP Performance Metrics, V. Paxson, G. Almes, J. Mahdavi, M. Mathis, May 1998

[StallingSNMP] SNMP, SMPv2, SNMPv3 and RMON 1 and 2, William Stallings, Third Edition

[netflow] http://www.cisco.com/warp/public/732/Tech/netflow/

tcpdump [tcpdump] http://www.tcpdump.org/

[rfc2678] IPPM Metrics for Measuring Connectivity, J. Mahdavi, V. Paxson, September 1999

[rfc2679] A One-way Delay Metric for IPPM, G. Almes, S. Kalidindi, M. Zekauskas, September 1999

[rfc-ipdv] IP Packet Delay Variation Metric for IPPM, C. Demichelis, P. Chimento, April 2002

[rfc2681] A Round-trip Delay Metric for IPPM, G. Almes, S. Kalidindi, M. Zekauskas, September 1999

[rfc2680] A One-way Packet Loss Metric for IPPM

G. Almes, S. Kalidindi, M. Zekauskas, September 1999

[rfc3148] A Framework for Defining Empirical Bulk Transfer Capacity Metrics, M. Mathis and M. Allman, July 2001

[iperf] http://dast.nlanr.net/Projects/Iperf/

[pathrate] http://www.cis.udel.edu/~dovrolis/bwmeter.html

[pipechar] http://www-didc.lbl.gov/pipechar/

[sprobe] http://sprobe.cs.washington.edu/

[AMP] http://watt.nlanr.net/AMP/

[NIMI] http://ncne.nlanr.net/nimi/

[EDG] http://eu-datagrid.web.cern.ch/eu-datagrid/

[IEPM-BW] http://www.slac.stanford.edu/comp/net/bandwidth-tests/html/slac_wan_bw_tests.html

[SACK] http://www.icir.org/floyd/sacks.html

[FastRetranmits] http://citeseer.nj.nec.com/floyd95tcp.html

[pathprobe] http://www.psc.edu/~web100/pathprobe/

[DataTAG] http://datatag.web.cern.ch/datatag/

[MB-NG] http://www.mg-ng.net

[ECN] http://www.icir.org/floyd/ecn.html

[ParallelTCP] http://www-iepm.slac.stanford.edu/monitoring/bulk/

[MulTCP] http://www.cs.ucl.ac.uk/staff/P.Gevros/multcp.html

[GridFTP] http://www.globus.org/datagrid/gridftp.html

[GARA] http://citeseer.nj.nec.com/foster99distributed.html

[HSTCP] http://www.icir.org/floyd/papers/draft-floyd-tcp-highspeed-00c.txt

[IBP] http://loci.cs.utk.edu/ibp/

[PGM] PGM Reliable Transport Protocol Specification, T. Speakman, J. Crowcroft, J. Gemmell et al. December 2001

[rude] http://cvs.atm.tut.fi/rude/

[NetworkCost] http://server11.infn.it/netgrid/task-dg/task2/DataGrid-07-D7-3-0113-1-5.pdf

[bbftp]	http://doc.in2p3.fr/bbftp/
[tsunami]	http://kb.indiana.edu/data/aebh.html
[udpmon]	<not available>
[traceroute]	http://www.traceroute.org/