Java-Based Bandwidth Measurement Tool for xDSL Networks

Liang Cheng and Ivan Maršić, *Member, IEEE*

Abstract -- In xDSL networks, accurate bandwidth measurement is useful for network management and traffic engineering, such as isolating line faults and verifying guaranteed QoS specifications, where ISP and xDSL providers may be involved. This paper presents a novel design and implementation of Java based tools for accurately measuring network bandwidth in xDSL networks, including asymmetric upstream and downstream cases. Considering ATM traffic shaping for ABR and UBR service classes in xDSL deployments, a stepwise algorithm is designed to minimize its effect on the bandwidth measurements. The algorithm is lightweight in terms of traffic overhead introduced. The accuracy of the algorithm is achieved by implementing an original traffic generator with stable performance.

Index Terms -- xDSL networks, network management, accurate bandwidth measurement, asymmetric network, Java, quality of service.

I. Introduction

The growth in deployment of high bandwidth applications over the Internet has created a need for reliable and fast local loops for both home-user communities and corporate communities, who connect to the Internet through a dial-in ISP (Internet Service Provider). In most cases, a serial modem solution that offers the maximum bandwidth of up to 56 Kbps is not sufficient. Therefore, various technologies, such as xDSL and cable TV modems, have emerged to achieve high-speed connectivity to the Internet.

xDSL refers to different variations of DSL (Digital Subscriber Line), such as ADSL (Asymmetric DSL), HDSL (High bit-rate DSL), and RADSL (Rate Adaptive DSL). For example, ADSL enables receiving data at rates up to 6.1 Mbps (of a theoretical 8.448 Mbps). More typically, individual ADSL connections will provide from 512 Kbps to 1.544 Mbps downstream and about 128 Kbps upstream.

In xDSL networks, accurate bandwidth measurement is useful for network management and traffic engineering, such as isolating line faults and verifying guaranteed QoS (Quality of Service) specifications, where ISP and xDSL providers may be involved. Traffic load can be balanced based on the measurements of available bandwidth in xDSL networks. Moreover, once we identify a bottleneck link, we can add extra capacity to enhance the network performance.

We present a bandwidth measurement algorithm that is lightweight in terms of traffic overhead introduced. The accuracy of the algorithm is achieved by implementing an original traffic generator with stable performance.

The paper is organized as follows. Section II briefly reviews the xDSL deployment architecture. Section III introduces bandwidth measurement methodology. Next, Section IV presents our stepwise bandwidth measurement algorithm. Section V describes the Java implementation, which is particularly suited for user-friendly deployment. Section VI presents the evaluation results and Section VII concludes the paper.

II. xDSL Deployment Architecture

Local phone companies in the United States are installing xDSL lines since 1998. A DSL line can carry simultaneously both data and voice signals and the data part of the line is continuously connected. The deployment architecture is abstracted in Figure 1.

In case of ADSL, the customer premises equipment is an ATU-R (ADSL Termination Unit – Remote) or an ADSL modem. An ATU-C (ADSL Termination Unit – Central Office) is located at a Central Office end of the phone company which provides the xDSL service. The ATU-C terminates multiple subscriber loops and connects the ATU-R to the Internet through a Gateway Router. The deployment and maintenance of the line between the ATU-R and the Gateway Router forms the responsibility of the phone company and, as already mentioned, there is a need to measure the actual physical line speed of the ADSL link for the reasons of network management and traffic engineering.

III. Bandwidth Measurement Methodology

A. General Methodology

The methodology for the bandwidth measurement in xDSL networks is based on the packet-pair technique with FIFO

Department of Electrical and Computer Engineering and the Center for Advanced Information Processing (CAIP), Rutgers University, Piscataway, NJ 08854-8058 USA. *{chengl, marsic}@caip.rutgers.edu*

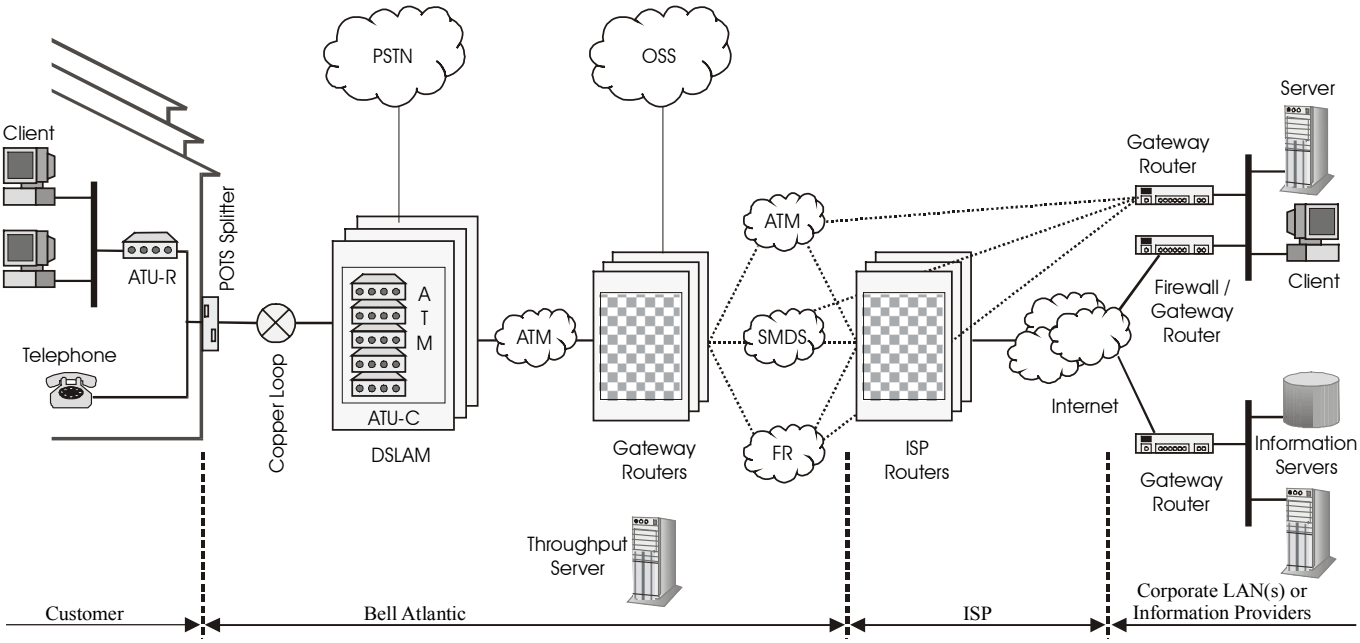

Figure 1. xDSL deployment architecture.

queueing network model. Various forms of the packet-pair technique are studied by Bolot [1], Carter and Crovella [2], Paxson [3], and Lai and Baker [4]. Its essential idea is using inter-packet time to estimate the characteristics of the bottleneck link. If two packets, e.g., ICMP (Internet Control Message Protocol) probe packets, travel together so that they are queued as a pair at the bottleneck link with no packet intervening between them, then their inter-packet spacing is proportional to the processing time required for the bottleneck link to transmit the second packet of the pair (Figure 2). We modified this technique to apply it to xDSL and asymmetric DSL (ADSL). In our method, the probing packets can be the payload packets or the explicit ICMP probe packets.

B. Asymmetric DSL (ADSL) Case

The asymmetric nature of the ADSL network makes it necessary to have different measurements methodologies for upstream and downstream cases. The design is predicated upon the assumption that the upstream bandwidth is much lower that the downstream bandwidth.

1) Upstream Methodology

A fixed number, N, of UDP (User Datagram Protocol) packets of uniform size, P bytes, are sent from the customer's computer (client) at a rate slightly higher than the nominal bottleneck bandwidth of the ADSL network. The slightly higher rate (about 10%) is necessary to saturate the pipe. A server process on the Throughput Server (see Figure 1), echoes back the packets as they arrive at the server end. The time difference, T, between the arrival of the first packet at the client end and the arrival of the last packet at the client end is measured and the upstream bottleneck speed is computed as: $N \times P \times 8/T$ Kbps.

2) Downstream Methodology

A traffic generator at the Throughput Server generates a downstream traffic. A receiving process at the client measures the arrival times of the packets. Because of the asymmetric nature of ADSL (upstream bandwidth is smaller than the downstream bandwidth), the client computes the downstream bandwidth using the same equations as for the upstream case instead of echoing the traffic back.

C. Traffic Generator with Stable Performance

Since the accuracy of the bandwidth measurement depends

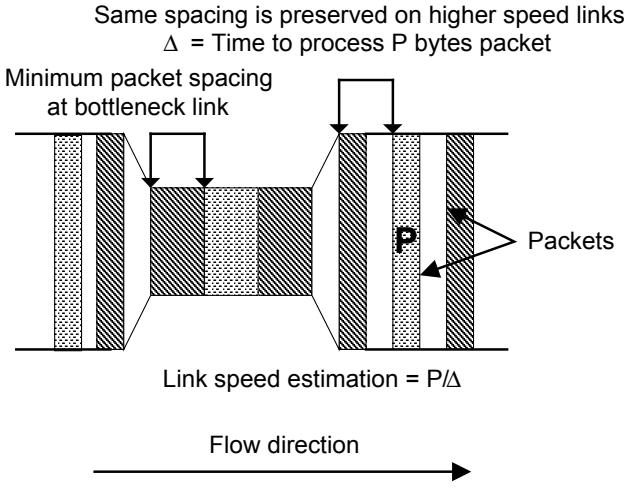

Figure 2. Packet-pair technique for bandwidth measurement.

on whether the traffic generator can generate traffic at a rate slightly higher than the nominal bottleneck bandwidth of the xDSL network, it is crucial to implement it with stable performance.

A straightforward method of generating packets at a known rate employs the following algorithm:

```
// Initialize:
delay = # of packets * packet size in bits / rate;

for (k=0; k < # of packets; k++)
{
     sendPacket();
     sleep(delay);
}
```

The above pseudo-code though seems to be correct in theory does not yield the expected behavior. Owing to the scheduling mechanisms of operating systems and coarse granularity of timers, the packets are not necessarily sent at a uniform rate with uniform spacing. Consider a case when the call to `sleep()` returns and the process is scheduled out. Instead of sending a packet immediately, the packet is sent only when the OS reschedules the process. This results in bursts in traffic generating.

To overcome this problem an alternative way of generating traffic at the desired rate was devised. The algorithm achieves the expected behavior, i.e., it stably generates a smooth flow of traffic at the expected rate. The pseudo-code for the improved algorithm is as follows:

```
// Initialize:
fraction = 0;
stopSleep = getTimeMillis();

do
{
    startSleep = stopSleep;
    sleep(1);      // sleep for 1 ms
    stopSleep= getTimeMillis();

    //pps: the # of packets to be sent per second
    packetCount = (int) ( fraction + pps *
          (stopSleep - startSleep) / 1000 );
    fraction = (fraction + pps * (stopSleep -
        startSleep) / 1000 ) - packetCount;

    upLimit = i + packetCount;
    if (upLimit > # of packets)
         upLimit = # of packets;
    for ( ; i < upLimit; i++)
         sendPacket();
} while(i < # of packets)
```

In computing the number of packets to be sent we divide by 1000 and truncate the decimal part. Thus, we may end up sending less number of packets than necessary to obtain the accurate measurement. For this purpose we maintain `fraction` and take the error into account in the next round.

The packet size is computed as:

P = IP header + UDP header + MAC header + payload

The default payload size is 1400 bytes. The payload is a ZIP-compressed file to avoid the effects of the compression on the V90 analog modem.

IV. Stepwise Algorithm for ATM Networks

A. Effects of ATM Traffic Shaping

xDSL deployments are typically done over ATM to provide guaranteed quality of service. The ATU-Cs in xDSL network are grouped into a DSLAM (DSL Access Module) unit that terminates the multiplexed traffic into an ATM switch.

There are two service categories for best effort traffic in ATM networks specified by ATM Forum: UBR (Unspecified Bit Rate) and ABR (Available Bit Rate) services. UBR service does not have any guarantee with regard to the delay, loss or bandwidth but applications should be able to handle the fluctuations in these parameters by using error-correction and flow control techniques. On the contrary ABR service guarantees the loss rate, which is achieved by exchanging resource management cells between the source and sink ATM nodes.

Based on the synchronization speed of ADSL modem, the ATM switch sets the service class for the virtual circuit to a class to which a particular customer has been mapped. In general commercial xDSL deployments, the ABR service class is chosen.

With the ABR service class setting, an increase in the data rate beyond the provisioned bandwidth results in heavy packet loss. This is caused by the ATM traffic shaping. Its effect on the bandwidth measurement algorithm is that it distorts the results because the accuracy of packet-pair technique depends on the success of receiving back-to-back packet pairs. Therefore, we designed a stepwise bandwidth measurement algorithm to minimize the effect of ATM traffic shaping.

B. Stepwise Bandwidth Measurement Algorithm

The algorithm consists of at least two steps. The same algorithm presented in the previous section can be used for both the downstream and upstream measurements. The traffic generator should be stable to avoid bursty traffic.

During the first step, a small number of packet-pair sequences, e.g., 5, are sent back-to-back from the client to the server. This small number of packets will likely not contribute significantly to packet loss, as most ATM networks are capable of handling such small bursts. The

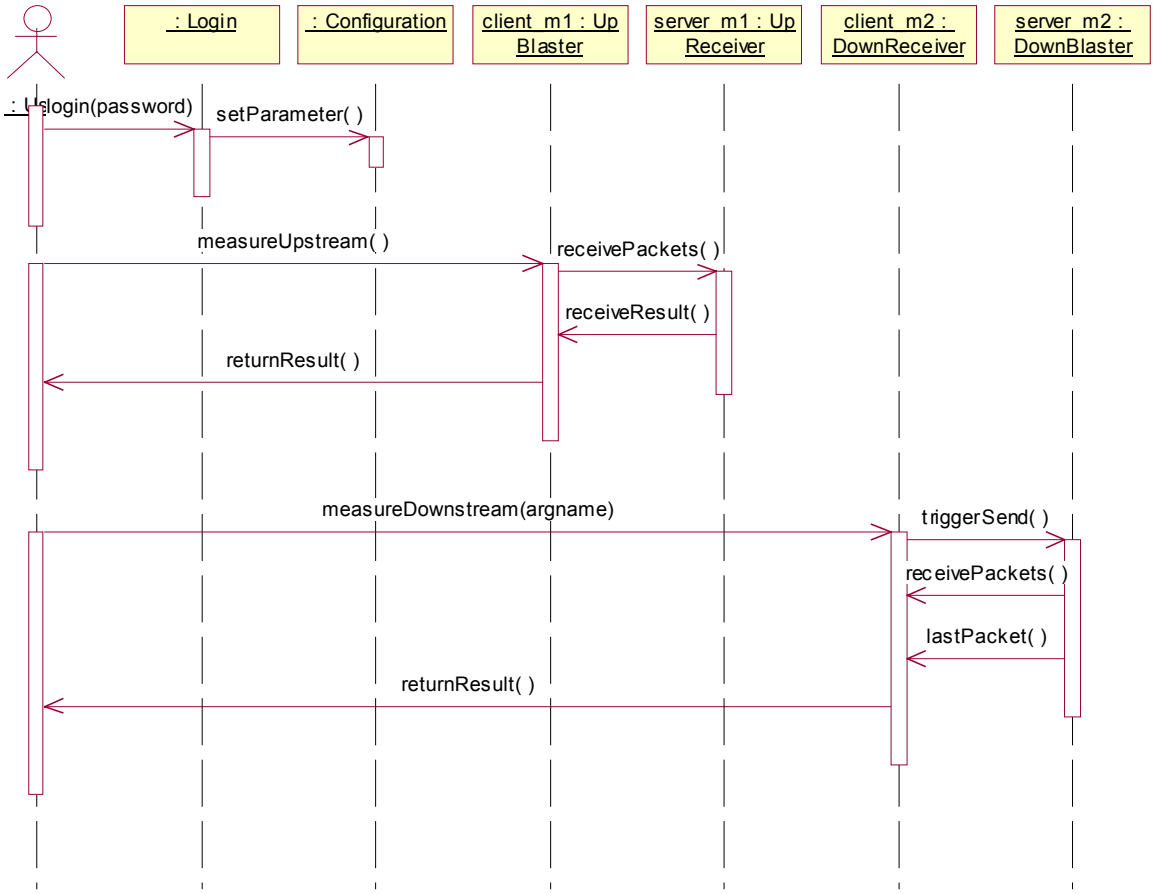

Figure 3. UML sequence diagram for interactions between the user and the bandwidth measurement system.

computed result is used in the second step as the trial bandwidth of the xDSL link.

The following step(s) assumes that the accurate xDSL bandwidth is close to the trial result obtained from the first step. In fact, only packet loss can impair the result. If the packet loss in the first step is minimal or zero, which has also been taken into account by our implementation, then the result of the first trial is accurate.

The second and the following steps use larger number of packet pairs in the measurement to ensure that the results are convergent and in the end consistent. Had it not been for the first step which approximately determined the bandwidth, the sustained higher-than-provisioned traffic would result in a considerable loss and would deteriorate the results.

We run the algorithm several times and report the maximum measured value as the bottleneck bandwidth.

V. JAVA IMPLEMENTATION

Our bandwidth measurement tool includes both server side and client side implementation. The client side of the tool is implemented as a Java applet, which can be loaded in a Web browser. Java applet is used for client side implementation so that (*i*) the client software does not need to be physically installed on the user's computer and it can be downloaded and run from the user's web browser, and (*ii*) upgrades to the client software can be easily facilitated by this feature.

On the server side, the Java programming language is used for implementation because of its multi-platform inter-operability.

Figure 3 illustrates the sequence of interactions between the user, the client software, and the server software. The user logs in, configures the tool, and initiates the upstream and downstream bandwidth measurement of xDSL links.

A. Components of Client Software

1) Authentication

The client software asks the user for user name and password and then validates the information from the server to check whether the user is allowed to use this tool. Figure 4 shows the login interface.

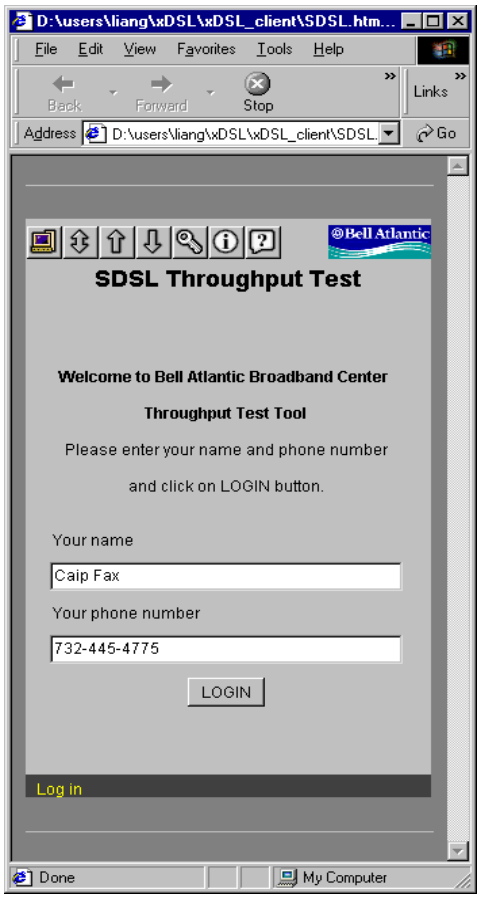

Figure 4. Login interface of the bandwidth measurement tool.

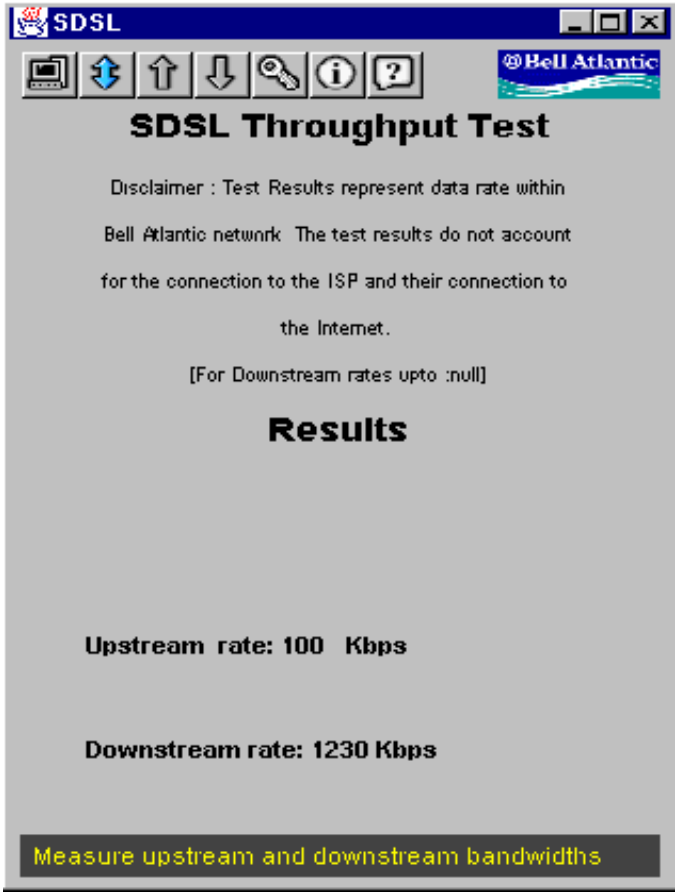

Figure 5. Bandwidth measurement result panel.

2) Tool Configuration

The client software presents an interface for the user to configure parameters for the bandwidth measurement, such as probe packet size, number of probes packets per measurement, number of stop packets, etc.

3) Traffic Generator

Since probe packets need to be generated and sent to the server in the upstream bandwidth measurement, a stable traffic generator using UDP is implemented.

4) Measurement

The measurement component implements stepwise algorithm for upstream bandwidth measurement. The measurement results are displayed in the user interface as shown in Figure 5. The results are also reported back to the server for further data analysis and history log maintenance.

B. Components of Server Software

1) Authorization

The server software receives user name and password from the client software, checks this information against the user database and, if confirmative, authorizes the user to start a measurement session.

2) Traffic Generation

Since probe packets need to be generated and sent to the client in the downstream bandwidth measurement, a stable traffic generator needs to be implemented. Again, UDP packets are used instead of ICMP packets because of Java's lack of support for raw sockets. The generator is activated by a trigger packet from the client. The last packet from the server is marked as the "stop packet" so the client knows when the stream of packets from the server ends. Since UDP is an unreliable protocol, the stop packet can be lost. For this reason we also set a timer when the first packet is received (default value is 5 sec), which will be used as substitute for the time of the stop packet arrival.

3) Echo Server

The UDP Echo Server for upstream measurements waits for Echo Request packets on a specific server port and echoes back the received packet back to the client. Essentially it acts as a rate generator generating packets at a rate closely matching the upstream bandwidth. Since the standard port 7 service of Unix servers discards Echo Request packets beyond a certain rate as a security measure, this standard Echo server cannot be used in our application. A version of the Echo Server has been coded to handle throughput rates as high as 640Kbps which satisfies the Bell Atlantic ADSL network Upstream Throughput rates.

4) Measurement

The measurement component implements the stepwise bandwidth measurement algorithm described in Section IV.

5) Logging Information for Data Analysis

The Log Server collects upstream and downstream measurement results and stores them into the database for future data analysis. Logs also enable the technician at the Central Office to instantaneously verify the results of the test being performed by the customer.

VI. EVALUATION

Figure 6 compares the performance of our bandwidth measurement tool to that of a popular tool called Pathchar [5]. The latter uses ICMP error packets in estimating the link characteristics and also relies on injecting UDP packets to measure the link characteristics. For the purpose of experiments the bottleneck link bandwidth was controlled in a laboratory testbed network by introducing a High-Speed Serial Interface (HSSI) between the two gateways. The endpoints for the test were located across the two ends of the ADSL link or the HSSI. Figure 6 shows an almost linear curve for our tool whereas Pathchar shows significant deterioration starting at about 1 Kbps.

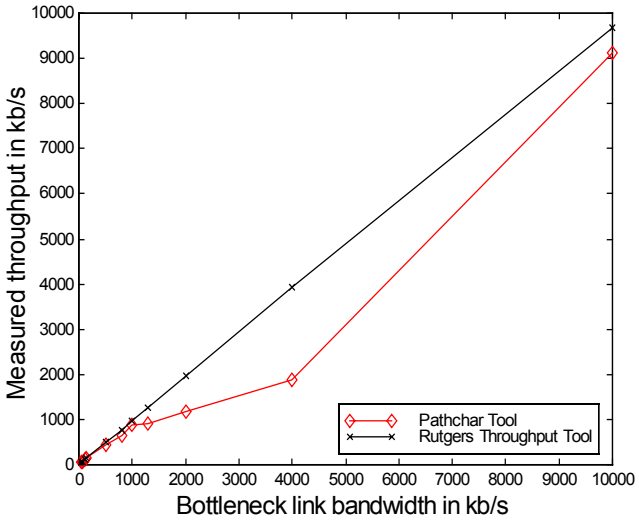

Figure 6. Comparison of tools for throughput estimation. The HSSI clock is only able to accurately set the link speeds up to 4000 Kbps. Consequently there is a gap from 4000 Kbps to 10000 Kbps, which is the link speed for a 10Mbps Ethernet LAN.

VII. CONCLUSION

This paper presents a design and implementation of Java based tools for accurately measuring network bandwidth in xDSL networks, including asymmetric upstream and downstream cases. Considering ATM traffic shaping for ABR and UBR service classes in xDSL deployments, a stepwise algorithm is designed to minimize its effect on the bandwidth measurements. The algorithm is lightweight in terms of traffic overhead introduced. Furthermore, the accuracy of the algorithm is achieved by implementing an original traffic generator with stable performance. Evaluation experiments have proved that the tool can achieve accurate bandwidth measurement in xDSL networks.

Acknowledgments

This research is supported by grants from Verizon, Inc. (formerly Bell Atlantic), Cisco, Inc., and DARPA Contract No. N66001-96-C-8510, and by the Rutgers Center for Advanced Information Processing (CAIP).

References

[1] J. C. Bolot, "End-to-End Packet Delay and Loss Behavior in the Internet," *Proceedings of the ACM SIGCOMM '93 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 1993.

[2] R. L. Carter and M. E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," *Technical Report* TR-96-006, Department of Computer Science, Boston University, March 1996. http://www.cs.bu.edu/faculty/crovella/papers.html

[3] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*, Ph.D. Thesis, University of California, Berkeley, April 1997.

[4] K. Lai and M. Baker, "Measuring Bandwidth," *Proceedings of IEEE INFOCOM*, 1999.

[5] A. Downey, "Using Pathchar to Estimate Internet Link Characteristics," Proceedings of the ACM SIGCOMM '99 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Cambridge, MA, pp.241-250, August/September 1999.