Design Principles for Accurate Passive Measurement

John Cleary, Stephen Donnelly, Ian Graham, Anthony McGregor and Murray Pearson

Abstract - Experiences with measuring high-bandwidth telecommunications traffic are described. This work has occurred as part of the WAND project which aims to provide large and reliable teletraffic traces. It is argued that to achieve this, special purpose hardware as well as careful attention to the details of measurement are required. The design of a series of measurement boards operating up to OC-48 is detailed. Experiences with using these boards are described and contrasted with experiences using software based measurement on standard PCs. The uses to which the traces are put are briefly alluded to and lessons for the measurement process drawn from them. The conclusion drawn from our experiences is that standard hardware and operating systems have fundamentally different aims from measurement and can only be used if great care is taken.

Index Terms – passive measurement, measurement hardware, measurement techniques

I. Introduction

Approximately three years ago the WAND research group began a program to measure ATM traffic. We were sharply constrained by cost and decided to start by reprogramming some ATM NIC cards. This paper is largely based on our experience as we have broadened this work to include IP-based non-ATM networks and the construction of our own hardware. We have learned a number of lessons in this work, rediscovering along the way some of the hard discipline that all observational scientists must submit to.

This paper follows the lifetime of data captured from the net through to processing and analyzing it. At each step of this process there are recurring themes that need to be addressed.

The first of these is capacity. The bandwidth of modern networks and the need to capture data over long times imposes stringent demands at every level on both bandwidth and storage capacity. This can be ameliorated by careful use of specialised hardware at critical points and by filtering data so that only what is essential is handed to the next stage.

John Cleary, Stephen Donnelly, Ian Graham, Anthony McGregor and Murray Pearson, Department of Computer Science, University of Waikato, Hamilton, New Zealand,
{jcleary| sfd| ian| tonym| mpearson}@cs.waikato.ac.nz.

The next theme is confidence in the results, that they reflect actual network behavior. An example of the type of problems that can arise is loss of data. Of more importance than ensuring that that loss does not occur is having independent checks in place to detect loss.

Another important aspect of confidence is the need to maintain an audit trail of how data was captured and what has been done to it. Much of the data that we capture is archived and may be reprocessed long after initial capture. Interpretation at this later date requires that careful and detailed information be kept of where and how the data was captured. Also, because there is always the possibility for anomalous and incorrect behaviors in the capturing software and hardware it is necessary to be meticulous about recording which versions of software and hardware were used to capture the data as well as any options that were set during the capturing process. Any later processing of the data also needs to be recorded. For example, in much of the work we do the timestamps recorded on captured data are corrected after the fact. It is essential the software used to do the correction is recorded (and that copies of the software be kept as well).

The final theme that we will consider is access and security. The presence of large amounts of recorded traffic on disk is a juicy target for those who would like to snoop on networks - we have done the hardest part of hacking, that is, gaining access to the network. Security is dealt with by: omitting sensitive data, by encrypting parts of the data, and by maintaining secure access to archived data.

II. Wire Level

The networks that are most interesting to analyse are those that carry large amounts of aggregated traffic. These links are also the most important to analyse from the point of view of telecommunications and service suppliers because they are costly and central to their operations. This means that measurement is being done on critical physical interfaces. Thus even the act of tapping into the wire or fibre can be come a serious operation. This requires a good relationship with the owner of the physical interface.

There are typically two ways of tapping into high-speed links. One is to use a switch or router to duplicate the traffic to an output port that is used solely for measurement. The other is to tap into the physical wire directly. Diverting traffic has the disadvantage that in some cases it distorts

traffic by increasing the demands on internal communication channels, so that it may not accurately reflect what is on any actual physical link. For example, some routers provide a SPAN port to which data can be directed from the other ports. Unfortunately, in the worst case the total amount of traffic that is being redirected can exceed the capacity of the SPAN port as well as possibly interfering in the internal traffic of the router.

An even more extreme example was a recent experience with a simple ethernet switch that we assumed just repeated all traffic that passed through it. However, some strange results from using it caused us to check this assumption and we found that in some cases it was delaying packets by up to 200 milli-seconds [1]. Clearly it was using some form of buffering (and probably had a bug as well). It is not always clear when such phenomena should be seen as bugs in the measurement process or as idiosyncrasies of the particular network being measured. However, the general lesson is that the environment of the measurement point needs to be well understood if clear interpretations of results are to be made.

Physically tapping into a wire has the advantage that it can be done without distorting the measurement process, it is guaranteed to be measuring what is on an actual link. The main disadvantage is that it requires disrupting the link briefly, and that it may alter signal levels on the link. We have used both techniques depending on circumstances.

As described below we like to make use of GPS for precise timing signals on our measurement hardware. This usually requires that there be an antenna installed on a roof and that a cable be taken from the GPS antenna to the machine room. This turns out to be remarkably difficult. In some cases, for security reasons, it is impossible. In others it is just expensive and awkward. This issue has become so difficult in practice that we are currently investigating alternatives to GPS including: using more accurate oscillators (including temperature controlled oscillators); synchronizing with SONET clocks; and using timing signals from cell phone base stations.

III. HARDWARE MEASUREMENT

A. Board Design

Our initial experiments in ATM data capture were made with re-programmed network interface cards (NICs). However, we found that this was not an ideal solution. The principal problem is that NICs are not designed for the accurate time-stamping of cell or packet arrivals. Our design aim was to time ATM cells with an accuracy and resolution that was approximately one tenth of a cell time, that is better than 275 nano-seconds at OC3, and 70 ns at OC12. With commercial NICs we were not able to get to within two orders of magnitude of this accuracy. The other major problem was that no single manufacturer produced a range of NICs that would give us consistent results for different network speeds and protocols. So, we took the decision to design and build our own hardware. This started with the Dag1 card that sat on top of an ATML PCI-bus NIC. Since this time there have been two further versions of the Dag cards released known as the Dag2 and Dag3. Both of these cards incorporate their own Processor and PCI interface. At present a Dag 4 design is underway to further extend the Dag series of cards. Our design aims in the Dag series have been to provide:

- a range of data capture boards with a consistent architecture, capable of handling data rates from 10 Mbps Ethernet to ATM and POS at OC48 and beyond;
- on board intelligence to filter the data before it is passed to the host processor;
- industry-standard interfaces — PCI and Compact PCI;
- a programmable and re-configurable structure to give the greatest flexibility to each design;
- highly accurate timing of cell or packet arrivals, referenced to a universal time standard;
- low cost.

The general architecture of the Dag series is shown in Figure.1. The Dag card is linked into a network by an optical coupler or resistive matching network that non-intrusively duplicates all the cells or packets on the network link.

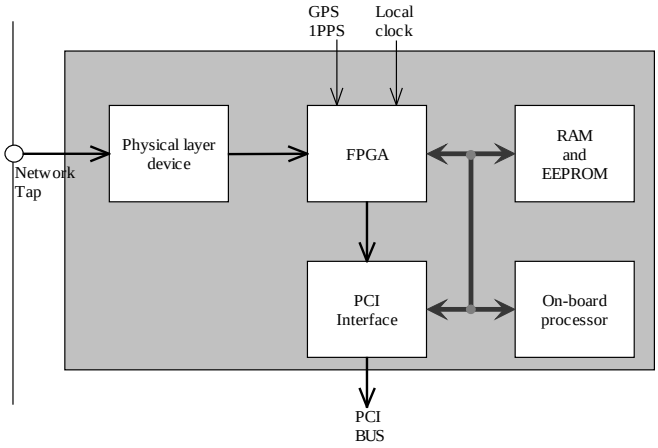

Figure 1. Dag Architecture.

At the heart of the Dag cards is a large Field programmable gate array (FPGA) that is used to:

- generate accurate timestamps for each cell/packet arrival;
- parallelise bytes arriving from the physical interface device into 32-bit words for transfer over the PCI bus;
- perform filtering and preprocessing of packet/cell data.
- buffer timestamps and cell data for transfer over the PCI bus;
- provide the PCI bus interface for the card
- provide dropped cell/packet counters to help

The FPGA's used are made by XILINX™ and are SRAM based meaning that they can be reprogrammed as required. The advantage of this approach is that is possible to download different images. For example, the Dag3.2 was

initially intended to receive and analyze data on OC3 and OC12 ATM links. A simple reprogramming of the FPGA allows the board to measure both ATM and POS traffic. Recently, we have also programmed the board to act as a data source for network testing.

In addition to the FPGA, the Dag2 and Dag3 cards also contain a processor and memory that can be used for filtering and processing data. In the Dag2 we have used ARM7 processors, the higher speed boards, Dag 3 and up, use a 233 MHz StrongARM. However even a 233 MHz processor cannot execute many instructions in the approximately 170 ns cell time of an OC48 network! One typical example of data filtering is where we need only to record packet headers for AAL5 encapsulated IP packets, on an ATM link. In almost all cases the entire header is contained with the first cell of each AAL5 pdu. This first cell is not marked in any way, but the last cell in each AAL5 pdu is marked by having a particular bit set in the *payload type* field of the cell header. AT OC3 and OC12 the StrongARM processor is quite fast enough to maintain state for each active virtual circuit, and to select only the first cell in each pdu for transfer to the PC. AT OC48 the processor is not fast enough, and has to be assisted with table lookup firmware within the FPGA.

Each board has the ability to receive periodic timing pulses which can be used to synchronize the timestamp clocks on multiple Dag cards. When used with a GPS antenna these pulses enable data to be time-stamped to an accuracy of ±250ns to UTC. This is sufficient for us to time delays through ATM switches and to see timing jitter caused by SONET framing. In some cases the timing pulses are used only for local synchronization. One board is designated the master and sends pulses to the other boards. One important use of this is to synchronize boards measuring the two directions of bi-directional ATM circuits. The relative timing between the boards is then accurate although they drift by approximately ±0.5ppm to UTC, that is, a drift of about 1ms in a 15 minute run.

B. PC Issues

After data has been captured on a measurement card it needs to be transferred to the PC for post-processing and recording. In most modern PC's this is likely to take place over a PCI bus. A PCI bus has a maximum theoretical transfer rate of 132Mbytes/sec. However in reality we have observed transfer rates of at most 40-50 Mbytes/sec. The main reason for this is that there is a maximum time (up to 256 bus cycles) that a device can have access to the PCI bus after which it is forced to surrender it. Once a device has lost access to the bus it must wait some minimum time before it can gain access to it again. If however other devices such as disk drives etc wish to use the PCI bus then the measurement card will have to wait its turn. Because of the possibility that data may be lost while it is being transferred over a PCI bus some mechanism is required to detect this loss. In the case of the DAG cards a counter is incremented every time a word of data is lost. Our experience has shown that a PCI bus is easily able to do a full capture of an OC3 stream while we have found particular care is required to perform full capture on an OC12 link. In the case of OC48 a faster bus than that typically supplied in a standard PC will be required.

Once the data has been transferred across the PCI bus it will be stored into the PC memory. We have not found the speed of PC memory to be a issue in the measurement process as it tends to be significantly faster than the speed of the PCI bus. However the amount of memory contained in a typical PC poses real problems. For example, 128Mbytes of memory will be filled in just over 2.5 seconds if capturing all data at OC12. Measurements over longer time periods will be required for many of our projects meaning that the data has to be moved immediately to secondary storage.

IDE disks are the most common type of disk drive found in commodity PCs. The IDE specification defines a maximum transfer rate of 22Mbytes/sec. However we have found that the maximum bandwidth that can be achieved using a standard IDE drive is 5 – 6 Mbytes/sec. This is insufficient to perform full data capture of even OC3 rates. In these cases either expensive arrays of disk drives that employ interleaving techniques or some form of filtering has to be applied to the data before being written to disk.

IV. SOFTWARE MEASUREMENT

Using the hardware measurement techniques outlined above is expensive and complex compared with using a standard PC with an off the shelf PC and NIC to do measurements. For these reasons we have made significant use of purely software based measurements. Such systems have to take a great deal of care to minimise the errors associated with timestamp generation.

The fundamental problem with modern operating systems and associated hardware is that they have goals that are often in direct contradiction to the needs of measurement. Thus network protocols below the TCP level are inherently unreliable. So in any desperate or complex situation the software can just give up and throw away the packet. This is undoubtedly the right thing for an operating system to do as it can significantly reduce the complexity of the kernel code. In contrast measurement needs to be fanatical about not missing any packets and at the very least must report the fact that packets have been lost. An operating system is not at all interested in reporting such loss as there is nothing that the higher levels of software can do about them.

The first source of error can be introduced by the way that many standard NIC cards transfer data over a PCI bus. Because timestamps are not generated until after the packet has been transferred any buffering that occurs in the NIC card adds errors to the timestamp generated.

Secondly, many NIC cards buffer data to minimise the overhead associated with gaining access to the PCI bus. Thus early packets appear to be delayed and timestamps are

bunched together. This can be a particular problem when measuring inter-arrival times.

Furthermore in the process of developing the ethernet version of the Dag cards we were using an industry standard chip set for the physical interface to the ethernet. I turned out that at 10Mbit/sec (but not at 100Mbit/sec) there was a subtle timing problem in the chip set. This resulted in between one in a hundred and one in a thousand packets being lost. This would be unnoticeable in normal operation but was very intrusive in the measurement process (particularly when debugging and ensuring the hardware was functioning correctly).

Once the packet or cells have arrived in the PC's memory the length of time it takes the PC to generate the timestamp is another potential source of error. This time can be very variable and depends on a number of factors such as the CPU speed, other processing load and even the version of the operating system. For example in one version of LINUX we found cases where packets were given timestamps with a smaller value than packets that had arrived before them.

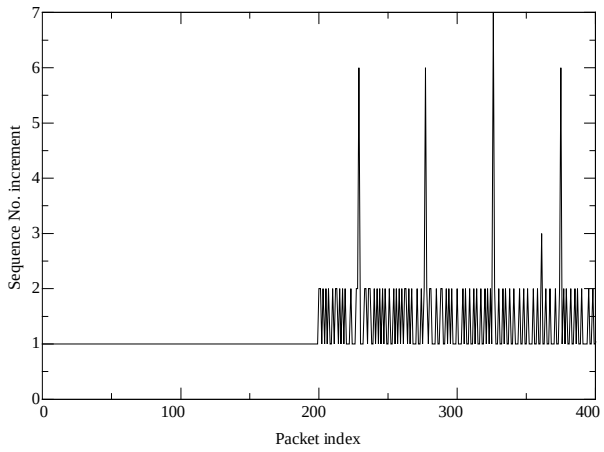

Figure 2 Sequence numbers verses packet index for a measurement on a 100Mbit/sec Ethernet link

One of the aims of Linux is to improve the average throughput of network traffic. To do this extensive use is made of internal buffering. The down side of this is that in exceptional cases, for example when buffers fill up, additional delays and even loss of packets can occur. Figure 2 shows such loss in a measurement done on a 100Mbit/sec ethernet link. The packets were all 1400bytes and generated at a total rate of 60Mbit/sec. The graph shows the difference in sequence numbers of successive packets *received* at the user level. If all were going well then this would always be one. As can be seen errors occur after the first 200 packets and single packets are dropped regularly after that. As well after approximately every 25 packets a group of 5 or 6 packets are lost. We believe that this behavior was caused by a buffer of 200 entries first filling up and then periodically needing to allocate additional memory. The memory allocation required sufficient processing time that a group of packets was lost before it could be completed.
It might be thought that providing a GPS interrupt every second (as was done for the Dag cards) to the PC would help with the accuracy of timing. But this does not help with the problems described above of delays in the NIC cards and kernel level interrupts and processing. Furthermore, great care has to be taken to ensure that additional errors do not creep in as part of the GPS synchronization process itself. For example, there can be a significant time lag before the interrupt from the GPS timer is serviced leading to an inaccuracy in the synchronized clock.

A careful study of the use of software measurement techniques shows that it is possible to significantly improve them. However, the process is far from trivial. Most of our work has taken place in the context of the Linux operating system where the source of the code is available. Without this it would be impossible to understand the operating system effects introduced into the measurement process let alone do anything about them. (Incidentally, Linux is fundamentally no better suited to measurement than any other operating system, but its transparency and modifiability make it by far the best choice if serious work is to be done).

Following is a list of things that were done to ensure that Linux gave good repeatable measurements:

- do the timestamping in kernel space rather than at the user level (for example by using the libpcap library);
- running the machine in single user mode;
- avoiding disk I/O wherever possible (this is difficult when the measurement results themselves must be written to disk);
- not running any active packet generation on the measurement machine;
- disabling time "improvement" facilities such as NTP;
- recompiling the kernel with increased internal buffer sizes;

The point of this list is not so much what was done as the fact that a careful process is needed to ensure reliable measurements. A different list will be needed for each operating system, and indeed we found the behaviors of Linux at this level differed even between successive releases.

Even with these measures in place we still encountered problems with lost and delayed packets at high traffic volumes. The end result of our efforts was measurements that were repeatable to levels below 10 micro-secs. Clearly, measuring to these levels is not a trivial task and much larger errors occur if naive user level measurements are taken.

V. Passive and Active Measurement

Often a distinction is made between *passive* and *active* measurements on networks. In active measurement packets are introduced into the network specifically to make a measurement. For example, the *ping* program sends an ICMP packet, and together with its reply measures the round trip time to a particular host. In contrast passive measurements only use existing traffic, nothing extra is introduced into the network.

A careful examination of passive and active measurement techniques [2] shows that the two techniques are not as different as might be supposed. In particular it is important to use the best *measurement* technology irrespective of how the active packets are generated. Thus all the comments above also apply to measuring the active data. As well there are some additional effects to be wary of.

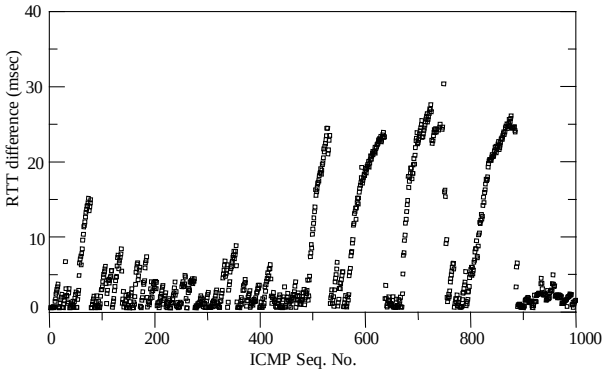

Figure 3 Difference in the round trip times for two machines measuring ping requests generated by one of the machines

For example, it is common to use ping from a single machine that both makes the measurements and generates the traffic. This is very dangerous as the generation process can easily interfere with the measurement process . The end result is displayed in Figure 3. This shows the difference in measured round-trip times between a machine running ping and doing its own timing in user space and a separate machine snooping on the same link using carefully optimized software based measurements (as described above). As can be seen the two processes differ by up to 30milli-secs and are seldom closer than 5milli-secs.

Another difficulty with active measurements is that often they attempt to generate a steady evenly spaced sequence of probe packets. However, the measurement process and buffering interferes with this and we have observed that the outgoing packets are often bunched together giving a most uneven sequence. The same careful procedures as for measurement as needed to ensure regularly spaced outgoing traffic.

VI. Software vs Hardware

In the end the biggest thing that is achieved by using hardware based measurement is peace of mind. The hardware is sufficiently reliable and capable of detecting malfunctions that measured behavior can be confidently ascribed to the network rather than the measurement equipment. Having measurements accurate to less than a micro-second to UTC is overkill for many measurement applications, however, it is not the accuracy that is important but rather knowing that there are not unexpected occasional glitches on the order of tens of milli-seconds in the results. In cases where it is too expensive to use dedicated measurement hardware its use to calibrate and check the software measurements can be very useful.

Another aspect of hardware based measurement is that it is easier to guarantee that it will work under situations of extreme load. The hardware design we use is in most aspects clocked and has to be able to complete all necessary logic within one clock. Thus a period of high load does not look very different to the hardware or in cases where external devices like PCI busses are used it is always possible to check if failure has occurred.

VII. Examples

In this section we will briefly discuss the uses that are being made of the teletraffic data that we are collecting. In each case we will draw conclusions about what is needed in terms of measurement capability and the implications that has for the measurement process.

A. Simulation

One of the biggest uses that we have been making of the data is to re-inject measured traffic into a simulation. [3], [4]. There are two aspects of this usage that are important.

Firstly, it is desirable to have large volumes of traffic. Otherwise if there is less traffic than is needed for the simulation then it is necessary to replicate the measured traffic and re-use it many times in the same simulation. While this is possible it runs the risk of causing correlation between different parts of the data and distorting the results. Thus, it is best to capture large volumes of data whether from a single high capacity link or over a loner period from a lower bandwidth link.

Secondly, simulations are often used to predict rare event such as buffer overflow or packet loss. Such rare events tend to occur about times of peak traffic. However, it is precisely in times of such peak flow that software measurement is most likely to lose packets or show other errors. Thus any errors at such times of extreme load are likely to be amplified in subsequent simulation and analysis.

B. Voice Over IP

Measurement of Voice Over IP (VOIP) traffic is difficult because it is not easy to recognize such traffic just from port usage or internal headers [Cutis00]. Such recognition is best done when long complete sequences are available including control as well as data packets. Thus it is important when making VOIP measurements to have long continuous data series. Times of the order of ten to 30

minutes are necessary to be sure of capturing the start and end of a significant number of voice sessions. In many interesting measurement points VOIP is a small (but possibly rapidly growing) part of the total traffic. Thus to be sure of capturing a significant number of sufficiently long sessions a very large raw trace needs to be captured.

Another important aspect of VOIP data is the trends in its growth over periods of years (this is to answer the question of whether or not it is likely to swamp the internet eventually). This places the emphasis on keeping historical and archived traces and making sure that they are comparable across years in terms of their reliability and accuracy.

C. TCP Time Analysis

A current project is disentangling the different contributions of network latency, network queuing, and server delay in HTTP and other TCP traffic [5]. To do this it is necessary to see a statistically significant number of different pieces of a TCP session. For example, the initial SYN-ACK pair at the start of a TCP session is very important as it provides a round trip time almost completely free of CPU processing overhead on the server machine. But such a pair only occurs once at the start of a TCP session. Thus it has been found that in order to get enough measurements to be statistically reliable measurements need to be done over long times and of large amounts of data.

D. Self Similarity

Another project has been attempting to make measurements of the self-similarity and the statistical nature of tele-traffic [6], [7], [8], [9], [10], [11]. In order for this to be reliable large amounts of data are needed. For example, we have successfully captured a whole week of traffic from one source and are going to attempt to capture some months of continuous data. The number of wavelet co-efficients that are computed rises logarithmically with the length of the trace. So to get more terms requires these very long measurement periods.

E. CRC Generation

In the discussion above we have emphasized the use of passive rather than active measurement. (Although the hardware described is capable of injecting as well as measuring traffic in practice this is significantly more difficult to do as well as potentially disrupting the system that is being measured). In the technique described below we have been able to do one way measurements of delays using only existing traffic [Graham98].

In order to measure cell delay we need to be able to identify cells at the entry and exit points of the network we are testing. Conventionally this is done by injecting special test and measurement cells into the network, these cells have some characteristic that distinguishes them from cells in the normal traffic. However, this technique has two disadvantages. Firstly, an active device is required to inject new cells into the network, secondly the very injection of cells into a network may change the parameters that we are trying to measure. Therefore it would be better if we were able to use the existing traffic in the network for measurement. To do this we need a way of characterizing cells in the existing traffic flow, and this must be based on the cell payload, as the cell header will change as the cell moves through switches. It is not practical to use the entire cell payload; there would be too high a volume of traffic back from the monitor to the matching station, matching would be slow as all twelve words from each cell would have to be matched, and the network user might rightly be concerned about security if their traffic was being copied and re-transmitted. Thus we have investigated the used of a 32-bit CRC calculated over the cell payload to characterize the cell. This has the advantages of being easy to calculate in hardware, of reducing the data to be transmitted to the analysis site from 48 bytes to 4, and of being an irreversible operation, in that the cell payload cannot be recovered from the CRC.

We have chosen to implement the AAL5 CRC32 in our hardware. This CRC is calculated in parallel with cell storage in the buffer as each 32-bit word is received, and the CRC is transferred to the cell buffer at the end of the cell time. The CRC transfer adds only one cycle of the FPGA clock (24 MHz in this implementation) to the time taken to receive a cell.

In order to match one cell stream against another in the analysis software, and thus measure delay through a network, each payload CRC from the cell stream at the entry to the network is compared with CRCs recorded at the exit point. This comparison is carried out for exit cells within a time window relative to the timestamp of the cell recorded at the entry point. The size and position of this window depends on the circumstances of the measurement.

For example, if we are measuring delay through a switch that has a minimum delay of 20 msec, the window will initially be set to start at say time 0, and may run forward to 1 msec. If we are measuring the delay between two points on the Internet separated by a light-speed travel time of 40 msec the window will start at 40 msec, and go forward to about one second. The width of the window is a compromise between increasing the probability of correctly matching cells delayed for a very long time, and reducing the probability of a false match.

False matches can occur if two cells with the same payload CRC are transmitted during the width of the time window. To study the distribution of CRC values we have recorded several million cells from NFS traffic between a workstation and its file server. We have found that the overall number of duplicate CRCs is very small, less than one per cent of the total. However, the important statistic for judging whether or not cell matching will be accurate is the number of duplicates that occur within the time window used for matching. Investigations into this statistic are still proceeding, using recordings of different types of network traffic.

However, at present we conclude that the AAL5 CRC is suitable for cell matching, and that there is a very low probability of false matches for delays in the region of 0 - 1 second.

VIII. CONCLUSIONS

Reliable measurement capable of supporting later analysis and conclusions requires considerable care. It is not likely to succeed just by naively taking standard off the shelf software and hardware. We have emphasized the use of special purpose hardware to achieve the requisite levels of accuracy and reliability. It is possible, albeit difficult, to make reliable measurements on standard software and hardware provided care has been taken to deal with the idiosyncrasies of the operating system and hardware and driver combinations. In any case it is a very good idea to calibrate the software back against a reliable hardware system.

The first main advantages of hardware it that it is capable of providing reliable and verifiable timing, and for one way measurements, timing that can be synchronized to UTC. The second main advantage it hat it is easier to guarantee and validate measurements in cases of extreme and peak loads. The third advantage is that it is possible to capture the very large volumes of data necessary for many of the analytic and simulation uses of the measured data. Further details of the WAND measurement project can be found at http://atm.cs.waikato.ac.nz.

IX. ACKNOWLEDGEMENTS

We would like to acknowledge the contributions of all the members of the WAND research group for providing analysis software and traffic traces. This work was supported by grant UOW-809 from the Public Good Science Fund of New Zealand, and by a grant from New Zealand Telecom.

X. REFERENCES

[1] Donnelly, S.F. Personal Communication, 1999.

[2] Short Term Behaviour of Ping Measurements, MSc Thesis, Department of Computer Science, University of Waikato, July 1999.

[3] Pearson, M. and McGregor A. " A Simulation Study of Network Architectures to Support HTTP Traffic on Symmetric High-bandwidth Delay Circuits" in Proceedings of the Asia-Pacfic Web Conference (APWEB'99), pp 19 – 25, 1999.

[4] Curtis J., Cleary, J.G., McGregor, A. and Pearson, M. "Measurement of Voice over IP Traffic", in PAM2000, Workshop on Passive and Active Networking, New Zealand, 2000.

[5] Martin, H.S. McGregor, A. and Cleary J.G. "Analysis of Internet Delay Times" in PAM2000, Workshop on Passive and Active Networking, New Zealand, 2000.

[6] Darryl Veitch and Patrice Abry, A wavelet based joint estimator of the parameters of long-range dependence," IEEE Trans. Inf. Th. special issue on "Multiscale Statistical Signal Analysis and its Applications", vol. 45, no. 3, April 1999.

[7] P. Abry and D. Veitch, Wavelet analysis of long-range dependent traffic," IEEE Trans. on Info. Theory, vol. 44, no. 1, pp.2 -15, 1998.

[8] P. Abry, P. Flandrin, M.S. Taqqu and D. Veitch. Wavelets for the analysis, estimation and synthesis of scaling data. To appear in Self-Similar Network Traffic and Performance Evaluation, K. Park and W. Willinger, eds, Wiley Interscience, 1999.

[9] Matthew Roughan, Darryl Veitch, and Patrice Abry, On-line estimation of parameters of long-range dependence," in IEEE GLOBECOM'98, Sydney, Australia, Nov. 1998, pp. 3716 - 3721.

[10] D. Veitch, P. Abry, P. Flandrin and P. Chainais. In_nitely divisible cascade analysis of network traffic data," to appear, proceedings of ICASSP 2000, Istanbul Turkey, June 2000.

[11] Matthew Roughan, Darryl Veitch, Martin Ahsberg, Hans El-gelid, Maurice Castro, Mick Dwyer, Patrice Abry "Real-Time Measurement of Long-Range Dependence in ATM Networks" in PAM2000, Workshop on Passive and Active Networking, New Zealand, 2000.

[12] [Graham98] Ian D. Graham, I.D., Donnelly, S., Martin, S., Martens, J. and Cleary, J. Nonintrusive and Accurate Measurement of Unidirectional Delay and Delay Variation on the Internet, Proceedings, INET'98 Conference, July 1998