Asian Journal of Information Technology 5 (12): 1313-1318, 2006

A Java-Based Estimation System for Real-Time Internet Access

Seifedine Kadry
Beirut-Lebanon Allenby Street, P.O.Box: 110-435 009613700512

Abstract: This study presents a Java-based real-time Internet access estimation tool for Quality of Service (QoS) in Internet accesses for Multimedia applications (JEQoSIM). It is specially aimed for real-time multimedia applications which use the User Datagram Protocol (UDP). The system is capable of estimating access capacity, available bandwidth and delay as the critical end-to-end QoS parameters for this kind of applications. The algorithm used for QoS estimations is one-way and is based on the packet train technique. Real-time QoS estimation elements are distributed among a central server and the Internet end user. The central server contains a UDP packet bursts server and a web server that hosts the Java applet that implements the UDP packet bursts client. JEQoSIM has been validated using several commercial Internet accesses with different technologies: Asymmetric Digital Subscriber Loop (ADSL), General Packet Radio Service (GPRS) and Universal Mobile Telecommunications System (UMTS).

Key words: QoS, internet access, real-time, multimedia

INTRODUCTION

From its beginning, Internet has experienced a huge increase in the number of users, services and data transferred. Different Internet access characteristics lead to very diverse levels of Quality of Service (QoS)[1], which can have a great impact on service performance, particularly on real-time ones.

There are several players interested in estimating QoS, namely network operators, Internet Service Providers (ISP) and end users. Network operators are concerned about network planning; ISPs want to ensure a certain degree of service for the end user, who finally wants to assess the QoS obtained. Thus, these three agents can greatly benefit from a system designed to estimate QoS, making it possible for them to compare what is theoretically offered with what is actually obtained from an Internet access. This is especially important for end users, given the fact that over the last years, problems derived from poor Internet accesses are at the top positions in the number of complaints to consumer associations. Moreover, QoS can also be considered from different points of view: security, performance, speed, reliability, overall user impression, etc. As a result, the complex set of elements that influences QoS makes its measurement a difficult task.

Several QoS-related network parameters estimation tools have been designed through the last years, being bandwidth one of the most widely measured parameters. Other parameters such as delay or packet loss rate are also frequently used, but to a lesser extent.

A review of some bottleneck link bandwidth estimation techniques is presented within[2-4]. However, other estimation tools use a more direct approach to bandwidth estimation, especially the common real-time bandwidth speed tests[5,6]. The majority of these systems measure the time required to transfer one or several fixed-size files to different servers in order to calculate bandwidth. Nevertheless, this method has a major drawback: Only the bandwidth for Transmission Control Protocol (TCP) file transfers is estimated. Real-time applications, on the other hand, are usually transmitted using the Real-time Transport Protocol (RTP)[7], which in turn uses the User Datagram Protocol (UDP), so existing bandwidth speed tests are not well suited to this type of applications.

In this context, this study presents a Java-based real-time QoS estimation system for Internet accesses specially aimed at real-time multimedia applications called evaluation of QoS in Internet accesses for Multimedia applications (JEQoSIM). It is capable of estimating access capacity, available bandwidth and delay as QoS parameters using UDP packet trains. It has been developed using Java, so it can be easily and quickly accessible for the end user.

MATERIALS AND METHODS

As has been stated in the introduction, the majority of the publicly available real-time bandwidth speed tests use TCP files transfers as the basis to estimate bandwidth[5,8]. This approach, however, has several drawbacks:

- Only the bandwidth for TCP file transfers can be estimated. UDP-based applications, mainly real-time ones, are not considered.
- The bandwidth estimation process is highly intrusive and it is frequently required that the user does not send any other network traffic while the bandwidth speed test is being carried out. This is not a realistic situation since typical Internet users generate different traffics at the same time and the access capacity is a value not as useful as the available bandwidth[8].
- Usually, delay and packet loss rate are not considered. There are specific tools that take them into account, but they are not intended for the non-expert Internet user[9,10].

Metrials: In the communication path there is usually a link that sets QoS parameters and it is commonly called the bottleneck link[2,8,11]. Different estimation tools focused on discovering bandwidth in the bottleneck link (also called the bottleneck bandwidth) use measurement methods that can be classified into passive[10,12] and active[3,4]. Active measurement methods can be further divided into those that measure Round Trip Time (RTT)[13] and those that only measure one traffic direction (One-Way)[3]. The most used protocols in these measurement systems are UDP, TCP and Internet Control Message Protocol (ICMP).

Once the different measurement acquisition methods have been presented, it is very important to identify the most relevant QoS parameters for real-time applications. Two common and ubiquitous parameters are used to measure QoS levels are bandwidth and delay[2,8,9,14,15]. These two parameters have been selected for JEQoSIM because they make it possible for the client to check the performance of his Internet access, especially when it is used for real-time communications. As this kind of communications mainly uses RTP, which in turn uses UDP, this is the protocol selected for the estimations.

The bandwidth estimation algorithm selected for JEQoSIM is One-way and it is based on the transmission, in both directions of communication, of bursts of k UDP packets with constant packet Size (S) (packet trains).

Bandwidth estimation algorithm: Given a path between two network end points that includes n links L_1, L_2, … L_n with bandwidths BW_1, BW_2, …, BW_n, the Bottleneck Bandwidth (BBW) can be defined as:

$$BBW = \min (BW_1, BW_2 \ldots Bw_n) \quad (1)$$

Next, given a link L_i with bandwidth BW_i and traffic load TL_i, the available bandwidth (ABW) in the link is defined as:

$$ABW_i = BW_i - TL_i \quad (2)$$

We present a *two-step algorithm* in order to obtain the *available bandwidth*. In the *first step*, we send the packets in the burst as close in time as possible, that is, with the minimum gap between them. In such conditions, other packets sharing the link are not likely to merge with the so closely ones in the burst. As explained in[13], when the burst crosses a link with less bandwidth, the packet spacing becomes higher (the packet rate becomes smaller). This increment in the packet spacing is preserved when the burst crosses higher speed links Fig. 1, allowing us to measure the *bottleneck* link capacity at the reception of the burst as the sum of the length of the packets received in response to the burst, divided into the time between the reception of the first and the last answer (3). The *second step* Fig. 2 consists on sending the packets in the burst at a rate equal to the nominal capacity just obtained. Now, a packet spacing increase will be due to other packets in the link merging with the burst.

This increase allows us to estimate the *available bandwidth* in the bottleneck, by using the formula shown in (3) again.

$$\text{Bottleneck bandwith} = \frac{((n-1)\times \text{packet\_size})}{(t_n - t_1)} \quad (3)$$

An important factor to consider is the number of packets to include in the burst, as well as the size of these packets. The aim is to obtain a good estimation of the links but being as less intrusive for the network as possible. In fact, the greater the number of packets in the burst and the greater their size, the more intrusive the method. But having more packets in the burst implies more accurate estimations. Studies carried out by other authors[11] show that using five packets by burst allows reaching a trade-off between good bandwidth estimation and little bandwidth required for the estimation method.

The parameters that characterize this algorithm (packet length, number of packets per burst, packet spacing in a burst and time between bursts) are fully configurable to select those better suited for each particular scenario.

It is important to note that this estimation method is much less intrusive than traditional bandwidth speed tests because transmits less information (UDP) than other methods (TCP) and produces acceptable results with a

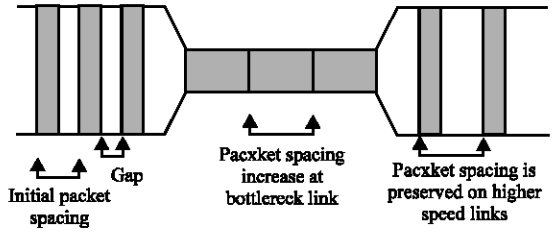

Fig. 1: First step: packet burst through a bottleneck link

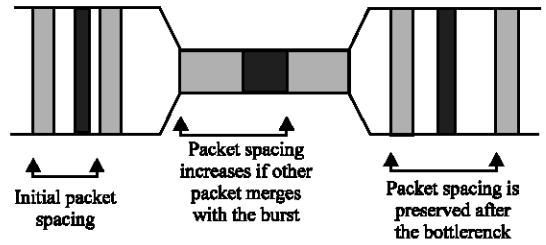

Fig. 2: Packet burst sent in the second step

minimum bandwidth waste. It is also capable of estimating bandwidth under realistic circumstances, i.e. when the user is generating other network traffics, which makes the value of ABW a crucial parameter in order to decide whether a particular real-time application can be used in conjunction with other traffics.

Java technology: Java has been selected as the underlying technology for JEQoSIM because it is platform-independent and widely used in Internet. A Java applet is responsible for the client side of the system, that in turn communicates with a Java application running in the web server from which the applet has been downloaded, making the appropriate measurements. Java applets have their specific security restrictions and limitations[16], especially regarding time accuracy, but they do not cause important estimation errors when working with low speed accesses (up to 1 Mbps in the downlink). Moreover, in a real application there is a clock Granularity (G), which can be defined as the maximum time interval in which the system clock measures the same moment. Thus, it is possible to divide the time in discrete intervals (timeslots) and to consider that the exact timeslot where a frame has arrived is known (instead of knowing its exact arrival time t). Therefore, the time difference between the last (t_k) and the first (t_1) frame is unknown, but the number of intervals (n) between their arrivals is known.

Time precision depends on clock resolution and affects time synchronization and parameter acquisition accuracy. In Java it is determined by the Java Virtual Machine (JVM) implementation in each operating system and computer architecture[17]. This resolution problem increases when S is small or the access rate is high, because the measured time intervals can be very small. Nevertheless, if more time resolution were needed, Java Native Interface (JNI) could be used to add C++ code[17].

System architecture: JEQoSIM has been developed according to the scenario presented in Fig. 3 as can be seen; there is a central node where users come to get their QoS measurements. This central node contains a web server that hosts Hypertext Mark-up Language (HTML) pages and the Java applet that implements the client-side application to be displayed in a Java-compatible browser. In addition, a Network Time Protocol (NTP) server and an UDP bursts server (the server that receives the UDP packet bursts and replies to them) that takes the appropriate measurements are installed.

The data flow diagram of Fig. 4 shows the process of making QoS estimation with JEQoSIM. When a user loads the main web page, an applet is downloaded showing three different versions: simple, advanced (for advanced users) and monitoring (designed to do scheduled QoS estimations). Then, the applet sends the user identifier and the server confirms it. The next step for the applet is to exchange NTP messages with the server in order to be synchronized. As soon as the time offset between the server and the client is corrected, TCP communications are used to establish the burst parameters (number of bursts, frames per burst, frame length, etc.). When the server processes those parameters, the UDP bursts client is accepted or refused through the TCP connection.

If the answer is affirmative, several UDP bursts are sent in the uplink and in the downlink.

To notice the end of the UDP bursts, two TCP "End of Burst" messages are sent. Once these TCP messages reach the client and the server, both of them exchange their measurements using TCP. By this way, the results can be displayed by the applet in the user's browser and stored by the server for further processing.

Evaluation tests

Test scenarios: JEQoSIM has been validated in commercial Internet accesses. This study presents several evaluation results obtained with the following commercial accesses:

Asymmetric Digital Subscriber Loop (ADSL): 300 kbps in the uplink and 1 Mbps in the downlink, with 10% guaranteed in the contract. The capacity is greater than that of an analog modem, but only a percentage of it is available. Information is transmitted using fixed size Asynchronous Transfer Mode (ATM) cells[11]. As its name indicates, both capacity and available bandwidth are asymmetrical.

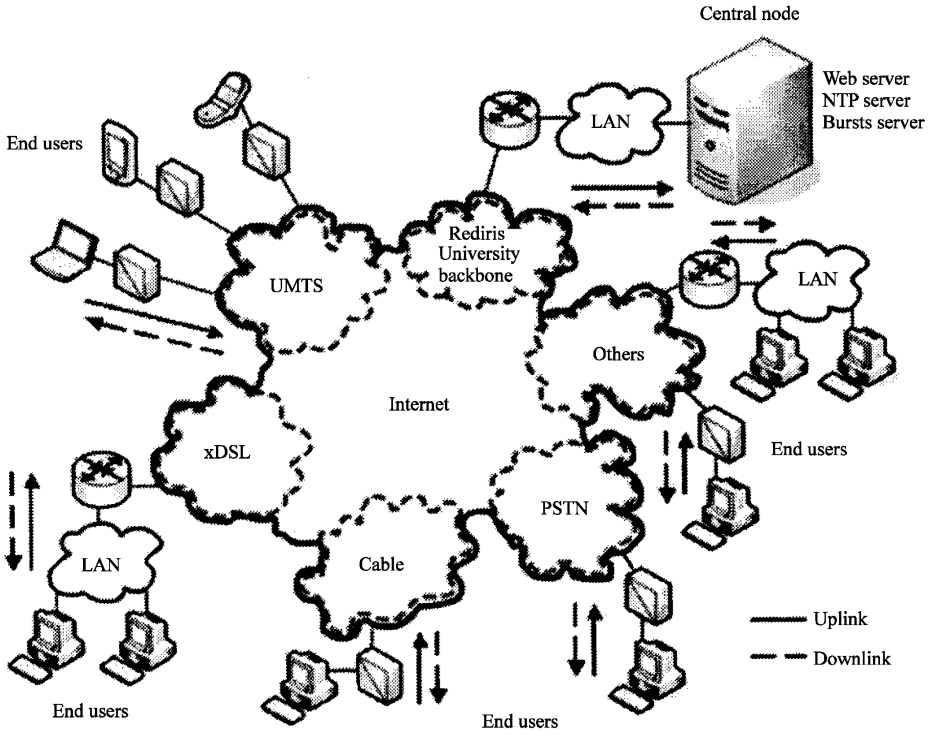

Fig. 3: General network scenario

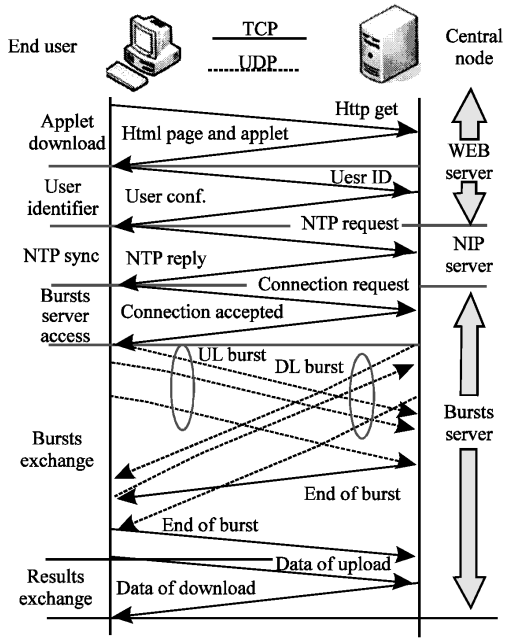

Fig. 4: Data flow diagram

General Packet Radio Service (GPRS) and Universal Mobile Telecommunication System (UMTS): These accesses are shared between several users. However, available bandwidth should remain almost constant, but its value can vary depending on the radio link conditions. Delay is greater than in the other accesses due to channel coding and interleaving.

It is important to remark that a typical user of JEQoSIM only knows the access parameters given by his ISP (access capacity and guaranteed bandwidth), but this is not enough in order to characterize the behavior of the access in a working situation. The bandwidth available to a particular user may vary through the time in a particular access, since ISPs only guarantee a certain percentage of it. As a result, real tests with commercial accesses can produce more significant results. A bandwidth monitoring process would be of special interest and for that reason JEQoSIM has the monitoring option.

Test parameters: The results presented in the next section correspond to several tests that consisted of:

- Number of bursts sent: 48 bursts in both uplink and downlink.
- Time between two consecutive bursts: 1 min.
- Variable frame size (S): 100, 400 and 1000 bytes of UDP data without overhead.

Table 1: Bottleneck bandwidth

			Uplink		Downlink	
Access	S bytes	K (Frames)	μ	σ	μ	σ
	100	5	257.882	11.809	794.169	98.902
		10	256.428	5.779	775.509	94.095
ADSL-1	400	5	258.871	3.572	794.539	43.033
		10	258.521	1.982	800.368	21.685
	1000	5	281.976	1.726	883.752	14.665
		10	281.969	0.904	882.600	7.081
	100	5	320.338	14.669	986.507	122.854
		10	318.531	7.179	963.327	116.884
ADSL-II	400	5	320.131	2.454	991.109	26.853
		10	319.820	1.026	100.082	8.032
	1000	5	319.829	1.958	1002.388	16.634
		10	319.820	1.026	1001.082	8.032
	100	5	27.118	18.278	38.216	19.993
		10	24.911	6.940	33.375	15.493
GPRS	400	5	22.825	5.581	27.577	10.593
		10	20.785	4.121	25.497	7.645
	1000	5	20.628	3.234	30.697	4.647
		10	20.472	3.060	30.877	3.490
	100	5	52.545	9.551	101.450	26.328
		10	56.360	7.193	117.470	23.258
UMTS	400	5	61.532	2.885	132.809	23.890
		10	62.122	1.419	126.481	11.707
	1000	5	63.158	2.894	122.392	11.805
		10	62.460	2.662	125.506	7.552

- Packets per burst: k=5 and k=10 have been chosen.
- Test conditions: No competing traffic, in order to measure BBW instead of ABBW.
- Different tests over the same link have been interleaved to concur at the same hour of the day.

RESULTS AND DISCUSSION

The results obtained in the evaluation tests are presented in Table 1. It shows the mean (μ) and the standard deviation (σ) of the BBW for each of the tests presented in the previous section.

In general, if S and k are low (S = 100 and k = 5) in order for the QoS estimations not to be very intrusive, σ increases. On the other hand, if S and k are high (S = 1000 and k = 10), σ decreases but the QoS estimations are more intrusive for the network.

The following points discuss the relevant aspects of the test results for each technology in more detail:

ADSL: Depending on the value of S used, the BBW at the IP layer (ADSL-I) varies. ADSL-II results have been obtained by taking into account ATM headers and represent bandwidth at the ATM layer. The variations in the value of μ depending on S in ADSL-I do not appear in ADSL-II. Finally, the percentage of the contract bandwidth that the ISP is really providing can be calculated. In all cases, the downlink and uplink reach almost 100% of the contract.

GPRS and UMTS: In these accesses, μ results using S = 100 are different from the rest. In order for the QoS estimation can be less reliable using this value of S. Furthermore, delay is much greater than in the previous wired accesses.

CONCLUSION

A Java-based real-time QoS estimation system specially aimed at real-time multimedia applications has been developed to evaluate Internet accesses. This system is especially useful for End users who want to estimate the quality of their Internet access and check its performance regarding the use of real-time multimedia applications. The usefulness of this system has been evidenced in the evaluation of commercial Internet accesses, since ISPs offer wide QoS ranges that can vary through the time. Evaluation results show that the number of packets per burst and the packet size have a big influence on the estimated QoS, so it is very important to study the particular technologies in depth, obtaining a suitable characterization of each access. The results obtained for ADSL, GPRS and UMTS accesses fit the expected ones, but further research is required in order to obtain a more complete characterization of these accesses.

JEQoSIM only provides QoS estimations between the end user and a central server, but real-time applications are frequently used in a peer-to-peer basis, so a modification of the application in order to take

measurements not only between the user and the central node, but also directly between two users is being considered.

REFERENCES

1. Vogel, A., B. Kerhervé, G. Von Bochmann and J. Gecsei, 1995. Distributed Multimedia and QoS: A Survey, IEEE Multimedia, pp: 10-18.
2. Curtis, J. and T. McGregor, 2001. Review of Bandwidth Estimation, Techniques, Dept. Computer Science, Univ. of Waikato.
3. Lai, K. and M. Baker, 2001. Nettimer: A Tool for Measuring Bottleneck Link Bandwidth, Proc. USENIX Symposium on Internet Technologies and Systems.
4. Downey, A.B. and C. College, 1999. Using pathchar to estimate Internet link characteristics, Proc. SIGCOMM, pp: 241-250.
5. Bandwidth Speed Test. URL:www.bandwidthplace.com/speedtest.
6. Test de velocidad del acceso a Internet, URL: www.aui.es/aui_test.
7. Schulzrinne, H., S. Casner, R. Frederick, V. Jacobson, 2003. RTP: A Transport Protocol for Real-Time Applications, Internet RFC 3550.
8. Hu, N. and P. Steenkiste, 2003. Evaluation and Characterization of Available Bandwidth Probing Techniques, IEEE J. Select. Areas Commun., pp: 879-894.
9. Bolot, J.C., 1993. Characterizing End-to-End Packet Delay and Loss in the Internet, J. High Speed Networks.
10. Cooperative Association for Internet Data Analysis (CAIDA), CAIDA Tools, URL: www.caida.org/tools.
11. Mark Crovella and Robert Carter, 1995. Dynamic Server Selection in the Internet, Proceedings of the 3rd. IEEE HPCS '95.
12. NLANR Measurement and Network Analysis, URL: http://moat.nlanr.net.
13. Wang, Z., A. Zeitoun and S. Jamin, 2003. Challenges and Lessons Learned in Measuring Path RTT for Proximity-based Applications, Passive and Active Measurement Workshop.
14. Lai, K. and M. Baker, 1999. Measuring Bandwidth, Proc. IEEE INFOCOM'99.
15. Dovrolis, C., P. Ramanathan and D. Moore, 2001. What do packet dispersion techniques measure Proc. IEEE Infocom.
16. Applet Security, URL:http://java.sun.com/sfaq/#summary.
17. Roubtsov, V., My kingdom for a good timer Reach sub millisecond timing precision in Java, "URL: www.javaworld.com"