

Overcoming Precision Limitations in Adaptive Bandwidth Measurements

James L. Alberi, Allen McIntosh, Marc Pucci, and Thomas Raleigh

Telcordia Technologies

445 South St.

Morristown, NJ 07960

{jla, mcintosh, marc, tom}@research.telcordia.com

Abstract—Available-bandwidth measurements determine the amount of spare capacity across a network connection without requiring access to intervening network elements, hence their value in supporting multi-provider service-level agreements and isolating network bottlenecks. However, examination of the various forms of available-bandwidth measurements currently being developed reveals areas for improvement in the way these metrics are measured. To obtain accurate results, existing measurement methodologies require precision in both the generation and capture of carefully structured sequences of packets. The Internet Monitoring Platform (IMP), described herein, addresses these concerns and expedites our comparisons of various available-bandwidth measurement techniques. IMP leverages operating system and hardware improvements, and incorporates a generalized form of packet sequencing that facilitates new measurement techniques.

I. INTRODUCTION

Internet measurements have progressed beyond basic delay, delay variance and loss metrics to include more involved metrics such as available bandwidth. Current research in this area in the Internet measurement community has focused on measurements using adaptive dispersion techniques. These techniques need to inject controlled traffic into the network and measure how it has been changed when it gets to the other end. Properties of the injected traffic are altered based on measurement history. Available bandwidth, link speed or other quantities of interest are inferred from the measurements. The advantage of this approach is that it does not require the cooperation of the transport provider; the disadvantage is that the inference step is difficult [2].

Measuring available bandwidth requires precise timing and generation of sequences of packets, while varying the inter-packet spacing and packet size. Unfortunately, achieving acceptable precision in user-level code involves compensating for operating system context switches and preemption, contention for low-level resources, timing loops, control channels between remote components and other complications. These concerns can distract an analyst from the science at the heart of a measurement technique.

To alleviate these problems, we have constructed a measurement platform infrastructure (IMP) where we can develop a number of available bandwidth techniques under controlled conditions and with the added flexibility of easily altering the specific characteristics of each variation on a theme. This paper describes IMP, discusses the available-bandwidth techniques that we are applying and presents preliminary results for one of them.

II. PLATFORM ARCHITECTURE

The IMP is an active measurement probe that runs on the Linux/X86 platform. Most of the system is written in Java.

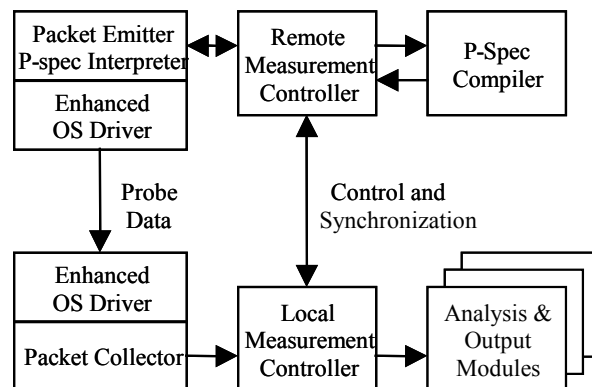


Figure 1. The IMP architecture.

Figure 1 illustrates the general design of the system. All measurement platforms behave in a symmetric manner, with multiple platforms coordinating measurements between nodes as described by a master configuration file. The system takes care of multiplexing connection requests to conserve network resources. It also handles restarts and failures.

We include a permanent, separate control path between the emitting and collecting side of a measurement to provide the flexibility of greater control over the measurement stream. For a simple measurement, such as a VoIP probe, the probe sequence remains constant across bursts, usually with only a sequence number changing. In such cases, the control channel provides simple start/stop access. However, in available bandwidth examples, we need to alter the packet sizes and inter-packet gap during the measurement process, and the control channel provides this capability.

III. SPECIFYING PACKET PROBE SEQUENCES

The characteristics of the sequence of packets that make up a probe will vary with each measurement algorithm and can even change while a single measurement is occurring. Rather than create a number of specialized packet generators, we have designed a generalized packet emitter that can be programmed and tuned in real-time to adapt to the changing requirements of a measurement algorithm.

The emitter accepts a packet specification, called a P-spec, and customizes itself to the described sequence as required by a new collector. P-specs are written in a packet description language that specifies the size, contents, inter-packet relationships and synchronization of the stream. When processed, they allow the tuning of arbitrary parameters while a measurement is in progress. An ASCII P-spec is sent by a collector and converted into a machine-code representation by the emitter. At appropriate times, the

```

fleets_per_burst=3
pkts_per_fleet=20
burst=3
*:(
  WAIT(fleet++)      send fleet index to collector and await response
                    reset new values for gap, size and delay
    gap=arg0
    size=arg1
    delay=arg2
  p1=0
  p3=pkts_per_fleet
  fleets_per_burst:(
    p0=burst          generate multiple packet fleets
                    packet word 0: burst number
                    packet word 2: packet in fleet
    p2=0
    pkts_per_fleet:(
      PACKET()        generate a packet descriptor
      p2+=1
    )
    p1+=1            packet word 1: fleet in burst
    DELAY(delay)     inter-fleet delay
  )
)

```

Figure 2. P-spec for an example asymptotic dispersion technique (ADT) probe. The probe sends bursts of fleets of packets, with the packet size, inter packet gap and inter burst delay adjusted by the collecting process according to the analysis algorithm.

emitter interprets this code into a packet burst described by a set packet descriptors that contain the details of each packet to be transmitted. This set of descriptors is then passed to an enhanced kernel driver to transmit this sequence of packets. Since the kernel does not interpret the P-spec directly, but only operates on a fixed set of descriptors, we do not introduce time varying operations that would interfere with the accuracy of the inter packet gaps.

Strictly periodic trains for basic measurements such as delay, loss and jitter are trivially described by a simple packet specification. Synchronization instructions represent points where the OS driver would return control to the generator for inter-processor communication. In this manner, the behavior of the interaction between the emitter and the operating system is one of a co-routine relationship, where only enough packet descriptors to generate (typically) one burst are evaluated and passed to the kernel output driver at a time.

The descriptions of more complex trains are more involved but provide the ability to adjust packet sizes, spacing and content in between bursts, as directed by the packet collector. A sample P-spec is shown in Figure 2. The WAIT directive passes the current fleet index back to the requestor as an indicator that the last packet of the fleet has been transmitted. It then awaits a continuation directive with possibly updated parameters to initiate the next cycle. These can alter the gap, packet size and inter-burst delay of the subsequent probe as determined by the algorithm.

IV. GUARANTEEING MEASUREMENT ACCURACY

We believe that network probes in general and active probes in particular require an absolute time base for the measurements described previously. Once this requirement is satisfied, time stamping of packets on input and output should occur as close to arrival or departure as possible to eliminate timing inaccuracy arising from processing latency.

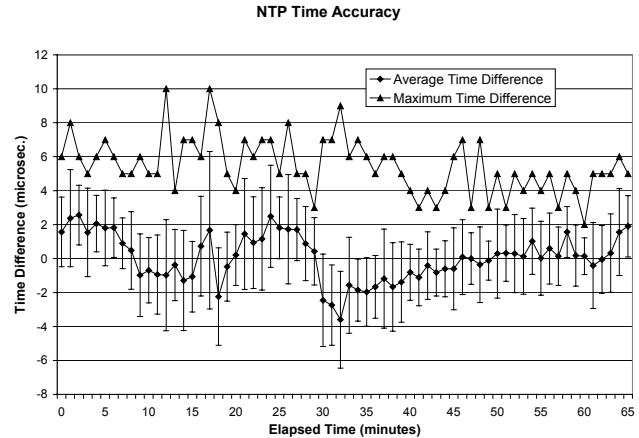


Figure 3. Plot of the NTP recorded time versus a reference time base.

Other authors have developed measurement systems synchronized with absolute time bases. However, IMP provides this functionality with minimal additional hardware by using mostly software solutions based on kernel modules for the Linux operating system. Therefore, the enhancements and features are available in IMP at modest additional cost to the user and with uncomplicated installation.

Packet Generation and Detection

We have also adhered to the goal of using the least specialized hardware in packet generation and input time stamping by using commodity 100-megabit Ethernet controllers (NICs) to generate and receive test packets. Other systems implement algorithms at the user level to try to detect and compensate for context switches and other delays that arise from the operating system IMP uses Linux kernel modules, which are immune to context switches at the user level, to mark packets at the kernel input and output queues adjacent to the NIC – a technique that is the most accurate short of resorting to customized hardware. The measurements shown in Section V validate this technique.

Time Base

The absolute time of each IMP measurement node depends on an inexpensive, commodity global positioning system (GPS) that emits coarse timing data in character-based ASCII and sub-microsecond time marks with a one pulse per second output. We are currently using kernel-based NTP to discipline the system time, which yields the timing accuracy shown in Figure 3.

These data are recorded by comparing the NTP times, which have a granularity of one microsecond, with the times measured by a GPS time code generator system that is used as a reference. We are currently working on a lightweight GPS-based timing system that is implemented with kernel modules and that does not use the complex algorithms of NTP.

V. APPLICATION

The literature contains several types of adaptive dispersion techniques for measuring available bandwidth:

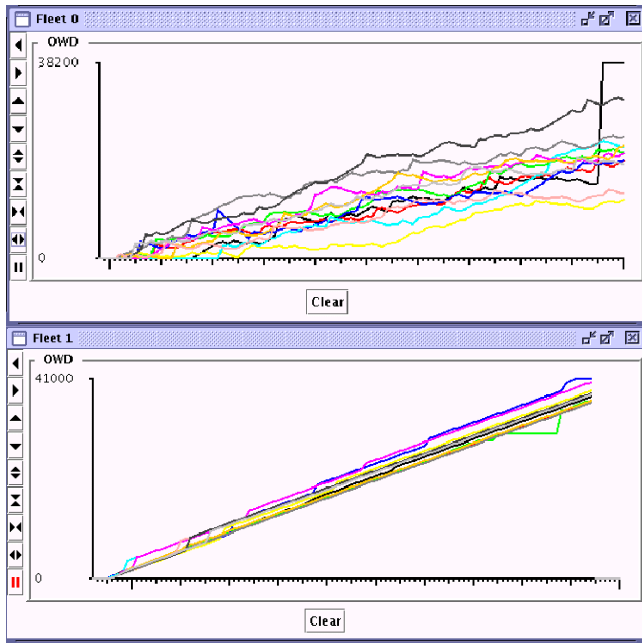


Figure 4. Send times (X-axis) versus receive times (Y-axis) of 12 packet trains set by *Pathload* using time-synchronized systems. Scales differ slightly due to differences in packet spacing.

- *Pathload* [3], which sends packet trains across the network at a fixed bit rate. By adjusting the transmission rate, *pathload* tries to bracket the available bandwidth.
- *Pathchirp* [1], which sends shorter packet trains, and increases exponentially the instantaneous bandwidth of the train by decreasing the packet spacing within a train.
- Packet pair methods [4][5], which vary the size and spacing of one or more pairs of packets.

All of these methods need high quality timing data to produce accurate results. On inspection, the code that implements these methods is seen to devote significant effort to tasks such as achieving the desired inter-packet spacing (most operating system schedulers do not provide fine enough granularity) and detecting data artifacts arising from context switches at both sender and receiver. Our initial tests showed these techniques to be inadequate, hence the impetus to design and build IMP. With its well designed infrastructure, IMP alleviates some of the practical problems with adaptive dispersion techniques as currently conceived and allows the experimenter to concentrate on the more fundamental aspects of these measurements.

The immediate goal of this work is to compare the accuracy and operational usefulness of adaptive dispersion techniques for measuring available bandwidth and ultimately to use IMP as a vehicle for research in novel measurement techniques of network parameters via active probing. Our first experiments in measuring available bandwidth were done with *pathload*, which had the only publicly available implementation at the start of this work.

To add *pathload* to IMP, we converted the analysis code to Java and incorporated it into an IMP collector module. We replaced the packet generation with an IMP P-spec and reused our generic packet emitter. Time-stamping was inherent in the IMP kernel module enhancements. We left the code to detect bad data in place both for purposes of comparison (we have packet generation code that does not use the hardware) and for testing. Ultimately algorithms for detecting bad trains will be services offered within IMP, but we first need a better idea of what constitutes best practice.

We observed significant decreases in measurement variability as we added the specialized IMP components to the system. The upper and lower plots in Figure 4 show the send and receive times of 12 packet trains sent as part of two invocations of *pathload*. The packet trains shown in the upper plot were generated using software timing loops and timestamps were generated by user-level code. The packet trains in the lower plot were generated with IMP using kernel-level assistance, and timestamps were also generated by the kernel. Systematic errors due to variability in the packet timing are obviously lessened in the lower plot. We are currently working to improve the performance of IMP even more and will report these results plus our experience with the other adaptive dispersion techniques mentioned in this abstract.

VI. CONCLUSION

We have constructed a flexible Internet measurement platform that addresses many of the necessary but annoying requirements of converting a measurement algorithm into a measurement system. We have applied this platform to both basic and complex measurement techniques, and have used it to evaluate and study different variations on adaptive bandwidth monitoring. The power of the features included in this platform allows us to incorporate new measurements quickly, with increased accuracy and repeatability.

VII. REFERENCES

- [1] Vinay Ribeiro, Rudolf Riedi, Richard Baraniuk, Jiri Navratil, Les Cottrell, pathChirp: Efficient Available Bandwidth Estimation for Network Paths, *Proceedings of PAM 2003, Workshop on Passive and Active Measurements*, April 2003, <http://moat.nlanr.net/PAM2003/-PAM2003papers/3824.pdf>.
- [2] Constantinos Dovrolis, Parameswaran Ramanathan and David Moore, What do Packet Dispersion Techniques Measure?, *Proceedings of IEEE INFOCOM, April 2001*, pp. 905–914, <http://www.ieee-info-com.org/2001/paper/661.ps>.
- [3] Manish Jain and Constantinos Dovrolis, End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput, *SIGCOMM 2002*, <http://www.acm.org/sigcomm/sigcomm2002/papers/e2ebw.pdf>.
- [4] Jiri Navratil and R. Les Cottrell, ABwE: A Practical Approach to Available Bandwidth Estimation, *Proceedings of PAM 2003, Workshop on Passive and Active Measurements*, April 2003 <http://moat.nlanr.net/PAM2003/PAM2003papers/3781.pdf>.
- [5] Hyuk Lim, Chao-Ju Hou, and Chong-Ho Choi, "End-to-end measurement of available bandwidth based on a packet pair delay model," submitted to IEEE INFOCOM 2004, July 2003