DETECTING AND MEASURING ASYMMETRIC LINKS IN AN IP NETWORK

Wenyu Jiang,
Columbia University
wenyu@cs.columbia.edu

Timothy F. Williams
Network Peripherals Inc.
twilliams@NPINY.com

Abstract

The rapid growth of Internet traffic has made it important to pinpoint network bottlenecks. Measuring link bandwidths can help identifying such bottlenecks. Existing bandwidth measurement tools such as Pathchar, Bing and Bprobe assume symmetric links. But as ADSL lines, cable modems, satellite links, and 56K modems [1] gain popularity, it becomes necessary to consider asymmetric links. We present an algorithm that can measure each hop's link bandwidth in both directions in an IP network. Hence it is trivial to detect link asymmetry. We performed several experiments to validate our algorithm. We also discuss factors that can adversely affect the precision and/or correctness of bandwidth measurement, and suggest some possible solutions.
Keywords: network measurement, bandwidth measurement, asymmetric link

1 Introduction

It is valuable to identify bottlenecks in a network for congestion control and performance tuning. When a user sees a slow Web access, as in Figure 1, he would want to know: is it due to queuing congestion at some locations, or the slow speed of link A, B, C, D, or some other links in the Internet?

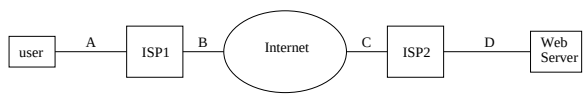


Figure 1: Slow connection to a Web server

Link speed consists of raw bandwidth (its full capacity) and available bandwidth (after sharing). We focus on measuring raw bandwidth.

Various bandwidth estimation tools exist today. Pathchar [Jac97] and Bing [Bey95] measure *per hop* bandwidth by analyzing a packet's Round Trip Time (RTT) linearity with respect to its size. Bprobe [CC96] and Tcpanaly [Pax97] measures *bottleneck* link bandwidth based on packet-pair technique. They assume links to be symmetric. Recent deployment of ADSL, cable modems, satellite links and 56K modems, however, is changing this assumption.

The ability to measure asymmetrical links is useful in discovering performance bottlenecks, validating service guarantees in terms of bandwidth, etc.

In the next section, we present an algorithm that measures *per hop* link bandwidth in both directions. Section 3 shows real word experiments we performed. Section 4 discusses some factors that may alter the result. Before conclusion, we compare the existing tools in Related Work.

2 The Measurement Algorithm

Our method is also based on RTT linearity analysis, like Pathchar and Bing. It is a mathematical combination of the technique used by Pathchar and the one used by Bing. We start with an symmetric link algorithm, which turns out to be the same as Bing, but we were not aware of Bing when we developed it. Then, we extend it to consider asymmetric links.

2.1 Algorithm 1: Symmetric case

The most common way to measure RTT is send an ICMP echo [Pos81a] packet and wait for a reply, as in the *ping* program. That's what we will use here.

Let T_i denote the RTT for an ICMP echo/reply packet of B bytes, v_i denote link speed between hop $i-1$ and hop i, where i is the number of hops between sender (hop 0) and receiver (hop i).

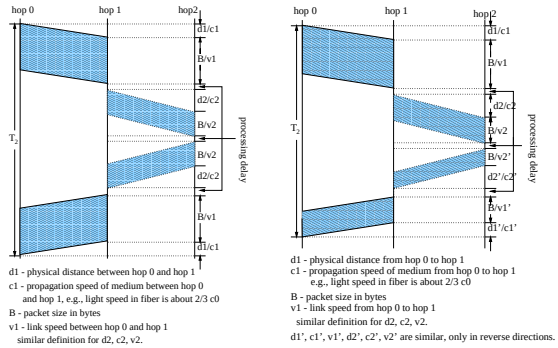
Then for $i = 2$, as in Figure 2 (a),

$$T_2 = \frac{2B}{v_1} + \frac{2B}{v_2} + const$$

where *const* is a constant that depends on i. It is the sum of propagation delay, processing delay, and queuing delay. Without loss of generality, we assume the queuing delay is 0.

So,
$$T_i = \sum_{j=1}^{i} \frac{2B}{v_j} + const = B \sum_{j=1}^{i} \frac{2}{v_j} + const$$

[1] A 56K modem downloads at 56 kb/s, and uploads at 33.6 kb/s. In fact, even 14.4k modems are often asymmetric, as we shall see in section 3.1



(a) symmetric case (b) asymmetric case

Figure 2: RTT for 2 Hops

Let k_i denote slope of linear function $T_i(B)$, then

$$k_i = \sum_{j=1}^{i} \frac{2}{v_j} \Longrightarrow k_i - k_{i-1} = \frac{2}{v_i}$$
$$\Longrightarrow v_i = \frac{2}{k_i - k_{i-1}} \qquad (1)$$

Formula (1) computes symmetric link bandwidths. To obtain k_i, we send packets of different sizes, creating sample points for function $T_i(B)$. Then, a linear regression is all that's needed to obtain k_i.

2.2 Effects of Asymmetric Links

Formula (1) does not work with asymmetric links. To derive the correct formula, let us first look at the Figure 2 (b), and see what T_i becomes.

When $i = 2$, $\quad T_2 = \frac{B}{v_1} + \frac{B}{v_1'} + \frac{B}{v_2} + \frac{B}{v_2'} + const$

Generally, $\quad T_i = B\sum_{j=1}^{i}(\frac{1}{v_j} + \frac{1}{v_j'}) + const$

$$\Rightarrow k_i = \sum_{j=1}^{i}(\frac{1}{v_j} + \frac{1}{v_j'}) \Rightarrow k_i - k_{i-1} = \frac{1}{v_i} + \frac{1}{v_i'}$$

Formula (1) becomes $\quad \hat{v_i} = \frac{2}{k_i - k_{i-1}} = \frac{2}{\frac{1}{v_i} + \frac{1}{v_i'}} \qquad (2)$

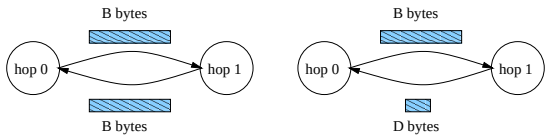
So $\hat{v_i}$ is the *harmonic mean* of the real v_i and v_i'.

2.3 Algorithm 2 - The final algorithm

Although Formula (2) does not give v_i, if we know either v_i or v_i', the other is immediately solvable. $\hat{v_i} \neq v_i$ because a packet's RTT is the sum of transit time in both directions ($RTT = T_{to} + T_{from}$), which depends on link speed in both directions and packet size B, as in Figure 3 (a). If we can force the reply packet to have a constant size D, as in Figure 3 (b), then T_{from} no longer depends on B but only on v_i', for any i. Since v_i' are constant for any i, T_{from} is also a constant. So v_i is directly solvable.

In an IP network, the most obvious case for Figure 3 (b) is the ICMP time-exceeded message [Pos81a]. Therefore our final algorithm is:

1. send an echo packet to ith router with TTL $= i$, this generates a normal reply;



(a) B bytes to/from (b) B bytes to, D from

Figure 3: RTT components

2. send another echo packet to $i+1$th router with TTL $= i$, at ith router, it will bounce back as an ICMP time-exceeded message [2], call this RTT T_i'.

Then from Figure 4, for $i = 2$, we have

$T_2 = \frac{B}{v_1} + \frac{D}{v_1'} + \frac{B}{v_2} + \frac{D}{v_2'} + const = \frac{B}{v_1} + \frac{B}{v_2} + const.$

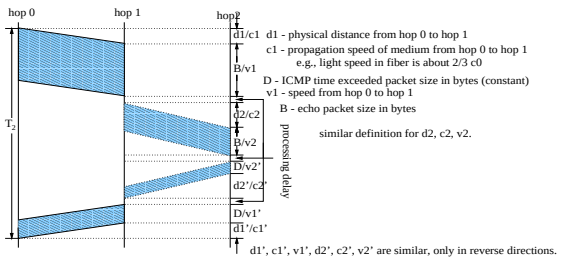


Figure 4: 2 Hop, asymmetric with time exceeded

$$T_i' = \sum_{j=1}^{i}(\frac{B}{v_j} + \frac{D}{v_j'}) + const = B\sum_{j=1}^{i}\frac{1}{v_j} + const \qquad (3)$$

As we predicted, T_i' is not dependent upon link speed in the returning direction.

$$k_i' = \sum_{j=1}^{i}\frac{1}{v_j} \Longrightarrow k_i' - k_{i-1}' = \frac{1}{v_j}$$
$$\Longrightarrow v_j = \frac{1}{k_i' - k_{i-1}'} \qquad (4)$$

Similarly, we have

$$k_i - k_{i-1} = \frac{1}{v_j} + \frac{1}{v_j'} = k_i' - k_{i-1}' + \frac{1}{v_j'}$$
$$\Longrightarrow v_j' = \frac{1}{k_i - k_{i-1} - (k_i' - k_{i-1}')} \qquad (5)$$

Therefore, Formula (4) and (5) compute the speed of ith link in both directions.

2.4 Measurment using Algorithm 2

To get v_i and v_i', we send packets of different sizes (B), and measure their RTTs T_i, T_i'. To eliminate queuing delay noise, we use the simplest approach by choosing a min for a given B and i. Then a linear regression of $T_i(B)$ and $T_i'(B)$ gives us v_i and v_i'.

Pathchar uses *ensemble min* [Jac97], but it is not explained. We assume it means taking all the values

[2] When ith hop is an end host, there would be no $i+1$th router. We can send a UDP packet with an unused destination port number, forcing an ICMP port unreachable message [Pos81a], which is also constant size. This was first used by *traceroute*, and also in Pathchar.

that are close to the min. We find that for a stable (low delay variation) network such as ethernet, *ensemble min* shows better results, but in satellite links, where delay variation is low in percentage but high in absolute value, the plain min seems to be better. A detailed analysis and critique of different statistical methods is not the focus of this paper.

2.5 Effects of Hidden Hops

When $\exists > 1$ links between hop $i-1$ and hop i, Formula (4) and (5) will be incorrect, because they increase RTTs proportionally to B. Examples of hidden hops are ethernet bridges, switches, VSAT hubs [Tan96], ATM network underneath IP, etc. Let's say there are m links inside an IP hop i, and the link speed is v_{ij} for $j = 1, ..., m$. Then, our algorithm will compute v_i as:

$$\hat{v_i} = (\sum_{j=1}^{m} \frac{1}{v_{ij}})^{-1} \qquad (6)$$

Apparently $\hat{v_i} < v_i$, meaning an under-estimation. Say a 10 Mb/s ethernet and a 100 Mb/s fast ethernet is inside an IP hop, $\hat{v_i}$ will be $(\frac{1}{10}+\frac{1}{100})^{-1} = 9.09$ Mb/s. If both were 10 Mb/s ethernets, $\hat{v_i}$ would be $(\frac{1}{10}+\frac{1}{10})^{-1} = 5$ Mb/s. An extra hidden link of the same speed will cut the result by half.

3 Real World Experiments

We validated Algorithm 2 with real world tests on 4 types of links. All the k's have the unit msec/byte.

Unfortunately, due to practical reasons, we do not know the actual speeds of many links we tested. In that case, they are marked with a "?".

1. Dial-up modem mixed with Ethernet

We did the first test using a Windows 95 laptop computer with 2 network interfaces: a 14.4 kb/s dialup modem [3], and a 10 Mb/s ethernet card. As in Figure 5 (a) (b) (c), machine A, B, C are the test initiator, laptop, and dialup server, respectively. All of them are on the same ethernet, but the routing tables on both the laptop and the test initiator were properly modified to control the routes. The modem link layer protocol is PPP. Here $i = 2$.

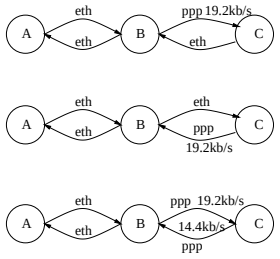
(1) Outgoing: PPP Dialup, Incoming: Ethernet

k_1	k_1'	k_2	k_2'
0.004369	0.002171	0.520233	0.5160

$$\Longrightarrow v_2 = \frac{1}{k_2'-k_1'} = \frac{1}{0.513829}$$
$$= 1.946 \text{ kB/s } [4] \Leftrightarrow 19.46 \text{ kb/s}$$

[3] Win95 Dialup Networking can only specify 9.6 kb/s and 19.2 kb/s. This results in a download speed of 19.2 kb/s and an upload link of 14.4 kb/s, which is both interesting and useful to our experiment.

[4] kB is kilo-Byte. In asynchronous transmission, typical setting is 1 start bit, 1 stop bit, no parity, i.e., 10 signaling bits per data byte. So 1.946 kB/s is equivalent to 19.46 kb/s



(a) Dial-up outgoing and ethernet incoming

(b) Ethernet outgoing and dial-up incoming

(c) Dial-up outgoing and incoming

Figure 5: Test Network for PPP Dial-up

$$\Longrightarrow v_2' = \frac{1}{k_2-k_1-(k_2'-k_1')} = \frac{1}{0.515864-0.513829}$$
$$= \frac{1}{0.002035} = 491.4kB/s = 3.93 \text{ Mb/s}$$

	v_2	v_2'
measured	1.946 kB/s	3.93 Mb/s
actual	1.92 kB/s	3.67 Mb/s ?

(2) Outgoing: Ethernet, Incoming: PPP Dialup

k_1	k_1'	k_2	k_2'
0.004347	0.002193	0.715773	0.003932

	v_2	v_2'
measured	4.6 Mb/s	1.41 kB/s
actual	3.67 Mb/s ?	1.44 kB/s

(3) Outgoing: PPP Dialup, Incoming: PPP Dialup

k_1	k_1'	k_2	k_2'
0.004348	0.002176	1.2274	0.5160

	v_2	v_2'
measured	1.946 kB/s	1.41 kB/s
actual	1.92 kB/s ?	1.44 kB/s

Note: Our measured ethernet speed is not 10 Mb/s because we are on a switched ethernet, which distorts the result as explained in section 2.5. Also, with the same signal to noise ratio, a slow PPP link can inject much higher noise in time measurement to an ethernet, that's why we got 2 quite different results: 3.9 Mb/s and 4.6 Mb/s. This effect is often called noise amplification [5].

2. Cable Modem

Host C below has a cable modem connection (download/upload is from host C's perspective).

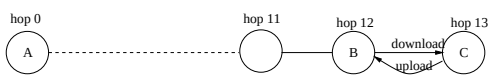


Figure 6: Asymmetric link with Cable modem

k_{12}	k_{12}'	k_{13}	k_{13}'
0.0083	0.00325	0.0264	0.0060

[5] The ideal value should be about 3.67 Mb/s, which is v_1 and v_1', because in our test it is known that path (A,B) and path (B,C) should pass through the same switches.

	v_{13}	v'_{13}
measured	2.91 Mb/s	521 kb/s
actual	3 Mb/s ?	512 kb/s ?

Unfortunately we don't know its actual speed. It may be 3Mb/s download and 512 kb/s upload, both peak speed, a typical offering from Cable providers.

We have also made an interesting comparison with Bing. We ran a Bing test (http://web.cnam.fr/cgi-bin/traceroute.html) on the cable modem host. Since Bing uses essentially our Algorithm 1, according to Formula (2), Bing will measure $\hat{v_i}$, the *harmonic average* of v_i and v'_i.

Therefore, we predict Bing's result to be $(\frac{1}{521} + \frac{1}{2910})^{-1} = 884$ kb/s for cable modem. The actual Bing test ranges from 716 to 992 kb/s, averaging 843kb/s, well in expectation.

3. ADSL Line

The path is same as Figure 6, except $i = 15$.

k_{14}	k'_{14}	k_{15}	k'_{15}
0.00748	0.0038	0.1021	0.028

	v_{15}	v'_{15}
measured	331 kb/s	113.6 kb/s
actual	320 kb/s ?	128 kb/s ?

We don't know its real speed. From the result, it could be 320 kb/s download and 128 kb/s upload.

We repeated an similar Bing comparison test on the ADSL host. our prediction is $(\frac{1}{113.6} + \frac{1}{331})^{-1} = 170$ kb/s. Actual test result ranges from 164 to 173 kb/s, averaging 168 kb/s, a nice match.

4. Satellite Link

The path here is similar to Figure 6, except $i = 12$. There is an 8 Mb/s satellite link from USA to Belgium. Unfortunately, we don't know the speed of returning link, could be a 155Mb/s OC-3 ATM link, or a 22Mb/s partial ATM link. Usually test results for very fast links are untrustable, since the RTT difference between a min-sized and max-sized packet would be only a few micro-seconds, a 10 ms jitter of queuing delay would easily mess up the results.

k_{11}	k'_{11}	k_{12}	k'_{12}
0.005919	0.002582	0.007833	0.00444

	v_{12}	v'_{12}
measured	4.3 Mb/s	143 Mb/s
actual	4.44 Mb/s ?	155 Mb/s ?

Measured v_{12} is 4.3 Mb/s, only about half of 8 Mb/s. Again there must be some hidden bridge(s). If we conjecture a 10 Mb/s ethernet is in between, since routers are more likely to have an ethernet interface as opposed to a satellite one, our predicted result as shown in next section, would be $1/(1/8 + 1/10) = 4.44$ Mb/s, which is very close to 4.3 Mb/s.

$v'_{12} = 143$ Mb/s may indicate a 155 Mb/s ATM link, but since there is enough noise to distort the measurement, we can only conclude it is a fast link.

4 Measurement Issues

Many factors interfere with bandwidth measurement, many of which were mentioned in Pathchar. Here we classify them into Random distortions, Deterministic distortions, and Potential fatal problems.

1. Random distortions

• *Noise* - Also mentioned in Pathchar, it usually comes from queueing delay variation, which can easily exceed 100ms. For a 100Mb/s link, even a 100 μs noise can be devastating.

In addition, as seen in Section 3.1, it is hard to measure a fast link (say ethernet) behind a slow link (say PPP), due to noise amplification.

2. Deterministic distortions

• *Hidden hops* - As discussed in 2.5, they distort measurements: $\hat{v_i} = (\sum_{j=1}^{m} \frac{1}{v_{ij}})^{-1}$

However, it remains a challenge to discover them. Bottleneck bandwidth measurement do not suffer from hidden hops, because it measures (pipe-lined) throughput. So an approach to solving this problem may be sending packets in a pipe-lined fashion. However, our initial analysis (not shown here) suggests this approach works only if link speeds are monotonically decreasing from left to right.

• *Signaling speed vs. Data speed* - An OC-3 ATM link has a signaling speed of 155 Mb/s, but its data speed is 48/53 of 155 Mb/s (48 bytes data, 53 bytes cell). Our Algorithm will measure only data speed.

In Bing [Bey95] it suggests: send packets at small incremental sizes, to detect the small jump in RTT when an extra cell is used. But on an OC-3 link, the jump is 53*8/155 M = 2.74 μs, barely noticeable and easily overwhelmed by noise.

An extreme case is SLIP/PPP dialup link. Nearly all asynchronous links use 1 start bit, 1 stop bit with no parity, therefore a 14.4kb/s modem will have a 1.44kB/s data speed. Even worse, we cannot use the above trick for ATM here, because the cell here is simply 1 byte. Therefore we must interpret our results with great care, especially when the measured speed is slow (on the order of 10kb/s).

• *Peculiarity of SLIP and PPP* - Both have the notion of escape bytes [Ste93]: PPP by default escapes all bytes less than 0x20, 0x01 becomes 0x7d21.

This can distort measurement greatly. Initially our program fills the ICMP echo packet with mostly bytes of 0, and our results for the PPP test was about half of the real speed. We also tested Pathchar

with our PPP link which has 19.2 kb/s download speed. Pathchar consistently reported around 11.3 kb/s average.

3. Potential fatal Problems

• *Asymmetry of paths* - Our algorithm can solve asymmetric links, but it assumes the network path is symmetric. If the returning path is completely different, the result would clearly be unreliable.

It is hard to detect such asymmetry. IP's record route option [Pos81b] only records 9 routers. Traceroute shows only outgoing path. We think there is no simple solution for it.

• *Multiple paths* - This was also mentioned in Pathchar, when there are more than one way to reach the destination and come back, the measurement again is clearly questionable.

One possible way to detect this is by analyzing RTT distribution, if multiple paths do exist, it will show up as several densely distributed regression lines. However, if there are too many paths available, this method will not scale.

5 Related Work, Comparisons

1. Tcpanaly [Pax97] calculates *bottleneck* link speed by measuring One-way Transit Time (OTT) as opposed to RTT. It is based on packet pair technique, which measures only the spacing of OTT and does not require a globally synchronized clock.
Pros: (1) OTT is not affected by asymmetric links (2) OTT measurement has less noise, as it is not affected by queueing delays in the returning direction.
Cons: (1) measures only one direction at a time (2) To obtain OTT, you must run monitoring tools (e.g.,tcpdump) at both ends, which requires extra human intervention. (3) If we want to extend Tcpanaly to measure *per hop* bandwidth, then monitoring tools must run on all intermediate routers, an arrangement hard to get by in a multi-ISP Internet.

2. Pathchar uses essentially step 2 of our Algorithm 2, so it can correctly measure outgoing link speed. To measure both directions, you must run Pathchar from host A to B, and then from B to A. This is less demanding than Tcpanaly, but still more restrictive than our technique.

3. Bing measures *per hop* link bandwidth, using essentially our Algorithm 1. We have already compared Algorithm 2's results with actual and predicted Bing's results, as in Section 3.2 and 3.3.

4. Bprobe measures *bottleneck* link speed also using packet-pair technique, but it analyzes spacing of RTTs instead of OTTs as in Tcpanaly. We haven't done any testing using Bprobe because it is only ported to SGI Irix, which we don't have access to. However, we can predict that Bprobe will measure the thinner of the two thinnest links in each direction, due to its packet pair technique.

6 Conclusions

We describe an algorithm that can measure *per hop* asymmetric raw link bandwidth. It requires only one end to do the measurement. It works by sending ICMP echo packets with varying TTLs and sizes, to cause ICMP reply packets and other times ICMP time-exceeded packets. Then we perform linear regression on RTTs and packet sizes, one for echo/reply case, the other for echo/time-exceeded case. This produces 2 linear equations with 2 unknowns, giving a solution to link speeds in both directions.

Our approach is in essence a combination of Bing and Pathchar, but it was developed independently.

Link bandwidth estimation is not simple. Many practical issues affecting the result. Bridges, switches, hubs, peculiarity of PPP/SLIP links, ATM links, path instability and asymmetry, are all obstacles to good results. We mentioned some solutions for part of the issues, but there remains a lot of challenge to achieve good measurement.

References

[Bey95] Pierre Beyssac. Bing - Bandwidth pING. 1995. At http://web.cnam.fr/reseau/bing.html.

[CC96] Robert L. Carter and Mark E. Crovella. Measuring bottleneck link in packet-switched networks. Technical Report BU-CS-96-006, Boston University, 1996.

[Jac97] Van Jacobson. pathchar – a tool to infer characteristics of Internet paths. In *Mathematical Sciences Research Institute (MSRI)*, April 1997. software at ftp://ftp.ee.lbl.gov/pathchar/.

[Pax97] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California at Berkeley, May 1997.

[Pos81a] J. Postel. Internet control message protocol. Request for Comments (Standard) 792, Internet Engineering Task Force, September 1981.

[Pos81b] J. Postel. Internet protocol. Request for Comments (Standard) 791, Internet Engineering Task Force, September 1981.

[Ste93] W. R. Stevens. *TCP/IP Illustrated: The Protocols*, volume 1. Addison-Wesley, Dec 1993.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.