Using pathchar to estimate Internet link characteristics

Allen B. Downey
Colby College
Waterville, ME 04901
downey@colby.edu

To appear in ACM SIGCOMM '99

Abstract

We evaluate `pathchar`, a tool that infers the characteristics of links along an Internet path (latency, bandwidth, queue delays). Looking at two example paths, we identify circumstances where `pathchar` is likely to succeed, and develop techniques to improve the accuracy of `pathchar`'s estimates and reduce the time it takes to generate them. The most successful of these techniques is a form of adaptive data collection that reduces the number of measurements `pathchar` needs by more than 90% for some links.

1 Introduction

`pathchar` is a new tool, written by Van Jacobson at Lawrence Berkeley Laboratory (LBL), that tries to infer the characteristics of individual links along an Internet path by measuring the round trip time of packets sent from a single host. An alpha version of `pathchar` is available for FreeBSD, Linux, OSF and Solaris from `ftp://ftp.ee.lbl.gov/pathchar/`. We explain the basic mechanism and evaluate its accuracy on two paths whose link characteristics are known.

Based on these observations, we propose techniques to improve the accuracy of `pathchar` and to reduce the number of measurements (and time) it takes to generate its estimates. The contributions of this paper are

- An evaluation of `pathchar` and some insight into when it can or can not be expected to be useful.
- A technique for generating intervals for the estimates `pathchar` generates. Although these intervals do not always contain the nominal (correct) values, their size conveys useful information about the accuracy of the estimates.
- A technique for determining dynamically the number of measurements needed to achieve a given interval size.

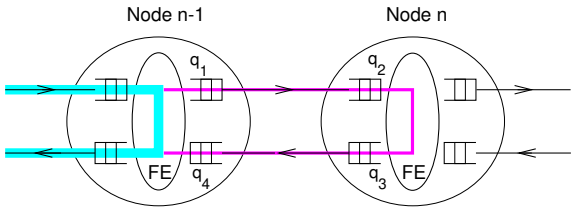

Figure 1: Network model.

1.1 Methodology

Using `pathchar`'s verbose option we collected measurements of two paths, one from rocky.colby.edu (on the campus of Colby College in Waterville, Maine) to emerald.mint.net (also in Waterville), the other from rocky to mach5.sdsc.edu (at the San Diego Supercomputer Center). We refer to these as the MINT and SDSC datasets.

Based on `pathchar`'s documentation, we wrote a program that processes these datasets and generates estimates of the link characteristics. This software allows us to test alternatives and extensions to `pathchar`'s techniques using the same data.

Our example paths, and the 11 links that comprise them, are not intended to be a statistical sample of the Internet (cf. Paxson's empirical studies [3, 4]). But the structure of these paths is probably typical of many—a fast local network connected through a comparatively slow link to the Internet proper, connected to the destination site through another LAN. We expect our experiences to be applicable to a large class of potential `pathchar` users.

2 Background

Van Jacobson presented `pathchar` at the Mathematical Sciences Research Institute (MSRI) in April 1997 [1]. The following description is based on slides he presented there.

Like `traceroute`[1], `pathchar` takes advantage of the **time-to-live** field (ttl) in an IP packet. The ttl determines how many links a packet can traverse before it expires. If a router receives a packet that has expired, it drops the packet and sends an ICMP error packet back to the sender. The source address of the error packet indicates which router the outgoing packet reached before expiring. By setting the ttl to

[1] For a survey of network measurement tools including `traceroute` see the Cooperative Association for Internet Data Analysis (CAIDA) web page `http://www.caida.org/Tools/`

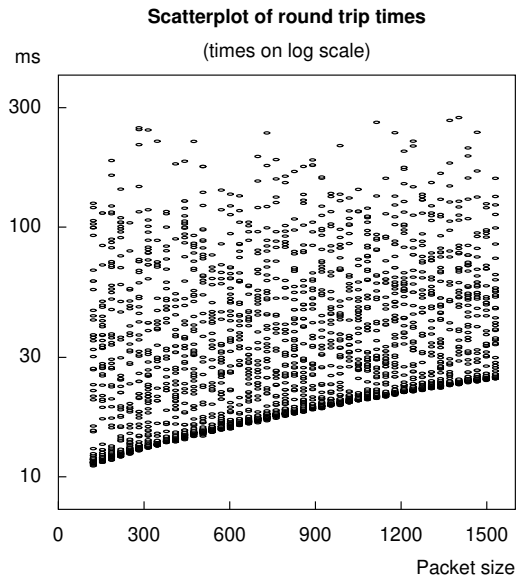

Figure 2: Scatterplot of round trip times versus packet size, 45 sizes, 64 probes each. The vertical axis is on a log scale, which is why the lower envelope appears convex, although it is a nearly straight line.

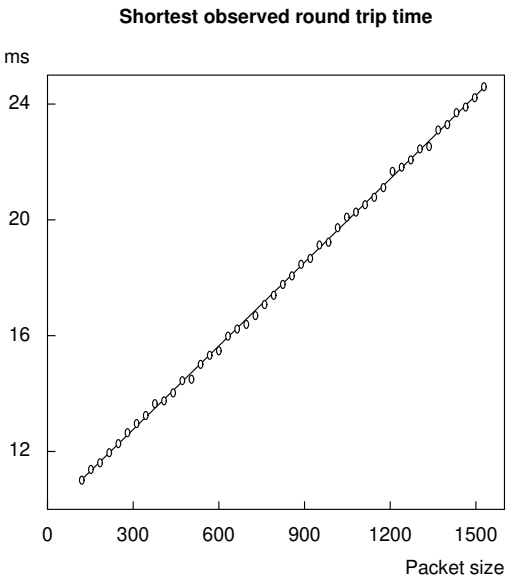

Figure 3: Shortest observed round trip time (SORTT) versus packet size. The line shows the linear least squares fit to the data.

a value n, it is possible to find the address of the nth router in the path.

`pathchar` works by sending out a series of probes with varying values of n and varying packet sizes. For each probe it measures the time until the error packet is received. By performing statistical analysis of these measurements, `pathchar` infers the latency and bandwidth of each link in the path, the distribution of queue times, and the probability that a packet is dropped.

The analysis is based on the network model in Figure 1. Before a packet leaves the $(n-1)$th node, it waits in queue to get onto the outgoing link. The time it spends on the network—transit time—is a linear function of the packet size, where the two parameters are the latency and bandwidth: $lat + size/bw$.

At node n, the packet waits in queue again until the router processes it and generates the error packet. The error packet waits in queue at node n, then returns to node $n-1$ with transit time $lat + error_size/bw$, where $error_size$ is the size of the ICMP error packet (56 bytes [5]).

Finally, it waits in queue at node $n-1$. The round trip time (rtt) from the $(n-1)$th to the nth node and back is:

$$\begin{aligned} rtt &= q_1 + (lat + packet_size/bw) + q_2 + forward \\ &+ q_3 + (lat + error_size/bw) + q_4 \end{aligned} \quad (1)$$

where the values q_i are random variables representing the queue times and $forward$ is the time it takes the forwarding engine to process the packet.

To simplify this expression `pathchar` makes three assumptions: (1) the size of the error packet is small enough that $error_size/bw$ is negligible, (2) the forward time is negligible, and (3) if we make a large number of measurements of a given path, eventually one of the probes will make the round trip with negligible queue delays. Eliminating the negligible terms yields:

$$rtt = (lat + packet_size/bw) + lat \quad (2)$$

This equation is the basis of the analysis `pathchar` uses to estimate link characteristics.

2.1 Statistical analysis

This section uses a sample data set to demonstrate the analysis `pathchar` performs. We use the following terms to refer to various data structures.

probe: a single measurement of an rtt for a given packet size and number of hops.

Sample: a set of probes with a given packet size.

Link: a set of Samples pertaining to a given link, spanning a range of packet sizes.

Path: a set of Links pertaining to a given path, in order from origin to destination.

Sample, Link, and Path are written with upper-case letters to indicate that they refer to a data structure, and to avoid confusion with their common use.

Minimum-filtering

Figure 2 shows a scatterplot of 2880 probes at 45 different sizes, from 120 to 1528 bytes, taken from the 6th link of the SDSC dataset. Each point represents a single probe; each column represents a Sample; the whole graph represents one Link.

Within each Sample, `pathchar` uses the shortest observed rtt to estimate the minimum possible rtt. Because we use the phrase "shortest observed rtt" frequently, we abbreviate it **SORTT**.

In each column there are many data points near the minimum, suggesting that packets have a reasonable chance of traversing the path without delay. This observation implies that `pathchar` can find the minimum rtt with a small number of probes at each packet size.

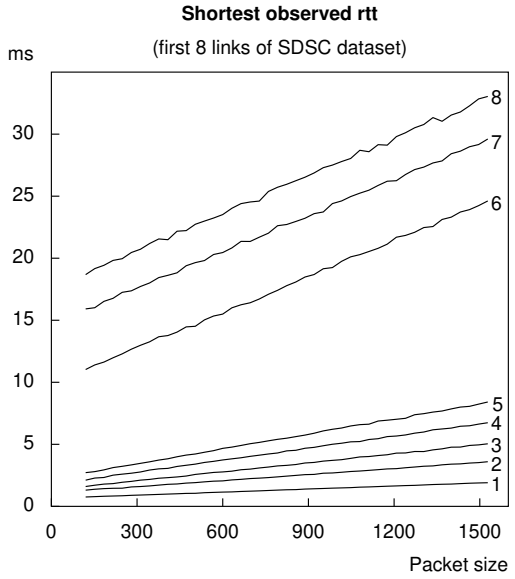

Figure 4: Shortest observed round trip times (SORTTs) for the first 8 links of the SDSC dataset.

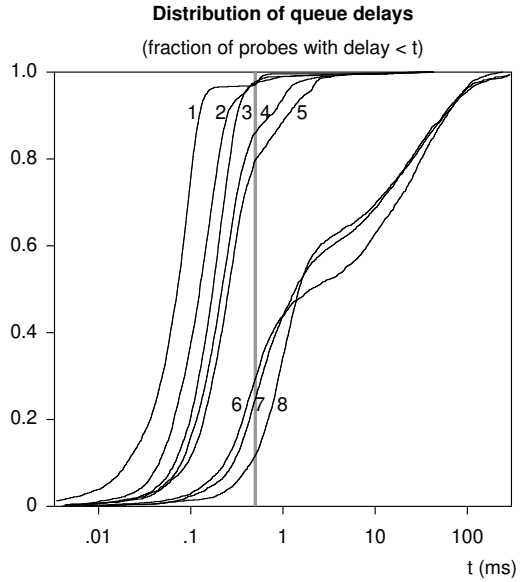

Figure 5: Distributions of cumulative queue delays, SDSC dataset.

The SORTTs from each column, plotted in Figure 3, form a straight line, in accordance with the two-parameter model of transit times (Equation 2).

Curve-fitting

Because the data fall so close to a line, it is easy to estimate parameters by a least squares fit. The interpretations of these **cumulative parameters** are:

- the latency from the first node to the nth node and back, and
- the marginal cost of sending an additional byte along the outgoing path.

Figure 4 shows the SORTTs for the first 8 links of the SDSC dataset. The line labeled 6 is the same line shown in Figure 3. In each case, the data fit a straight line well.

Differencing

The nice thing about the cumulative parameters is that they add: the parameters of a path are the sum of the parameters of the links. Thus, given estimated cumulative parameters, `pathchar` finds link parameters by subtraction.

For example, to find the latency of the 6th link, we subtract the intercepts of line 6 and line 5 (9.88 ms − 2.22 ms = 7.66 ms). According to Equation 2, this difference is equal to twice the latency, so the estimated link latency is 7.66/2 = 3.83 ms.

To find the bandwidth, we subtract the two slopes (9.61 μs/B − 4.02 μs/B = 5.6 μs/B). According to Equation 2, this difference is the inverse of the bandwidth, so the estimated link bandwidth is 1.43 Mb/s.

Deconvolution

If we assume that the SORTT in each column is the minimum possible rtt, then the additional time the other probes spend must be due to queueing and other nondeterministic delays.

Once `pathchar` fits a line to the SORTTs, it calculates the minimum possible rtt for each packet size and subtracts it from each probe. Aggregating these excess times, `pathchar` estimates the distribution of the total queue delay along the path to the nth node and back.

Figure 5 shows these distributions (empirical cumulative distribution functions) for the SDSC dataset. The vertical gray line is at 0.5 ms; where the distributions cross this line indicates the probability of observing an rtt within 0.5 ms of the minimum. As the length of the path grows, this probability drops quickly. This probability is relevant because it indicates how many probes are necessary to see an rtt near the minimum.

These distributions are cumulative, but unlike the cumulative parameters, they do not add in a simple way. Rather, each distribution is the convolution of the distributions for the prior links. In Section 7 we address the problem of deconvolving them.

2.2 Accuracy

Using `pathchar`'s techniques we estimated characteristics of the first 8 links of the sample path (Table 1). We chose the first 8 links because they provide examples of some of `pathchar`'s successes and failures while avoiding some complications in the subsequent links. We address the complications in Section 3.3.

The first 5 links are 10Mb/s Ethernets on Colby's campus. In each case, `pathchar`'s estimate is within 4% of the nominal value.

The next link is the T1 that connects Colby to the rest of the world. The estimated bandwidth, 1.43 Mb/s, is reasonably close to the nominal bandwidth, 1.536 Mb/s. It is surprising that it is not more accurate, though, since it is generally easy to measure the bandwidth of a slow link.

The link from bordercore2 to core4, according to an MCI representative, is an OC-3 with nominal bandwidth 155 Mb/s. Actually, OC-3 links are implemented as 3 distinct OC-1 bitstreams at 51.8 Mb/s. Each packet is sent down one of the three pipes in round robin fashion. Thus, from

link	host	address	latency est. (μs)	bandwidth est. (Mb/s)	nominal bandwidth
1	hub.colby.edu	137.146.194.17	328	9.74	10
2	routing.colby.edu	137.146.238.194	222	9.93	10
3	network.colby.edu	137.146.238.209	116	10.1	10
4	general.colby.edu	137.146.112.132	220	9.65	10
5	routing.colby.edu	137.146.238.146	223	10.4	10
6	bordercore2.cht.mci.net	166.48.64.25	3.83 ms	1.43	1.536
7	core4.WestOrange.mci.net	204.70.4.77	2.41 ms	71.3	3 × 51.8
8	sprint3-nap.WestOrange.mci.net	204.70.10.226	1.45 ms	24.1	44.7

Table 1: Estimated latencies and bandwidths of the first 8 links of the SDSC path, with nominal bandwidths reported by the network service provider.

`pathchar`'s point of view it is impossible to distinguish an OC-3 from and OC-1 or an OC-12. Nevertheless, `pathchar` can estimate the bandwidth of the OC-1, although in this case it is not very accurate (38% too high).

The next link is a DS3, which is also made up of a number of parallel pipes (28 DS1s, which are themselves made up of 24 DS0s). In this case, though, the packet is broken into parallel bitstreams, so that the bandwidth experienced by a single packet should be the nominal bandwith of the link, 44.7 Mb/s. Again, the estimate is not very accurate (46% too low).

2.3 Noise

These examples demonstrate the fundamental difficulty of `pathchar`—the higher the bandwidth, the harder it is to measure. The estimated bandwidth is based on the slope of the rtt curve, which captures the difference between the rtt of the largest and smallest packets. The higher the bandwidth, the smaller this difference.

Looking at the 6th and 7th hops, we see that the *difference* between the rtt for the largest and smallest packets is small compared to the noise in the data. The estimated slopes of the lines are 9.614 and 9.726 μs/B, so the difference between them is only 0.112 μs/B, meaning that the difference between the rtt of the smallest and largest packets is only about 150 μs. By comparison, the largest outlier in Figure 3 is 175 μs from the fitted line.

This problem is compounded by the need to subtract adjacent estimates. In general, the calculated difference between estimated values is less accurate than the values themselves. As a result, the accuracy of the estimated slopes needs to be considerably better than the accuracy required for the estimated bandwidths.

To make matters worse, minimum filtering amplifies measurement error by selecting extreme values. We tried several techniques to mitigate this effect:

- Instead of using the minimum from each Sample, we tried other summary statistics, including the 2nd percentile.
- For the curve-fit, we used iteratively-weighted least squares (IWLS) to dilute the effect of outliers.
- We tried to model the distribution of queue times and estimate the minimum round trip time as a parameter of the distribution model.

None of these techniques reliably improved `pathchar`'s bandwidth estimates. This problem is the focus of our ongoing work.

3 Modeling errors

Like all models, `pathchar`'s network model omits many details of the real world. This section discusses some of them and their effect on the estimates.

3.1 Omissions that affect latency

One omission we have already mentioned is the assumption that the size of the error packet is negligible. As a result the latency estimates are a little too high. Given the actual size of the error packet, it is easy to adjust the estimated latency by subtracting $error_size/estimated_bandwidth$. In most cases this adjustment is insignificant.

A second omission is the assumption that forwarding time is negligible. Actually, since `pathchar`'s estimates are based on the difference between measurements, forwarding time drops out as long as it is the same for all routers. Only the variation in forwarding time from router-to-router will affect the estimated latency.

A related issue is the possibility that forwarding time varies over time. Possible causes of variation include MAC delays and time the router spends processing updates. These transient delays will be eliminated by minimum filtering as long as they occur infrequently.

A more serious problem is the possibility that the return path is not the same as the outgoing path. In this case, it is no longer true that the two latencies in Equation 1 are the same. As a result, the estimated latencies can be wildly wrong. Unfortunately, `pathchar` can neither detect nor deal with this possibility.

Most of these factors do not affect `pathchar`'s bandwidth estimates because they do not discriminate between large and small packets.

3.2 Omissions that affect bandwidth

One problem that we have already mentioned is the existence of links that are made up of parallel pipes. If these links put a whole packet into one pipe, rather than dividing it up, they will appear to have the bandwidth of a single pipe.

Another problem is caused by probes that exceed the MTU of a link. The MTU is the Maximum Transfer Unit, the largest packet that can be sent without being fragmented. A probe that exceeds the MTU will be fragmented into some number of smaller packets, and an error packet will be sent as soon as the *first* packet arrives at the nth router. The resulting rtt curve levels off at the MTU, distorting the curve-fit and tending to make the bandwidth estimate too high.

Setting the don't fragment flag of the outgoing packets does not help, since it does not cause the packet to exceed the MTU of a link; rather, it causes an ICMP Unreachable Error (Fragmentation Required) at the first link whose MTU is exceeded [5].

3.3 Route alternation

A final factor that is omitted from `pathchar`'s model, and that causes significant problems, is route alternation. Route alternation is the tendency for the path between two hosts to change over time, typically oscillating over a small set of possibilities. Paxson has shown that route alternation is common on the Internet [4]. Rapidly-oscillating routing can make it impossible for `pathchar` to generate useful estimates, not only for the oscillating link but also for the links beyond it. The reason is that once `pathchar` has characterized the $(n-1)$th link of a path, it assumes that all measurements of the nth link follow the same first $n-1$ links. If that is not the case then subtracting the characteristics of the two links becomes meaningless.

Fortunately, Paxson found that the majority (91%) of Internet routes persist for hours, long enough for `pathchar` to make its measurements. Unfortunately, there are a significant number of routes that alternate more quickly, and these routes do create problems. For example, the 9th link of the path from Colby to SDSC alternates among at least three routes. The current version of `pathchar` tries to characterize the first route it finds, and rejects probes that follow another route. If it happens to encounter a low-probability route first, it can waste a lot of probes. Also, if the route it characterizes is not the shortest route, estimates for subsequent links might be inaccurate. This problem affected the SDSC dataset, and is the reason we restricted the discussion to the first 8 links.

4 Interval estimates

It is always dangerous to generate estimates without confidence intervals, and especially so for `pathchar`, since some estimates are much more accurate than others.

Although the linear least squares fit described in Section 2.1 produces error estimates for the parameters, these estimates have no statistical meaning in the context of the estimated network characteristics. The least-squares fit only "knows" about the SORTT for each packet size; it doesn't take into account how many probes were sent or what the probability was of traversing the path without incurring any queue delays. Developing a statistical model that uses this information to generate an error bound would be formidable.

An alternative is to use a technique from non-parametric statistics: divide a large sample into smaller samples, and look at the variation in the estimated parameters among the subsamples.

Using the same data from the previous section, we divided each Sample into two, containing the even- and odd-numbered probes. Each subsample contains 32 probes at each of 45 sizes for each link. When we calculate the difference between adjacent Links, we get four estimates for each parameter by taking

- the difference between the even samples,
- the difference between the odd samples,
- the difference between the evens from one and the odds from the other, and vice versa.

link	low bw (Mb/s)	high bw (Mb/s)	nominal bw (Mb/s)
1	9.68	9.78	10
2	9.85	10.1	10
3	10.0	10.2	10
4	9.65	9.86	10
5	9.96	10.4	10
6	1.42	1.43	1.536
7	56.5	72.4	3×51.8
8	23.5	36.6	44.7

Table 2: Intervals for the estimated bandwidth of the first 8 links of the SDSC dataset.

Taking the largest and smallest estimatess of the four, we form an interval for the estimated characteristics.

Table 2 shows the intervals for the bandwidth of the first 8 links of the path. The intervals for the first 5 links are narrow (less than 4% of the estimated value), indicating that the estimated value is consistent, if not perfectly accurate. In some cases, but not all, the interval contains the nominal bandwidth of the link. The estimated bandwidth of the T1 (link 6) is very consistent, although somewhat lower than the nominal bandwidth (1.536 Mb/s).

The difficult links, as before, are the OC-3 and the DS3. The good news is that the intervals correctly indicate that these estimates are not accurate. The width of the interval is 31% of the nominal value in one case and 29% in the other. The bad news is that in both cases the interval does not contain the nominal bandwidth. Thus, while these intervals give some measure of the uncertainty of the estimates, they are still somewhat optimistic.

In Section 6 we use these intervals as a criterion for convergence, in order to choose the number of probes at each link adaptively.

5 More sizes vs. bigger samples

With a given number of probes, it is not obvious whether it is better to make measurements of a wide variety of packet sizes or to make a large number of measurements at each size. The advantage of a large number of measurements is that there is a better chance of observing the minimum possible rtt. The advantage of making measurements over many packet sizes is that more data points are used for curve-fitting (and fewer are discarded by minimum-filtering).

To evaluate this tradeoff, we collected a large dataset and divided it into subsamples in various ways, using different packet sizes and different numbers of measurements. The MINT dataset contains 512 probes at each of 64 sizes, from 88 to 1348 bytes. The measurements were made between Tuesday 7 July, 1998 at 10:39 EDT and Wednesday 8 July, 1998 at 03:59 EDT. Table 3 shows the links along this path and the estimated bandwidth for each.

We divided the data into 16 subsets along several different axes. In one case we use all 64 packets sizes, but divide the probes into 16 sets (by taking every 16th probe). In another case we use only 4 different sizes, but keep all 512 probes at each size. The headings in Table 4 show the various ways to partition the dataset; for example, 4×512 means 4 sizes and 512 probes per size.

For each data subset, we estimated the bandwidth for each link, and then calculated the relative error for the 16

link	host	address	latency est. (μs)	bandwidth est. (Mb/s)	nominal bandwidth
1	hub.colby.edu	137.146.194.17	323	9.68	10
2	network.colby.edu	137.146.238.194	219	10.1	10
3	router5.colby.edu	137.146.238.210	195	10.0	10
4	router1.colby.edu	137.146.112.209	115	9.9	10
5	general.colby.edu	137.146.112.132	209	9.8	10
6	network.colby.edu	137.146.238.146	218	10.1	10
7	bordercore2.cht.mci.net	166.48.64.25	3.68 ms	1.43	1.536
8	bordercore1.Boston.mci.net	166.48.60.1	108	56.7	3 × 51.8
9	mint.Boston.mci.net	166.48.60.18	3.37 ms	1.43	1.536
10	emerald.mint.net	204.254.98.9	78	8.8	10

Table 3: Estimated characteristics of the MINT path.

subsets, where the relative error is the absolute difference between the estimated and nominal bandwidth, divided by the nominal bandwidth. Each entry in the table is the average of the 16 errors. For the first link in the path, the 4×512 dataset yields estimates that are off by 3.1% on average; the 64×32 dataset is off by 2.8% on average.

The remaining links are similar—the datasets that use more packet sizes consistently yield better estimates than the ones that have more probes at each size. The discrepancy is particularly significant for the 8th link, which is an OC-3. The errors for this link are bigger across the board than for the other links, but the best estimates come from using many packet sizes.

The last link is omitted from this table because of what seems to be a bug in the alpha version of `pathchar`. For this link, `pathchar` used a different set of packet sizes than it used for the other links. This discrepancy does not affect the other calculations, but it does break our system for splitting the data into subsets.

We performed a similar experiment with the SDSC path and found similar results—more sizes is better than more probes per size.

5.1 Consistency

This experiment also gives us an opportunity to evaluate the consistency of `pathchar`'s measurements. Using the 32×64 dataset, we examined the 16 estimates for each link. In general, estimates for low bandwidth links are more accurate and more consistent. For example, the 16 estimates for the second Ethernet are:

9.82	9.93	9.94	9.95	9.98	9.99	10.0	10.0
10.1	10.1	10.1	10.1	10.1	10.1	10.2	10.2

The largest and smallest values differ by less than 4%.

For the OC-3 (link 8) the estimates are:

31.4	33.8	38.9	39.2	43.1	44.0	46.6	51.9
65.1	83.5	86.6	99.8	111	115	117	148

The median of these values, 58.5 Mb/s is close to the nominal value of the OC-1 pipes that make up the OC-3, which is 51.8 Mb/s. Nevertheless, the range is large, indicating that a single estimate with this sample size is unreliable.

The measurements of the T1s (links 7 and 9) are interesting because they are very consistent, within 4% of 1.43 Mb/s, but not accurate. The nominal bandwidth is 1.536 Mb/s.

Our conclusion is that these links are not providing the nominal bandwidth, but we have no explanation for this behavior. Framing bits were a likely suspect, but they are already accounted for in the nominal bandwidth (the line rate is 1.544 Mb/s). In Table 4, where we compare the accuracy of different sample sizes, we use 1.43 Mb/s as the nominal value for the two T1 links.

link	4×512	8×256	16×128	32×64	64×32
1	3.1%	3.0%	2.9%	2.9%	**2.8%**
2	1.8	2.1	1.1	**0.83**	0.85
3	2.9	1.7	1.8	1.6	**1.4**
4	4.6	3.7	2.5	2.0	**1.3**
5	4.5	3.2	3.0	2.4	**1.9**
6	4.9	3.8	3.7	**2.4**	2.9
7*	1.2	1.0	0.88	**0.47**	0.49
8	252	69	564	60	**42**
9*	2.7	1.5	1.8	**0.77**	1.1

Table 4: Mean relative error of the bandwidth estimated for the MINT dataset. Each column represents a different way of partitioning the data. The heading 4×512 indicates a dataset with 4 packet sizes and 512 probes at each size. The lowest value in each row appears in bold. The asterisks indicate that these errors are based on the median estimate rather than the nominal bandwidth (see Section 5.1).

5.2 Latency

So far we have said little about the accuracy of the latency estimates, since end-to-end latency is the sum of several factors and we do not have nominal values for the links we measured.

On the other hand, there are a number of links that appear in both sample paths. We can use these measurements to assess the consistency of the estimates, if not their accuracy. In each case, the two measurements differ by less than 5%.

In both paths the latency of the first link is higher than the latency of the other Ethernets. The excess, about 100 μs, might be the time it take the router to generate the ICMP error packet. For the subsequent links, this time is eliminated by subtraction of adjacent latencies.

link	number of probes	estimated and (nominal) bandwidth	converge criterion
1	296	9.6 (10)	0.5 %
2	296	10.1 (10)	1.7 %
3	296	9.6 (10)	2.3 %
4	296	10.5 (10)	3.5 %
5	888	9.8 (10)	4.1 %
6	888	9.7 (10)	2.2 %
7	888	1.43 (1.54)	1.6 %
8	37835	63.3 (51.8)	100 %
9	4440	1.44 (1.54)	6.5 %
10	37839	8.7 (10)	40 %

Table 5: Adaptive data collection, MINT dataset.

6 Adaptive data collection

One thing that is clear from the example paths is that some links require more measurements than others to generate an accurate characterization. For the first link in the MINT dataset (a 10 Mb/s Ethernet), an 8×16 sample is sufficient to estimate bandwidth within 3%. Even across several hops, an 8×16 sample is enough to characterize the T1 link to within 2%. But for fast links with many intervening hops, like the OC-3, even the 64×512 dataset (32,768 probes) is off by 9%.

The current release of `pathchar` uses the same number of probes for each link of the path. In practice, this means that if the sample size is large enough to characterize the most difficult link, many probes are being wasted on the easy links. We set out to design an adaptive data collection system that uses only as many probes as necessary.

In the following experiments, we simulated adaptive data collection by collecting a large dataset (74×512) in the usual way (the same number of probes for all links), and then using an adaptively-selected subset of the data. We start out with a small number of samples for each link, and then make "new" measurements by incrementally including additional data. If the estimated characteristics of a link seem to have converged, we move on to the next link and discard the remaining measurements. We hope that this process will discover the minimum number of samples required to reach a given accuracy.

This simulation distorts the time interval between probes somewhat, but we do not expect that to affect the results, because on the time scale of minutes queue delays are noisy and uncorrelated (Section 7). It should not matter whether there is a delay between the measurement of one link and the next.

6.1 Detecting convergence

In Section 4 we described a simple way to calculate an interval for the estimated characteristics. We divide the sample into two halves and estimate the parameters for each subsample. Since each estimate is based on the difference between adjacent links, we can use the two subsamples to generate four estimates for each link. The **convergence criterion** is the range of the four estimates divided by the smallest estimate; in other words, the difference between the smallest and largest, expressed as a percentage of the smallest. We consider a link converged if this criterion is below 10%.

link	number of probes	estimated and (nominal) bandwidth	converge criterion
1	540	9.82 (10)	1.6 %
2	180	9.07 (10)	1.6 %
3	180	10.76 (10)	3.4 %
4	180	10.10 (10)	6.1 %
5	180	10.73 (10)	6.7 %
6	180	1.24 (1.54)	5.2 %
7	2809	-11.13 (51.8)	46.1 %
8	2829	24.15 (44.7)	55.4 %

Table 6: Adaptive data collection, SDSC dataset.

link	number of probes	estimated and (nominal) bandwidth	converge criterion
1	540	9.8 (10)	1.6 %
2	180	9.1 (10)	1.6 %
3	180	10.8 (10)	3.4 %
4	180	10.1 (10)	6.1 %
5	180	10.7 (10)	6.7 %
6	2871	1.42 (1.54)	0.67 %
7	2809	71.3 (51.8)	28.3 %
8	2829	24.2 (44.7)	55.4 %

Table 7: Retroactive data collection, SDSC dataset.

Based on the result from Section 5, we collected measurements at a large number of packet sizes (74 sizes ranging from from 88 to 1548). We started with 2 probes at each packet size and added new measurements at each size until the estimates converged or we exhausted the supply of previously-collected data (256 evens and 256 odds). Each time we added new measurements, we doubled their number, starting with 2, 4, 8, etc.

Table 5 shows the number of data points used to characterize the links in the MINT dataset. For each link, the table shows the estimated bandwidth and the width of the interval.

In many cases, two probes per packet size is sufficient for the link to converge. Thus, the total number of samples for many links is 296 (2 subsets, 2 probes per size, 74 sizes) or 888 (6 probes per size). The 8th and 10th links fail to converge even using all the available data.

6.2 Retroactive data collection

Although adaptive data collection works well on the MINT dataset, it fails for the SDSC dataset (Table 6). The estimated bandwidth of the 7th link fails to converge even using all the available data, and the best estimate is negative, which is a pretty good indication that something is wrong.

The problem is that the estimate for the 7th link depends on the data for the 6th link, and in this case the 6th link has converged on a value that is close to the nominal value, but not close enough to yield an accurate *difference* between the 6th and 7th links (see Section 2.1).

A simple solution is to collect additional measurements of the 6th link at the same time we are working on the 7th. For every 4 samples of the current link, we make one additional

link	n	μ_1 (ms)	μ_2
1	37843	0.21	2.9
2	37832	0.29	3.5
3	37850	0.40	4.3
4	37846	0.55	4.7
5	37842	0.79	5.5
6	37838	0.93	6.3
7	37808	11.6	1020
8	37793	8.6	917
9	37346	29.9	1690
10	37777	8.8	374

Table 8: Estimated moments of the queue delays for the MINT dataset.

measurement of the prior link. Table 7 shows the result of this technique, which we call **retroactive data collection**.

In this case we have to use all the available data for the 6th link, which yields an improved bandwidth estimate with a very narrow interval. Improving the characterization of the 6th link eliminates the problematic negative bandwidth of the 7th. The characterizations of the 7th and 8th links are still inaccurate, but they are no worse than in Table 1.

Retroactive data collection slightly improves the estimates from the MINT dataset. In both cases it requires more probes than simple adaptive collection, but still far fewer than would be required if we collected the same number of probes for each link.

6.3 Directed adaptive collection

We tried an alternate form of adaptive data collection that reduces the number of probes required to achieve a given accuracy. The basic idea is to use the residuals of the curve-fitting step to direct data collection. If the SORTT for a given packet size is above the fitted curve, we assume that we have not observed the minimum possible rtt, and collect more measurements at that size. Conversely, if a SORTT is below the fitted curve we can avoid wasting measurements on it.

This technique is effective—it reduces the number of measurements needed without reducing accuracy—but it might make it more difficult to make a strong claim about the statistical validity of the result, since an initial error might lead to a self-fulfilling prophesy, at least in theory. In practice, that does not seem to be a problem.

7 Deconvolution

In addition to inferring the physical characteristics of a link, `pathchar` tries to characterize the distribution of queue delays, reporting the mean and hinge (ratio of the interquartile distance to the median).

Predicting the delay suffered by an individual packet is difficult, because conditions change quickly and even recent measurements seldom predict the future [6]. But knowing the distribution of queue delays might make it possible to predict aggregate performance.

Unfortunately, we seldom have the luxury of observing the queue delays of single link; except for the first link, all our observation are the sum of many queue delays (see Figure 1 and Equation 1). In order to discern the delays imposed by a given link, we have to deconvolve the observed distributions of adjacent links.

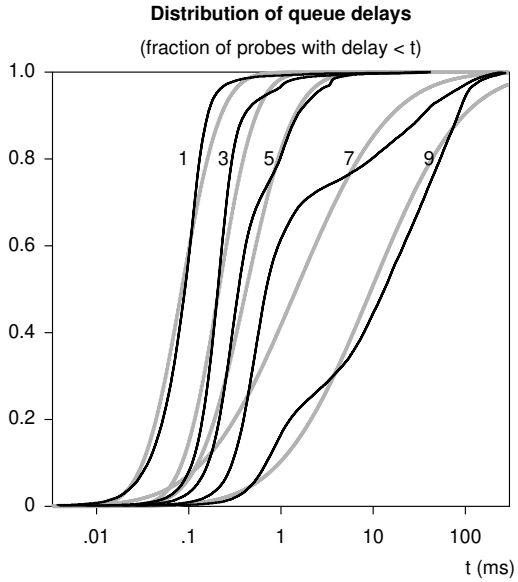

Figure 6: Distributions of cumulative queue delays from the MINT dataset, shown on a log scale along with fitted log-normal distributions.

If x is the total queue delay of the first $n-1$ links of a path, and y is the queue delay of the nth link, then clearly the total delay of the first n links is the sum of x and y, which we call z. If we knew x and z, therefore, it would be trivial to find y. But because of the way `pathchar` works, we can only measure x or z, never both for the same packet.

Nevertheless, by making many measurements of x and z, we can estimate the distribution of each, and by deconvolution we can infer the distribution of y. The simplest form of deconvolution is the subtraction of moments. This is the technique `pathchar` uses.

If we know F_X, the distribution of X, and F_Y, the distribution of Y, we can calculate the first three moments of F_Z, the distribution of the sum $Z = X + Y$, just by adding the moments of F_X and F_Y. This is true for all distributions and does not depend on the independence of X and Y. Using this property, `pathchar` estimates the moments of F_Y by subtracting the moments of F_X from the moments of F_Z.

Since negative queue delays are impossible, the first three moments must increase monotonically from one link to the next. For example, the mean queue delay to the nth node cannot be less than the mean queue delay to the $(n-1)$th node.

Table 8 shows the first two moments of the queue delays from the MINT dataset. For each link, we fit a curve to the SORTTs and estimate the queue delay by subtracting the fitted time from each measurement. After discarding a small number of negative values (about 1%), we have about 38000 measurements per link.

For the first 7 links, the moments increase monotonically, but then things break down. The mean queue delay for the 8th link is less than the mean for the 7th link, and the 10th is less than the 9th. There are several possible explanations:

- The estimated moments are not the true moments of the underlying distributions. Estimating the moments of distributions with long tails is notoriously difficult; a few outlying values can have a large influence on the

	Mean			Lognormal mean			Median		
link	early	late	(% diff)	early	late	(% diff)	early	late	(% diff)
1	0.196	0.199	(+1.5)	0.107	0.114	(+6.7)	0.083	0.091	(+9.2)
2	0.293	0.275	(-6.2)	0.181	0.175	(-3.7)	0.141	0.141	(-0.5)
3	0.346	0.428	(+23.7)	0.241	0.289	(+19.8)	0.184	0.200	(+8.8)
4	0.566	0.502	(-11.4)	0.433	0.384	(-11.4)	0.278	0.245	(-11.8)
5	0.748	0.799	(+6.8)	0.634	0.668	(+5.3)	0.330	0.326	(-1.1)
6	1.13	0.732	(-35.1)	1.03	0.587	(-42.9)	0.434	0.324	(-25.4)
7	15.8	7.52	(-52.3)	18.1	3.41	(-81.1)	0.890	0.576	(-35.2)
8	7.51	9.85	(+31.1)	3.26	5.24	(+60.8)	0.713	0.759	(+6.4)
9	37.1	23.5	(-36.6)	64.6	35.0	(-45.8)	21.9	8.47	(-61.2)
10	10.7	5.84	(-45.2)	10.5	4.41	(-58.1)	1.52	0.905	(-40.3)

Table 9: Estimated moments of the queue delays for the MINT dataset.

calculated estimates.

- The distributions of queue delays are not stationary. Traffic conditions may have changed between the measurement of the $(n-1)$th and the nth links. `pathchar` collects all the measurements from each link before moving on to the next, which means that there are significant delays (15–45 minutes for this dataset) between each set of measurements. It is possible for traffic conditions to change during this interval.

Both of these explanations turn out to be true. The next two sections discuss them in more detail.

7.1 Moment estimation

The queue delay distributions can be modeled reasonably well by the lognormal distribution [2]. Figure 6 shows five representative links from the MINT dataset. In each case, it is clear that the observed distribution is not strictly lognormal; nevertheless, the general shape of these curves indicates that the lognormal model is a reasonable choice.

In an empirical study of wide-area TCP connections, Paxson found a variety of other characteristics that fit the lognormal model [3]. We do not know whether there is an underlying mechanism that relates these observations, or whether they only reflect the versatility of the lognormal model.

Using the lognormal model to estimate the moments of the distributions yields very different results, especially for the second moment. A likely explanation is that the conventional estimates are more sensitive to outliers; a small number of large queue delays ($\sim$ 300 ms) can raise the mean significantly and have a huge effect on the second moment. Because the lognormal model is less sensitive to outliers, it does a better job of describing these distributions.

Nevertheless, the alternate moments have the same problem as the conventional estimates: the moments sometimes decrease from link to link in a way that is impossible if the distribution of queue delays is stationary. We conclude that the distributions are, in fact, changing while `pathchar` is running.

7.2 Non-stationarity

To get an idea of how the distribution of queue delays varies, we divided the MINT dataset into two subsets, containing the first 18000 samples from each link and the second 18000.

The entire dataset took over 5 hours to collect, so the elapsed time between each early subset and the corresponding late subset is on the order of tens of minutes. Table 9 shows the summary statistics of the two subsets, including the mean, the first moment of the lognormal model, and the median.

For many of the links, the parameters of the distribution change by more than 30%, which implies that recent history is not a very good predictor of even the near future. In the longer term the predictions are likely to be worse. As a result, it may not be worthwhile for `pathchar` to characterize the distribution of queue times at all.

8 Conclusions

Based on our evaluation of the alpha version of `pathchar` we have found:

- Estimating link latencies is relatively easy, since latencies in wide-area networks are large compared to `pathchar`'s measurement errors.
- Estimating bandwidths is harder, because the difference in round-trip time between the largest packet and the smallest is small compared to the measurement errors. The higher the bandwidth, the more difficult it is to estimate.
- The key to getting a good bandwidth estimate is to send enough probes that one of them traverses the entire path without incurring any queue delays. As the length of the path increases, the number of probes required increases quickly.
- A single busy link, by imposing queue delays on the majority of probes, makes it difficult to resolve the characteristics of links on the other side. On the other hand, slow links are not necessarily a barrier to accurate measurement, as long as their performance is consistent.
- `pathchar`'s attempt to characterize the distribution of queue times for individual links may be in vain, since network traffic conditions change significantly while `pathchar` is running.

Of the new techniques we tested, only adaptive data collection seems to work well: it greatly reduces the amount of data required, without affecting the accuracy of the estimated characteristics. None of the techniques we tried significantly improved `pathchar`'s accuracy.

Our experiments have uncovered a few anomalies.

- The measured bandwidth of T1 links is consistently within a few percent of 1.43 Mb/s, when the nominal bandwidth is 1.536 Mb/s.
- The measured bandwidth of the DS3 is consistently too low, with a median value close to half of the nominal bandwidth.

It is not easy to explain these results, and we cannot attribute them to noise. In order for a fast link to masquerade as a slow link, it has to impose delays that are proportional to the packet size. Random noise would not yield estimates that are always too low, or as consistent as the T1 measurements.

Acknowledgements

Many thanks to k claffy at CAIDA, Jason Lavoie at MINT, Greg Miller at MCI, Evi Nemeth and Daniel McRobb at CAIDA, Rich Wolski at the University of Tennessee, Mark Crovella at Boston University, the SIGMETRICS reviewers and program committee, the SIGCOMM reviewers and program committee, and Van Jacobson for writing `pathchar`.

References

[1] Van Jacobson. pathchar—a tool to infer characteristics of Internet paths. Presented at the Mathematical Sciences Research Institute(MSRI); slides available from `ftp://ftp.ee.lbl.gov/pathchar/`, April 1997.

[2] Norman L. Johnson and Samuel Kotz. *Continuous univariate distributions-1.* John Wiley & Sons, 1970.

[3] Vern Paxson. Empirically-derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking*, 2(4):316–336, August 1994.

[4] Vern Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(3):601–615, 1996.

[5] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: the Protocols.* Addison-Wesley, 1994.

[6] Rich Wolski. Dynamically forecasting network performance using the network weather service. *Cluster Computing*, 1998. Also available from `http://www.cs.ucsd.edu/users/rich/publications.html`.