LinkWidth: A Method to measure Link Capacity and Available Bandwidth using Single-End Probes.

Sambuddho Chakravarty
Department of Computer Science
Columbia University
sc2516@cs.columbia.edu

Angelos Stavrou
Department of Computer Science
Columbia University
angel@cs.columbia.edu

Angelos D. Keromytis
Department of Computer Science
Columbia University
angelos@cs.columbia.edu

We introduce LinkWidth, a method for estimating capacity and available bandwidth using single-end controlled TCP packet probes. To estimate capacity, we generate a train of TCP RST packets "sandwiched" between two TCP SYN packets. Capacity is obtained by end-to-end packet dispersion of the received TCP RST/ACK packets corresponding to the TCP SYN packets. Our technique is significantly different from the rest of the packet-pair-based measurement techniques, such as *CapProbe, pathchar* and *pathrate,* because the long packet trains minimize errors due to bursty cross-traffic. TCP RST packets do not generate additional ICMP replies preventing cross-traffic interference with our probes. In addition, we use TCP packets for all our probes to prevent some types of QoS-related traffic shaping from affecting our measurements.

We extend the Train of Packet Pairs technique to approximate the available link capacity. We use pairs of TCP packets with variable intra-pair delays and sizes. This is the first attempt to implement this technique using single-end TCP probes, tested on a wide range of real networks with variable cross-traffic. We compare our prototype with *pathchirp* and *pathload,* which require control of both ends, and demonstrate that in most cases our method gives approximately the same results.

1. INTRODUCTION

The scale and complexity of the Internet makes the task of understanding and analyzing its properties extremely difficult. The diffuse style of administration and operation means that even such basic information as topology or link capacity is spread across multiple entities with little incentive to share it. This knowledge, however, is necessary for the design and development of better management tools, communication protocols, and mechanisms that are network-related. As a concrete example, there is little information available on the distribution of access-link capacities and available bandwidth for end-users (*e.g.,* how fast can the average user send/receive data from the network). Although our interest is motivated by our work on distributed denial of service defense mechanisms [6, 11], such information is more generally useful. For example, if end users can estimate this information for different network paths then, given a choice between different mirrors, the user can choose the one with the best access characteristics.

This recognition has spurred research in this area, with several efforts focusing on estimating available capacity between two hosts [4, 8, 12, 9, 10], and measuring the capacity of an arbitrary network link respectively [7, 4, 10, 3]. Most of these tools (eg. PathLoad , PathChirp etc.) use a technique called Self Loading of Periodic Streams. In most cases it requires co-ordiation from both ends of a path / link. The sender attempts to increase the sending rate to reach a point wherein the one way delay (OWD) of the packets goes on increasing. This indicates a state wherein the sender is attempting to pump out packets faster, but has utilised the entire avaialble capacity and the packets are now being queued at the intermediate forwarding hosts. The faster the packets are queued the more is the OWD expericed at the reciever.In general, the tools for estimating installed capacity allow for *single-ended control, i.e.,* they can be used from a single host to measure link or path capacity. However, all previous tools on estimating available capacity require *two-ended control*, *i.e.,* that the two endpoints of a path collaborate in the measurement. Thus, the utility of such tools is limited, given our goal to conduct large-scale and/or opportunistic measurements of arbitrary links and paths.

We present *LinkWidth*, a novel technique for measuring of installed and available capacity for IP paths. In contrast to previous work, which uses *Self-Loading of Periodic Streams* (SLoPS) as the basic measurement methodology, we use a hybrid approach that uses both SLoPS and a *Train of Packet-Pair* (TOPP) approach to measure both installed capacity and available capacity using single-ended control.

The Train of Packet Pairs is quite similar to SLoPS. Infact rather than measuring the OWD of packets arriving the receiver, TOPP goes of increasing the sending rate the sender to reach a point wherein it has exceeded the available capacity and and the packets get queued at the intermediate forwarding hosts and the receiver rate is fixed at a constant value (the available capacity). Thus, while the sender is still sending within the available capacity, the receiving rate is close to the sending rate (assuiming ideal case of no cross packets resulting in probe packets to be queued in routers). Thus the ratio of sending rate to the receiving rate is close to unity.In the process of increasing the sending rate, when it increases to a value above the available capacity the the receiving rate gets fixed to available capacity and the ratio of sending to receiving rate grows linearly.

In case of LinkWidth , we use a somewhat hybrid technique. The sending rate is not increased linearly as in case of TOPP but we use a binary search to converge to a value of sending rate which is within the end-to-end installed capacity and at the same time use the ratio of received dispersion to sending dispersion to converge with certain convergence parameter. This process is more detail in

later sections of this paper.

In contrast to previous work, LinkWidth does not require two-side control to accurately estimate available capacity, and is thus suitable for use in both a large-scale survey of Internet access links and for opportunistic network-path available-capacity estimation to arbitrary destinations.

The basic idea of LinkWidth though derived from TOPP doesn't vary the sending rate lineaerly. It varies it between 0 and C (the installed capacity) to determine the value of avaliable capacity . The search algorithm is a simple binary search over the entire range. The selection of mid-point and range limits contingent upon convergence parameters which again is a function of the ratio of the end-to-end dispersion of the received and sent train. This is described in detail in later sections. We evaluate LinkWidth using a variety of tests, both over controlled environments (with several topologies and degrees of cross-traffic burstiness) and on the Internet. We compare the results obtained via LinkWidth with the ground truth (when known) and against the measurements obtained from other tools. We thereby conclude emperically that for most cases we are accurately able to determine the capacity and closely reach the bottleneck available capacity when compred other known tools.

In summary, the contributions of our work are:

- A new, hybrid technique for single-ended estimation of installed capacity and available capacity of network links and paths, without collaboration from the network provider(s) or the remote endpoint.
- An implementation of LinkWidth in the form of a tool for Linux 2.6.
- An experimental evaluation and validation of LinkWidth using a battery of tests in both controlled network environments and on the Internet.

Outline of the paper. The remainder of this paper is organized as follows. Section 2 outlines the description of differnet terminologies that that we use. Thereby in section 3 we describe how we propose a new technique based on some existing techniques for measuring capacity and available capacity. It discusses it lenght how the technique is being used in LinkWidth to converge to values of capacity and available capacity. Section 4 describes the various experiments we conducted and the results that we obtained from those experiments. Finally we conclude the paper in section 5 with a summary which outlines the technique LinkWidth employs to measure the capacity and available capacity and how it fares when pitted against some of the existing tools/techniques.

2. CAPACITY DEFINITIONS

2.0.1 Bottleneck Link Capacity

Bottleneck link capacity represents the data transmission rate of the slowest forwarding element of a network path between a sender and a receiver. However, of interest to an application is then the available capacity on the path, i.e. the capacity the application can achieve when sharing path with cross traffic.

The available capacity for an application depends on the behavior of many factors including the application, the protocols, the characteristics of the cross traffic and the configuration routers. The variation in cross traffic makes measurement of available cpacity an even more elusive goal to measure. **The bottleneck capacity is also the end-to-end path capacity along an Internet path between any two nodes on an Internet path.**

2.0.2 Available Capacity

Available Capacity as defined by Melander, Bjorkman et. al [1] is the unutilized part of the network end-to-end capacity (which may or may not be the unutilized part of the bottleneck link). This is explained later as the hidden bottleneck problem in the next section. Also identified are three metrics that can be used to characterize the available capacity of a path: *proportional share, the surplus and the protocol dependent available capacity*.

2.0.3 Proportional Share

Proportional Share capacity is defined in terms of the capacity available to a sender depending upon the kind of protocol it is using. It is defined depending upon the Information Rate that is achieved by the sender with all the protocol overheads such as handshake / signalling packets, headers , ACK / NAK packets and connection termination packets.

2.0.4 Surplus Available Capacity

Surplus Share available capacity is defined as the capacity which is achievable by an application , such that it is just enough not to cause cross traffic packet to be dropped. It defines traffic to be rather non-aggressive such that the cross traffic packets aren't dropped. This situation is very difficult to accurately guarantee and achieve a situation in which the packets sending rate is of the sender is just about enough of what is available to it without causing the cross traffic packets to be dropped. This situation arises when the cross traffic packets arent dropped by the sender's packets. The cross traffic behavior is not know. Non uniformly bursty cross traffic such as SSH and WWW may not always allow the sender to maintain a steady throughput , thereby causing it to constantly change its sending rate. Thus a moving average of the sender's sending rate over a time window can be used to thought of as a crude approximation to the available capacity of the sender. To the best of the authors' knowledge there is no such tool available which can accurate measure the surplus available capacity of an end to end IP path between two hosts.

2.0.5 Available Bottleneck Link Capacity

Available Bottleneck Link Capacity is the measureable unutilized link capacity. For an IP path connecting two hosts, the end-to-end available bottleneck link capacity is defined as the maximum achievable capacity by any of the hosts. This is always less than the installed maximum link capacity; infact it is bounded above by the maximum available link capacity. This definition may seem (and also is) ambiguous. For an already congested link, a new flow of of probe packets from a host could result in some other flow to loose packets. Moreover by pushing faster the packets of this new flow could itself be lost. This may be a situation wherein the sender has reached the available bottleneck link capacity. This explaination seems enough to describe that limiting point of the sending rate which we define as the available bottleneck link capacity is not very well defined and rather fuzzy. It depends on the queue lenght of the queues of the intermeditate forwarding routers, the applications and their burstinesses, the congestedness of the network and other related factors. For the sake of convenience , we use the term available bottleneck link capacity and available capacity interchangably for the remainder of this paper.

2.1 Existing Tools and Techniques

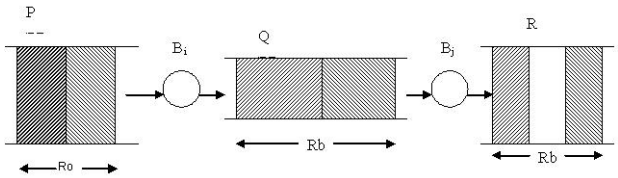

Figure 1: Basic Packet Pair Dispersion Technique

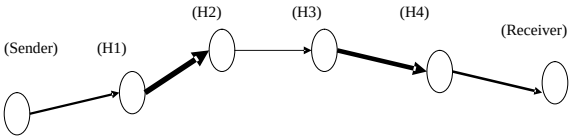

Figure 2: Multihop Path with Varying Link Capacities

2.1.1 End-to-End Capacity Estimation

Quite a few techniques have been proposed for measurement of end-to-end capacity of an Internet path. Most techniques however make use of the old principles of packet pair and packet train dispersion.

2.1.2 Packet Pair Dispersion Technique

The packet pair dispersion technique involves sending of back-to-back packet pair of appropriate size over a multihop path as shown in figure (Figure 1). The basic idea behind packet pair technique is sending two back-to-back packets to the destintion and measuring the time gap between the reception of the last bit of the first packet and the last bit of the second packet at the receiver (also known as the the end-to-end dispersion). The packet length measured in bits divided by this time dispersion is taken as the end-to-end installed capacity of the packet (Figure 1). Perhaps one of the oldest tools and widely known tool to accurately measure the end-to-end capacity is Patchar [2, 4]. (Figure 1) gives us an overview of how end-to-end capacity is measured using packet pair dispersion technique.

In most cases this train would be send over a path with varying link capacities as shown in figure 2. The end-to-end gap of the received packet pair at the destination equals the measure of the dispersion caused by the bottleneck link (cite the initial packet pair dispersion paper). Thus the packet length when divided by the end-to-end dispersion measures the bottleneck capacity.

From figure 2 we see that $(H1 - H2) > (H3 - H4) > (H4 - Receiver) > (Sender - H1) > (H2 - H3)$ Upon reception of the packet at the receiver the packet pair undergoes dispersion (spreads out in time) at the narrow link $(H2 - H3)$. The end - to - end capacity (C) can thus be measured as :

$$C = L/Max(R_1, R_2, ..R_m)$$

(Assuming L to be the length (in bits) of the packet ; measured from the end of the first packet to the end of the last packet. $Max(R_1, R_2, ...R_m)$ gives the maximum dispersion of the packet pair from end-to-end of the entire internet path).

2.1.3 Packet Train Dispersion Technique

Packet Train Dispersion Technique, although very similar to the packet pair dispersion technique uses not a single packet pair but an entire train of packets. The main idea remains the same as that of end-to-end dispersion in the packet pair dispersion technique.

Here the value of dispersion of the packet train is measured from the end of the first packet to the end of the last packet.

$$C = L * (n - 1)/max(R1, R2, ...Rm)$$

Thus packet train has its obvious advantages over the packet pair method. IP , inherently unreliable connectionless protocol, fragments large size packet and may even reorder them. This is especially proven to be true for large sized packets [5]. This results is large gaps being introduced between the packet pairs; which results in capacity underestimation. On the other hand , as explained in [5], smaller packets suffer from the problem that two smaller back-to-back packets result in capacity overestimation. This is particularly the case when the first packet is queued longer than the second packet. Such effects result from the behavior of routers and varying sizes and flow rates of the cross-traffic case of packet trains. The train being very large (large number of packets) , variations in measured end-to-end gap of the received train is averaged out for large train lengths.

2.2 Available Capacity Estimation

Techniques for measuring available capacity have arisen from the commonly known techniques for capacity estimation i.e. packet train and packet pair techniques. Available capacity measurement starts with the sender sending packets with an initial dispersion and the receiver receiving these packet streams and measuring the received dispersion. As long as the received dispersion R_m equals R_o, the sending dispersion, we can say that we are sending within the available capacity as there is no queuing delay experienced by the packet train and hence the packet train is being sent within the available capacity. When $R_m > R_o$ we know we have exceeded the sending rate to more than the available capacity A as decribed in [1]. Once again, as pointed out earlier in [5] these queuing delays could be results of packets sizes as well. Thus a packet train of large number of closely spaced packets could average out the the effect of under and overestimation

Since the average available capacity changes over time it is important to quickly converge to a particular value. This is especially true for applications that use available capacity measurements to adapt their transmission rate. In contrast, the capacity of a path typically remains constant for long time intervals, e.g., until routing changes like distance vector broadcasts or link state updates occur. Therefore the installed capacity of a path does not need to be measured as quickly as the available capacity.

The most commonly used available capacity measurement techniques are Self-Loading of Periodic Streams (SLoPS) and Train of Packet Pairs (TOPP).

2.2.1 Self - Loading of Periodic Streams (SLoPS)

SLoPS is a rather recent methodology for measuring end-to-end available capacity [10]. The source sends a number of equal-sized packets (a periodic packet stream) to the receiver at a certain rate R. If the stream rate R is greater than the path's available capacity A, the stream will cause a short term overload at the queue of the narrow link. One way delays (OWDs) of the probing packets will keep increasing as each packet of the stream queues up at the tight link. On the other hand, if the stream rate, R is lower than the available capacity A, the probe packets will pass through the narrow path without causing an increasing backlog at the tight link and their one way delays will not increase. Figure 3 shows variation of

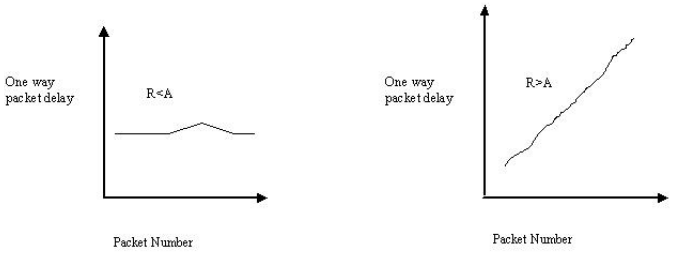

Figure 3: Variation of One Way Packet Delay versus Number of Packets Sent

one way packet delay against packets sent (packet numbers)

Intuitively speaking , when using SLoPS the sender attempts to bring the stream rate R close to the available capacity A, following an iterative algorithm. The sender probes the path with successive packet trains of different rates, while the receiver notifies the sender about the one-way delay trend of each stream.

Next we describe another very similar technique **Train of Packet Pairs (TOPP)**. The idea behind TOPP is taken as a concept for designing LinkWidth.

2.2.2 Train of Packet Pair (TOPP)

Train of Packet Pair (TOPP) sends trains packet pairs at gradually increasing rates from the source to the destination. Assume a packet pair from a source to destination with an initial dispersion of R_o. The probe packets have size of L bytes and thus the offered rate of the packet pair is $O = L/R_o$. If $O \leq A$, TOPP assumes that the packet pair will arrive at the receiver with the same rate it had at the sender, i.e. $M = O$ .If $O > A$, the measured rate at the receiver will be $M < O$. It has been pointed emprically in [1] that the recived rate M is a fraction of the sending rate O when the sending rate O exceeds the available capacity A. Typically $M = (O/(O + X)) * L$

The very idea of TOPP doesn't vary much from SLoPS. We use neither SLoPS nor TOPP verbatim. Our technique proposes a modification of the orginal TOPP wherein a binary search technique is used to converge to an estimated value of available capacity depending upon two convergence parameters θ and ϵ.

Thus,

$$M = (O/(O + X)) * L$$
$$or \;\; O/M = (O + X)/C$$
$$or \;\; O/M = (O + C - A)/C$$
$$\boxed{or \;\; O/M = O/C + (1 - A/C)}$$

This gives a linear variation of O/M versus O in the case where $O > A$. This is illustrated in figure (Figure 4)

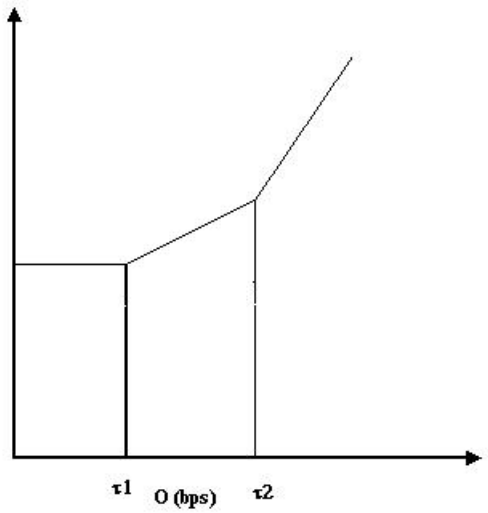

Figure 4: Variation of One Way Packet Delay versus Number of Packets Sent

Here the graph displays two possible available bottleneck link capacities($\tau 1$ and $\tau 2$), i.e. the least available bottleneck capacity and the next value of the available bottleneck capacity values. A major constraint of this method is that it assumes that in case of a multi-hop path; the values of C_i and or A_i must be unique all along the path.

3. PROPOSED METHODOLOGY

We propose capacity estimation using a modified version of the existing packet train-dispersion / recursive packet train method. This section starts with a description of the Recursive Packet Train (RPT) method [3] and then explains how we have modified it to incorporate a more accurate procedure to give more accurate results. Our implementation, which we call LinkWidth, involves modification of the existing technique.

3.1 Pathneck (Using Recursive Packet Trains)

Pathneck proposed by Ninigin Hu et. al., used Recursive Packet Train (RPT) technique to locate bottleneck links and measure the end-to-end installed bottleneck capacity. The recursive packet train uses a train of back to back UDP packet (called load packets) which are appended and prepended by another train of UDP packets called measurement packets. Figure 5 describes the arrangement of packets in the pathneck arrangement.

Evident from the arrangement of packets in RPT, the TTL values of the head and tail measurement packets would decrement to zero at each hop of the train. This would give back ICMP TTL Expired packets back to the sender. The time dispersion of two received TTL expired at the source from the same forwarding host would give a good approximation of the dispersion of the packet train from end-to-end as measured at the destination.

3.2 The TCP version of RPT

We modified the RPT method originally proposed by Ninigin Hu et. al [3] to incorporate TCP SYN packets in place of UDP/ICMP packets as was described in the previous section. We use TCP SYN packets being sent to a TCP port on which most operating systems under most circustances have no service running and thus result in TCP RST+ACK packets being sent back as reply.

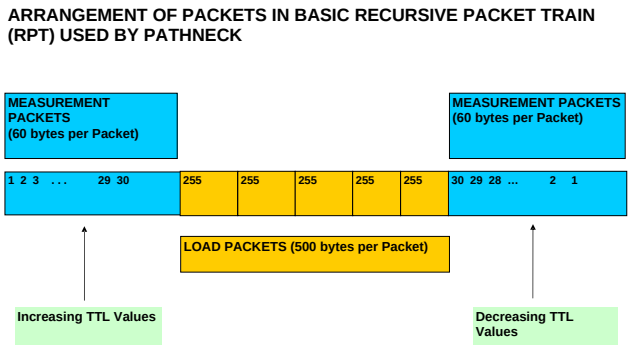

Figure 5: Arrangement of packets in RPT Pathneck

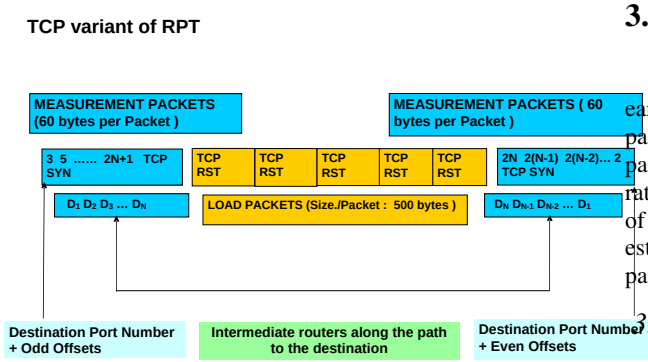

Figure 6: TCP based variant of RPT

From 6 , we see that each of the consecutive SYN segments in the arrangement is sent to each of the consecutive routers / forwarding hosts along the path to the destination. Starting with a certain base port number BASE-PORT, each of the consecutive packets are sent to even destination port numbers BASE-PORT+2, BASE-PORT+4, BASE-PORT+6, BASE-PORT+$(2 * N)$. Thus , packet sent to BASE-PORT+2 is sent to router 1 , BASE-PORT+4 is sent to router 2 , BASE-PORT+6 is sent to router 3 and so on to BASE-PORT+$(2 * N)$ is sent to the destination hop. This arrangement forms the head measurement packets; followed by the TCP RST load packets; and finally the tail measurement packets where the packets with destination port numbers odd port numbers BASE-PORT+2 $* N + 1$, BASE-PORT+2 $* (N - 1) + 1$, ... , BASE-PORT+5, BASE-PORT+3, being sent to the destination hosts D_N,...D_2,D_1 respectively. The reason for using such port numbering is to match up each of the head measurement packet's TCP RST+ACK reply (from even TCP port number) with each of the tail measurement packet's TCP RST+ACK reply packet(from the next consecutive odd TCP port number). Thus the measured time gap between two such consecutive TCP RST+ACK packet gives the end-to-end dispersion of the entire train measured for that particular router.

There are two inherent advantages of the above arrangement:-

1. Most of the single ended controlled methods for measuring the capacity and / or available capacity make use of ICMP TTL expired packets. Many of the commercially available routers give very low priority to ICMP packet generation / forwarding resulting is large end-to-end gaps being inserted between the ICMP TTL expired packets. This leads to capacity underestimation. We avoid, as much as possible, such a situation using TCP packets.

2. The load packets used here are TCP RST packets which don't result in the ICMP Destination Unreachable packets which used to be generated in case of older UDP/ ICMP based techniques such as Pathneck and Capprobe. This avoids reverse cross traffic for the forward probe traffic.

3.3 TCP Based Train of Packet Pairs (TOPP) - Available Capacity Estimation

A certain variation of the Train of Packet Pair method, discribed earlier, is used to measure the available capacity of an end-to-end IP path. A single ended control technique, requires the sender to send packets at varying sending rates and to measure the varing receiving rates to converge to a value of available capacity. The arrangement of packet train is quite similar to that used for installed capacity estimation. However the packet sizes for the measuement and load packets remains the same.

3.3.1 Binary Search Procedure Used to Compute the Available Capacity

Based on a capacity / sending rate range O_{MIN} and O_{MAX} , the available capacity converges to a value $O_{MIN} \leq O_i \leq O_{MAX}$. The program uses a binary search method to iteratively split this range to either converge at a mid value O_{MID} for which the absolute value of the size of the range, i.e. $|O_{MAX} - O_{MIN}|$ becomes less than or equal a granularity parameter θ. Thereby the mean value of the range is reported as the available capacity. Otherwise the value of the difference $|O_{MID}/M - 1|$ is compared to a convergence parameter ϵ. If $|O_{MID}/M - 1| > \epsilon$ then our sending rate $O > A$. Thus we set $O_{MAX} = O_{MIN}$ and repeat the experiment. Else if $|O_{MID}/M - 1| < \epsilon$ then we are within the available ca-

pacity and can go on increasing our sending rate further by setting $O_{MIN} = O_{MID}$ and thereby find a new O_{MID} sending repeating our binary search procedure.

The following algorithm summarizes the procedure to compute available capacity

1. Start with a range of sending range 0 bps - C bps the bottleneck capacity (O_{MIN}) = 0 bps and (O_{MAX}) = C bps.
2. If $|O_{MAX} - O_{MIN}| < \theta$ then report $A = (O_{MIN} + O_{MAX})/2$ and stop the search procedure
3. Perform the experiment by sending the train of packet pair with appropriate sending rate and compute $O/M = R_m/R_o$; R_o = end - to - end dispersion of the train at the sender side ; R_m = end - to - end dispersion of the train at the receiver side
4. If $|R_m/R_o - 1| < \epsilon$ (sending rate is within available capacity and we can try sending faster) , set $O_{MIN} = O_{MID}$ and go back to step 2 Else if $|R_m/R_o - 1| > \epsilon$ (we are above the available capacity and hence need to back off) , set $O_{MAX} = O_{MID}$ and go back to step 2

3.4 Implementation Methodology

The methodology used is simple. It involves merging the two techniques described above, namely the **TCP based RPT** derived from a non-TCP method, with the **TCP oriented TOPP implementation.**

LinkWidth is implemented from an existing implementation of **PATHNECK**. UDP / ICMP TTL Expired Packets used for measuring the end-to-end dispersion of the Train of Packet Pair at the destination is replaced with a TCP based implemented. The head measurement packets , the load packets and the tail measurement packets are such as that described earlier (Figure 6). The head measurement packets are so arranged such that the first head packet and the last tail measurement packet are destined to the first hop in the path. The second head and the penultimate tail packet are destined to the second host and so on till we reach the case in which the last head packet and the first tail packet are destined for the final hop along the path. Also, the destination port numbers assigened according to as described earlier. The head measurement packets are send to even port numbers while the tail measurement packets are send to the next consecutive odd port numbers numbers. This as per the the description of RPT based TCP as described earlier. The entire attempt is to replicate the UDP / ICMP arangement and functionaly using TCP SYN / RST packets.

The following are the input parameters that LinkWidth takes :-

- O_{MIN} = Minimum sending rate (default is 0)
- O_{MAX} = Maximum sending rate (C)
- θ = granularity parameter
- ϵ = convergence parameter
- $payloadSize$ = Payload Size
- $npackets$ = Number of packets per train
- rt = Inter-Pair gap
- Inter-Train delay
- Output file

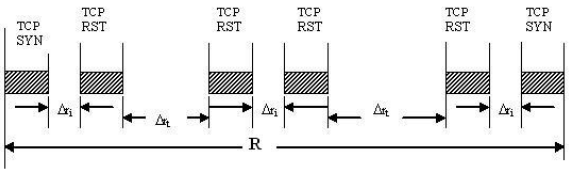

Figure 7: Arrangement of Packets in TOPP

3.5 Arrangement of Packets in the TCP Based TOPP

- R = End-to-end dispersion of the train
- m = Number of packet pairs per train
- δr_i = Intra pair gap for a given sending rate $O_i, 1 \leq i \leq K$, assuming we converge to the available capacity within K trains
- Δr_t = Fixed Inter pair gap for each train
- $packetSendTime$ = time to send (push out) one packet onto the wire.

Thus ,

$$\boxed{R = (m * \Delta r_i) + (m-1) * \Delta r_t + 2 * m * packetSendTime}$$

While the sending rate of the train is within the available capacity, the sending and receiving dispersion are almost the same. When the sending rate O exceeds the true available capacity A of the path, the received dispersion R_m of the train fixes to a certain value, while the sending dispersion, R_o of the train goes decreasing. Thus we have reached a point where the ratio of receieved dispersion to the sending dispersion (R_m/R_o which in turn equals O/M) goes on increasing (Figure 4).

3.6 Computation of Intra - Pair Gaps

From figure (Figure 7) above , the intra - pair gaps can be computed as follows:-
trainLen = The number bytes from the end of the first packet to the end of the last packet (as described by most of the packet train methods).

From the previous section we have the formula for end-to-end gap of the sending train as

$$\boxed{R = (m * \Delta r_i) + (m-1) * \Delta r_t + 2 * m * packetSendTime}$$

$(trainLen*8)/((m-1)*\Delta r_t + m*\Delta r_i + (2*m*packetSendTime)) = O$

Solving for Δr_i , given that the values of the other variables are known and approximating $m - 1$ as m we get

$$\boxed{\Delta r_i = (trainLen * 8)/(O * m) - \Delta r_t - 2 * packetSendTime}$$

Thus, for each train Δr_i value is computed anew. O varies between $O_{MIN}(0)$ and O_{MAX} (C - the bottleneck capacity). The

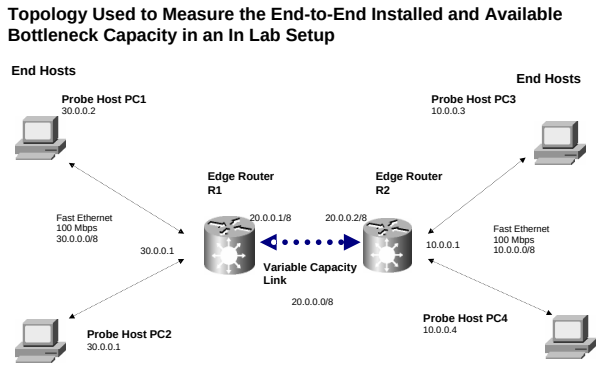

Figure 8: Basic Experimental Testbed

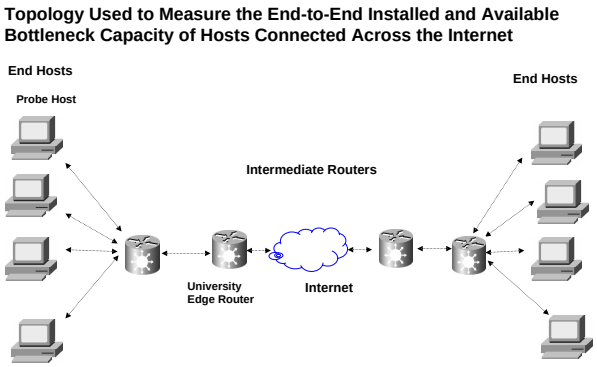

Figure 9: Measuring Capacity of Hosts Connected to the Interent

sending train rate is varied by varying this intra-pair gaps (Δr_i) while the the value of inter-pair gap Δr_t stays fixed (It is accepted as an input from the user program).

4. EXPERIMENTS AND RESULTS

LinkWidth has been tested over an inlab testbed as well as over the Internet.

4.1 Network Topology for Lab Experiments

The following topology (Figure 9) was used for the measuring the capacity available capacity of hosts connected accross the interet.

4.1.1 Test bed configuration

CPUs of PC:Intel Celeron 2.0 Ghz.
OS: Linux 2.4.27
Packages: Knoppix
Network Interface Card: Integrated 10/100 Ethernet Adaptors
MAC: Switched 100Mbps Ethernet

Figure 8 describes how to experimental testbed was setup

4.2 Network Topology for Hosts Conneted Accross the Interent

The following topology (Figure 9) was used for the measuring the capacity available capacity of hosts connected accross the interet.

4.3 Experiements

4.3.1 In Lab Experiments

For the in-lab experiments the topology was setup as desribed in the previous subsection (Figure 8). DITG (Distributed Internet Traffic Generator) tool was used to generate the cross traffic. LinkWidth was used to measure the end-to-end installed and available capacity for both constant bit rate and variable bit rate traffic (both TCP and UDP cross traffic). DITG tool was run on PC1 and PC4. Traffic shaping was used over the interfaces of both the routers.

Thus, the aim was to measure this slow link capacity in case of both no cross traffic and with definite measureable cross traffic and measuring the capacity and available capacity using LinkWidth and thereby comparing these results with what we measure when using other tools.

As evident from the results we are not able to observe the differece in surplus available capacity and the bottleneck available capacity. This is due to the fact that on each physical link there is only one sender which is not always the case for real the internet (in case of the interent ,many users would be connected to the same ISP via one access router's physical link). Thus, tools like IPERF which measure TCP and UDP capacity , and also employ the inherent flow control, somewhat closely approximate the surplus available capacity.In case of LinkWidth the program achieves the actual sending rate by setting the process scheduling policy to SCHED_FIFO (real time FIFO scheduling) with highest scheduling priority. This may result in a sending rate way over the surplus bottleneck capacity as there is no co-ordination / signalling with any other host on the network. The single end controlled sender may thus end up sending packets very aggressively tthereby resulting in the cross traffic packets to be dropped.

4.3.2 Results from In Lab Experiements

It is emperically evident from our results of installed and available capacity are more fine grained than those achieved by other tools that measure installed and avaible capacity.

figure 10 and figure 11 show a comparison of how installed capacity and available bottleneck capacity measurement using LinkWidth fare against the other known techniques. These comparison are for shown or for the results obtined in controlled lab setup.

In the next subsection we describe how we tested LinkWidth to measure installed and available capacity for hosts connected to the internet

Results of In Lab Experimentation with Variation in Cross Traffic Rate

CAPACITY(Mbps)	0 Mbps	2Mbps	6.4 Mbps	8Mbps	20 Mbps	16 Mbps
Cross Traffic: Simple UDP Streams (Constant Bit Rate)						
LinkWidth	67	66	66	66	57	57
Iperf	56	54.8	50	48	37.3	43
Pathrate	300-600/1000	NA	NA	NA	NA	NA
Capprobe	2.13	2.124	2.07	2.09	0.016	2.05
Pathneck	42	38	37	26	18	19
Pathchar	65	35	25	30	NA	NA
Cross Traffic : Simple UDP Streams (Constant Bit Rate)						
AVAILABLE CAPACITY(Mbps)	**0 Mbps**	**2Mbps**	**6.4 Mbps**	**8Mbps**	**20 Mbps**	**16 Mbps**
LinkWidth : $\theta = 1Mbps$	67	61	60	54	49	49
LinkWidth : $\theta = 0.5Mbps$	67	64	57	56	46	51
IGI	65	60	57	54	43	40
PathChirp	55	34	30	37	30	22
PathLoad	46-70	(reports interrupt colasence)				
Cross Traffic : Exponentially Varying Variable Bit Rate						
LinkWidth : $\theta = 1Mbps$	66	60	58	59	52	49
LinkWidth : $\theta = 0.5Mbps$	66	57	57	60	46	53
IGI	65	65	65	45	43	44
PathChirp	48	48	39	39	25	20
PathLoad	46-70	NA	NA	33	NA	NA

4.3.3 Measuring Capacity and Available Capacity of Hosts Connected to the Interent

A bunch of planetlab hosts were probed for capacity and available capacity from a selected probing machine and the results were compared with IPERF which measures the TCP and UDP capacity / throughput [12]
Next the same experiment was repeated accross three hosts connected to the internet. Two of these hosts are physically located in two different Universities while third is a privately owned computer connected to the Interent through a local Earthlink ISP.
The results were compared against IPERF for measuring available capacity (which is somewhat a gross approximation of surplus avaiable capacity). The installed capacity was measured LinkWidth and also using Capprobe [5] and [4].
As evident from the results and also explained in the previous subsections , LinkWidth in many cases successfully sends probes way above the absolute surplus available capacity (uses aggressive sending).This may be resulting in cross traffic packets to be queued / dropped. Thus the measurements are more closer to the bottleneck available capacity than to the surplus available capacity.

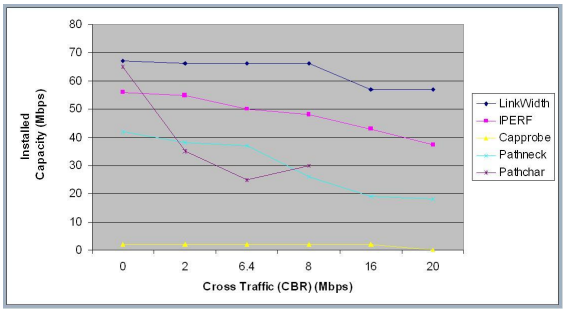

Figure 10: Comparison of Installed Capacity Measurement : LinkWidth vs Other Tools

5. CONCLUSION

LinkWidth accurately measures the installed capacity and gives a measure of the bottleneck available capacity which to quite an extent varies from the results / metrics obtained from other tools (some of which we use to benchmark our tools' performance).

[1] mentions two kinds of available banwidth definition - Surplus Available Capacity (Passively measured to the degree that the cross traffic packets are not dropped due to congestion from the sending packet trains) and the other is the Available Bottleneck Capacity (which is the maximum achievable without regarding the affect on the cross traffic on the path but the traffic sending rate being affected by the traffic reaching the router and being delayed due to queuing at the routers). Thus we see that tools like iperf which require control from both ends use a method similar to SLOPS to converge to a knee point where the sending and receiving rates are just about the same.This is the closest the tool can converge to surplus available capacity. What IPERF measured can be grossly accepted as the surplus available capacity, such measurements using LinkWidth are possible only in controlled / experimental set-up and the cross traffic is not affected as much as in real networks like the Internet.

Measurements of the upper bound of capacity (the installed capacity) , C , gives consistent and accurate results as evident from the results. However the measurement of available capacity (bottleneck available capacity), varies when measured over a network such as the internet where there may most likely be more than one user connected through the same physical link to the same router. Thus the only way we could make sure that the sending train didn't force the cross traffic to get dropped was to send the trains non - aggressively (by introducing large inter train delays). Thus there are circumstances where we send fast enough to swamp the cross traffic thereby leading the cross traffic to be dropped. This happens because our mechanism incorporates no flow control TCP (unlike tools like IPERF which use TCP connections to co-ordinate between the sender and the receiver).

LinkWidth by default would measure the Available Bottleneck

Measuring Installed and Available Capacity of Geographically Dispersed Hosts Connected to the PlanetLab Network

Destination	IPerf(Mbps)	Capacity(Linkwidth)(Mbps)	Available Capacity(Linkwidth)(Mbps)
orbpl1.rutgers.edu	38	90	30
pads21.cs.nthu.edu.tw	2.28	87	3.5
planetlab1.unineuchatel.ch	2.34	96	20
planetlab2.singapore.equinix.planet-lab.org	0.75	46	0.726
bonnie.ibds.uka.de	2.39	91	15
planetlab2.simula.no	2.5	83.6	19
planet4.cs.huji.ac.il	1.25	9	1.27
planet1.att.nodes.planet-lab.org	5	89	19
csplanetlab1.kaist.ac.kr	1.20	87	15
system19.ncl-ext.net	0.8	2	0.73
dschinni.planetlab.extranet.uni-passau.de	0.431	85	0.534
node1.lbnl.nodes.planet-lab.org	6	92	10
planet2.ics.forth.gr	2	96	6
planet.cc.gt.atl.ga.us	7	96	6.5
planet2.winnipeg.canet4.nodes.planet-lab.org	2.19	88	7.5
planet2.learninglab.uni-hannover.de	25	93	0.4
planet2.cs.rochester.edu	25	93	0.4
pl1.ucs.indiana.edu	9	91	13
thu1.6planetlab.edu.cn	0.166	90	30
planetlab-01.naist.jp	20	80	40
planet1.scs.stanford.edu	5	82	45
planet1.calgary.canet4.nodes.planet-lab.org	10	93	73

Measuring Installed Capacity of Geographically Dispersed Hosts Connected to the Internet

Destination Host	Pathchar(Mbps)	Capprobe(Mbps)	Linkwidth(Mbps)
franc.cis.upenn.edu	42	15	80
139.91.70.72	38	ICMP echo requests are filtered	95
64.131.172.50	5	9	4
franc.cis.upenn.edu	42	12	70
139.91.70.72	35	ICMP echo requests are filtered	95
64.131.172.50	6	13	5

Measuring Available Capacity of Geographically Dispersed Hosts Connected to the Internet

Destination Host	Iperf(Mbps)	Linkwidth(Mbps)
franc.cis.upenn.edu	0.3	0.5
139.91.70.72	1.03	0.435
64.131.172.50	2.5	0.344
franc.cis.upenn.edu	20	0.3
139.91.70.72	0.7	0.4
64.131.172.50	2	0.5

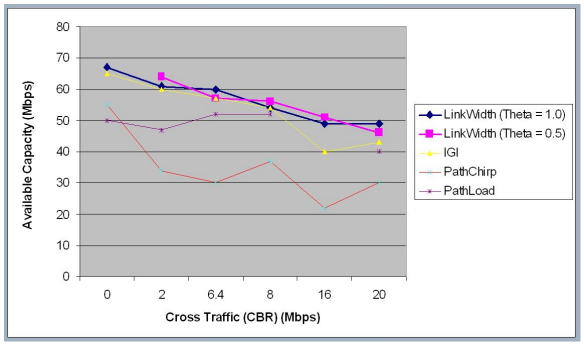

Figure 11: Comparison of Available Capacity Measurement : LinkWidth vs Other Tools

Capacity which doesn't take into consideration what effect it would have on cross-traffic. Rather it relies on current state of the intermediate routers along the path to the destination. Thus the Available Bottleneck Capacity which the sender is able to achieve such that the ratio of receiving to sending rate and the range of the packet sending rate is withing two respective convergence parameter (ϵ and θ respectively.

6. REFERENCES

[1] B.Melander, M.Bjorkman, and P.Gunningberg. New end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *Proceedings of GLOBECOM 2000*, November 2000.

[2] A. Downey. Using pathchar to estimate internet link characteristics. In *Proceedings of ACM SIGCOMM 1999*, August 1999.

[3] N. Hu, L. E. Li, Z.M.Mao, P. Steenkiste, and J. Wang. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. In *Proceedings of ACM SIGCOMM 2004*, August 2004.

[4] V. Jacobson. PATHCHAR. `http://www.caida.org/tools/utilities/others/patchar/`, 1997.

[5] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi. Capprobe: A simple technique to measure path capacity. In *Proceedings of ACM SIGMETRICS 2004*, August 2004.

[6] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM*, pages 61–72, August 2002.

[7] B. Mah. Pchar. `http://www.kitchenlab.org/www/bmah/Software/pchar`, 1997.

[8] M.Jain and C.Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. In *Proceedings of ACM SIGCOMM 2002*, August 2002.

[9] N.Hu and P.Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, August 2003.

[10] R. Prasad, M.Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation:metrics, measurement techniques, and tools. In *Proceedings of IEEE Network 2003*, August 2002.

[11] A. Stavrou and A. Keromytis. Countering DoS Attacks With Stateless Multipath Overlays. In *Proceedings of the* 12^{th} *ACM Conference on Computer and Communications Security (CCS)*, pages 249–259, November 2005.

[12] A. Tirumala, F. Qin, J. Dugan, J. Feguson, and K. Gibbs. IPERF. `http://dast.nlanr.net/projects/Iperf/`, 1997.