Chapter 4

Methodology

In this chapter we discuss the methodology used to make our routing measurements. We begin with the software we used: the `npd` network probe daemon, the `npd_control` program used to drive the measurements, and the `traceroute` utility for measuring Internet paths. We then discuss the utility of sampling at exponentially distributed intervals, including the "PASTA Principle," which provides the underlying statistical validity of our measurements. In § 4.4 we then address which aspects of our data are plausibly representative of Internet traffic and which are not.

In our analysis we also attempt to draw some conclusions as to which differences between our datasets reflect significant changes in Internet conditions over time. To do so, we give in § 4.5 an overview of *Fisher's exact test* for determining whether the frequencies with which a property is observed in two different datasets is consistent with the null hypothesis of a single underlying probability of observing the property. If the frequencies observed are inconsistent with this hypothesis, then we conclude that the probability of observing the property *changed* between the two datasets, reflecting a corresponding change in Internet conditions. Finally, in order to use Fisher's test, we need to make an independence assumption that is not entirely accurate. § 4.6 discusses why this assumption remains tenable.

4.1 Experimental apparatus

We conducted our experiment as follows. First we recruited a number of Internet sites (detailed in Tables I and II) to participate in the study. Each site ran a "network probe daemon" (`npd`) that provides measurement services, as described in the Appendix. To measure the route from Internet host A to host B, a program called `npd_control`, running on our local workstation, would connect to `npd` on host A and request that it trace the route to host B using `traceroute`. The `npd` on A would then do so and send the results back to `npd_control`. In this fashion, we could run a single script on our local workstation to orchestrate any number of simultaneous route measurements. The script (which we programmaticly generated) would run `npd_control` in the background to conduct a single measurement, sleep until the time for the next measurement, run another `npd_control` in the background to conduct that measurement, and so on. Each measurement comprised a single `traceroute` from a randomly selected site to another randomly selected site.

This setup gave our experiment a single point of failure, namely our local workstation, but also the benefit of a single point of administration, which greatly simplified the task of keeping the experiment running correctly as we added new participating sites. Fortunately, during the entire measurement period the workstation never crashed or required rebooting, so the measurements proceeded uninterrupted.

For our first set of measurements, termed $\mathcal{R}_1$, we tuned the script driving the measurements so that each site would measure routes at an average rate of one every two hours, to minimize network load. Two exceptions were the `austr` and `korea` sites. They instead made measurements at lower rates of one every four hours and one every eight hours, in deference to the heavily loaded trans-Pacific network links that their traffic had to cross.

While using the same rate for each site meant each site had a consistent measurement load, as we added new participating sites to our study, the sampling rate of *pairs* of sites decreased. This inhomogeneity, however, does not present any particular difficulties for our sampling methodology, a point we address in § 4.3.

For the second set of measurements, $\mathcal{R}_2$, we made measurements at two different, fixed rates. The majority (60%) of the measurements were made with a mean inter-measurement interval of 2 hours, while the remainder were made with a mean interval of about 2.75 days. The bulk of the $\mathcal{R}_2$ measurements were also *paired*, meaning we would measure the path $A \Rightarrow B$ and then immediately measure the path $B \Rightarrow A$. We discuss the reasons for these changes in methodology in § 7.4 and § 8.4.

4.2 The `traceroute` Utility

`Traceroute` is a program written by Van Jacobson to trace the different hops comprising a route through the Internet [Jac89]. In this section we discuss the operation of the tool, as its particulars have direct impact on our routing measurements.

4.2.1 The Time To Live field

All packets sent using the Internetwork Protocol (IP) contain in their headers a *Time To Live* (TTL) field [Po81a]. In the original IP design, this field was meant to limit the amount of time that a packet could exist inside the network, to prevent packets from endlessly circulating around routing loops (and eventually clogging up the entire network). The TTL header field is 8 bits wide and is interpreted as the time in seconds remaining until the packet must be discarded. Each internetwork router must decrement the field by the amount of time required to process the packet (including queueing), or by 1 second, whichever is larger. Thus, the TTL limits packets to at most 255 hops through the network,[1] and a lifetime of at most 255 seconds.

If upon decrementing the TTL field a router observes that the TTL has reached zero, then it must not forward the packet but instead discard it as being too old. When it discards a packet for this reason, it must[2] then send back an Internet Control Message Protocol (ICMP; [Po81b]) message informing the sender of the packet that it was dropped due to an expired lifetime.

[1] This is plenty in today's Internet. Routes of more than 30 hops are rare (§ 6.7.5). But if much longer routes became commonplace, then the limited size of the TTL field could render parts of the Internet unable to communicate with other parts.

[2] This "must" is actually a very strong "should." [Ba95] states that the router must generate the message, but can provide a per-interface option to disable generation, provided the option defaults to generation enabled.

While the original IP standard states that TTL is a *time* [Po81a], in reality virtually all Internet routers only decrement the TTL by 1 per hop, regardless of the processing time, often for reasons of performance. Acknowledging this *de facto* behavior, the current standard for Internet routers only requires that routers decrement the TTL by 1 per hop, while allowing them the option to decrement by more to account for processing time [Ba95]. Part of the motivation for this relaxation of the TTL requirement is to aid the workings of `traceroute`.

4.2.2 How `traceroute` works

The heart of `traceroute` is clever exploitation of the TTL field, as follows. To trace the route to a remote host H, `traceroute` first constructs a packet with H as its destination but with the TTL field initialized to 1. When this packet reaches the first hop in the path to H, the router decrements the TTL field, notices that it is zero, and sends back an ICMP message to this effect. The ICMP message includes in its own header the address of the router sending the message, which lets `traceroute` identify the hop 1 router as that address.

`Traceroute` then sends a packet to H with the TTL field initialized to 2, and, similarly, gets back an ICMP message identifying the hop 2 router. It proceeds in this fashion until it receives a reply from H itself, and at that point it has elucidated the entire path to H. (Note that it has *not* also elucidated the path from H to the host running `traceroute`. The two are not necessarily the same, as we demonstrate in Chapter 8.)

We will refer to the packets `traceroute` sends with adjusted TTL's as *probes*, and those with an initial TTL of n as "hop n" probes. Here is an example of the output from `traceroute`, tracing the path from a host at the University of Colorado at Boulder (`ucol`, as explained in Table I) to one at the San Diego Supercomputing Center (`sdsc`).

```
traceroute to rintrah-fddi.sdsc.edu (198.17.46.57),
  30 hops max, 40 byte packets
 1  128.138.209.2  2 ms  2 ms  2 ms
 2  128.138.138.1  14 ms  4 ms  3 ms
 3  144.228.73.113  44 ms  39 ms  53 ms
 4  144.228.73.82  218 ms  207 ms  147 ms
 5  134.24.66.100  234 ms *  85 ms
 6  198.17.46.57  85 ms  63 ms  67 ms
```

By default, `traceroute` sends three probes for each hop. The probes are sent serially, each waiting until `traceroute` receives an answer for the previous one. For each hop, `traceroute` reports the number of hop, the IP address of the corresponding router, and the time in milliseconds it took to receive the reply. We note, however, that these times are often exceptionally noisy, because part of the total round-trip time includes the delay incurred at the router in generating an ICMP response to the exceptional event of an expired TTL. This delay can be quite large of the router is busy with other, higher priority tasks.

A reply time of "`*`," such as shown for hop 5, corresponds to a *lost* packet. Either the `traceroute` probe or the corresponding ICMP message was dropped by the network (or perhaps the ICMP message was not generated—see § 6.1, and also below). `Traceroute` waits 5 seconds for a reply before deciding that it will not be getting one.[3]

[3]Most versions of the `traceroute` documentation erroneously give this time as 3 seconds.

The first line of the output indicates "`30 hops max`," meaning that `traceroute` will stop sending probes after trying to elicit the 30th hop. This behavior is important because, as we will see in § 6.3.1, the Internet sometimes contains routing loops that would allow packets to circulate all the way up to the maximum of 255 hops, wasting considerable network resources. For our study we always used the default of 30 hops maximum (only very rarely did this prevent us from measuring the full path between sites in our study; see § 6.7.5), and the default of three probes per hop.

We can translate the IP addresses to hostnames in order to visualize the route more clearly:

```
 1  cs-gw-discovery.cs.colorado.edu  2 ms  2 ms  2 ms
 2  cu-gw.colorado.edu  14 ms  4 ms  3 ms
 3  sl-ana-3-s2/4-t1.sprintlink.net  44 ms  39 ms  53 ms
 4  sl-univ-ca-1-s0-t1.sprintlink.net  218 ms  207 ms  147 ms
 5  sdsc-ucop-mci.cerf.net  234 ms *  85 ms
 6  rintrah.sdsc.edu  85 ms  63 ms  67 ms
```

We see that the first two hops occur inside the University of Colorado at Boulder; then the packets are forwarded on to SprintLink, traveling first to Anaheim, CA, then up to Oakland, California (the University of California Office of the President), and finally back down along CERFNET to San Diego.

4.2.3 `Traceroute` limitations

When using `traceroute` there are several limitations and measurement difficulties that one must bear in mind. In the previous section we showed an example of a `traceroute` from Colorado to San Diego that went quite smoothly, suffering only a single packet loss. In contrast, consider the following `traceroute`, between the same two hosts:

```
traceroute to rintrah.sdsc.edu (198.17.47.57),
  30 hops max, 40 byte packets
 1  128.138.209.2  10 ms  0 ms  0 ms
 2  128.138.138.1  0 ms  0 ms  0 ms
 3  129.19.248.61  10 ms 129.19.254.45  10 ms 129.19.248.61  30 ms
 4  192.52.106.1  60 ms  60 ms  70 ms
 5  140.222.96.4  60 ms *  50 ms
 6  140.222.88.1  70 ms  60 ms  60 ms
 7  140.222.8.1  60 ms  50 ms  60 ms
 8  140.222.16.1  70 ms  70 ms  70 ms
 9  140.222.135.1  60 ms  70 ms  70 ms
10  198.17.47.2  4720 ms !H *  5100 ms !H
```

Here are the corresponding hostnames:

```
traceroute to rintrah.sdsc.edu (198.17.47.57),
  30 hops max, 40 byte packets
 1  cs-gw-discovery.cs.colorado.edu  10 ms  0 ms  0 ms
 2  cu-gw.colorado.edu  0 ms  0 ms  0 ms
 3  129.19.248.61 10 ms ncar-cu.co.westnet.net 10 ms 129.19.248.61 30 ms
 4  enss.ucar.edu  60 ms  60 ms  70 ms
 5  t3-3.cnss96.denver.t3.ans.net  60 ms *  50 ms
```

```
 6  t3-0.cnss88.seattle.t3.ans.net  70 ms  60 ms  60 ms
 7  t3-0.cnss8.san-francisco.t3.ans.net  60 ms  50 ms  60 ms
 8  t3-0.cnss16.los-angeles.t3.ans.net  70 ms  70 ms  70 ms
 9  t3-0.enss135.t3.ans.net  60 ms  70 ms  70 ms
10  enss.sdsc.edu  4720 ms !H *  5100 ms !H
```

The first thing we notice is that this route is longer than the previous one, and more circuitous, traveling over ANSNET (instead of SprintLink) through Denver and Seattle before arriving in California.

We also notice that the router at hop 3, `129.19.248.61`, does not have a corresponding hostname registered in the Domain Name System (DNS; [MD88]). While most routers have hostnames associated with their IP addresses, we found that not all do. In this case, we could identify the router's location from its network prefix (`129.19`), as Colorado State University in Boulder, Colorado.

Furthermore, for hop 3 `traceroute` reports not just one IP address but *multiple* addresses. What happened was that the first hop 3 probe was routed via the router with IP address `129.19.248.61`, while the second one went via a *different* router, `129.19.254.45` (this one has a hostname, `ncar-cu.co.westnet.net`). The third one went via the same router as the first one, `129.19.248.61`. Routing variation such as this can occur due to "load balancing," in which the upstream router (hop 2 in this case) alternates the downstream links it uses to forward packets in an effort to spread load among them and avoid overloading either one. We investigate the effects of such routing, which we term "fluttering," in detail in § 6.6.

Hop 3 also illustrates the more general principle that *packets do not always take the same route*. It also can be difficult to determine whether two routes are equivalent. For example, it may be that `129.19.248.61` is indeed an interface on the same `ncar-cu.co.westnet.net` router, but one that happens not to have a hostname associated with it. Or it may be a physically distinct router.

Because Internet routes can change between successive probe packets, we need to also realize that *we have no guarantee that probes of different hops take the same route as previous probes.* For example, from the above we might conclude that the first hop 3 probe took the route `cs-gw-discovery.cs.colorado.edu` → `cu-gw.colorado.edu` → `129.19.248.61`, and the second took the route `cs-gw-discovery.cs.colorado.edu` → `cu-gw.colorado.edu` → `ncar-cu.co.westnet.net`. But for all we know the upstream route could have changed between the end of the hop 2 probes and the beginning of the hop 3 probes, and the hop 3 packets may have been routed via Alaska at the first two hops! The only "guarantees" we can have that the route has not changed are: (1) consistency with other measurements of the same path (for example, in multiple measurements we always see the same routers for hop 2 and hop 3), and (2) self-consistency within the route. For example, if we find that hop $n + 1$ is geographically distant from hop n, and we know the network lacks a link between those two locations, then we would conclude that a routing change occurred upstream from hop $n + 1$. Some examples of this behavior are given in § 6.5 and § 6.6.1.

In general, if a route appears self-consistent and shows no sign of multiple routing for any of its hops, then we assume that it is indeed self-consistent, and treat the route as a valid measurement of the path to the remote host.

Another anomaly to discuss in the example above is the 10th hop:

```
10  enss.sdsc.edu  4720 ms !H *  5100 ms !H
```

Here "`!H`" indicates that `traceroute` received an ICMP "Host unreachable" message from the router `enss.sdsc.edu`. This means that the router knows that the host cannot be presently reached. Another diagnostic `traceroute` can generate is "`!N`," indicating that it received an ICMP "Network unreachable," the counterpart message indicating an entire network is unreachable (e.g., due to a failed link). We observed only two of these in all of our measurements.

Note also that the 3rd probe packet reports a round-trip time (RTT) of 5,100 msec, even though `traceroute` supposedly only waits 5 seconds to receive a reply. `Traceroute`'s timer, however, is not fine-grained, so due either to the timer's granularity, or to delays in scheduling the `traceroute` process for execution, `traceroute` received the reply before it decided to time out the probe.

Another limitation to keep in mind is that `traceroute` elicits the route as seen at the *IP network layer*. Each hop reported gives the next IP router in the path from the source to the destination. Often, IP routers are connected to one another using simple "link layer" technologies such as Ethernets or point-to-point links, with trivial topologies. Increasingly, however, the link layer technologies, for example ATM or Frame Relay, themselves have more complicated topologies, and are capable of routing packets within a link layer mesh that itself has multiple hops. `traceroute` cannot measure routing at this layer, because the TTL mechanism (§ 4.2.2) is present only at the higher (IP) layer. For example, in our second dataset we found a route with the following two successive hops:

```
gw1.scl1.alter.net
107.hssi4/0.gw1.mia1.alter.net
```

The first hop is in Santa Clara, California, and the second in Miami, Florida. It turns out that there is no direct physical connection between these two routers, but rather a Frame Relay mesh [Lid96], a fact that we could not have surmised from the `traceroute` measurement of the route.

Another potential source of measurement error arises in older (4.3 BSD-derived) routers incorrectly setting the TTL in their ICMP replies. As explained in the `traceroute` documentation ([Jac89]), these routers would erroneously use for the ICMP reply the TTL of the incoming packet that triggered the reply. For `traceroute` probes, this is a disaster, because the reply being triggered is precisely "TTL expired," so the ICMP replies would be sent back using a TTL of 0, too (and thus never reach us). Since such routers consistently fail to return an ICMP reply to the sender, they are a form of "unresponsive" router, for which we analyze our measurements in § 6.1.

A more subtle measurement problem occurs due to routers that are configured to *rate limit* generation of ICMP messages. For example, some routers will send at most one ICMP message each second. Such behavior is specifically encouraged in §4.3.2.8 of [Ba95], as a means of conserving both network bandwidth and router resources. In § 6.2 we analyze our measurements for the presence of rate-limiting routers, and find that, in general, only endpoint hosts (and not routers internal to the Internet) appear to be presently limiting their ICMP generation rate.

Another issue regarding `traceroute` concerns its use of the User Datagram Protocol (UDP; [Po80]). In order to associate the ICMP replies it receives with the probe packets it previously generated, `traceroute` must construct packets that manage to record identifying information in just the first 8 bytes of the transport layer header, as that is all of the original packet returned in an ICMP message. It does this by using for its probe a UDP packet, which it sends to a (hopefully) non-existent port on the remote host H. The information `traceroute` needs to record the identifying information is coded in the port number in the UDP header.

Some network sites, however, have "firewalls" in place to filter incoming network traffic for security purposes [CB94]. These firewalls may decide that the incoming UDP packet does not appear destined to any of the services the site wishes to make publicly available to the Internet, so the firewall drops the packet without returning an ICMP Time Exceeded message. Thus, firewalls can generate an effect similar to lost packets (`traceroute` never receives a reply for a given hop, or beyond it). It was easy to identify such sites, as `traceroutes` to them consistently stopped short at the same router (§ 6.7.4). For our analysis of the data, we considered any `traceroute` reaching a firewall router as having successfully reached the host.

`Traceroute`'s use of UDP packets raises another measurement issue. When `traceroute` traces the route to an IP address A, it determines that it has elicited the full route whenever it receives a "UDP Port Unreachable" ICMP reply, *even if the reply did not come from a router identifying itself as address A*. Some hosts (and indeed all routers) have multiple IP addresses associated with them, so it is possible when tracing the route to address A to receive a reply from address B. When this happens, it indicates that A and B are both addresses for the same host (even though their associated hostnames might not reveal this).[4]

It is sometimes possible to use this `traceroute` feature to determine whether two IP addresses correspond to the same host. For example, the name associated with `134.55.12.231` is `llnl3-e-stub.es.net`, while the name associated with `134.55.6.71` is `llnl-lc3-3.es.net`. Both of these names have DNS "A" records for the corresponding addresses, and no extra records, so *a priori* we might assume that the two addresses/hostnames refer to two separate machines. However, depending on the state of ESNET routing, it is possible for a traceroute to `llnl3-e-stub.es.net` to be "answered" by `llnl-lc3-3.es.net`, indicating that they are indeed the same machine. This test is not guaranteed to work, though. It depends on the machine's algorithm for deciding what IP address to put in its ICMP reply, and on which interface the incoming UDP probe packet arrives (which in turn depends on the current routing).

4.3 Exponential sampling

We use the term "measurement" to denote the full process of running the `traceroute` utility; that is, the attempted tracing of the entire route between a source host and a destination host. In our experiment we devise our measurements of Internet routes so that the time intervals between consecutive measurements are independent and exponentially distributed.

Using independent and exponentially distributed intervals between measurements gains two important (and related) properties. The first is that the measurements correspond to *additive random sampling* [BM92]. Such sampling is unbiased because it samples all instantaneous signal values with equal probability.

The second important property is that the measurement times form a Poisson process. This means that Wolff's *PASTA principle*—"Poisson Arrivals See Time Averages"—applies to our measurements: asymptotically, the proportion of our measurements that observe a given state is equal to the amount of time that the Internet spends in that state [Wo82].

[4] Note also that sometimes the route to address A is different than the route to address B! For our measurements, this only occurred for `mbone.ucar.edu`, for which the route to one of its addresses is one hop longer (and a strict superset) of the route to the other address. We accommodated this difference in our analysis by considering a `traceroute` that reached the endpoint of the shorter route as having traveled successfully to the host.

Two important points regarding Wolff's theorem are (1) the observed process does *not* need to be Markovian; and (2) the Poisson arrivals need not be *homogeneous*[5] [Wo82, § 3]. This second point is particularly important for our study, because our measurement rate varied, as discussed in § 4.1.

The only requirement of the PASTA theorem is that the observed process cannot *anticipate* observation arrivals. For any interarrival distribution other than independent exponentials, the process can anticipate observation times to some degree because the instantaneous probability of an arrival changes with the length of time since the last observation. For the exponential distribution, however, the probability remains constant, a consequence of the distribution's "memoryless" property. Thus, the theorem fundamentally requires independent exponential intervals between measurements, which argues strongly for the use of exponential sampling in practice.

There is one respect in which our measurements fail the "lack of anticipation" requirement. Even though we schedule our observations to come at independent, exponentially distributed intervals, the network *can* anticipate arrivals to a certain extent. In particular, *when the network has lost connectivity between the site running* `npd_control` (§ 4.1) *and a site potentially conducting a* `traceroute`*, the network can predict that* no *measurement will occur.* Thus, while the times at which we *attempted* to measure the network satisfy the PASTA requirements, the times for which we *successfully* measured the network do not in this regard. The effect of this imperfect sampling is a tendency to *underestimate* the prevalence of network connectivity problems, as discussed further in § 5.2.

The main use we make of the PASTA theorem is as follows. If we make n observations of Internet routing, of which k find state S and $n-k$ find some other state, then because of PASTA we are on firm ground making the assumption that the unconditional probability of observing state S is approximately k/n. Furthermore, if $k \ll n$, we argue that we can consider the observations as independent, and hence can apply a Fisher's exact test (§ 4.5) to test for significant differences among sets of observations. We discuss this independence assumption further in § 4.6.

4.4 Which observations are representative?

In this section we discuss what sort of observations we can make of the Internet for which our samples are plausibly representative of Internet behavior in general, and those for which we would not consider our samples representative.

37 Internet hosts participated in our routing study. This is a miniscule fraction of the estimated 6.6 million Internet hosts as of July, 1995 [Lo95], so clearly the behavior we observe that is due to the particular endpoint hosts in our study is not representative.[6]

The 37 endpoint hosts were from 34 different networks, again a miniscule fraction of the more than 50,000 known to the NSFNET in April, 1995 [Me95a]. So, again, any behavior we observe due to the particular endpoint ("stub") networks in our study is not persuasively representative.

On the other hand, we argue that the *routes* between the 37 hosts are plausibly representative, because they include a non-negligible fraction of the *autonomous systems* (AS's) which

[5]That is, the arrival rate can vary over time, as long as the interarrival distribution remains exponential and the arrivals remain independent of each other and of the observed process.

[6]Furthermore, the sites were self-selected (usually, though not always, because someone at the site had an interest in wide-area networking) and skewed to universities.

together comprise the Internet. Recall that AS's are administrative entities that manage routing for a collection of networks, using unspecified protocol(s), and that routing *between* AS's is done using the Border Gateway Protocol. We expect the different routes within an AS to have similar characteristics (e.g., prevalence of pathologies, or routing stability), because they fall under a common administration. We therefore argue that sampling a significant number of AS's lends representational weight to a set of measurements.

To determine the number of AS's in the Internet, we proceeded as follows. In January, 1996, we obtained a BGP routing table dump from the AS border router `kasina.sdsc.edu`, located at the San Diego Supercomputer Center (SDSC) [7]. The routing table lists all the destinations (networks, more or less) known to the router, i.e., its view of the Internet. For each of those destinations, the table includes a list of AS's over which routing information for the destination traveled to `kasina.sdsc.edu`. The view of Internet routing given directly by this table is skewed by SDSC's particular location in the Internet. However, virtually all of the routing reflects disparate AS's connecting to SDSC's network service provider, MCI, at many different points. So, if we exclude MCI itself from our subsequent analysis, then the remainder of the routing gives us a much broader view, namely that seen by MCI at its many interconnection points.

All in all, the routes in the table included 1,031 AS's for 33,824 distinct destinations. From this we estimate that the Internet presently has about 1,000 active AS's. (As of August, 1995, about 6,600 had been assigned [DISA95].) The routes in our study traversed 85 of these, or about 8%.

An important point, however, is that not all AS's are equal—some are much more prominent in Internet routing than others. We devised a "weight" to associate with AS's as follows. For each AS, we counted the number of times it occurred in the BGP table in a path to a remote destination. The AS's weight then is the ratio of the number of its occurrences to the total number of occurrences of any AS.[8]

The weights obtained in this fashion are skewed towards the view of the Internet as seen by SDSC, and indeed two AS's had weight 25%: AS 145 ("NSFNET-CORE") and AS 3561 ("MCI-RESTON"), because virtually every route known to the SDSC router goes through these two. But the next AS has a weight of only 5% (AS 1239, "SprintLink"), because the majority of the routes do not go through it. So we adjusted for the SDSC-skewed perspective by removing the first two AS's from the set and recomputing the weights. After this adjustment, we find that the AS's sampled by the routes we measured represent, by weight, about 52% of the Internet routes. We take this as an indication that we did indeed sample a significant subset of the large-scale variation in Internet routes, and our observations of those routes are plausibly representative of Internet routing as a whole.

4.5 Testing for significant differences

Because we have measurements taken at two points in time—the end of 1994 and the end of 1995—we have an opportunity to assess a number of aspects of the measurements in the two datasets for the degree to which they reflect significant differences. We can then interpret these

[7] Many thanks to Hans-Werner Braun of SDSC for suggesting and facilitating this.

[8] Better would probably be to weight by traffic volume. Unfortunately, the statistics necessary for doing so are not available.

differences (or lack of differences) as indicating how the Internet changed (remained unchanged) over the course of 1995. While having just two points in time offers only the most crude form of trend, it is still far better than simply assuming that characteristics of the Internet do not change, particularly given evidence of major changes over time as discussed in our previous work [Pa94b, Pa94a].

The potential changes we will attempt to assess concern the frequency with which we observe different Internet phenomena (for example, routing loops). Suppose that, out of two representative samples from $\mathcal{R}_1$ and $\mathcal{R}_2$ of n_1 and n_2 observations, respectively, we find that subsets of size k_1 and k_2 exhibit some property $\mathcal{P}$. We wish to gauge whether finding k_1 instances of $\mathcal{P}$ out of n_1 samples in $\mathcal{R}_1$ is statistically consistent with finding k_2 instances out of n_2 samples in $\mathcal{R}_2$. If consistent, then we do not have evidence of a significant change between $\mathcal{R}_1$ and $\mathcal{R}_2$. But if the findings are inconsistent, then we interpret the difference as due to a change in the prevalence of $\mathcal{P}$: either the likelihood of $\mathcal{P}$ increased during 1995, if $\frac{k_2}{n_2} > \frac{k_1}{n_1}$, or decreased, if $\frac{k_2}{n_2} < \frac{k_1}{n_1}$.

To test for statistically significant differences, we use *Fisher's exact test*. The discussion of the test we now present follows that of Rice [Ri95]. Let K_1 denote a random variable giving the number of instances of $\mathcal{P}$ observed in $\mathcal{R}_1$, N_1 the total number of observations in $\mathcal{R}_1$, and K_2 and N_2 the same for $\mathcal{R}_2$. Let $K = K_1 + K_2$ and $N = N_1 + N_2$ correspond to the totals across both datasets.

The key observation of Fisher's test is that, if the likelihood of observing $\mathcal{P}$ is the same in the two datasets, then we can view the problem as: for K total instances of $\mathcal{P}$ out of N observations, how likely is it that K_1 of them would have fallen into $\mathcal{R}_1$, given that $\mathcal{R}_1$ comprises N_1 observations? With this rephrasing of the problem, we have that

$$P[K_1 = k_1 | N_1 = n_1, K = k, N = n] = \frac{\binom{n_1}{k_1}\binom{n-n_1}{k-k_1}}{\binom{n}{k}}. \tag{4.1}$$

The numerator of Eqn 4.1 corresponds to the number of ways that k instances of $\mathcal{P}$ can be distributed, among a partition of n total observations, into two sets of n_1 and $n_2 = n - n_1$ observations, given that the first set of observations includes k_1 instances of $\mathcal{P}$. The denominator corresponds to the total number of ways that k instances can be distributed over n observations, not subject to any conditioning. The ratio then gives the probability of observing k_1 instances in $\mathcal{R}_1$, given the size of $\mathcal{R}_1$, the total number of instances of $\mathcal{P}$, the size of the combined sample pool, and the null hypothesis that $\mathcal{R}_1$ and $\mathcal{R}_2$ are constructed using independent draws without replacement from the combined sample pool.

Armed with Eqn 4.1 for the probability of observing exactly k_1 instances, we can then construct a *rejection region* corresponding to values of k_1 that we would be unlikely to observe if the null hypothesis is indeed correct. We use a *two-sided* region, meaning that it includes both values of k_1 that are too low to be likely, and values that are too high. To construct the region, we find the maximum k_l and minimum k_u for which

$$\begin{aligned} P[K_1 \leq k_l | N_1 = n_1, K = k, N = n] &\leq \frac{\alpha}{2} \\ P[K_1 \geq k_u | N_1 = n_1, K = k, N = n] &\leq \frac{\alpha}{2}. \end{aligned}$$

Given these values, we then have

$$P[K_1 \leq k_l \text{ or } K_1 \geq k_u | N_1 = n_1, K = k, N = n] \leq \alpha.$$

So, given the null hypothesis, K_1 will fall into the rejection region by chance with probability α or smaller. By using $\alpha = 0.05$, using this test we will erroneously reject the null hypothesis at most 5% of the time. Consequently, if K_1 falls into the rejection region, we conclude with confidence 95% that the null hypothesis is incorrect, and indeed there was a significant change in the prevalence of $\mathcal{P}$ between $\mathcal{R}_1$ and $\mathcal{R}_2$.

All that remains to use this test is to specify how to find k_l and k_u. For a given κ, we have

$$P[K_1 \leq \kappa | N_1 = n_1, K = k, N = n] \quad = \quad \binom{n}{k}^{-1} \sum_{i=\max(0,k-n_2)}^{\kappa} \binom{n_1}{i}\binom{n-n_1}{k-i},$$

where $n_2 = n - n_1$. So to find k_l we simply carry out the summation for $\kappa = 0, \ldots, \min(n_1, k)$ and note the largest value of κ for which the probability is $\leq \frac{\alpha}{2}$.[9]

The procedure for finding k_u is analogous.

4.6 A note on independence

The argument in the previous section assumes that our measurements are observing independent events. This is not quite true for our measurements. Using Poisson sampling means that the measurement *arrivals* are independent. However, the observations *themselves* (what each independently scheduled measurement observes) are not independent: any temporal correlations in the observed process will be faithfully reflected in the observations.

However, we will be applying the methodology in § 4.5 to *rare* events, such as the observation of pathological routing conditions. These rare events are generally *not* clustered in time, so the approximation that observations of them are independent is a good one.

[9]Here and in the equation, the $\min$ and $\max$ operators are to exclude values of κ that are impossible because they require more than n_2 instances of $\mathcal{P}$ in the second set of the partition, or fewer than 0.