

Network Working Group
Internet Draft

V. Paxson, Lawrence Berkeley National Lab
G. Almes, Advanced Network & Services
J. Mahdavi, Pittsburgh Supercomputer Center
M. Mathis, Pittsburgh Supercomputer Center

Expiration Date: July 1998

February 1998

Framework for IP Performance Metrics
<[draft-ietf-ippm-framework-02.txt](#)>

1. Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months, and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet Drafts as reference material or to cite them other than as ``work in progress''.

To learn the current status of any Internet Draft, please check the ``1id-abstracts.txt' listing contained in the Internet Drafts shadow directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Table of Contents

1.	STATUS OF THIS MEMO.....	1
2.	INTRODUCTION.....	2
3.	CRITERIA FOR IP PERFORMANCE METRICS.....	4
4.	TERMINOLOGY FOR PATHS AND CLOUDS.....	4
5.	FUNDAMENTAL CONCEPTS.....	5
5.1	Metrics.....	5
5.2	Measurement Methodology.....	6
5.2	Measurements, Uncertainties, and Errors.....	8
6.	METRICS AND THE ANALYTICAL FRAMEWORK.....	9
7.	EMPIRICALLY SPECIFIED METRICS.....	11
8.	TWO FORMS OF COMPOSITION.....	12
8.1	Spatial Composition of Metrics.....	12
8.2	Temporal Composition of Formal Models and Empirical Metrics.....	13
9.	ISSUES RELATED TO TIME.....	14
9.1	Clock Issues.....	14
9.2	The Notion of "Wire Time".....	17
10.	SINGLETONS, SAMPLES, AND STATISTICS.....	19
10.1	Methods of Collecting Samples.....	20
10.1.1	Poisson Sampling.....	21
10.1.2	Geometric Sampling.....	22
10.1.3	Generating Poisson Sampling Intervals.....	22
10.2	Self-Consistency.....	23
10.3	Defining Statistical Distributions.....	25
10.4	Testing For Goodness-of-Fit.....	26
11.	AVOIDING STOCHASTIC METRICS.....	27
12.	PACKETS OF TYPE P.....	28
13.	INTERNET ADDRESSES VS. HOSTS.....	29
14.	STANDARD-FORMED PACKETS.....	29
15.	ACKNOWLEDGEMENTS.....	30
16.	SECURITY CONSIDERATIONS.....	31
17.	APPENDIX.....	31
18.	REFERENCES.....	37
19.	AUTHORS' ADDRESSES.....	37

[2.](#) Introduction

The purpose of this memo is to define a general framework for particular metrics to be developed by the IETF's IP Performance Metrics effort, begun by the Benchmarking Methodology Working Group (BMWG) of the Operational Requirements Area, and being continued by

the IP Performance Metrics Working Group (IPPM) of the Transport Area.

We begin by laying out several criteria for the metrics that we adopt. These criteria are designed to promote an IPPM effort that

will maximize an accurate common understanding by Internet users and Internet providers of the performance and reliability both of end-to-end paths through the Internet and of specific 'IP clouds' that comprise portions of those paths.

We next define some Internet vocabulary that will allow us to speak clearly about Internet components such as routers, paths, and clouds.

We then define the fundamental concepts of 'metric' and 'measurement methodology', which allow us to speak clearly about measurement issues. Given these concepts, we proceed to discuss the important issue of measurement uncertainties and errors, and develop a key, somewhat subtle notion of how they relate to the analytical framework shared by many aspects of the Internet engineering discipline. We then introduce the notion of empirically defined metrics, and finish this part of the document with a general discussion of how metrics can be 'composed'.

The remainder of the document deals with a variety of issues related to defining sound metrics and methodologies: how to deal with imperfect clocks; the notion of 'wire time' as distinct from 'host time'; how to aggregate sets of singleton metrics into samples and derive sound statistics from those samples; why it is recommended to avoid thinking about Internet properties in probabilistic terms (such as the probability that a packet is dropped), since these terms often include implicit assumptions about how the network behaves; the utility of defining metrics in terms of packets of a generic type; the benefits of preferring IP addresses to DNS host names; and the notion of 'standard-formed' packets. An appendix discusses the Anderson-Darling test for gauging whether a set of values matches a given statistical distribution, and gives C code for an implementation of the test.

In some sections of the memo, we will surround some commentary text with the brackets {Comment: ... }. We stress that this commentary is only commentary, and is not itself part of the framework document or

a proposal of particular metrics. In some cases this commentary will discuss some of the properties of metrics that might be envisioned, but the reader should assume that any such discussion is intended only to shed light on points made in the framework document, and not to suggest any specific metrics.

[3.](#) Criteria for IP Performance Metrics

The overarching goal of the IP Performance Metrics effort is to achieve a situation in which users and providers of Internet transport service have an accurate common understanding of the performance and reliability of the Internet component 'clouds' that they use/provide.

To achieve this, performance and reliability metrics for paths through the Internet must be developed. In several IETF meetings criteria for these metrics have been specified:

- + The metrics must be concrete and well-defined,
- + A methodology for a metric should have the property that it is repeatable: if the methodology is used multiple times under identical conditions, the same measurements should result in the same measurements.
- + The metrics must exhibit no bias for IP clouds implemented with identical technology,
- + The metrics must exhibit understood and fair bias for IP clouds implemented with non-identical technology,
- + The metrics must be useful to users and providers in understanding the performance they experience or provide,
- + The metrics must avoid inducing artificial performance goals.

[4.](#) Terminology for Paths and Clouds

The following list defines terms that need to be precise in the development of path metrics. We begin with low-level notions of a

path into relevant pieces.

host A computer capable of communicating using the Internet protocols; includes "routers".

link A single link-level connection between two (or more) hosts; includes leased lines, ethernet, frame relay clouds, etc.

router A host which facilitates network-level communication between hosts by forwarding IP packets.

path A sequence of the form $\langle h_0, l_1, h_1, \dots, l_n, h_n \rangle$, where $n \geq 0$, each h_i is a host, each l_i is a link between h_{i-1} and h_i , each $h_1 \dots h_{n-1}$ is a router. A pair $\langle l_i, h_i \rangle$ is termed a 'hop'. In an appropriate operational configuration, the links and routers in the path facilitate network-layer communication of packets from h_0 to h_n . Note that path is a unidirectional concept.

subpath

Given a path, a subpath is any subsequence of the given path which is itself a path. (Thus, the first and last element of a subpath is a host.)

cloud An undirected (possibly cyclic) graph whose vertices are routers and whose edges are links that connect pairs of routers. Formally, ethernet, frame relay clouds, and other links that connect more than two routers are modelled as fully-connected meshes of graph edges. Note that to connect to a cloud means to connect to a router of the cloud over a link; this link is not itself part of the cloud.

exchange

A special case of a link, an exchange directly connects either a host to a cloud and/or one cloud to another cloud.

cloud subpath

A subpath of a given path, all of whose hosts are routers of a given cloud.

path digest

A sequence of the form $\langle h_0, e_1, C_1, \dots, e_n, h_n \rangle$, where $n \geq 0$, h_0 and h_n are hosts, each $e_1 \dots e_n$ is an exchange, and each $C_1 \dots C_{n-1}$ is a cloud subpath.

5. Fundamental Concepts

5.1. Metrics

In the operational Internet, there are several quantities related to the performance and reliability of the Internet that we'd like to know the value of. When such a quantity is carefully specified, we term the quantity a metric. We anticipate that there will be separate RFCs for each metric (or for each closely related group of metrics).

In some cases, there might be no obvious means to effectively measure the metric; this is allowed, and even understood to be very useful in some cases. It is required, however, that the specification of the metric be as clear as possible about what quantity is being specified. Thus, difficulty in practical measurement is sometimes allowed, but ambiguity in meaning is not.

Each metric will be defined in terms of standard units of measurement. The international metric system will be used, with the

following points specifically noted:

- + When a unit is expressed in simple meters (for distance/length) or seconds (for duration), appropriate related units based on thousands or thousandths of acceptable units are acceptable. Thus, distances expressed in kilometers (km), durations expressed in milliseconds (ms), or microseconds (us) are allowed, but not centimeters (because the prefix is not in terms of thousands or thousandths).
- + When a unit is expressed in a combination of units, appropriate related units based on thousands or thousandths of acceptable units are acceptable, but all such thousands/thousandths must be grouped at the beginning. Thus, kilo-meters per second (km/s) is allowed, but meters per millisecond is not.
- + The unit of information is the bit.
- + When metric prefixes are used with bits or with combinations

including bits, those prefixes will have their metric meaning (related to decimal 1000), and not the meaning conventional with computer storage (related to decimal 1024). In any RFC that defines a metric whose units include bits, this convention will be followed and will be repeated to ensure clarity for the reader.

- + When a time is given, it will be expressed in UTC.

Note that these points apply to the specifications for metrics and not, for example, to packet formats where octets will likely be used in preference/addition to bits.

Finally, we note that some metrics may be defined purely in terms of other metrics; such metrics are call 'derived metrics'.

[5.2.](#) Measurement Methodology

For a given set of well-defined metrics, a number of distinct measurement methodologies may exist. A partial list includes:

- + Direct measurement of a performance metric using injected test traffic. Example: measurement of the round-trip delay of an IP packet of a given size over a given route at a given time.
- + Projection of a metric from lower-level measurements. Example: given accurate measurements of propagation delay and bandwidth for each step along a path, projection of the complete delay for the path for an IP packet of a given size.
- + Estimation of a consituent metric from a set of more aggregated measurements. Example: given accurate measurements of delay for a given one-hop path for IP packets of different sizes, estimation of propagation delay for the link of that one-hop path.

- + Estimation of a given metric at one time from a set of related metrics at other times. Example: given an accurate measurement of flow capacity at a past time, together with a set of accurate delay measurements for that past time and the current time, and given a model of flow dynamics, estimate the flow capacity that would be observed at the current time.

This list is by no means exhaustive. The purpose is to point out the variety of measurement techniques.

When a given metric is specified, a given measurement approach might be noted and discussed. That approach, however, is not formally part of the specification.

A methodology for a metric should have the property that it is repeatable: if the methodology is used multiple times under identical conditions, it should result in consistent measurements.

Backing off a little from the word 'identical' in the previous paragraph, we could more accurately use the word 'continuity' to describe a property of a given methodology: a methodology for a given metric exhibits continuity if, for small variations in conditions, it results in small variations in the resulting measurements. Slightly more precisely, for every positive epsilon, there exists a positive delta, such that if two sets of conditions are within delta of each other, then the resulting measurements will be within epsilon of each other. At this point, this should be taken as a heuristic driving our intuition about one kind of robustness property rather than as a precise notion.

A metric that has at least one methodology that exhibits continuity is said itself to exhibit continuity.

Note that some metrics, such as hop-count along a path, are integer-valued and therefore cannot exhibit continuity in quite the sense given above.

Note further that, in practice, it may not be practical to know (or be able to quantify) the conditions relevant to a measurement at a given time. For example, since the instantaneous load (in packets to be served) at a given router in a high-speed wide-area network can vary widely over relatively brief periods and will be very hard for an external observer to quantify, various statistics of a given metric may be more repeatable, or may better exhibit continuity. In that case those particular statistics should be specified when the metric is specified.

Finally, some measurement methodologies may be 'conservative' in the sense that the act of measurement does not modify, or only slightly

attempts to measure. {Comment: for example, in a wide-area high-speed network under modest load, a test using several small 'ping' packets to measure delay would likely not interfere (much) with the delay properties of that network as observed by others. The corresponding statement about tests using a large flow to measure flow capacity would likely fail.}

5.3. Measurements, Uncertainties, and Errors

Even the very best measurement methodologies for the very most well behaved metrics will exhibit errors. Those who develop such measurement methodologies, however, should strive to:

- + minimize their uncertainties/errors,
- + understand and document the sources of uncertainty/error, and
- + quantify the amounts of uncertainty/error.

For example, when developing a method for measuring delay, understand how any errors in your clocks introduce errors into your delay measurement, and quantify this effect as well as you can. In some cases, this will result in a requirement that a clock be at least up to a certain quality if it is to be used to make a certain measurement.

As a second example, consider the timing error due to measurement overheads within the computer making the measurement, as opposed to delays due to the Internet component being measured. The former is a measurement error, while the latter reflects the metric of interest. Note that one technique that can help avoid this overhead is the use of a packet filter/sniffer, running on a separate computer that records network packets and timestamps them accurately (see the discussion of 'wire time' below). The resulting trace can then be analysed to assess the test traffic, minimising the effect of measurement host delays, or at least allowing those delays to be accounted for. We note that this technique may prove beneficial even if the packet filter/sniffer runs on the same machine, because such measurements generally provide 'kernel-level' timestamping as opposed to less-accurate 'application-level' timestamping.

Finally, we note that derived metrics (defined above) or metrics that exhibit spatial or temporal composition (defined below) offer particular occasion for the analysis of measurement uncertainties, namely how the uncertainties propagate (conceptually) due to the derivation or composition.

6. Metrics and the Analytical Framework

As the Internet has evolved from the early packet-switching studies of the 1960s, the Internet engineering community has evolved a common analytical framework of concepts. This analytical framework, or A-frame, used by designers and implementers of protocols, by those involved in measurement, and by those who study computer network performance using the tools of simulation and analysis, has great advantage to our work. A major objective here is to generate network characterizations that are consistent in both analytical and practical settings, since this will maximize the chances that non-empirical network study can be better correlated with, and used to further our understanding of, real network behavior.

Whenever possible, therefore, we would like to develop and leverage off of the A-frame. Thus, whenever a metric to be specified is understood to be closely related to concepts within the A-frame, we will attempt to specify the metric in the A-frame's terms. In such a specification we will develop the A-frame by precisely defining the concepts needed for the metric, then leverage off of the A-frame by defining the metric in terms of those concepts.

Such a metric will be called an 'analytically specified metric' or, more simply, an analytical metric.

{Comment: Examples of such analytical metrics might include:

propagation time of a link

The time, in seconds, required by a single bit to travel from the output port on one Internet host across a single link to another Internet host.

bandwidth of a link for packets of size k

The capacity, in bits/second, where only those bits of the IP packet are counted, for packets of size k bytes.

routeThe path, as defined in [Section 4](#), from A to B at a given time.

hop count of a route

The value 'n' of the route path.

}

Note that we make no a priori list of just what A-frame concepts will emerge in these specifications, but we do encourage their use and urge that they be carefully specified so that, as our set of metrics develops, so will a specified set of A-frame concepts

technically consistent with each other and consonant with the common understanding of those concepts within the general Internet

community.

These A-frame concepts will be intended to abstract from actual Internet components in such a way that:

- + the essential function of the component is retained,
- + properties of the component relevant to the metrics we aim to create are retained,
- + a subset of these component properties are potentially defined as analytical metrics, and
- + those properties of actual Internet components not relevant to defining the metrics we aim to create are dropped.

For example, when considering a router in the context of packet forwarding, we might model the router as a component that receives packets on an input link, queues them on a FIFO packet queue of finite size, employs tail-drop when the packet queue is full, and forwards them on an output link. The transmission speed (in bits/second) of the input and output links, the latency in the router (in seconds), and the maximum size of the packet queue (in bits) are relevant analytical metrics.

In some cases, such analytical metrics used in relation to a router will be very closely related to specific metrics of the performance of Internet paths. For example, an obvious formula $(L + P/B)$ involving the latency in the router (L), the packet size (in bits) (P), and the transmission speed of the output link (B) might closely approximate the increase in packet delay due to the insertion of a given router along a path.

We stress, however, that well-chosen and well-specified A-frame concepts and their analytical metrics will support more general metric creation efforts in less obvious ways.

{Comment: for example, when considering the flow capacity of a path, it may be of real value to be able to model each of the routers along the path as packet forwarders as above. Techniques for estimating the flow capacity of a path might use the maximum packet queue size as a parameter in decidedly non-obvious ways. For example, as the maximum queue size increases, so will the ability of the router to

continuously move traffic along an output link despite fluctuations in traffic from an input link. Estimating this increase, however, remains a research topic.}

Note that, when we specify A-frame concepts and analytical metrics, we will inevitably make simplifying assumptions. The key role of these concepts is to abstract the properties of the Internet components relevant to given metrics. Judgement is required to avoid making assumptions that bias the modeling and metric effort toward

one kind of design.

{Comment: for example, routers might not use tail-drop, even though tail-drop might be easier to model analytically.}

Finally, note that different elements of the A-frame might well make different simplifying assumptions. For example, the abstraction of a router used to further the definition of path delay might treat the router's packet queue as a single FIFO queue, but the abstraction of a router used to further the definition of the handling of an RSVP-enabled packet might treat the router's packet queue as supporting bounded delay -- a contradictory assumption. This is not to say that we make contradictory assumptions at the same time, but that two different parts of our work might refine the simpler base concept in two divergent ways for different purposes.

{Comment: in more mathematical terms, we would say that the A-frame taken as a whole need not be consistent; but the set of particular A-frame elements used to define a particular metric must be.}

7. Empirically Specified Metrics

There are useful performance and reliability metrics that do not fit so neatly into the A-frame, usually because the A-frame lacks the detail or power for dealing with them. For example, "the best flow capacity achievable along a path using an [RFC-2001](#)-compliant TCP" would be good to be able to measure, but we have no analytical framework of sufficient richness to allow us to cast that flow capacity as an analytical metric.

These notions can still be well specified by instead describing a

reference methodology for measuring them.

Such a metric will be called an 'empirically specified metric', or more simply, an empirical metric.

Such empirical metrics should have three properties:

- + we should have a clear definition for each in terms of Internet components,
- + we should have at least one effective means to measure them, and
- + to the extent possible, we should have an (necessarily incomplete) understanding of the metric in terms of the A-frame so that we can use our measurements to reason about the performance and reliability of A-frame components and of aggregations of A-frame components.

[8.](#) Two Forms of Composition

[8.1.](#) Spatial Composition of Metrics

In some cases, it may be realistic and useful to define metrics in such a fashion that they exhibit spatial composition.

By spatial composition, we mean a characteristic of some path metrics, in which the metric as applied to a (complete) path can also be defined for various subpaths, and in which the appropriate A-frame concepts for the metric suggest useful relationships between the metric applied to these various subpaths (including the complete path, the various cloud subpaths of a given path digest, and even single routers along the path). The effectiveness of spatial composition depends:

- + on the usefulness in analysis of these relationships as applied to the relevant A-frame components, and
- + on the practical use of the corresponding relationships as applied to metrics and to measurement methodologies.

{Comment: for example, consider some metric for delay of a 100-byte packet across a path P , and consider further a path digest $\langle h_0, e_1, C_1, \dots, e_n, h_n \rangle$ of P . The definition of such a metric might include a conjecture that the delay across P is very nearly the sum of the

corresponding metric across the exchanges (e_i) and clouds (C_i) of the given path digest. The definition would further include a note on how a corresponding relation applies to relevant A-frame components, both for the path P and for the exchanges and clouds of the path digest.}

When the definition of a metric includes a conjecture that the metric across the path is related to the metric across the subpaths of the path, that conjecture constitutes a claim that the metric exhibits spatial composition. The definition should then include:

- + the specific conjecture applied to the metric,
- + a justification of the practical utility of the composition in terms of making accurate measurements of the metric on the path,
- + a justification of the usefulness of the composition in terms of making analysis of the path using A-frame concepts more effective, and
- + an analysis of how the conjecture could be incorrect.

[8.2.](#) Temporal Composition of Formal Models and Empirical Metrics

In some cases, it may be realistic and useful to define metrics in such a fashion that they exhibit temporal composition.

By temporal composition, we mean a characteristic of some path metric, in which the metric as applied to a path at a given time T is also defined for various times $t_0 < t_1 < \dots < t_n < T$, and in which the appropriate A-frame concepts for the metric suggests useful relationships between the metric applied at times t_0, \dots, t_n and the metric applied at time T . The effectiveness of temporal composition depends:

- + on the usefulness in analysis of these relationships as applied to the relevant A-frame components, and
- + on the practical use of the corresponding relationships as applied to metrics and to measurement methodologies.

{Comment: for example, consider a metric for the expected flow capa-

city across a path P during the five-minute period surrounding the time T , and suppose further that we have the corresponding values for each of the four previous five-minute periods t_0 , t_1 , t_2 , and t_3 . The definition of such a metric might include a conjecture that the flow capacity at time T can be estimated from a certain kind of extrapolation from the values of t_0 , ..., t_3 . The definition would further include a note on how a corresponding relation applies to relevant A-frame components.

Note: any (spatial or temporal) compositions involving flow capacity are likely to be subtle, and temporal compositions are generally more subtle than spatial compositions, so the reader should understand that the foregoing example is intentionally naive.}

When the definition of a metric includes a conjecture that the metric across the path at a given time T is related to the metric across the path for a set of other times, that conjecture constitutes a claim that the metric exhibits temporal composition. The definition should then include:

- + the specific conjecture applied to the metric,
- + a justification of the practical utility of the composition in terms of making accurate measurements of the metric on the path, and
- + a justification of the usefulness of the composition in terms of making analysis of the path using A-frame concepts more effective.

[9. Issues related to Time](#)

[9.1. Clock Issues](#)

Measurements of time lie at the heart of many Internet metrics. Because of this, it will often be crucial when designing a methodology for measuring a metric to understand the different types of errors and uncertainties introduced by imperfect clocks. In this section we define terminology for discussing the characteristics of clocks and touch upon related measurement issues which need to be

addressed by any sound methodology.

The Network Time Protocol (NTP; [RFC 1305](#)) defines a nomenclature for discussing clock characteristics, which we will also use when appropriate [[Mi92](#)]. The main goal of NTP is to provide accurate timekeeping over fairly long time scales, such as minutes to days, while for measurement purposes often what is more important is short-term accuracy, between the beginning of the measurement and the end, or over the course of gathering a body of measurements (a sample). This difference in goals sometimes leads to different definitions of terminology as well, as discussed below.

To begin, we define a clock's "offset" at a particular moment as the difference between the time reported by the clock and the "true" time as defined by UTC. If the clock reports a time T_c and the true time is T_t , then the clock's offset is $T_c - T_t$.

We will refer to a clock as "accurate" at a particular moment if the clock's offset is zero, and more generally a clock's "accuracy" is how close the absolute value of the offset is to zero. For NTP, accuracy also includes a notion of the frequency of the clock; for our purposes, we instead incorporate this notion into that of "skew", because we define accuracy in terms of a single moment in time rather than over an interval of time.

A clock's "skew" at a particular moment is the frequency difference (first derivative of its offset with respect to true time) between the clock and true time.

As noted in [RFC 1305](#), real clocks exhibit some variation in skew. That is, the second derivative of the clock's offset with respect to true time is generally non-zero. In keeping with [RFC 1305](#), we define this quantity as the clock's "drift".

A clock's "resolution" is the smallest unit by which the clock's time is updated. It gives a lower bound on the clock's uncertainty. (Note that clocks can have very fine resolutions and yet be wildly

inaccurate.) Resolution is defined in terms of seconds. However, resolution is relative to the clock's reported time and not to true time, so for example a resolution of 10 ms only means that the clock updates its notion of time in 0.01 second increments, not that this

is the true amount of time between updates.

{Comment: Systems differ on how an application interface to the clock reports the time on subsequent calls during which the clock has not advanced. Some systems simply return the same unchanged time as given for previous calls. Others may add a small increment to the reported time to maintain monotonic increasing timestamps. For systems that do the latter, we do **not** consider these small increments when defining the clock's resolution. They are instead an impediment to assessing the clock's resolution, since a natural method for doing so is to repeatedly query the clock to determine the smallest non-zero difference in reported times.}

It is expected that a clock's resolution changes only rarely (for example, due to a hardware upgrade).

There are a number of interesting metrics for which some natural measurement methodologies involve comparing times reported by two different clocks. An example is one-way packet delay [\[AK96\]](#). Here, the time required for a packet to travel through the network is measured by comparing the time reported by a clock at one end of the packet's path, corresponding to when the packet first entered the network, with the time reported by a clock at the other end of the path, corresponding to when the packet finished traversing the network.

We are thus also interested in terminology for describing how two clocks C1 and C2 compare. To do so, we introduce terms related to those above in which the notion of "true time" is replaced by the time as reported by clock C1. For example, clock C2's offset relative to C1 at a particular moment is $T_{c2} - T_{c1}$, the instantaneous difference in time reported by C2 and C1. To disambiguate between the use of the terms to compare two clocks versus the use of the terms to compare to true time, we will in the former case use the phrase "relative". So the offset defined earlier in this paragraph is the "relative offset" between C2 and C1.

When comparing clocks, the analog of "resolution" is not "relative resolution", but instead "joint resolution", which is the sum of the resolutions of C1 and C2. The joint resolution then indicates a conservative lower bound on the accuracy of any time intervals computed by subtracting timestamps generated by one clock from those generated by the other.

If two clocks are "accurate" with respect to one another (their relative offset is zero), we will refer to the pair of clocks as "synchronized". Note that clocks can be highly synchronized yet arbitrarily inaccurate in terms of how well they tell true time. This point is important because for many Internet measurements, synchronization between two clocks is more important than the accuracy of the clocks. The is somewhat true of skew, too: as long as the absolute skew is not too great, then minimal relative skew is more important, as it can induce systematic trends in packet transit times measured by comparing timestamps produced by the two clocks.

These distinctions arise because for Internet measurement what is often most important are differences in time as computed by comparing the output of two clocks. The process of computing the difference removes any error due to clock inaccuracies with respect to true time; but it is crucial that the differences themselves accurately reflect differences in true time.

Measurement methodologies will often begin with the step of assuring that two clocks are synchronized and have minimal skew and drift. {Comment: An effective way to assure these conditions (and also clock accuracy) is by using clocks that derive their notion of time from an external source, rather than only the host computer's clock. (These latter are often subject to large errors.) It is further preferable that the clocks directly derive their time, for example by having immediate access to a GPS (Global Positioning System) unit.}

Two important concerns arise if the clocks indirectly derive their time using a network time synchronization protocol such as NTP:

- + First, NTP's accuracy depends in part on the properties (particularly delay) of the Internet paths used by the NTP peers, and these might be exactly the properties that we wish to measure, so it would be unsound to use NTP to calibrate such measurements.
- + Second, NTP focuses on clock accuracy, which can come at the expense of short-term clock skew and drift. For example, when a host's clock is indirectly synchronized to a time source, if the synchronization intervals occur infrequently, then the host will sometimes be faced with the problem of how to adjust its current, incorrect time, T_i , with a considerably different, more accurate time it has just learned, T_a . Two general ways in which this is done are to either immediately set the current time to T_a , or to adjust the local clock's update frequency (hence, its skew) so that at some point in the future the local time T_i' will agree with the more accurate time T_a' . The first mechanism introduces discontinuities and can also violate common assumptions that timestamps are monotone increasing. If the host's clock is set backward in time, sometimes this can be easily detected. If the

clock is set forward in time, this can be harder to detect. The

skew induced by the second mechanism can lead to considerable inaccuracies when computing differences in time, as discussed above.

To illustrate why skew is a crucial concern, consider samples of one-way delays between two Internet hosts made at one minute intervals. The true transmission delay between the hosts might plausibly be on the order of 50 ms for a transcontinental path. If the skew between the two clocks is 0.01%, that is, 1 part in 10,000, then after 10 minutes of observation the error introduced into the measurement is 60 ms. Unless corrected, this error is enough to completely wipe out any accuracy in the transmission delay measurement. Finally, we note that assessing skew errors between unsynchronized network clocks is an open research area. (See [\[Pa97\]](#) for a discussion of detecting and compensating for these sorts of errors.) This shortcoming makes use of a solid, independent clock source such as GPS especially desirable.

[9.2](#). The Notion of "Wire Time"

Internet measurement is often complicated by the use of Internet hosts themselves to perform the measurement. These hosts can introduce delays, bottlenecks, and the like that are due to hardware or operating system effects and have nothing to do with the network behavior we would like to measure. This problem is particularly acute when timestamping of network events occurs at the application level.

In order to provide a general way of talking about these effects, we introduce two notions of "wire time". These notions are only defined in terms of an Internet host *H* observing an Internet link *L* at a particular location:

- + For a given packet *P*, the 'wire arrival time' of *P* at *H* on *L* is the first time *T* at which any bit of *P* has appeared at *H*'s observational position on *L*.
- + For a given packet *P*, the 'wire exit time' of *P* at *H* on *L* is the first time *T* at which all the bits of *P* have appeared at *H*'s observational position on *L*.

Note that intrinsic to the definition is the notion of where on the

link we are observing. This distinction is important because for large-latency links, we may obtain very different times depending on exactly where we are observing the link. We could allow the observational position to be an arbitrary location along the link; however, we define it to be in terms of an Internet host because we anticipate in practice that, for IPPM metrics, all such timing will be constrained to be performed by Internet hosts, rather than specialized hardware devices that might be able to monitor a link at locations

where a host cannot. This definition also takes care of the problem of links that are comprised of multiple physical channels. Because these multiple channels are not visible at the IP layer, they cannot be individually observed in terms of the above definitions.

It is possible, though one hopes uncommon, that a packet *P* might make multiple trips over a particular link *L*, due to a forwarding loop. These trips might even overlap, depending on the link technology. Whenever this occurs, we define a separate wire time associated with each instance of *P* seen at *H*'s position on the link. This definition is worth making because it serves as a reminder that notions like **the** unique time a packet passes a point in the Internet are inherently slippery.

The term wire time has historically been used to loosely denote the time at which a packet appeared on a link, without exactly specifying whether this refers to the first bit, the last bit, or some other consideration. This informal definition is generally already very useful, as it is usually used to make a distinction between when the packet's propagation delays begin and cease to be due to the network rather than the endpoint hosts.

When appropriate, metrics should be defined in terms of wire times rather than host endpoint times, so that the metric's definition highlights the issue of separating delays due to the host from those due to the network.

We note that one potential difficulty when dealing with wire times concerns IP fragments. It may be the case that, due to fragmentation, only a portion of a particular packet passes by *H*'s location. Such fragments are themselves legitimate packets and have well-defined wire times associated with them; but the larger IP packet corresponding to their aggregate may not.

We also note that these notions have not, to our knowledge, been previously defined in exact terms for Internet traffic. Consequently, we may find with experience that these definitions require some adjustment in the future.

{Comment: It can sometimes be difficult to measure wire times. One technique is to use a packet filter to monitor traffic on a link. The architecture of these filters often attempts to associate with each packet a timestamp as close to the wire time as possible. We note however that one common source of error is to run the packet filter on one of the endpoint hosts. In this case, it has been observed that some packet filters receive for some packets timestamps corresponding to when the packet was **scheduled** to be injected into the network, rather than when it actually was **sent** out onto the

network (wire time). There can be a substantial difference between these two times. A technique for dealing with this problem is to run the packet filter on a separate host that passively monitors the given link. This can be problematic however for some link technologies. See [[Pa97](#)] for a discussion of the sorts of errors packet filters can exhibit. Finally, we note that packet filters will often only capture the first fragment of a fragmented IP packet, due to the use of filtering on fields in the IP and transport protocol headers. As we generally desire our measurement methodologies to avoid the complexity of creating fragmented traffic, one strategy for dealing with their presence as detected by a packet filter is to flag that the measured traffic has an unusual form and abandon further analysis of the packet timing.}

[10](#). Singletons, Samples, and Statistics

With experience we have found it useful to introduce a separation between three distinct -- yet related -- notions:

- + By a 'singleton' metric, we refer to metrics that are, in a sense, atomic. For example, a single instance of "bulk throughput capacity" from one host to another might be defined as a singleton metric, even though the instance involves measuring the timing of a number of Internet packets.
- + By a 'sample' metric, we refer to metrics derived from a given singleton metric by taking a number of distinct instances

together. For example, we might define a sample metric of one-way delays from one host to another as an hour's worth of measurements, each made at Poisson intervals with a mean spacing of one second.

- + By a 'statistical' metric, we refer to metrics derived from a given sample metric by computing some statistic of the values defined by the singleton metric on the sample. For example, the mean of all the one-way delay values on the sample given above might be defined as a statistical metric.

By applying these notions of singleton, sample, and statistic in a consistent way, we will be able to reuse lessons learned about how to define samples and statistics on various metrics. The orthogonality among these three notions will thus make all our work more effective and more intelligible by the community.

In the remainder of this section, we will cover some topics in sampling and statistics that we believe will be important to a variety of metric definitions and measurement efforts.

[10.1.](#) Methods of Collecting Samples

The main reason for collecting samples is to see what sort of variations and consistencies are present in the metric being measured. These variations might be with respect to different points in the Internet, or different measurement times. When assessing variations based on a sample, one generally makes an assumption that the sample is "unbiased", meaning that the process of collecting the measurements in the sample did not skew the sample so that it no longer accurately reflects the metric's variations and consistencies.

One common way of collecting samples is to make measurements separated by fixed amounts of time: periodic sampling. Periodic sampling is particularly attractive because of its simplicity, but it suffers from two potential problems:

- + If the metric being measured itself exhibits periodic behavior, then there is a possibility that the sampling will observe only part of the periodic behavior if the periods happen to agree (either directly, or if one is a multiple of the other). Related

to this problem is the notion that periodic sampling can be easily anticipated. Predictable sampling is susceptible to manipulation if there are mechanisms by which a network component's behavior can be temporarily changed such that the sampling only sees the modified behavior.

- + The act of measurement can perturb what is being measured (for example, injecting measurement traffic into a network alters the congestion level of the network), and repeated periodic perturbations can drive a network into a state of synchronization (cf. [FJ94]), greatly magnifying what might individually be minor effects.

A more sound approach is based on "random additive sampling": samples are separated by independent, randomly generated intervals that have a common statistical distribution $G(t)$ [BM92]. The quality of this sampling depends on the distribution $G(t)$. For example, if $G(t)$ generates a constant value g with probability one, then the sampling reduces to periodic sampling with a period of g .

Random additive sampling gains significant advantages. In general, it avoids synchronization effects and yields an unbiased estimate of the property being sampled. The only significant drawbacks with it are:

- + it complicates frequency-domain analysis, because the samples do not occur at fixed intervals such as assumed by Fourier-transform techniques; and

- + unless $G(t)$ is the exponential distribution (see below), sampling still remains somewhat predictable, as discussed for periodic sampling above.

10.1.1.1. Poisson Sampling

It can be proved that if $G(t)$ is an exponential distribution with rate λ , that is

$$G(t) = 1 - \exp(-\lambda * t)$$

then the arrival of new samples *cannot* be predicted (and, again, the sampling is unbiased). Furthermore, the sampling is asymptoti-

cally unbiased even if the act of sampling affects the network's state. Such sampling is referred to as "Poisson sampling". It is not prone to inducing synchronization, it can be used to accurately collect measurements of periodic behavior, and it is not prone to manipulation by anticipating when new samples will occur.

Because of these valuable properties, we in general prefer that samples of Internet measurements are gathered using Poisson sampling. {Comment: We note, however, that there may be circumstances that favor use of a different $G(t)$. For example, the exponential distribution is unbounded, so its use will on occasion generate lengthy spaces between sampling times. We might instead desire to bound the longest such interval to a maximum value dT , to speed the convergence of the estimation derived from the sampling. This could be done by using

$$G(t) = \text{Unif}(0, dT)$$

that is, the uniform distribution between 0 and dT . This sampling, of course, becomes highly predictable if an interval of nearly length dT has elapsed without a sample occurring.}

In its purest form, Poisson sampling is done by generating independent, exponentially distributed intervals and gathering a single measurement after each interval has elapsed. It can be shown that if starting at time T one performs Poisson sampling over an interval dT , during which a total of N measurements happen to be made, then those measurements will be uniformly distributed over the interval $[T, T+dT]$. So another way of conducting Poisson sampling is to pick dT and N and generate N random sampling times uniformly over the interval $[T, T+dT]$. The two approaches are equivalent, except if N and dT are externally known. In that case, the property of not being able to predict measurement times is weakened (the other properties still hold). The N/dT approach has an advantage that dealing with fixed values of N and dT can be simpler than dealing with a fixed λ but variable numbers of measurements over variably-sized intervals.

[10.1.2](#). Geometric Sampling

Closely related to Poisson sampling is "geometric sampling", in which external events are measured with a fixed probability p . For example, one might capture all the packets over a link but only record

the packet to a trace file if a randomly generated number uniformly distributed between 0 and 1 is less than a given p . Geometric sampling has the same properties of being unbiased and not predictable in advance as Poisson sampling, so if it fits a particular Internet measurement task, it too is sound. See [CPB93] for more discussion.

10.1.3. Generating Poisson Sampling Intervals

To generate Poisson sampling intervals, one first determines the rate λ at which the singleton measurements will on average be made (e.g., for an average sampling interval of 30 seconds, we have $\lambda = 1/30$, if the units of time are seconds). One then generates a series of exponentially-distributed (pseudo-)random numbers E_1, E_2, \dots, E_n . The first measurement is made at time E_1 , the next at time E_1+E_2 , and so on.

One technique for generating exponentially-distributed (pseudo-)random numbers is based on the ability to generate U_1, U_2, \dots, U_n , (pseudo-)random numbers that are uniformly distributed between 0 and 1. Many computers provide libraries that can do this. Given such U_i , to generate E_i one uses:

$$E_i = -\log(U_i) / \lambda$$

where $\log(U_i)$ is the natural logarithm of U_i . {Comment: This technique is an instance of the more general "inverse transform" method for generating random numbers with a given distribution.}

Implementation details:

There are at least three different methods for approximating Poisson sampling, which we describe here as Methods 1 through 3. Method 1 is the easiest to implement and has the most error, and method 3 is the most difficult to implement and has the least error (potentially none).

Method 1 is to proceed as follows:

1. Generate E_1 and wait that long.
2. Perform a measurement.
3. Generate E_2 and wait that long.
4. Perform a measurement.
5. Generate E_3 and wait that long.
6. Perform a measurement ...

The problem with this approach is that the "Perform a measurement" steps themselves take time, so the sampling is not done at times E_1 , E_1+E_2 , etc., but rather at E_1 , $E_1+M_1+E_2$, etc., where M_i is the amount of time required for the i 'th measurement. If M_i is very small compared to $1/\lambda$ then the potential error introduced by this technique is likewise small. As M_i becomes a non-negligible fraction of $1/\lambda$, the potential error increases.

Method 2 attempts to correct this error by taking into account the amount of time required by the measurements (i.e., the M_i 's) and adjusting the waiting intervals accordingly:

1. Generate E_1 and wait that long.
2. Perform a measurement and measure M_1 , the time it took to do so.
3. Generate E_2 and wait for a time E_2-M_1 .
4. Perform a measurement and measure M_2 ..

This approach works fine as long as $E_{i+1} \geq M_i$. But if $E_{i+1} < M_i$ then it is impossible to wait the proper amount of time. (Note that this case corresponds to needing to perform two measurements simultaneously.)

Method 3 is generating a schedule of measurement times E_1 , E_1+E_2 , etc., and then sticking to it:

1. Generate E_1 , E_2 , ..., E_n .
2. Compute measurement times T_1 , T_2 , ..., T_n , as $T_i = E_1 + \dots + E_i$.
3. Arrange that at times T_1 , T_2 , ..., T_n , a measurement is made.

By allowing simultaneous measurements, Method 3 avoids the shortcomings of Methods 1 and 2. If, however, simultaneous measurements interfere with one another, then Method 3 does not gain any benefit and may actually prove worse than Methods 1 or 2.

For Internet phenomena, it is not known to what degree the inaccuracies of these methods are significant. If the M_i 's are much less than $1/\lambda$, then any of the three should suffice. If the M_i 's are less than $1/\lambda$ but perhaps not greatly less, then Method 2 is preferred to Method 1. If simultaneous measurements do not interfere with one another, then Method 3 is preferred, though it can be considerably harder to implement.

10.2. Self-Consistency

A fundamental requirement for a sound measurement methodology is that measurement be made using as few unconfirmed assumptions as possible. Experience has painfully shown how easy it is to make an (often implicit) assumption that turns out to be incorrect. An example is incorporating into a measurement the reading of a clock synchronized

to a highly accurate source. It is easy to assume that the clock is therefore accurate; but due to software bugs, a loss of power in the source, or a loss of communication between the source and the clock, the clock could actually be quite inaccurate.

This is not to argue that one must not make *any* assumptions when measuring, but rather that, to the extent which is practical, assumptions should be tested. One powerful way for doing so involves checking for self-consistency. Such checking applies both to the observed value(s) of the measurement *and* the values used by the measurement process itself*. A simple example of the former is that when computing a round trip time, one should check to see if it is negative. Since negative time intervals are non-physical, if it ever is negative that finding immediately flags an error. **These sorts of errors should then be investigated!** It is crucial to determine where the error lies, because only by doing so diligently can we build up faith in a methodology's fundamental soundness. For example, it could be that the round trip time is negative because during the measurement the clock was set backward in the process of synchronizing it with another source. But it could also be that the measurement program accesses uninitialized memory in one of its computations and, only very rarely, that leads to a bogus computation. This second error is more serious, if the same program is used by others to perform the same measurement, since then they too will suffer from incorrect results. Furthermore, once uncovered it can be completely fixed.

A more subtle example of testing for self-consistency comes from gathering samples of one-way Internet delays. If one has a large sample of such delays, it may well be highly telling to, for example, fit a line to the pairs of (time of measurement, measured delay), to see if the resulting line has a clearly non-zero slope. If so, a possible interpretation is that one of the clocks used in the measurements is skewed relative to the other. Another interpretation is that the slope is actually due to genuine network effects. Determining which is indeed the case will often be highly illuminating. (See [\[Pa97\]](#) for a discussion of distinguishing between relative clock skew and genuine network effects.) Furthermore, if making this check is part of the methodology, then a finding that the long-term slope is very near zero is positive evidence that the measurements are probably not biased by a difference in skew.

A final example illustrates checking the measurement process itself for self-consistency. Above we outline Poisson sampling techniques, based on generating exponentially-distributed intervals. A sound measurement methodology would include testing the generated intervals to see whether they are indeed exponentially distributed (and also to see if they suffer from correlation). In the appendix we discuss and

give C code for one such technique, a general-purpose, well-regarded goodness-of-fit test called the Anderson-Darling test.

Finally, we note that what is truly relevant for Poisson sampling of Internet metrics is often not when the measurements began but the wire times corresponding to the measurement process. These could well be different, due to complications on the hosts used to perform the measurement. Thus, even those with complete faith in their pseudo-random number generators and subsequent algorithms are encouraged to consider how they might test the assumptions of each measurement procedure as much as possible.

[10.3](#). Defining Statistical Distributions

One way of describing a collection of measurements (a sample) is as a statistical distribution -- informally, as percentiles. There are several slightly different ways of doing so. In this section we define a standard definition to give uniformity to these descriptions.

The "empirical distribution function" (EDF) of a set of scalar measurements is a function $F(x)$ which for any x gives the fractional proportion of the total measurements that were $\leq x$. If x is less than the minimum value observed, then $F(x)$ is 0. If it is greater or equal to the maximum value observed, then $F(x)$ is 1.

For example, given the 6 measurements:

-2, 7, 7, 4, 18, -5

Then $F(-8) = 0$, $F(-5) = 1/6$, $F(-5.0001) = 0$, $F(-4.999) = 1/6$, $F(7) = 5/6$, $F(18) = 1$, $F(239) = 1$.

Note that we can recover the different measured values and how many times each occurred from $F(x)$ -- no information regarding the range in values is lost. Summarizing measurements using histograms, on the

other hand, in general loses information about the different values observed, so the EDF is preferred.

Using either the EDF or a histogram, however, we do lose information regarding the order in which the values were observed. Whether this loss is potentially significant will depend on the metric being measured.

We will use the term "percentile" to refer to the smallest value of x for which $F(x) \geq$ a given percentage. So the 50th percentile of the example above is 4, since $F(4) = 3/6 = 50\%$; the 25th percentile is -2, since $F(-5) = 1/6 < 25\%$, and $F(-2) = 2/6 \geq 25\%$; the 100th percentile is 18; and the 0th percentile is $-\infty$, as is the 15th

percentile.

Care must be taken when using percentiles to summarize a sample, because they can lend an unwarranted appearance of more precision than is really available. Any such summary must include the sample size N , because any percentile difference finer than $1/N$ is below the resolution of the sample.

See [\[DS86\]](#) for more details regarding EDF's.

We close with a note on the common (and important!) notion of median. In statistics, the median of a distribution is defined to be the point X for which the probability of observing a value $\leq X$ is equal to the probability of observing a value $> X$. When estimating the median of a set of observations, the estimate depends on whether the number of observations, N , is odd or even:

- + If N is odd, then the 50th percentile as defined above is used as the estimated median.
- + If N is even, then the estimated median is the average of the central two observations; that is, if the observations are sorted in ascending order and numbered from 1 to N , where $N = 2 \times K$, then the estimated median is the average of the (K) 'th and $(K+1)$ 'th observations.

Usually the term "estimated" is dropped from the phrase "estimated median" and this value is simply referred to as the "median".

[10.4](#). Testing For Goodness-of-Fit

For some forms of measurement calibration we need to test whether a set of numbers is consistent with those numbers having been drawn from a particular distribution. An example is that to apply a self-consistency check to measurements made using a Poisson process, one test is to see whether the spacing between the sampling times does indeed reflect an exponential distribution; or if the dT/N approach discussed above was used, whether the times are uniformly distributed across $[T, dT]$.

{Comment: There are at least three possible sets of values we could test: the scheduled packet transmission times, as determined by use of a pseudo-random number generator; user-level timestamps made just before or after the system call for transmitting the packet; and wire times for the packets as recorded using a packet filter. All three of these are potentially informative: failures for the scheduled times to match an exponential distribution indicate inaccuracies in the random number generation; failures for the user-level times indicate inaccuracies in the timers used to schedule transmission; and failures for the wire times indicate inaccuracies in actually

transmitting the packets, perhaps due to contention for a shared resource.}

There are a large number of statistical goodness-of-fit techniques for performing such tests. See [\[DS86\]](#) for a thorough discussion. That reference recommends the Anderson-Darling EDF test as being a good all-purpose test, as well as one that is especially good at detecting deviations from a given distribution in the lower and upper tails of the EDF.

It is important to understand that the nature of goodness-of-fit tests is that one first selects a "significance level", which is the probability that the test will erroneously declare that the EDF of a given set of measurements fails to match a particular distribution when in fact the measurements do indeed reflect that distribution. Unless otherwise stated, IPPM goodness-of-fit tests are done using 5% significance. This means that if the test is applied to 100 samples and 5 of those samples are deemed to have failed the test, then the samples are all consistent with the distribution being tested. If significantly more of the samples fail the test, then the assumption that the samples are consistent with the distribution being tested

must be rejected. If significantly fewer of the samples fail the test, then the samples have potentially been doctored too well to fit the distribution. Similarly, some goodness-of-fit tests (including Anderson-Darling) can detect whether it is likely that a given sample was doctored. We also use a significance of 5% for this case; that is, the test will report that a given honest sample is "too good to be true" 5% of the time, so if the test reports this finding significantly more often than one time out of twenty, it is an indication that something unusual is occurring.

The appendix gives sample C code for implementing the Anderson-Darling test, as well as further discussing its use.

See [Pa94] for a discussion of goodness-of-fit and closeness-of-fit tests in the context of network measurement.

11. Avoiding Stochastic Metrics

When defining metrics applying to a path, subpath, cloud, or other network element, we in general do not define them in stochastic terms (probabilities). We instead prefer a deterministic definition. So, for example, rather than defining a metric about a "packet loss probability between A and B", we would define a metric about a "packet loss rate between A and B". (A measurement given by the first definition might be "0.73", and by the second "73 packets out of 100".)

We emphasize that the above distinction concerns the **definitions** of **metrics**. It is not intended to apply to what sort of techniques we might use to analyze the results of measurements.

The reason for this distinction is as follows. When definitions are made in terms of probabilities, there are often hidden assumptions in the definition about a stochastic model of the behavior being measured. The fundamental goal with avoiding probabilities in our metric definitions is to avoid biasing our definitions by these hidden assumptions.

For example, an easy hidden assumption to make is that packet loss in a network component due to queueing overflows can be described as something that happens to any given packet with a particular proba-

bility. In today's Internet, however, queueing drops are actually usually **deterministic**, and assuming that they should be described probabilistically can obscure crucial correlations between queueing drops among a set of packets. So it's better to explicitly note stochastic assumptions, rather than have them sneak into our definitions implicitly.

This does **not** mean that we abandon stochastic models for **understanding** network performance! It only means that when defining IP metrics we avoid terms such as "probability" for terms like "proportion" or "rate". We will still use, for example, random sampling in order to estimate probabilities used by stochastic models related to the IP metrics. We also do not rule out the possibility of stochastic metrics when they are truly appropriate (for example, perhaps to model transmission errors caused by certain types of line noise).

[12.](#) Packets of Type P

A fundamental property of many Internet metrics is that the value of the metric depends on the type of IP packet(s) used to make the measurement. Consider an IP-connectivity metric: one obtains different results depending on whether one is interested in connectivity for packets destined for well-known TCP ports or unreserved UDP ports, or those with invalid IP checksums, or those with TTL's of 16, for example. In some circumstances these distinctions will be highly interesting (for example, in the presence of firewalls, or RSVP reservations).

Because of this distinction, we introduce the generic notion of a "packet of type P", where in some contexts P will be explicitly defined (i.e., exactly what type of packet we mean), partially defined (e.g., "with a payload of B octets"), or left generic. Thus we may talk about generic IP-type-P-connectivity or more specific

IP-port-HTTP-connectivity. Some metrics and methodologies may be fruitfully defined using generic type P definitions which are then made specific when performing actual measurements.

Whenever a metric's value depends on the type of the packets involved in the metric, the metric's name will include either a specific type or a phrase such as "type-P". Thus we will not define an "IP-

connectivity" metric but instead an "IP-type-P-connectivity" metric and/or perhaps an "IP-port-HTTP-connectivity" metric. This naming convention serves as an important reminder that one must be conscious of the exact type of traffic being measured.

A closely related note: it would be very useful to know if a given Internet component treats equally a class C of different types of packets. If so, then any one of those types of packets can be used for subsequent measurement of the component. This suggests we devise a metric or suite of metrics that attempt to determine C.

[13.](#) Internet Addresses vs. Hosts

When considering a metric for some path through the Internet, it is often natural to think about it as being for the path from Internet host H1 to host H2. A definition in these terms, though, can be ambiguous, because Internet hosts can be attached to more than one network. In this case, the result of the metric will depend on which of these networks is actually used.

Because of this ambiguity, usually such definitions should instead be defined in terms of Internet IP addresses. For the common case of a unidirectional path through the Internet, we will use the term "Src" to denote the IP address of the beginning of the path, and "Dst" to denote the IP address of the end.

[14.](#) Standard-Formed Packets

Unless otherwise stated, all metric definitions that concern IP packets include an implicit assumption that the packet is **standard formed**. A packet is standard formed if it meets all of the following criteria:

- + Its length as given in the IP header corresponds to the size of the IP header plus the size of the payload.

- + It includes a valid IP header: the version field is 4 (later, we will expand this to include 6); the header length is ≥ 5 ; the checksum is correct.
- + It is not an IP fragment.
- + The source and destination addresses correspond to the hosts in question.
- + Either the packet possesses sufficient TTL to travel from the source to the destination if the TTL is decremented by one at each hop, or it possesses the maximum TTL of 255.
- + It does not contain IP options unless explicitly noted.
- + If a transport header is present, it too contains a valid checksum and other valid fields.

We further require that if a packet is described as having a "length of B octets", then $0 \leq B \leq 65535$; and if B is the payload length in octets, then $B \leq (65535 - \text{IP header size in octets})$.

So, for example, one might imagine defining an IP connectivity metric as "IP-type-P-connectivity for standard-formed packets with the IP TOS field set to 0", or, more succinctly, "IP-type-P-connectivity with the IP TOS field set to 0", since standard-formed is already implied by convention.

A particular type of standard-formed packet often useful to consider is the "minimal IP packet from A to B" - this is an IP packet with the following properties:

- + It is standard-formed.
- + Its data payload is 0 octets.
- + It contains no options.

(Note that we do not define its protocol field, as different values may lead to different treatment by the network.)

When defining IP metrics we keep in mind that no packet smaller or simpler than this can be transmitted over a correctly operating IP network.

[15.](#) Acknowledgements

The comments of Brian Carpenter, Bill Cervený, Padma Krishnaswamy, Jeff Sedayao and Howard Stanislevic are appreciated.

[16.](#) Security Considerations

This document concerns definitions and concepts related to Internet measurement. We discuss measurement procedures only in high-level terms, regarding principles that lend themselves to sound measurement. As such, the topics discussed do not affect the security of the Internet or of applications which run on it.

That said, it should be recognized that conducting Internet measurements can raise both security and privacy concerns. Active techniques, in which traffic is injected into the network, can be abused for denial-of-service attacks disguised as legitimate measurement activity. Passive techniques, in which existing traffic is recorded and analyzed, can expose the contents of Internet traffic to unintended recipients. Consequently, the definition of each metric and methodology must include a corresponding discussion of security considerations.

[17.](#) Appendix

Below we give routines written in C for computing the Anderson-Darling test statistic (A2) for determining whether a set of values is consistent with a given statistical distribution. Externally, the two main routines of interest are:

```
double exp_A2_known_mean(double x[], int n, double mean)
double unif_A2_known_range(double x[], int n,
                           double min_val, double max_val)
```

Both take as their first argument, *x*, the array of *n* values to be tested. (Upon return, the elements of *x* are sorted.) The remaining parameters characterize the distribution to be used: either the mean ($1/\lambda$), for an exponential distribution, or the lower and upper bounds, for a uniform distribution. The names of the routines stress that these values must be known in advance, and *not* estimated from the data (for example, by computing its sample mean). Estimating the parameters from the data *changes* the significance level of the test statistic. While [\[DS86\]](#) gives alternate significance tables for some instances in which the parameters are estimated from the data, for our purposes we expect that we should indeed know the parameters in advance, since what we will be testing are generally values such as packet sending times that we wish to verify follow a known distribution.

Both routines return a significance level, as described earlier. This is a value between 0 and 1. The correct use of the routines is to pick in advance the threshold for the significance level to test; generally, this will be 0.05, corresponding to 5%, as also described above. Subsequently, if the routines return a value strictly less

than this threshold, then the data are deemed to be inconsistent with the presumed distribution, *subject to an error corresponding to the significance level*. That is, for a significance level of 5%, 5% of the time data that is indeed drawn from the presumed distribution will be erroneously deemed inconsistent.

Thus, it is important to bear in mind that if these routines are used frequently, then one will indeed encounter occasional failures, even if the data is unblemished.

Another important point concerning significance levels is that it is unsound to compare them in order to determine which of two sets of values is a "better" fit to a presumed distribution. Such testing should instead be done using "closeness-of-fit metrics" such as the λ^2 metric described in [[Pa94](#)].

While the routines provided are for exponential and uniform distributions with known parameters, it is generally straight-forward to write comparable routines for any distribution with known parameters. The heart of the A2 tests lies in a statistic computed for testing whether a set of values is consistent with a uniform distribution between 0 and 1, which we term $\text{Unif}(0, 1)$. If we wish to test whether a set of values, X , is consistent with a given distribution $G(x)$, we first compute

$$Y = G_{\text{inverse}}(X)$$

If X is indeed distributed according to $G(x)$, then Y will be distributed according to $\text{Unif}(0, 1)$; so by testing Y for consistency with $\text{Unif}(0, 1)$, we also test X for consistency with $G(x)$.

We note, however, that the process of computing Y above might yield values of Y outside the range $(0..1)$. Such values should not occur if X is indeed distributed according to $G(x)$, but easily can occur if it is not. In the latter case, we need to avoid computing the central A2 statistic, since floating-point exceptions may occur if any of the values lie outside $(0..1)$. Accordingly, the routines check for this possibility, and if encountered, return a raw A2 statistic of

-1. The routine that converts the raw A2 statistic to a significance level likewise propagates this value, returning a significance level of -1. So, any use of these routines must be prepared for a possible negative significance level.

The last important point regarding use of A2 statistic concerns n , the number of values being tested. If $n < 5$ then the test is not meaningful, and in this case a significance level of -1 is returned.

On the other hand, for "real" data the test **gains** power as n becomes larger. It is well known in the statistics community that real data almost never exactly matches a theoretical distribution,

Paxson et al.

[Page 32]

ID Framework for IP Performance Metrics February 1998

even in cases such as rolling dice a great many times (see [[Pa94](#)] for a brief discussion and references). The A2 test is sensitive enough that, for sufficiently large sets of real data, the test will almost always fail, because it will manage to detect slight imperfections in the fit of the data to the distribution.

For example, we have found that when testing 8,192 measured wire times for packets sent at Poisson intervals, the measurements almost always fail the A2 test. On the other hand, testing 128 measurements failed at 5% significance only about 5% of the time, as expected. Thus, in general, when the test fails, care must be taken to understand why it failed.

The remainder of this appendix gives C code for the routines mentioned above.

```
/* Routines for computing the Anderson-Darling A2 test statistic.
 *
 * Implemented based on the description in "Goodness-of-Fit
 * Techniques," R. D'Agostino and M. Stephens, editors,
 * Marcel Dekker, Inc., 1986.
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Returns the raw A^2 test statistic for n sorted samples
 * z[0] .. z[n-1], for z ~ Unif(0,1).
```

```

*/
extern double compute_A2(double z[], int n);

/* Returns the significance level associated with a A^2 test
 * statistic value of A2, assuming no parameters of the tested
 * distribution were estimated from the data.
 */
extern double A2_significance(double A2);

/* Returns the A^2 significance level for testing n observations
 * x[0] .. x[n-1] against an exponential distribution with the
 * given mean.
 *
 * SIDE EFFECT: the x[0..n-1] are sorted upon return.
 */
extern double exp_A2_known_mean(double x[], int n, double mean);

/* Returns the A^2 significance level for testing n observations
 * x[0] .. x[n-1] against the uniform distribution [min_val, max_val].

```

```

*
* SIDE EFFECT: the x[0..n-1] are sorted upon return.
*/
extern double unif_A2_known_range(double x[], int n,
                                   double min_val, double max_val);

/* Returns a pseudo-random number distributed according to an
 * exponential distribution with the given mean.
 */
extern double random_exponential(double mean);

/* Helper function used by qsort() to sort double-precision
 * floating-point values.
 */
static int
compare_double(const void *v1, const void *v2)
{
    double d1 = *(double *) v1;
    double d2 = *(double *) v2;

    if (d1 < d2)

```

```

        return -1;
    else if (d1 > d2)
        return 1;
    else
        return 0;
}

double
compute_A2(double z[], int n)
{
    int i;
    double sum = 0.0;

    if ( n < 5 )
        /* Too few values. */
        return -1.0;

    /* If any of the values are outside the range (0, 1) then
     * fail immediately (and avoid a possible floating point
     * exception in the code below).
     */
    for (i = 0; i < n; ++i)
        if ( z[i] <= 0.0 || z[i] >= 1.0 )
            return -1.0;

    /* Page 101 of D'Agostino and Stephens. */

```

```

    for (i = 1; i <= n; ++i) {
        sum += (2 * i - 1) * log(z[i-1]);
        sum += (2 * n + 1 - 2 * i) * log(1.0 - z[i-1]);
    }
    return -n - (1.0 / n) * sum;
}

double
A2_significance(double A2)
{
    /* Page 105 of D'Agostino and Stephens. */
    if (A2 < 0.0)
        return A2;    /* Bogus A2 value - propagate it. */

    /* Check for possibly doctored values. */

```

```

    if (A2 <= 0.201)
        return 0.99;
    else if (A2 <= 0.240)
        return 0.975;
    else if (A2 <= 0.283)
        return 0.95;
    else if (A2 <= 0.346)
        return 0.90;
    else if (A2 <= 0.399)
        return 0.85;

    /* Now check for possible inconsistency. */
    if (A2 <= 1.248)
        return 0.25;
    else if (A2 <= 1.610)
        return 0.15;
    else if (A2 <= 1.933)
        return 0.10;
    else if (A2 <= 2.492)
        return 0.05;
    else if (A2 <= 3.070)
        return 0.025;
    else if (A2 <= 3.880)
        return 0.01;
    else if (A2 <= 4.500)
        return 0.005;
    else if (A2 <= 6.000)
        return 0.001;
    else
        return 0.0;
}

double

```

```

exp_A2_known_mean(double x[], int n, double mean)
{
    int i;
    double A2;

    /* Sort the first n values. */
    qsort(x, n, sizeof(x[0]), compare_double);

```



```

    /* Assuming they match an exponential distribution, transform
    * them to Unif(0,1).
    */
    for (i = 0; i < n; ++i) {
        x[i] = 1.0 - exp(-x[i] / mean);
    }

    /* Now make the A^2 test to see if they're truly uniform. */
    A2 = compute_A2(x, n);
    return A2_significance(A2);
}

double
unif_A2_known_range(double x[], int n, double min_val, double max_val)
{
    int i;
    double A2;
    double range = max_val - min_val;

    /* Sort the first n values. */
    qsort(x, n, sizeof(x[0]), compare_double);

    /* Transform Unif(min_val, max_val) to Unif(0,1). */
    for (i = 0; i < n; ++i)
        x[i] = (x[i] - min_val) / range;

    /* Now make the A^2 test to see if they're truly uniform. */
    A2 = compute_A2(x, n);
    return A2_significance(A2);
}

double
random_exponential(double mean)
{
    return -mean * log1p(-drand48());
}

```

[AK96] G. Almes and S. Kalidindi, "A One-way Delay Metric for IPPM", work in progress, November 1997.

[BM92] I. Bilinskis and A. Mikelsons, Randomized Signal Processing, Prentice Hall International, 1992.

[DS86] R. D'Agostino and M. Stephens, editors, Goodness-of-Fit Techniques, Marcel Dekker, Inc., 1986.

[CPB93] K. Claffy, G. Polyzos, and H-W. Braun, ``Application of Sampling Methodologies to Network Traffic Characterization,' Proc. SIGCOMM '93, pp. 194-203, San Francisco, September 1993.

[FJ94] S. Floyd and V. Jacobson, ``The Synchronization of Periodic Routing Messages,' IEEE/ACM Transactions on Networking, 2(2), pp. 122-136, April 1994.

[Mi92] D. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis", [RFC 1305](#), March 1992.

[Pa94] V. Paxson, ``Empirically-Derived Analytic Models of Wide-Area TCP Connections,' IEEE/ACM Transactions on Networking, 2(4), pp. 316-336, August 1994.

[Pa96] V. Paxson, ``Towards a Framework for Defining Internet Performance Metrics,' Proceedings of INET '96, <ftp://ftp.ee.lbl.gov/papers/metrics-framework-INET96.ps.Z>

[Pa97] V. Paxson, ``Measurements and Analysis of End-to-End Internet Dynamics,' Ph.D. dissertation, U.C. Berkeley, 1997, <ftp://ftp.ee.lbl.gov/papers/vp-thesis/dis.ps.gz>.

19. Authors' Addresses

Vern Paxson <vern@ee.lbl.gov>
MS 50B/2239
Lawrence Berkeley National Laboratory
University of California
Berkeley, CA 94720
USA
Phone: +1 510/486-7504

Guy Almes <almes@advanced.org>
Advanced Network & Services, Inc.
200 Business Park Drive

Armonk, NY 10504

USA

Phone: +1 914/273-7863

Jamshid Mahdavi <mahdavi@psc.edu>

Pittsburgh Supercomputing Center

4400 5th Avenue

Pittsburgh, PA 15213

USA

Phone: +1 412/268-6282

Matt Mathis <mathis@psc.edu>

Pittsburgh Supercomputing Center

4400 5th Avenue

Pittsburgh, PA 15213

USA

Phone: +1 412/268-3319

