Strategies for Sound Internet Measurement

Vern Paxson

International Computer Science Institute
Berkeley, CA 94704 USA
vern@icir.org

ABSTRACT

Conducting an Internet measurement study in a sound fashion can be much more difficult than it might first appear. We present a number of strategies drawn from experiences for avoiding or overcoming some of the pitfalls. In particular, we discuss dealing with errors and inaccuracies; the importance of associating *meta-data* with measurements; the technique of calibrating measurements by examining outliers and testing for consistencies; difficulties that arise with large-scale measurements; the utility of developing a discipline for reliably reproducing analysis results; and issues with making datasets publicly available. We conclude with thoughts on the sorts of tools and community practices that can assist researchers with conducting sound measurement studies.

Categories and Subject Descriptors: C.2.5 [Local and Wide-Area Networks]: Internet

General Terms: Measurement, Experimentation

Keywords: Internet Measurement, Calibration, Reproducibility, Meta-data, Datasets

1. INTRODUCTION

Conducting a sound Internet measurement study is a difficult undertaking. Some of the hurdles are readily apparent: designing a meaningful experiment, securing permission to deploy the necessary apparatus, testing that the tools work correctly, reducing the raw data, finding illuminating ways to explore the data and present the results. Other difficulties are not so apparent, and yet can significantly undermine the soundness of the final results at different stages of the process.

To cope with these less obvious difficulties, experienced measurement practitioners have learned to incorporate a number of considerations into their efforts. In this paper we discuss a number of methodological strategies reflecting these considerations. The observations we make are not for the most part novel, and indeed the Internet simulation community wrestles with similar issues [9]. Rather, the discussion aims to help students (and others new to Internet measurement) avoid some of the pitfalls that practitioners have come to appreciate over time. (While we often draw upon our own work to illustrate various issues, this is merely for ease of exposition.)

The goal which these strategies target is *soundness*: developing confidence that the results we derive from our measurements are indeed well-justified claims. By this we mean that we have a solid understanding of the strengths and limitations of the measurement process on which we base our results; and, likewise, a solid understanding of the quality of the chain of analysis supporting the results.

The discussion strives to emphasize general principles rather than particular techniques and recommended tools. Doing so, we follow several themes. First, we consider the basic problem of dealing with imperfect measurement devices (§ 2), which can exhibit limitations both intrinsic to their design and in how we use them. We then in § 3 discuss issues that arise when dealing with a large volume of data, which is often the case for Internet measurement studies. In § 4 we emphasize the importance of imposing a systematic structure on the analysis process so that we can later accurately *reproduce* our analysis. Our final theme concerns issues that arise when making data publicly available (§ 5), which can be particularly important for Internet measurement studies for a number of reasons. We conclude in § 6 with thoughts on the sorts of tools and community practices that can assist researchers with conducting sound measurement studies.

2. DEALING WITH ERRORS AND IMPERFECTIONS

We begin with issues relating to the basic task of making measurements. Measurement tools exhibit several types of imperfections. We first discuss the fairly simple notion of *precision*, i.e., errors inherent in the basic design of a tool. This leads to the strategy of associating *meta-data* with measurements, which will be a recurring theme. We then look at errors incurred during the particular application of a tool (*accuracy*) and the critical danger of not measuring what we believe we are measuring (*misconception*). Finally, we discuss the general strategy of *calibration* as a means to detect, and sometimes correct, such errors.

2.1 Precision

Let's use the term *precision* to refer to the maximum exactness that a tool's design permits. For example, a clock that reports time in units of 1 μsec cannot record any distinction in time finer than a microsecond. However, the clock's *actual* precision might be much coarser.[1] For example, for some operating systems the clock actually advances only every 10 msec. Such clocks, even though they

[1]See [15] for an extensive discussion of measurement issues relating to clocks.

IMC'04, October 25–27, 2004, Taormina, Sicily, Italy.

report time apparently to 1 μsec precision, in fact have four orders of magnitude less precision. A key consideration therefore is whether when reporting measurements made with a tool, the report includes an indication of the tool's precision. If it does, then we can formulate *error bars* around results derived from the measurements in order to keep track of how the effect of limited precision propagates through our analysis.

For many forms of Internet measurement, precision can seem readily apparent, especially measurements regarding discrete quantities such as copies of packets, Web server log entries, or BGP routing feeds. For these, precision concerns what information is omitted and what information is kept. For example, copies of packet headers are clearly less precise than copies of entire packets.

However, this can sometimes get tricky, particularly when filtering is involved. An illustrative example comes from our experience with security monitoring. Often when we discover a compromised host, we start tracing all packets subsequently sent to the compromised host in order to detect attempts by the attacker to reaccess the host. Sometimes these traces show failed attempts by remote IP addresses—previously unassociated with the event—trying to access unusual ports on the host. Our initial response: this is the attacker trying to get to their back door! But we've forgotten that the trace is filtered to only traffic sent to the compromised host. A broader trace shows that the new remote IP address in fact scanned our site's entire address space on that port. Given this more precise measurement, we immediately can realize that the access we spotted was incidental, rather than fraught with intent.

Another major area where considerations of precision arise concerns measurements of *time*. Computer clocks have well-defined notions of precision [14], but it can be considerable work to ensure that these precisions are included in reports of Internet measurements, because it is often difficult to access the clock's internal assessment of its precision, and many popular measurement tools do not attempt to do so. A particular pitfall in this regard is the reporting of simplistic precisions, such as simply characterizing the *format* in which time is represented (e.g., the 1 μsec precision discussed above).

However, we must keep in mind a second question: *does the clock precision matter*? For many studies, the computer clock's precision of at least 10 msec, and likely better, is clearly good enough—because the study focuses on much larger time scales, say seconds and above—in which case we can forgo the considerable work of assessing the clock's precision. It is important—as with many of the strategies we develop in this paper—to not lose sight of this critical question of *when does the extra effort matter*. Unfortunately, there are no crisp rules for deciding here. However, clearly sometimes it does indeed matter. More generally, a sound measurement study should strive to include a discussion of whether or not it does matter.

2.2 Meta-data

The issue of tracking measurement precision relates to the broader theme of the *meta-data* associated with measurements traces.[2] Determining a measurement's precision is only half the battle; the other half is *preserving* the information during the course of analysis. It is easy to overlook the need to do this, and many convenient data formats lack a way to annotate measurements in this fashion. For example, for ASCII log formats such as those from Web servers it is often convenient to impose a simple, line-oriented structure to facilitate subsequent processing. Another example is the save file format of the popular `tcpdump` tool: while it includes a field "`sigfigs`" in the trace header for recording the "accuracy of timestamps," there is no API available for retrieving or setting the value.

On the other hand, an example between these two that works much better in this regard is Kohler's `ipsumdump` tool [11]: it reads network traffic (or `tcpdump` save files) and produces configurable ASCII output. While most of the output has a uniform format, it also includes annotation lines marked with an initial "`!`" that capture information such as when the trace was recorded, on which host, and with what options, as well as when drops occur during the tracing process (see below). In addition, by adopting a uniform format for the meta-data it remains easy to write post-processing tools for `ipsumdump`'s output, and also to introduce additional annotations beyond those the tool generates itself.

Finally, because good data is hard to gather, it can have a lifetime beyond what the researcher initially envisions. This means that we can find ourselves revisiting datasets in new contexts for which meta-data information that was not needed for the initial analysis is now important. Thus it can prove highly beneficial to retain meta-data information even when doing so is not of immediate benefit.

2.3 Accuracy

A second form of imperfection in measurement tools concerns their *accuracy*. A measurement is an abstraction of the phenomenon being measured; how well does the abstraction indeed match the actual phenomenon? Accuracy is a *much* bigger problem than precision (which we can view as a subset of accuracy) because measurements are prone to a wealth of different types of errors beyond those imposed by the basic limitations of the measurement apparatus. In addition, because accuracy errors come in many forms, it can require a great deal of diligence and care to assess their presence and magnitude.

To illustrate, consider the accuracy of packet filters. Conceptually, packet filters are simple: network traffic appearing on an attached link is reduced down to a subset by testing each packet against a filter. The matching subset is then recorded, either in its entirety, or just portions of it (for example, the TCP/IP headers, but not the TCP payload). But despite this apparent simplicity, packet filters can exhibit a wide range of problems:

First, they can fail to record all of the packets matching the filter that appeared on the link, termed a packet filter *drop*. Drops can occur due to a failure of the packet recorder (e.g., `tcpdump`) to keep up with the rate at which the packet filter accepts packets. They can also occur due to a failure of the filter to keep up with the rate at which the network tap sends it the raw packet stream, or because the tap fails to keep up with the raw packet stream on the link. These different failure modes have somewhat different implications in terms of their impact on analysis of the measurements, and they also require different forms of instrumentation in order for the measurement process to detect their presence, which complicates characterizing them.

Here again the meta-data problem can arise. Measurement tools are, unfortunately, often deficient in adequately recording such failure information. For example, `tcpdump` only produces an end-of-run summary of the *total* number of drops (both filtered packets and the raw stream, though not drops by the tap itself), so it is not possible to associate drops with the point in time at which they occurred. Furthermore, `tcpdump` does not record these values in the trace file but separately, so preserving the meta-data requires manually associating it with the trace. (Compare with

[2] Here we use "trace" in a broad sense, to mean a series of measurements we have recorded. The strategies we discuss are relevant for measurements of much broader scope than just fine-grained network traffic traces.

`ipsumdump`, which intermingles loss information with packets in its single output, addressing both these regards.)

As cataloged in grisly empirical detail in [22], packet filters can also record packets more than once (with different timestamps for the different copies), reorder the sequencing of packets (again, the timestamps do not reveal the reordering), fail to report dropped packets, falsely report dropped packets when in fact no drops occurred, and misfilter (making incorrect decisions on which packets match the filter[3]).

Whether these errors are common varies a great deal across different implementations and environments. However, since the Internet can both replicate packets and reorder them, we must take care in determining the true cause of duplicates and reorderings in traces, so it is important to understand the degree to which artifacts such as these can taint the accuracy of our measurements. The reader should note that we catalog these problems not to caution about the dangers of using packet filters in particular, but to point out the alarming diversity of inaccuracies potentially present in measurements in general. In addition, see § 2.5 for suggestions on general approaches to ferret out such problems.

Another basic area where issues of accuracy arise concerns clocks. The most basic clock inaccuracy, of course, is failure to be synchronized to true time. Other problems include abrupt jumps forward or backward, and advancing at erroneous rates [23].

Clock inaccuracies can present significant difficulties when analyzing behavior on fine time scales, or when comparing measurements made by different clocks. While algorithms exist to compensate for such errors [23, 18, 30], an increasingly viable alternative is to use clocks that are synchronized by GPS. Such synchronization can be highly beneficial compared with clocks synchronized using NTP, which can still exhibit forms of skew, drift, and jumps, as chronicled in [23]. One pitfall that arises when using GPS, however, is that we can be beguiled into simply assuming that the clocks must report correct time, rather than checking that in fact they do. The path from the highly-accurate GPS timing signal—through perhaps daisy-chaining, into the operating system kernel, perhaps out to the NIC or perhaps applied to an entire buffer's worth of packets at one time—and ultimately to the reported per-packet timestamps can introduce significant errors.[4] GPS is not the only possible answer here, either. Pasztor and Veitch make the key observation that for most network measurement, absolute time does not matter nearly as much as fidelity in the *rate* at which a clock advances, and they demonstrate the feasibility of obtaining a highly stable clock rate without requiring GPS synchronization [20].

One difference between issues of clock precision vs. accuracy is that the former can be known in advance, and hence is easier to incorporate into meta-data. The latter can require post-processing, which points up the utility of designing meta-data management mechanisms that accommodate dynamically generated attributes.

As with precision, we must keep in mind the question of whether a particular form of accuracy in fact matters for the measurement question at hand. Again, the same issues apply regarding using calibration for assessing errors and being aware that data taken today may be re-analyzed in a different context tomorrow.

Finally, the point of this discussion is not to induce consternation—*how can we ever trust our clocks?*—but rather to serve as one of the motivators for the importance of the calibration techniques discussed in § 2.5.

[3]For example, the `tcpdump` filter "`tcp[4:4] == 0 and tcp[4:4] > 0`" should never accept a packet since it matches a TCP sequence number that is both equal to zero and greater than zero. But in some `tcpdump` versions it in fact can accept packets due to a bug in `tcpdump`'s optimizer.

[4]Barford [4] once found a GPS clock setup that, due to subtle interface issues, routinely reported time exactly 1 second off from true time!

2.4 Misconception

Related to the problem of measurement tool accuracy, but different in terms of its basic cause—and also potentially of much greater importance—is the danger of *misconception*: errors in equating what we are actually measuring with what we *wish* to measure. The pitfalls relating to misconception take many forms. Along with our example above about misinterpreting attacker activity because we failed to account for a loss of precision due to packet filtering, consider:

- Measuring TCP packet loss by counting retransmitted packets, which risks overlooking the problem of packets retransmitted unnecessarily, or of packets replicated by the network (see [10] for a study that explicitly acknowledges this difficulty, and [3] for a study demonstrating that differences in the two rates can be quite significant).

- Assessing end-to-end Web transfer times but failing to account for hidden proxies. For an extreme example of this problem, Allman reported measuring a TCP session that took only 10 msec to establish a connection, transfer its data, and complete the three-way termination handshake ... with a host 100 msec away, that he subsequently discovered had been powered off! The paradox was due to a local hidden proxy completely intercepting and terminating the connection [1].

- Quantifying TCP throughput using transfers with large socket buffers and modest transfer sizes, such that application-level timing (e.g., measuring how long it takes to issue all of the `write` system calls) measures how long it takes to fill the kernel buffer, rather than how long it takes to transmit the data and receive an acknowledgment.

- Computing the distribution of TCP connection sizes by capturing SYN and FIN packets and using the difference between their sequence numbers to compute the size of each connection—but failing to recognize that the very largest connections on the monitored link might often already be underway when we start tracing, or have not terminated when we finish, and thus we will miss their SYNs or FINs and fail to include them.

- Characterizing global BGP reachability in terms of reachability as seen by a single, multi-hop BGP peer, without accounting for how outages in the multi-hop BGP peering session magnify into apparent global outages [28].

Each of these is an easy-to-make mistake. They arise from a mismatch between the mental models we use to abstract network behavior and the actual complexity of the beast. The last example is particularly of note, as it is an instance of the general problem of *vantage point* [22]: that the location of exactly where a measurement is performed can significantly skew the interpretation of the measurement, in quite non-apparent ways. Some vantage-point issues cannot be corrected without additional information, and in fact this leads to a fundamental problem in network intrusion detection of adversaries being able to exploit vantage-point ambiguities to evade security monitoring [25, 8].

Another broad class of misconception errors concerns the degree to which individual collections of Internet measurements are often

not representative. Numerous studies have established that Internet properties often vary a great deal both across different points in the network and across different points in time at the same place in the network [5]. While for some properties such as congestion, variation is to be expected, for others (e.g., median FTP item size [5]) it is quite surprising. A general strategy we can suggest here towards more sound Internet measurement is, if at all possible, to gather more than one type of dataset—either from a different location (this often proves the most fruitful) or from a different time. Having even just two datasets rather than one can prove illuminating and sobering in realizing that the phenomenon under study is more diverse than we had pictured.

Finally, we note that a major pitfall with problems of misconception is that they can be difficult for researchers to identify by themselves, since the problems arise out of our own incomplete mental models. Thus, it can be extremely helpful to seek out *early* peer review of a proposed measurement effort, particularly from peers with somewhat different perspectives.

2.5 Calibration

We now turn to a discussion of a set of techniques that can greatly help with detecting problems of inaccuracy, misconception, and errors in analysis. We term these, somewhat loosely, as *calibration* strategies. Four general ones are:

- Examining outliers and spikes.
- Employing self-consistency checks.
- Measuring facets of the same phenomenon different ways and comparing.
- Evaluating synthetic data.

In the discussion that follows,[5] it is good to keep in mind that the point of these strategies is not to achieve perfection in the correctness of our datasets, but rather to *build confidence* that we have a solid understanding of both our data and the processes by which we have measured and analyzed it.

2.5.1 Examining outliers and spikes

The first strategy stems from two important points: *(i)* outliers (unusually low or high values) and spikes (values that repeat a great deal) represent "corner cases" at the extremes of measurement where problems often manifest, and *(ii)* these corner cases are *easy to locate*, so we can leverage the diagnostic benefits of inspecting outliers and spikes without spending a great deal of effort.

While often spikes and outliers turn out to be genuine, and perhaps not unexpected, phenomena, they can also reflect measurement errors, analysis errors, or misconceptions. An example of a measurement error in this regard is analyzing a set of round-trip times (RTTs) to find that the smallest outliers are physically impossible given speed-of-light constraints, and instead realizing that the timings are due to an infelicitous clock adjustment.

An example of an analysis error would be the same scenario but the bad RTT is due to a mismatch in associating the outbound packet with the wrong reply for determining its round trip. Another example comes from our experiences computing connection sizes using the sequence numbers in TCP SYN/FIN/RST packets. For one connection, we found a size of 4,294,967,295 bytes. This seemed unlikely, given the duration of the connection, and upon further investigation we found that the size reflected a bug in our sequence number computation yielding a size of exactly $2^{32} - 1$.

An example of a misconception error caught by such techniques arose when we were analyzing Telnet connection arrivals to assess the degree to which they were well described using a Poisson model. One dataset had a spike of nearly 2,000 connections whose interarrivals were all 180.436 ± 0.002 seconds apart. These turned out to be due to a special-purpose host with an attached modem. Whenever a call came in to the modem, the host launched a Telnet connection to an on-line library catalog. But the modem had broken and continually sent a false signal indicating a call had come in. This led to repeated connections, each timing out after 180 sec. The misconception here is subtle: the error in our mental model was that we presumed we were measuring *human*-initiated activity, for which a Poisson model might indeed make sense. If we had not uncovered this misconception, we might have determined that the dataset was inconsistent with Poisson modeling—true enough—but missed the finding that if we removed the blatant machine-initiated activity, then the remainder was in fact well-modeled as Poisson.

2.5.2 Employing self-consistency checks

The strategy of employing self-consistency checks works by exploiting additional properties of the measured phenomenon to see if they agree with behavior reflected in the initial measurement. That is, test whether properties that *must* hold do in fact hold. Sometimes they will fail to hold because in fact our conception of why they "must" hold is incorrect, in which case unearthing exceptions can be enlightening for refining our mental model. But usually when they fail to hold, it is due to some sort of measurement or analysis error.

For example, consider the problem when tracing TCP traffic of determining whether the trace includes all of the traffic associated with a given connection, or whether some of the traffic is missing due to a measurement problem such as a packet filter drop. In this case, we can use the additional property of the TCP protocol that it is designed to be highly reliable. One facet of this strong reliability is that a TCP receiver should never send an acknowledgment for data it has not received. Since TCP acknowledgments are cumulative, this means that we can inspect each acknowledgment present in a trace to see whether at the point in time it was sent, all of the data up to the sequence number it acknowledges has indeed been seen previously in the trace. If not, then we have strong evidence that the tracing suffered from some packet filter drops, because we believe the alternative explanation that the TCP receiver really did acknowledge unreceived data highly improbable (however, see below).

Thus, by analyzing the deeper semantics of the traced traffic, we can develop a higher degree of confidence that the trace is indeed a sound measurement of the traffic; or, alternatively, we can locate portions of the trace that suffer from measurement errors.

2.5.3 Comparing multiple measurements

A third calibration strategy is measuring the same phenomenon different ways and comparing the results. A simple example of employing multiple measurements in this fashion is to run two separate packet monitors, to see if they agree on which packets were present in a captured stream. When they disagree, we can sometimes distinguish between packets genuinely lost by the network vs. packet filter drops by considering the directionality of a missing packet with respect to the monitors: if the downstream monitor sees a packet missed by the upstream monitor, then the upstream moni-

[5] Also, see [6] for discussion of an exemplary system that automates a number of calibration techniques ("trace sanitization") in the context of ongoing, very high volume packet measurement.

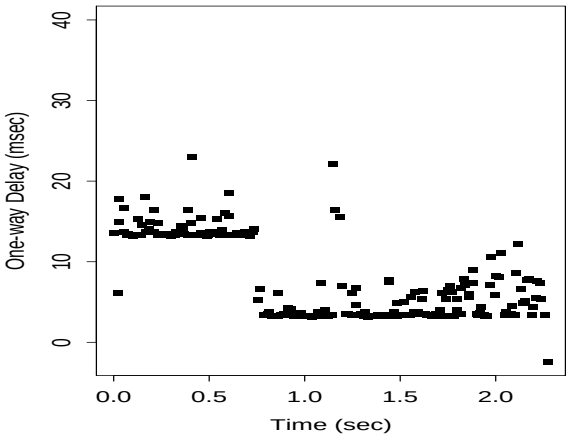

Figure 1: One-way transit time step that could be due to either a routing change or a clock adjustment.

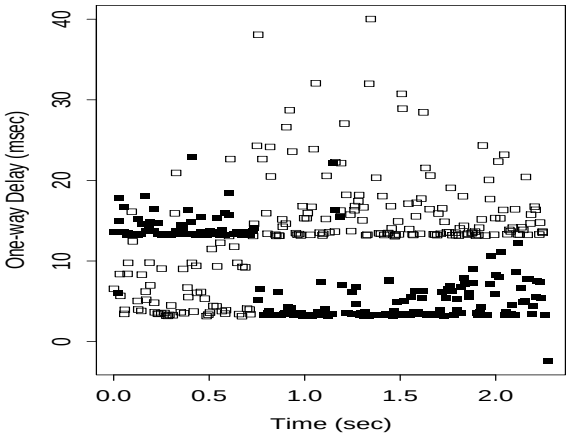

Figure 2: Incorporating additional measurements resolves the change as due to a clock adjustment.

tor very likely suffered a packet filter drop.[6] Note that this measurement needn't be heavyweight—the second "monitor" could simply be the receiver accounting for each packet it receives.

Calibrating via multiple measurements can also involve conducting additional measurement. For example, consider the problem of assessing the accuracy of packet filter timestamps when measuring network transit times. In Figure 1, taken from the data analyzed in [23], each solid square reflects a one-way transit time computed by subtracting the timestamp generated by the sender upon transmission of a given packet from the timestamp generated by the receiver upon arrival. At about 750 msec into the transfer, we see a sudden downward shift in the transit time. Such a step could be due to a routing change which has shortened the end-to-end path latency, or a clock adjustment at either the sender or the receiver.[7] The first is an interesting network event, the second merely a measurement glitch.

[6]It is important to note two other possibilities here: the packet, unbeknownst to us, took a different route through the network than we imagine, and thus bypassed the location of the first monitor (a form of *misconception*); or we may have confused two identical or nearly-identical packets.

[7]Note, both the sender's and the receiver's timestamps were monotone increasing, so there was no *obvious* clock adjustment.

Figure 2 illustrates the power of using additional measurements to calibrate [23]. Here we include measurements of the reverse path (hollow squares), computed in the same fashion. Using Occam's Razor, the additional measurements allow us to deduce the highly likely presence of a clock jump, rather than network phenomena such as a routing change, due to the equal-but-opposite character of the jump in the two directions, which matches what we would expect from a clock jump, but would require unusual network dynamics. (This example illustrates how sometimes calibration is rooted in *plausibility* rather than direct comparison, but that does not undermine its diagnostic power.)

It is also possible to apply this strategy not in terms of multiple measurements but rather in terms of multiple versions of our analysis. While we almost never have the resources to completely reimplement our analysis software solely for purposes of calibration, we can often find snippets of analysis that are easy to recompute using an entirely different approach. When we recognize such cases, they can provide a valuable check on the soundness of our software. For a simple example, suppose we are analyzing HTTP packet traces and part of what our software does after reassembling the TCP byte streams in the trace is generate a report of how many GET, HEAD, and POST requests are present. We could also compute these values by extracting the raw strings present in the packet trace file (e.g., using the Unix `strings` utility) and directly counting occurrences of the strings "`GET`", "`HEAD`", and "`POST`" within them. We would not expect an exact match between the counts (there might be additional instances of the strings that happen to be in HTTP items or other headers, or due to retransmitted packets), but we should find *at least* as many instances from the raw counts as our program reports from its refined analysis, and likely not too many more. If we find a discrepancy that seems a bit "funny" (one's intuition for what constitutes an anomaly in this regard develops with experience), then it merits further investigation to determine whether our software might be flawed.

We can apply a similar strategy (and often more cheaply) within a single analysis program by computing the same value multiple ways. For example, if we are reassembling TCP byte streams, then for each connection we can compute a running total of how many bytes in the stream we have processed. At the end of the connection, this total should match the difference between the SYN and FIN sequence numbers. Or, if comparing retransmitted with non-retransmitted packets, rather than keeping count of only retransmissions and total packets, and computing non-retransmissions as the difference, we can explicitly maintain counters for all three, and ensure that they agree when we finish.

In general, when applying calibration of some form to our measurements, we unfortunately must steel ourselves for considerable extra effort. First, there is the work to acquire additional measurements and devise and assess the self-consistency checks. For large datasets, this includes the effort necessary to *automate* the checks in some fashion, since manually checking (e.g., visually inspecting plots such as those shown above) rapidly becomes intractable faced with a large volume of data.[8]

Second, experience shows that very often when we do this extra work, we find that the measurements *do* in fact include inconsistencies or errors. We then need to gauge their significance or devise ways to soundly compensate for their presence. Sometimes we can get away with discarding measurements tainted with inconsistencies, providing that we first reason through whether doing so will impart a bias on our analysis of the remaining measurements. (For example, discarding traces that we find have packet filter drops will

[8]See [21] for an example of a tool that automates a number of such checks for packet traces.

often bias the remainder towards having lower traffic levels, since packet filter drops generally occur during periods of high traffic load.) Other times, we can attempt to remove the error, such as the timing adjustment algorithms developed in [23, 18, 30].

We need to also be prepared for the fact that sometimes calibration analysis is less crisp than the examples above. For example, if we find clear evidence of steady clock skew in one direction, but not in the other, how should we treat the possible error? (See [23] for an example of such.) It might still reflect a clock artifact, but might instead reflect some peculiar network behavior. Or, going back to our example of detecting packet filter drops by analyzing transport-protocol behavior, consider the surprising fact that TCP receivers have indeed been recorded acknowledging data never received! (See the section on "Crud Seen on a DMZ" in [24].) Fortunately, such ambiguities are often sufficiently rare that we can either present and analyze them separately, or simply remove them (noting this fact) because their total numbers are not sufficient to affect our overall analysis.

2.5.4 Evaluating synthetic data

A final calibration strategy for developing confidence in our analysis software is to test it using synthetic data. If the software processes text input (e.g., Web server logs), then this can be as easy as hand-editing some of our measurements to introduce changes (especially, manufacturing outliers and spikes) which we then test whether the software correctly processes. Other forms of input may require additional work, though we may be able to leverage additional tools in this regard. For example, the `tcpslice` utility available with some Unix systems can allow us to edit traces by extracting or gluing together separate sets of packets, and the `NetDuDe` (NETwork DUmp data Displayer and Editor) utility [12] provides a powerful visual editor for transforming packet traces in a wide variety of ways.

A related technique for verifying the correctness of analysis algorithms or the theories and modeling that provides their underpinnings is to use Monte Carlo simulations: when the analysis is based on the data having a particular form (for example, conforming to a given statistical distribution), simulate multiple random instances of that form to ensure that—at least if the statistical assumptions are correct—the analysis works properly. For example, for analysis that assumes Poisson event arrivals, not only should we test that the measurements are indeed well-modeled as Poisson, but we should also test that for statistically pure, synthetic Poisson data, the analysis produces precisely the results predicted by the theory.

3. DEALING WITH LARGE VOLUMES OF DATA

Depending on what is being studied, a collection of Internet measurement data can span many millions of measurements. This large scale leads to a number of potential difficulties to keep in mind.

The first problem is bumping into system limitations such as disk space, maximum file sizes, number of files on a volume, or directory search performance, that lengthen, in painfully mundane ways, the data analysis process. A related problem concerns the software system used for statistical analysis: many systems have upper bounds on the amount of data they can process before they began to thrash.[9] This problem can even manifest when generating plots to visualize data. For example, we've found that using a simple filter to strip out redundant points (ones that lie directly on top of one another) can greatly speed up rendering for some types of analysis.

A final form of "system limitation" concerns the utility of some types of statistical techniques. For example, it is well known in the statistics community that large datasets almost never have statistically exact descriptions, with a specific example being an experiment in which 26,306 throws of 12 dice failed a χ^2 test for fitting the predicted binomial distribution [13]. This is not due to flaws in the tests but rather that they are *too good*: they are able to detect minor deviations from statistical exactness, and given enough real data this will indeed manifest.

These difficulties can combine to lead to potentially enormous "edit–compile–debug" cycles when developing and applying analysis tools. This in turn runs the risk of hindering our thorough exploration of the available data. A general strategy that often helps a great deal here is to extract small subsets of the data and first analyze those in depth.[10] These can be selected randomly to avoid bias; more generally, it is highly recommended to select additional subsets fairly early on in order to get a sense of what sort of variations are present across the subsets. One of the main goals of such early analysis is to find properties that in fact hold across the subsets. When these appear well supported, we can then perform the honed analysis on the entire dataset (perhaps batched as a large number of additional subsets). Coupled with visualization techniques that allow us to compare the property across the subsets, this can provide an effective means for getting a handle on a very large dataset without becoming sidetracked by dealing with system limitations.

4. ENSURING REPRODUCIBLE ANALYSIS

A very important facet of conducting a sound measurement study—and one that is easy to overlook initially, as its import only becomes apparent later on—concerns structuring the processing of the measurement data to ensure that the analysis derived from the data is *reproducible*.

This need for a disciplined approach to reproducible analysis is well illustrated by the following experience, all too familiar to not only the author but also a number of colleagues with whom we've discussed this problem:

A researcher works on a measurement study at a feverish pace in order to submit the research to a conference. The work is deadline-driven, and the common mistake of underestimating the dismaying complexity of the measurement and analysis process compresses the overall effort into a period of intense immersion in understanding the data.

Later, the researcher receives feedback from the conference reviewers. Inevitably, a reviewer points out a facet of the analysis that would be more insightful if done in a slightly different fashion, or pushed a bit farther.

But now months have lapsed, or perhaps even more if the work was submitted for journal publication. Clearly, the researcher should address the reviewer's comment—doing so strengthens the work. The question, though, is how to go about doing so.

The natural response might be to simply modify the analysis scripts according to the reviewer's suggestion, crunch them against the data, and update the text with the revised results.

The more sound path, however, is for the researcher to first reassure themselves that they understand the details of how they

[9] From an informal poll of network measurement colleagues, the statistical systems they most commonly use are *R* [26], the related *S-PLUS* [27], and *Matlab* [7]. Quite a few reported using Perl or C programs either for additional analysis or for working around scaling problems in these systems.

[10] Note that these subsets are for the *initial* analysis, in order to help us hone the analysis procedures. We do not derive our *final* results from just the subsets!

reached the original findings in the paper in the first place. To do so, they re-run the analysis scripts against the data in order to reproduce the original numbers.

It is at this point—we know personally from repeated, painful experience—that trouble can begin, because the reality is that for a complex measurement study, the researcher will often discover that they cannot *reproduce the original findings precisely! The main reason this happens is that the researcher has now lost the rich mental context they developed during the earlier intense data-analysis period. Their ad hoc notes on how they treated the data—a catalog of the various measurement glitches, data removed as outliers, fudge factors applied to correct problems caught shortly before the deadline—contain holes and inconsistencies. They might find that they must have used somewhat different versions of the analysis scripts for different parts of the paper. They may also find that they made mistakes in producing the original text (such as re-rounding a number in a table that had already been rounded), which can now only be inferred.*

This can be a dismaying position in which to find oneself. Rectifying the discrepancies in order to soundly reproduce the original findings can require spending an exorbitant amount of time tracking down a host of minor details.

How serious are the hidden flaws? Are they really worth this sort of effort? Unfortunately, we often can't know without delving into them individually. From personal experience, they very often are minor in terms of their impact on the original results. Every now and then, however, they are quite serious (the most significant for us concerned the conference version of [24], for which we found when generating the final copy an off-by-a-factor-of-two error that meant the "high performance" claims in the paper had to be halved).

A vital observation here, however, is that this unhappy situation is not fundamentally unavoidable. While the degree to which large datasets are rife with weird eccentricities, and the sheer necessary scale of the analysis, present ample opportunities for confusion along the lines of what we sketched above, a key means for minimizing these problems concerns adopting a systematic analysis process that emphasizes reproducibility. Such a process aims to maintain an "audit trail" for the chain of analysis, starting from the raw data and eventually leading to the derived findings and plots. The process needs to include a notion of version control so that it is possible to understand how specific results were obtained at specific times, and what has changed in the analysis process since that time.

An example of such a process is to enforce the discipline of using a single master script that builds all analysis results from the raw data. (A similar practice is already common in the network simulation community, though there the analysis chain is often more tidy.) The script maintains all intermediary, reduced forms of the data as explicitly ephemeral. Accompanying the use of such a script is also the discipline of maintaining a notebook cataloging the different forms of data reduction and analyses performed, and to what effect, using a version control system to track changes to both the notebook and the scripts.

By structuring analysis around the use of such a master script we gain two major benefits. The first is to always be able to reproduce our results, minimizing the headaches described above when we need to reanalyze the data at a later point in time. The second is that we then have a way to explore the analysis of the data in a consistent fashion, so that we can both systematically incorporate new elements into our analysis and ensure we apply them coherently to the entire data set. We also can then use the version control system to fully *undo* analysis explorations that turn out to lead to dead ends.

While the benefits of using such a structured approach to analysis are large, the difficulties it brings with it concern the *efficiency* of the resulting analysis. We would really like to be able to express the dependencies between different scripts and different sets of intermediary results, so that when either the scripts or the reduced data change, the effects can be efficiently re-analyzed, recomputing only the necessary intermediary results rather than all of them. A significant hurdle to such an approach is developing mechanisms to express the dependencies at the right granularity. For example, experience with the popular Unix "make" utility is that its file-level granularity is too coarse. Another particularly challenging task is devising ways to apprehend what *changed* between the earlier analysis and the re-analysis. But if we can construct "change visualization" tools, then we can conduct much more effective analyses of large datasets.

5. MAKING DATASETS PUBLICLY AVAILABLE

One difficulty the Internet measurement research community faces is a dearth of publicly available datasets. These are needed to serve as a common framework used for different analyses; to confirm analyses conducted by other researchers; and to address the major problem of attaining representative measurements (from multiple sites, and from multiple periods of time, as discussed in [5]). However, building a public dataset repository faces formidable logistics [2].

In this section we look at some considerations for making datasets publicly available. As developed earlier in this paper, a basic requirement is that the measurements need to include rich associated meta-data. This needs to encompass the data's precision and issues affecting its accuracy; more generally, any information regarding the data that cannot be constructed from the data itself. For example, we would like traces of traffic seen on busy links to include comments along the lines of:

> *No packet loss information was recorded. The data was analyzed for sequencing holes—these exist, but it is not known if they reflect measurement drops or packet loss. A denial-of-service flood elevates packet levels and losses during the noon hour. This site's Internet access is bandwidth-limited by an OC-3 access link.*

so that the dataset carries with it the information necessary for understanding its particular structure.

Another form of meta-data that is highly helpful to include regards the analysis tools and scripts that have been previously applied to the data, to facilitate both reproducing these results and building on the earlier work. Yet another type of meta-data is auxiliary information associated with the measurement. For example, it often is highly useful to have a mapping of IP addresses $\leftrightarrow$ hostnames, or the routes between the measured hosts. These change over time; if the data fails to include them, they can be recreated only imperfectly at a later time.

These problems become particularly acute for *longitudinal* data, i.e., data gathered over long time frames like years. There is great value in such data as a way to understand not only how the Internet and its use has evolved in practice, but, more importantly, for identifying those things that do *not* change as the Internet evolves. ([5] refers to these unchanging elements as *invariants*, and discusses both their importance in terms of providing a foundation for understanding, and the difficulty of identifying them.)

The existing corpus of multi-year longitudinal data is very small, because it is difficult to sustain such measurement efforts over time.

From our experiences with amassing and working with a few such datasets, the most important advice we can give in this regard is to *periodically analyze* the ongoing measurements. The analysis here needn't be in depth, and can in fact be highly automated. The goal is not to derive new findings from the measurements but instead to exercise consistency-checking as a form of calibration. This process serves two roles:

- Discovering whether some facet of the measurement is broken (failing to deliver sound values).
- Driving early in the process of gathering the measurements the requirement of accumulating the meta-data necessary for checking the data's soundness.

Again, see [6] for an example in this regard of a thorough system for gathering very large volumes of measurements in an ongoing fashion.

A second, quite different problem we face with building up a larger set of publicly available traces concerns the reluctance or sometimes legal impediments to making data available, for reasons of privacy, security, and business sensitivities. This has led to the development of anonymization technologies, primarily in the context of removing packet contents and rewriting IP and transport headers [16, 29], but also recently with packet contents preserved but rewritten to remove sensitive information [19]. Secure anonymization[11] is a difficult problem because there are a large number of attacks that can be used to recover identities (see for example [19] for discussion), some of which are quite difficult to defend against unless we can accept degraded notions of identity (for example, mapping a block of IP addresses all to a single anonymized address).

An alternative paradigm to publishing traces is for data gatherers to instead accept "data reduction requests" [17]. The contributor keeps the raw data privately, but researchers send their data analysis software to the contributor, who then runs it against the raw data on behalf of the researcher and sends back the reduced results. This approach gives the contributor tighter control over the released data, as they do not have to devise a single, irrevocable anonymization policy, and they can hand-inspect the privacy implications of each set of derived results (though in practice data gatherers often lack the time required for this last).

There is another benefit to the data-reduction-request approach: the practice of having to send our data analysis software for *others* to run forces the development of portable analysis software and well-specified analysis steps. This software in turn can be made available for use in other contexts, including reproducing analyses if more raw data becomes available, and potentially encouraging the sharing of analysis technology.

A significant potential disadvantage of this approach, however, is that the data gatherers must find the required effort sufficiently tenable to participate. Furthermore, maintaining access to the data over time requires a significant commitment by the gatherer, even if the data collection itself was a one-time effort. Such data sources will therefore often lack longevity. This limited lifetime can then be at odds with the frequent need to revisit raw data as we debug our initial analysis or discover unexpected properties in the data that we then want to delve into more deeply.

[11]Note that anonymization requirements can vary a great deal from one environment to another. For example, in some environments, individual addresses are highly sensitive, while others might have a need to not disclose *types* of activity that reflect poorly on their institutions.

6. SUMMARY

We have presented several general strategies for conducting sound Internet measurement studies:

- Maintain comprehensive meta-data.
- Calibrate measurements by investigating spikes and outliers, testing for self-consistency, and comparing different measurements when the opportunity presents itself.
- Structure the analysis process to make it amenable to reproducibility.
- For large datasets, work initially on small subsets and assess variability across different subsets.
- When making long-running measurements, institute periodic, automated analysis of new measurements as a means of detecting when the process breaks, and also to ensure that the process includes the recording of adequate meta-data.
- The need to gain access to traces by sending data reduction programs to data gatherers can be used as an opportunity to develop data analysis tools that lend themselves to reproducibility and sharing.

These goals in turn suggest a number of tools and community practices worth pursuing:

- Data management in terms of using databases and version control.
- Scriptable analysis environments that support ease of exploration and also reproducibility. For large projects, these may require mechanisms for managing ephemeral intermediary results for speeding up the edit-compile-debug cycle.
- Tools (visualizations, test suites) to investigate differences, both between datasets, and between different versions of the same analysis on the same dataset.
- The electronic equivalent of a scientist's laboratory notebook for capturing the full details of the measurement and analysis process.
- Encouraging the publication of portable measurement management tools and environments.
- Encouraging the publication of measurement data.

Indeed, it is our belief that there is an important, currently neglected, opportunity here for Internet measurement funders to broadly enhance the efficacy and quality of the research that gets performed by directly supporting the development of more community tools along these lines.

Many of our suggestions add more work—at least initially—to the already labor-intensive process of conducting a measurement study. It is fair to question whether it really is worth the extra effort. If we value the soundness of our measurement results, then in our own experience the answer is clearly Yes, as often these techniques have in fact, in concert, uncovered significant flaws in our work; or their lack has led to significant subsequent headaches when trying to untangle the lengthy path from original measurement to final conclusion.

More generally, we would add that: *(i)* care in the measurement and analysis processes often makes us more thoughtful about the *meaning* underlying the analysis, too, leading to deeper insights from the overall effort; *(ii)* with time, the extra effort in carefully

scrutinizing these processes builds deeper overall confidence in the practitioner (especially for students) and offers opportunities for serendipity; and *(iii)* errors of various forms often *add up*, so a discipline that keeps them in check to the extent possible will indeed help keep us closer to the truth.

7. ACKNOWLEDGMENTS

This work has benefited from discussions with many colleagues over the years. I would particularly like to thank Andy Adams, Mark Allman, Paul Barford, Mark Crovella, Christophe Diot, Nick Duffield, Anja Feldmann, Sally Floyd, Steve Gribble, Mark Handley, Eddie Kohler, Balachander Krishnamurthy, Al Lee, Morley Mao, Jeff Mogul, Sue Moon, Jitendra Padhye, Jennifer Rexford, Stefan Savage, Scott Shenker, Neil Spring, Darryl Veitch, David Wetherall, Walter Willinger, and anonymous reviewers of previous versions of this work.

In addition, this work was funded in part by the National Science Foundation under grants NRT-0335290 and ITR/ANI-0205519.

8. REFERENCES

[1] M. Allman. Private communication. 2001.

[2] M. Allman, E. Blanton, and W. Eddy. A Scalable System for Sharing Internet Measurements. Proc. PAM 2002.

[3] M. Allman, W. Eddy and S. Ostermann. Estimating Loss Rates With TCP. ACM Performance Evaluation Review, 31(3). 2003.

[4] P. Barford. Private communication. 2000.

[5] S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking* 9(4):392–403. 2001.

[6] C. Fraleigh et al. Packet-Level Traffic Measurements from the Sprint IP Backbone. IEEE Network 17(6). 2003.

[7] A. Gilat. *MATLAB: An Introduction with Applications*. Wiley Text Books. 2003.

[8] M. Handley, C. Kreibich and V. Paxson. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. Proc. USENIX Security Symposium. 2001.

[9] J. Heidemann, K. Mills, and S. Kumar. Expanding Confidence in Network Simulation. *IEEE Network*, 15(5):58–63. 2001.

[10] S. Jaiswal et al. Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone. Proc. ACM SIGCOMM IMW, 2002.

[11] E. Kohler. *ipsumdump*. http://www.icir.org/kohler/ipsumdump/. 2002.

[12] C. Kreibich. Design and Implementation of Netdude, a Framework for Packet Trace Manipulation. Proc. USENIX/FREENIX, 2004.

[13] P. Martin-Löf. The Notion of Redundancy and Its Use as a Quantitative Measure of the Discrepancy between a Statistical Hypothesis and a Set of Observational Data. *Scandinavian Journal of Statistics*, 1(1), pp. 3-18. 1974.

[14] D. Mills. Network Time Protocol (Version 3): Specification, Implementation and Analysis. RFC 1305, Network Information Center, SRI International, Menlo Park, CA. March 1992.

[15] D. Mills. Modelling and Analysis of Computer Network Clocks. Technical Report 92-5-2, Electrical Engineering Department, Univ. Delaware. May 1992.

[16] Greg Minshall. *TCPdpriv* manual. http://www.acm.org/sigcomm/ITA/. 1996.

[17] J. Mogul. Trace anonymization misses the point. WWW 2002 panel on Web Measurements. http://www2002.org/presentations/mogul-n.pdf

[18] S. Moon, P. Skelly and D. Towsley. Estimation and Removal of Clock Skew from Network Delay Measurements. Proc. IEEE INFOCOM. 1999.

[19] R. Pang and V. Paxson. A High-level Programming Environment for Packet Trace Anonymization and Transformation. Proc. ACM SIGCOMM. 2003.

[20] A. Pasztor and D. Veitch. PC Based Precision Timing Without GPS. Proc. ACM SIGMETRICS. 2002.

[21] V. Paxson. Automated Packet Trace Analysis of TCP Implementations. Proc. ACM SIGCOMM. 1997.

[22] V. Paxson. Measurements and Analysis of End-to-End Internet Dynamics. Ph.D. dissertation. Univ. California, Berkeley. 1997.

[23] V. Paxson. On Calibrating Measurements of Packet Transit Times. *Proc. ACM SIGMETRICS '98.*

[24] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks* 31(23–24), pp. 2435–2463, Dec. 1999.

[25] T. Ptacek and T. Newsham, Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technical report, Secure Networks. http://www.icir.org/vern/Ptacek-Newsham-Evasion-98.ps. Jan. 1998.

[26] R Development Core Team. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. 2003.

[27] W. Venables and B. Ripley. *Modern Applied Statistics with S-PLUS.* Springer. 1999.

[28] L. Wang et al. Observation and Analysis of BGP Behavior under Stress. Proc. ACM SIGCOMM IMW. 2002.

[29] J. Xu, J. Fan, M. Ammar, S. Moon. Prefix-Preserving IP Address Anonymization: Measurement-Based Security Evaluation and a New Cryptography-based Scheme. Proc. ICNP. 2002.

[30] L. Zhang, Z. Liu, and C. Xia. Clock Synchronization Algorithms for Network Measurements. Proc. IEEE INFOCOM. 2002