# Improved Bandwidth Estimations using the IPv6 Hop-by-Hop Extension Header

M. Crocker, G. Lazarou, J. Picone and J. Baca
Intelligent Electronic Systems
Center for Advanced Vehicular Systems
Mississippi State University
{crocker, glaz, picone, baca}@cavs.msstate.edu

***Abstract*—Accurate measurement of bottleneck and available bandwidths in network paths is a problem that has intrigued researchers for years. Numerous tools and techniques have been developed to determine bottleneck and available bandwidths but none of these methods provide a simple solution that is accurate, efficient, and flexible. Such a solution is simply not practical for IPv4 networks. Fortunately, the next generation Internet Protocol, IP version 6, has the functionality necessary to implement feedback mechanisms to assist in accurate bandwidth estimations. In this paper we present a method to accurately and efficiently determine the network bandwidth in IPv6 networks through the use of a proposed timestamp option for the IPv6 hop-by-hop extension header.**

***Index Terms*—Internet, Protocols, IPv6, Bandwidth, Timestamp**

## I. Introduction

Accurately estimating the bottleneck and/or available bandwidths in a network path is essential for the correct and efficient operation of many Internet applications and protocols as well as network management applications. End-to-end flow control, server selection for downloads and streaming media, peer-to-peer host selection and content delivery, and multicast configuration protocols are a just a few examples where accurate bandwidth estimations are valuable. Accurate bandwidth estimations are also valuable to network designers and administrators to aid in network troubleshooting, capacity provisioning, and traffic engineering.

Knowing the capacity of a network path is valuable in a number of situations, but the best methods for measuring bandwidth are essentially limited due to the nature of the network. The basic problem with all the current measurement techniques is that they are trying to infer characteristics about a network that is not designed to reveal any information to data that traverses it; this is especially true of the Internet. The network simply transports data to its destination as best it can, which is why the Internet is considered a best effort network. There are no guarantees for data traveling through the Internet and there is no way to accurately predict how data will be handled. The only way to determine link capacities and utilization of the links is by examining how the network delivers a single packet or sequence of packets to the destination.

Despite these limitations, researchers have been able to devise methods that can measure bandwidths in a network, even the Internet, with some degree of accuracy. Each method has its own limitations and tradeoffs as we

will see in the following section but none of these methods provide a simple, accurate, efficient, and flexible solution. Such a solution is simply not possible with the current Internet Protocol. The next generation Internet Protocol, IPv6, has the potential to offer an improved solution through the use of timestamps.

In this paper, we suggest that the hop-by-hop extension header in IPv6 be used to hold timestamps to mark when a packet is received and/or sent at each hop through a network path. With this information, we can determine how a packet is handled at each hop as it traverses the network and therefore accurately calculate bottleneck and available bandwidths. The solution is simple, accurate, efficient, and can be used throughout a wide range of applications.

## II. Related Work

### *A. Definitions*

Before we can discuss related work in bandwidth measurements, we must first define bottleneck and available bandwidths. These two terms are often used incorrectly and interchangeably but are in fact describing two different network attributes. The bottleneck bandwidth between two endpoints of a network path is the maximum bandwidth that can be seen between the sender and receiver through that path in the absence of competing traffic. Simply put, the bottleneck bandwidth is determined by the link with the smallest capacity in a network path. This value does not change in time with the load and dynamics of the network but remains the same for as long as the communication links in the path do not change. On the other hand, available bandwidth is the maximum unused capacity available to a sender at a single point in time along the same path in the presence of competing traffic. This value will change in time as the available capacity of the network is utilized by traffic between other hosts.

Available bandwidth measurements are most useful in applications that need to respond to network conditions in order to provide a level of control and optimization. These applications adjust based on the load of the network to achieve the best performance as well as to be fair to competing traffic. Bottleneck bandwidth measures are most useful in network management situations but both measures are useful in applications needing to establish connections for long term communications.

### *B. Measurement Techniques*

Several different techniques for measuring bottleneck and available bandwidths have been developed and implemented by a number of different researchers over the past several years. We can first classify these techniques based on whether they measure bottleneck [1]-[3] or available bandwidths [4], [5] or based on the actual measurement technique such as hop-by-hop [1] or end-to-end [4].

We can further classify these techniques based on the probing methods used. Most forms of measurement use an active technique that sends packets through the network for the sole purpose of measuring the bandwidth. A few techniques use passive methods that use data packets from normal flows to obtain data about the network [3], [4].

Our proposed method can be used in either active or passive measurements but we will focus primarily on

active probing. In active probing, single or multiple probe packets are sent to gather data about the network. Single packet probing models such as [1] and [6] send a single packet or sequence of single packets with different characteristics through a particular network path. The correlation and timing of the reply packets is used with statistical analysis to calculate the bandwidth of the network.

All other active methods send two or more packets together in varying configurations and use the relationship between the packets to determine the network bandwidths. These packets can be sent in pairs of equal or varying sizes or as a group of packets. The method we propose uses the packet pair approach with packets of equal and varying sizes.

*C. Packet Pair*

The packet pair [6] concept works by sending two packets back-to-back through a network path. When the packets reach their destination, the difference in the arrival times and the size of the packet can give the bottleneck bandwidth estimation based on the following formula:

$$b = \frac{probe \quad packet \quad size}{gap \quad between \quad 2 \quad packets} \tag{1}$$

Since the probe packets are sent back-to-back, any separation in the arrival times at the destination is an indication of the bottleneck and/or available bandwidths in the network path. Unfortunately, this technique assumes that packet spacing is preserved as the packets traverse through each hop. Figure 1 presents a simple illustration where the separation of the two probe packets is not preserved through the network.

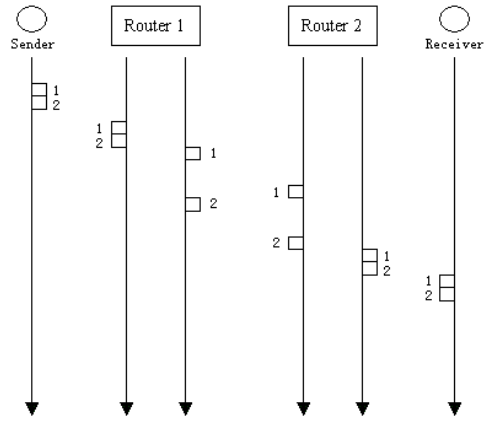


**Figure 1 – Incorrect bottleneck bandwidth calculation scenario**

The separation introduced at Router 1 is not preserved at Router 2 and so the receiver cannot discern a bottleneck in the path. The deterministic model in [2] can be used to filter out some of the erroneous measurements but even the measurements perceived to be accurate are susceptible to significant error due to cross traffic and other

unpredictable network behaviors. The simple fact is that none of the methods currently in use can consistently and accurately measure bandwidth without help from the network itself.

## III. IPv6

### *A. IPv6 Overview*

Internet Protocol version 6 (IPv6) [7] will soon replace the current Internet Protocol version 4 (IPv4) (although IPv4 may never completely go away [8]). Full deployment of IPv6 can be expected to be completed within the next 10 years or sooner depending on the push from other countries and government agencies, not to mention the demand due to the increase in Internet capable devices and the constant threat of router meltdown due to unmanageable routing tables [8]. Initial deployment of IPv6 has already begun with the establishment of 6bone [9], the IPv6 Internet backbone.

When IPv6 is fully deployed, the current tools and techniques for measuring bandwidths should still be applicable but some of the techniques will have to be reevaluated. IPv6 is different in many respects from IPv4, so much so that some of the assumptions may not hold true and the approaches may need to be adjusted. IPv6 primarily differs with IPv4 in that it offers expanded addressing, simplified header format, and improved extension and option support [8].

These differences should not present any immediate problems to the present IPv4 measurement techniques. The differences that will demand the need for reevaluation is the increased header size, MTU, and fragmentation properties of IPv6. The header size in IPv6 is fixed at 40 octets whereas the IPv4 header is variable between 20 and 40 octets. The minimum MTU for IPv6 is set to 1280 octets, more than double the 576 octets of IPv4. The tailgating technique described in [2] uses a smaller packet followed by a larger packet in order to give a higher probability that the two packets will be queued adjacently at each router. The increase in the IPv6 header size will reduce the minimum packet size and therefore reduce the large packet to small packet ratio. This ratio is critical in maintaining back-to-back queuing in network paths where adjacent link capacities increase by a factor greater than the lead/tailgate packet ratio. One would expect the larger MTU of IPv6 to increase the ratio but most researchers assume that packets of 1500 octets can be sent without fragmentation and so the ratio commonly used is 1500 octets to 40 octets.

One benefit of IPv6 is that it eliminates fragmentation at the router level. Hosts must not send packets larger than the MTU of the receiver or an ICMP error message will be returned to the host. The elimination of fragmentation at the router level was decided in order to reduce the complexity and improve efficiency for all routers in the Internet. This decision should prove to benefit bandwidth estimations as the overall network traffic will have a more consistent composition and may result in more accurate assumptions about the network.

### *B. Timestamps*

A timestamp option was initially defined for IPv4 in RFC 760 [10]. This original definition was a very limited

and incomplete definition that only introduced the timestamp specification as part of the DoD standard Internet Protocol. Not until RFC 781 [11] was the timestamp option completely specified. The developers of IPv4 understood the unpredictable behavior and variable delays that are a characteristic of packet switched networks. The timestamp option was meant to provide a solution for critical performance measurement in these networks. Unfortunately, the timestamp option was never very practical and lacked sufficient resources to be useful.

The primary reason the timestamp option failed is because IPv4 only allows up to 40 octets for options. This gives enough room for nine four byte timestamps after the flag fields are accounted for. Timestamps without an associated router are useless unless the network path never changes. To address this, the timestamp option defines a flag to record the timestamp and IP of the router, resulting in a total of only 4 router/timestamp pairs. Room for only four hops is not practical in Internet paths that commonly have a dozen or more hops. Also, we have no control or indication as to how accurate the timestamps are at a given router [12], further adding to the problem. On top of all these short comings, IPv4 routers tend to service packets with options more slowly than normal traffic since they are optimized to handle standard packets [8]. In theory, the IPv4 timestamp option promised to be a valuable feature but the lack or resources and hardware support resulted in the option being useless.

*C. IPv6 Extension Header*

IPv6 is much more suited to handle a timestamp option through the use of the hop-by-hop extension header. Extension headers in IPv6 are additional headers added after the main IPv6 header. There is no limit to the number or size (up to the MTU) of these headers. Some of the extension headers that are currently defined include the hop-by-hop header, routing header, fragment header, destination options header, authentication header, and encapsulating security payload header. The IPv6 specification allows for additional headers to be defined in the future as need arises. In addition, IPv6 routers only inspect packets that contain a hop-by-hop extension header, unlike IPv4 where every packet with an option must be inspected.

The hop-by-hop extension header offers a better structure than IPv4 options and has sufficient resources to be a viable option for gathering information about the network. When a router encounters a datagram with a hop-by-hop header, it must inspect the header and act according to the IETF specification for the option. The IETF currently has two options defined for the hop-by-hop header: jumbo payload option and router alert option. There have been a few other proposals for hop-by-hop options but they have been denied by the IPv6 working group within the IETF.

Two of the proposed options include a traceroute [13] option and an option for Connection/Link Status Investigation (CSI) [14]. The former option is based on the IPv4 record route option that provides a mechanism to record the forward path to a host. The IPv6 traceroute option included this same functionality along with the ability to record the return path using an ICMP reply. This draft was not accepted in its present form due to concerns about a possible Denial of Service attack. The working group agreed that work should continue but there were never any subsequent drafts.

The CSI draft proposed a mechanism to gather information about nodes along a communications path. The information that could be gathered included interface attributes and statistics such as IP address, speed, type, number of transmitted and received octets, number of transmitted and received packets, and number of discarded and erroneous packets. A packet passing through a router could use the CSI option to gather any of the available information for each interface the packet came in contact with. CSI also defined a timestamp option that could be inserted in a datagram for each interface as well.

The CSI option would have been an invaluable tool for investigating and analyzing links in communication paths, but it was not accepted for several reasons. The IPv6 working group saw the CSI option as a potential security and denial of service problem. They also commented that the majority of this information is considered proprietary by ISPs and they would not be willing to reveal that amount of detail about their network interfaces. Lastly, the working group felt that such functionality would be too complicated to implement for most routers, if it was possible at all.

At the time of writing this paper, there have not been any further drafts proposed to the IETF regarding timestamps in IPv6. We feel that although timestamps were not practical due to the limitations of IPv4, IPv6 has the ability to fully support timestamps and the benefits of a timestamp option are well justified. In the next section we present an IPv6 timestamp option that expands on the original IPv4 specification by incorporating some of the improved features of the CSI mechanism without presenting security, complexity, or proprietary technology issues.

## IV. IPv6 Timestamp Option

### *A. Overview*

The timestamp option we present takes attributes from RFC 781 and the CSI mechanism. In order for a timestamp option to be practical it will need the following capabilities:

- Timestamp with millisecond or better resolution.
- Specification of timestamp resolution and format
- Ability to record interface ID
- Adequate space to include all hops in a path.

Timestamps need to have a millisecond or better resolution in order to guarantee that each packet can be uniquely time stamped. That is, minimum size packets that are sent or received back-to-back need to have unique timestamps that is representative of the instance in time that they are handled. No two packets should ever have the same timestamp. Clocks need not be synchronized at each router since we can send multiple packets and use the relationship of the timestamps between packets to calculate network delays. We are not forcing any specific timestamp format due to the differences in hardware, so we must be able to indicate the format and resolution. The ability to record interface IDs is necessary to match timestamps with the appropriate hop, although with IPv6 we can specify a predetermined route using the routing extension header.

In addition to a timestamp option, ICMP messages may need to be defined to request and report timestamps. The ICMP ping message could be modified to copy the timestamp header from the request message into the reply message. In this manner, any host that is reachable with a ping message has the capability to return timestamp information about the incoming and return path to the sender. Integration into an ICMP message will eliminate the need for special software running on the destination host as well. The CSI mechanism defines ICMP messages to be used for link investigation but for the sake of brevity we will assume that a modified ICMP echo message exists that can carry the timestamp hop-by-hop extension to the destination host and back.

*B. Timestamp Hop-by-Hop Format*

The proposed timestamp option format is shown in Figure 2. This figure represents fields in the hop-by-hop extension header to specify desired handling of the timestamp option and it also includes fields for holding information about this packet.

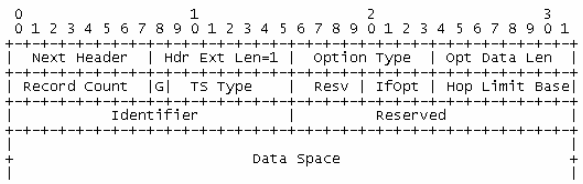


**Figure 2 – IPv6 Timestamp Header**

Each of the fields is described as follows:

**Option Type:** 8-bit integer identifying this option as the timestamp option. The IPv4 timestamp option type value is 68 but this cannot be used for IPv6. The IPv6 option must start with 001 indicating that routers should skip this option if they do not support it and that the data in this option may change en route to the destination.

**Opt Data Len:** 8-bit length of the timestamp option in number of octets. Option length starts from record count to the end of data space.

**Record Count:** 8-bit unsigned integer indicating the number of records contained in data space. Each router must increment this field after inserting a record in the data space. The position for the next record can be calculated using [Record Count] * [Record Length] + 12.

**TS Type:** The TS Type field indicates the desired timestamp behavior. This is an 8-bit field with the upper bit R specifying a high resolution timestamp (finer than 1 millisecond) or a normal resolution timestamp (1 millisecond or less). The TS Type currently has 3 values specified below that follow the flags field in RFC 791. The unused values for this field are reserved for future use.

0 – insert timestamp record only

1 – insert Internet address of the registering entity before the timestamp record

3 – the Internet address fields are pre-specified. An IP module only registers a timestamp if it matches its own

address with the next specified Internet address.

R = 0 indicates that the registering entity should use a normal resolution (1 millisecond or less) while R = 1 indicates a higher resolution finer than 1 millisecond is desired if available. Entities that are not capable of greater than 1 millisecond resolution should insert a normal timestamp and indicate the resolution in the timestamp record.

**IfOpt:** 4 bit integer indicating the interface where the timestamp should be recorded.

0 – reserved

1 – timestamp at incoming interface

2 – timestamp at outgoing interface

3 – timestamp at both interfaces

If the registering entity does not have the ability to timestamp immediately after receiving or before sending then this must be indicated in the timestamp record. The upper 4 bits of this option are reserved for future use.

**Hop Limit Base**: this field is a copy of the initial hop limit field in the IPv6 header. This value is not decremented at each hop but is used to calculate the Hop Number in the timestamp record.

**Identifier:** 16-bit unsigned integer used to distinguish this packet from other probes.

**Data Space:** This space contains the timestamp records as described in the next subsection.

*C. Timestamp Record*

Each timestamp record contains not only the timestamp, but also information about the timestamp including timestamp resolution, format, interface, hop number, internet address and count. The timestamp record format can be seen in Figure 3.

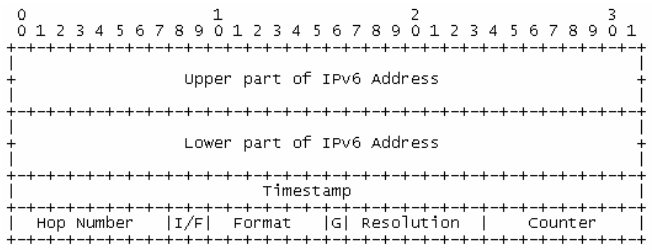


**Figure 3 – Timestamp Record**

The first two fields of the timestamp record consist of the upper and lower portions of the IPv6 address. These fields are not included as part of the timestamp record when a TS Type of 0 is specified in the options. When a TS Type of 3 is specified, then the address fields will be pre-filled by the sending host and entities that match the address field will insert the timestamp after the address.

**Timestamp:** 32-bit unsigned integer. The format of the timestamp is determined by the capabilities of the recording entity. If the R bit is set in the timestamp options then the recording entity must use a timestamp with a

resolution finer than 1 millisecond if possible.

**Hop Number:** 8-bit unsigned integer calculated using the formula [Hop Number] = [Hop Limit Base] – [Hop Limit].

**I/F:** Designates the interface where the timestamp was recorded.

=00 Neutral (independent of interface)

=01 Timestamp at incoming interface inspected

=10 Timestamp at outgoing interface

=11 Reserved

**Format:** Specifies the format of the recorded timestamp. Since we cannot guarantee that all hops will be using the same time format, this field will reveal the formatting of the timestamp at this hop. The only format currently defined is the number of milliseconds that have elapsed since midnight UTC. This format is specified with a value of 000001.

**Resolution:** The resolution of the timestamp is specified as a multiple or divisor of 1 millisecond. For a low resolution where the timestamp is in units greater than 1 millisecond, then G=0 and the timestamp is multiplied by [Resolution] to get the timestamp in units of milliseconds. When we have a high resolution greater than 1 millisecond, G=1 and the timestamp is divided by [Resolution].

**Counter:** 8-bit unsigned integer indicating the packet count at this hop. Each hop increments an internal counter for every packet sent. The lower 8 bits of this counter is placed into the counter field of the timestamp record. The counter field of two back-to-back probe packets can then be used to determine if the packets were indeed sent back-to-back at each hop.

## V. Bandwidth Measurements

In this section, we apply the timestamp option to measure both available and bottleneck bandwidths in a network path. With timestamps, we can measure available bandwidth quite trivially, but bottleneck bandwidth measurements require a little more work including techniques from methods used in IPv4.

### *A. Bandwidth*

Before we discuss our techniques for measuring bandwidth using timestamps, first we need to define bandwidth and how it relates to the transmission of packets. The capacity or bandwidth of a link can be defined using the following equation:

$$BW = \frac{b}{t} \tag{2}$$

Where $b$ is some number of bits and $t$ is some amount of time. Simply stated, the bandwidth is equal to the number of bits that can be transmitted per unit time. We can further expand this equation to the following:

$$BW = \frac{b}{t_2 - t_1} \tag{3}$$

Where $t_2$ is the final transmit time and $t_1$ is the beginning transmit time of some number bits $b$. Based on this equation, if we know the beginning and ending times for transmitting a sequence of bits, we can very easily calculate the capacity of a link in bits per unit time.

*B. Available Bandwidth*

As we stated earlier, available bandwidth is the unused capacity in a network path between two hosts. Available bandwidth can also be described as the maximum throughput a sender can achieve to a destination host without affecting competing traffic. Available bandwidth measurements are quite useful for network applications that want to send data at the highest rate possible without burdening the network by causing competing flows to lose and retransmit packets. In fact, applications using available bandwidth measurements to control their sending rate can be more efficient than TCP flows since TCP control mechanisms continually force the network beyond its limit before backing off.

To calculate available bandwidth, we employ (3) by using packet pairs to obtain the time interval for transmitting a packet at each hop. For a path of $n$ physical links $L_1, L_2, \ldots, L_n$, we send two back-to-back probes denoted by [fs], where f is the first packet and s is the second packet in the packet pair. The size of the probe packets should be representative of the data being sent by the application desiring the measurement. The available bandwidth for the entire path as seen by the sender is

$$BW = \min_{1 \le i \le n} \left( \frac{s(f)}{d(s)_i - d(f)_i} \right) \tag{4}$$

Where $s(f)$ is the size of the first packet and $d(s)$, $d(f)$ are the departure times of the second and first packets respectively. Figure 4 illustrates an example where the packet pair is sent through a path made up of five links. All links are of equal capacity except for link *L3*. This link actually has a higher capacity and can therefore transmit packets faster than all the other links. But due to congestion, other packets are queued behind packet *f* and as a result, packet *s* must wait in the queue for some time before being transmitted to the next hop. The departure times at *L3* then determines the maximum rate a sender can achieve through this path. Effectively, it takes a time of $d(s) - d(f)$ to transmit packet *f* across *L3* since the probe packets were separated at *L3* even though the packet pair was initially sent with no separation. If the traffic dynamics were to remain constant and the router had an infinite queue, any attempt to send packets at a faster rate than the rate observed at *L3* would only further congest the router and the throughput seen by the sender would drop. In a real world situation, the router would drop packets from the queue and all TCP and TCP-friendly flows would reduce sending rates and the abusive sender would eventually capture more of the available bandwidth.

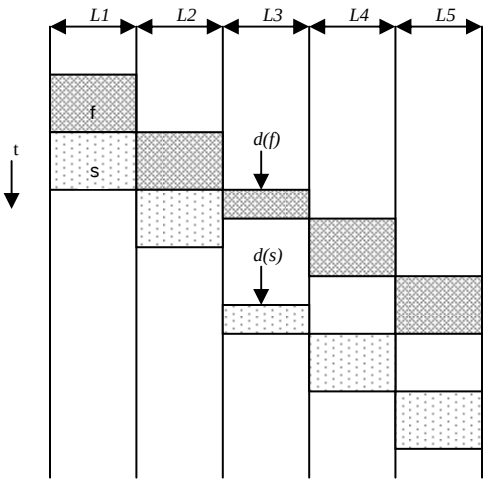


**Figure 4 – Packet-Pair Probe**

In this illustration, we can see that the separation *d(s) – d(f)* is preserved all the way to the destination. All of the current measurement techniques rely on the preservation of packet spacing from the host to the destination, but this is not practical due to the unpredictable behavior of the network. Time stamping packets at each hop will allow us to see the packet spacing at each link and the preservation of packet spacing will not be necessary. We can easily find the maximum spacing between our probe packets and calculate the available bandwidth.

*C. Bottleneck Bandwidth*

In the available bandwidth measurements, we rely on competing traffic to introduce separation between our probe packets to indicate the amount of bandwidth available through a network path. For measuring bottleneck bandwidths, we need to do just the opposite and prevent competing traffic from separating our probes. If two probe packets are queued back-to-back at each hop through a network path then the bottleneck bandwidth for this path can be calculated using (4). The departure time of the second packet is the completion time for sending the first packet and therefore the differences in these times is the interval it takes to send the first packet as given in (3).

In order to increase the probability of back-to-back queuing, we use the packet tailgating technique described in [2]. The tailgating technique sends a large pacer packet immediately followed by a much smaller tailgating packet. The tailgating packet has a much higher probability of being queued behind the pacer packet since the amount of time for other packets to be received and queued while the tailgating packet is being transmitted is small. Obviously, we cannot guarantee back-to-back queuing through a network path but we can perform repeated measurements until two packets are sent back-to-back as indicated by the Count field in the timestamp record.

While timestamps will make it possible to measure transmission time intervals and ultimately calculate link capacity, they will also increase the minimum size of tailgating packet and greatly reduce the probability of back-to-back queuing. In IPv4, a common scenario for a tailgating packet pair is a 1500 octet pacer packet and a 40 octet tailgating packet. This gives a ratio of 37.5 and means that a link can only be 37.5 times faster than the previous link to ensure the two packets are queued and sent back-to-back [2]. This ratio is substantially reduced for an IPv6

packet with the timestamp option. The minimum size for an IPv6 timestamp packet is 40 octets for the IPv6 header, 12 octets for the hop-by-hop extension header with the timestamp option and fields, and 8 octets for each timestamp record without recording IP addresses. For a 20 hop path, we would have a minimum size of 132 octets reducing the ratio to 11.3. This ratio reduces even more if interface addresses for each hop are recorded.

The most accurate but least efficient way to measure capacities is to measure each hop individually. We can do this by setting TS Type = 3 in the timestamp options and insert the IP address of the hop we are investigating. The resulting packet is a minimum of 76 octets and the ratio is around 20. This will limit the ability to accurately measure capacities in a network path where two consecutive link capacities increase by a factor greater than 20.

## VI. Simulations

In this section, we present results from network simulations using the proposed timestamp option and compare our results to an IPv4 method.

### *A. Experimental Setup*

We used the MLDesigner [15] simulation platform to create a network consisting of a single 20-hop path connecting two hosts *A* and *B*. Cross traffic at each hop was modeled using a Poisson process with packet sizes determined by an exponential random distribution with a mean packet size of 200 octets. The mean rate of the cross traffic flows ranged from 100 Kbps in one simulation scenario to 1 Mbps in another scenario. During the simulation, host *A* sends bandwidth probes through the network where they are time stamped at each hop and eventually received at host *B*. The recorded departure times are then processed to calculate the bandwidth for each hop.

We also performed simulations using the cartouche method of probing described in [16]. We chose the cartouche method since it is the most accurate bottleneck bandwidth measurement method in IPv4 and is similar to our method in that it performs active, end-to-end probing and uses a form of the packet pair technique.

### *B. Simulation Synopsis and Results*

Simulations were performed with the number of cross traffic flows ranging from 8 to 48 and the bandwidth of the bottleneck hop ranging from 10 Mbps to 100 Mbps. For the IPv6 probing simulations, probe sizes were 1500 octets for the lead packet and 76 and 132 octets for the tailgating packet. Bandwidth estimations were only made for a hop if the probe packet pair was sent back-to-back from that particular hop as indicated by the Count field in the timestamp record. An average was taken from 100 probe packet pairs and the results can be seen in Figure 5. Identical scenarios were used for the cartouche probing experiments using 1500 and 40 octet probes, with the bandwidth estimated from the average inter-arrival time of the probe packets at host *B*. The results of the cartouche probing are presented in Figure 6.

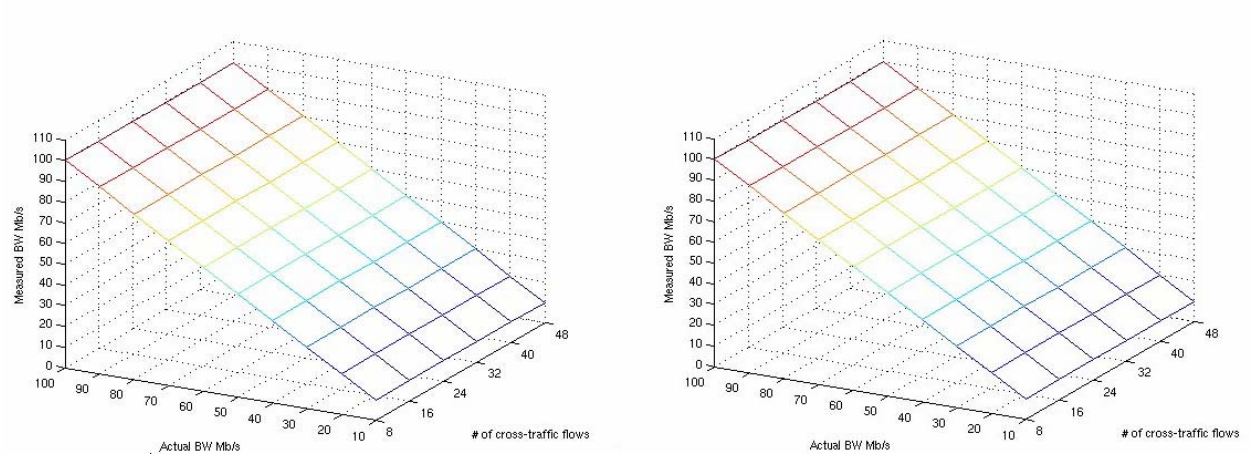


**Figure 5 – Measured BW vs Actual BW for a varying number of cross traffic flows using IPv6 Method. (left) CT Rate = 100 Kbps and Tail Packet = 132 Octets, (right) CT Rate = 1 Mbps and Tail Packet = 76 Octets.**

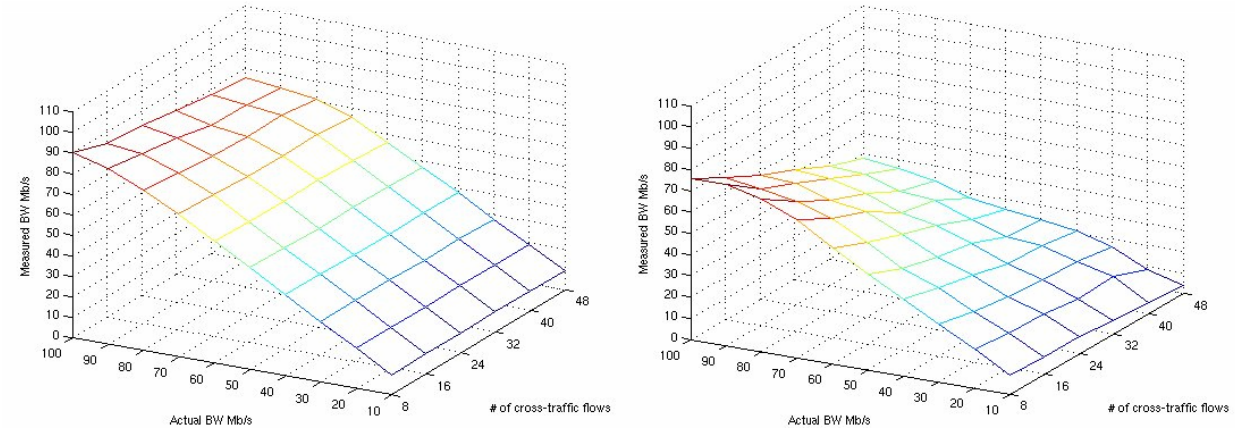


**Figure 6 – Measured BW vs Actual BW for a varying number of cross traffic flows using Cartouche Method. (left) CT Rate = 100 Kbps and r = 2, (right) CT Rate = 1 Mbps and r = 2.**

The results clearly show the increased accuracy of the IPv6 probing method compared to cartouche probing. Our method produced estimations that were 100% accurate compared to the actual bandwidth capacity where as the cartouche method produced estimations that were as low as 30% accurate. The cartouche probes experience compression and decompression as the probes encounter cross traffic at each hop resulting in artificial estimations. Our probing method is immune to inter-hop, cross traffic effects and can detect when the probes are not sent back-to-back at a hop.

While these simulations show favorable results, there are many factors in real world situations that may attribute to skewed measurements. Also, the clock resolution of the simulations can be as precise as needed, something that can not be said for clocks on routing equipment.

## VII. Conclusion

Bandwidth measurements are useful for a number of different applications and situations. Network applications can use information about network paths to select the best path and also decide how to efficiently use the available bandwidth in that path. Administrators and network designers can use bandwidth information to

perform load balancing, traffic engineering, and network optimization. Current techniques for measuring available and bottleneck bandwidths are overall acceptable in certain applications but they suffer in accuracy, simplicity, and efficiency.

The measurement techniques in IPv4, although inaccurate, should be applicable in IPv6, but IPv6's improved options provide the resources necessary to implement a timestamp option that will make these techniques simpler, accurate, and more efficient. Our simulations show that timestamps used in conjunction with IPv6 provide an accurate solution and are up to 70% more accurate than the best estimation method for IPv4.

The greatest obstacle to making our technique a reality is getting the IETF to accept a definition for a timestamp option and making it a standard. The IETF may have reservations for such an option due to hardware limitations and the ineffectiveness of the IPv4 timestamp. The IPv6 specification is constantly undergoing revisions and improvements so a timestamp option would be a low priority next to completely defining IPv6. Even so, we feel that a timestamp option would benefit not only bandwidth measurements, but other areas as well. Until there is feedback from the network itself, consistent accurate bandwidth measurements will be a difficult challenge to overcome.

## VIII. REFERENCES


[1] V. Jacobson. Pathchar: A Tool to Infer Characteristics of Internet Paths. ftp://ftp.ee.lbl.gov/pathchar.

[2] K. Lai and M. Baker. "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay," ACM SIGCOMM Computer Communication Review, vol.30, pp.283-294, Oct. 2000.

[3] K. Lai and M. Baker. "Nettimer: A tool for Measuring Bottleneck Link Bandwidth," in *Proc. of USITS*, pp. 123-134, Mar. 2001.

[4] D. Sisalem and H. Schulzrinne, "The Loss-delay Based Adjustment Algorithm: A TCP-friendly Adaptation Scheme", Workshop on Network and Operating System Support for Digital Audio and Video, 1998.

[5] M. Jain and C. Dovrolis. "End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," in *Proc. of ACM SIGCOMM*, pp. 295–308, Aug. 2002.

[6] J. C. Bolot. "End-to-end Packet Delay and Loss Behavior in the Internet," in *Proc. ACM SIGCOMM*, pp. 289-298, Sept.1993.

[7] Internet Engineering Taskforce and Internet Engineering Steering Group: IPv6 Working Group. RFC 2460: Internet Protocol, Version 6 (IPv6) Specification. [Online] Available: http://www.rfc-editor.org/rfc/rfc2460.

[8] P. Loshin, "IPv6 Theory, Protocol, and Practice", 2nd ed., San Francisco, CA: Morgan Kaufmann, 2004, pp. 22-23.

[9] IPv6 Internet Backbone. [Online]. Available: http://www.6bone.net

[10] Internet Engineering Taskforce and Internet Engineering Steering Group: Network Working Group. RFC 760: DoD standard Internet Protocol. [Online]. Available: http://www.rfc-editor.org/rfc/rfc760.

[11] Internet Engineering Taskforce and Internet Engineering Steering Group: Network Working Group. RFC 781: Specification of the Internet Protocol (IP) timestamp option. [Online] Available: http://www.rfc-editor.org/rfc/rfc781.

[12] W. Stevens, “TCP/IP Illustrated,” 1st ed., vol. 1, Indiana: Addison Wesley, 1994, pp. 95-96.

[13] A. Durand, L. Toutain, "IPv6 Traceroute Option," IPv6 Working Group Internet Draft, June 1997. [Online] Available: http://www.join.uni-muenster.de/Dokumente/drafts/draft-durand-ipv6-traceroute-00.txt

[14] H. Kitamura, "Connection/Link Status Investigation (CSI) for IPv6 Hop-by-Hop option and ICMPv6 messages Extension," IPng Internet-draft, NEC Corporation, Oct. 1999. [Online] Available: http://www.watersprings.org/pub/id/draft-ietf-ipngwg-hbh-ext-csi-02.txt

[15] ML Designer. MLDesign Technologies. http://www.mldesigner.com.

[16] Harfoush, K, Bestavros, A, Byers, J, “Measuring Bottleneck Bandwidth of Targeted Path Segments,” in *Proc. IEEE INFOCOM*, vol. 3, pp. 2079 – 2089, Mar. 2003.