QoS Estimators for Client-Side Dynamic Server Selection: Limitations and Keys

Javier Lafuente-Martínez
GTC-I3A-University of Zaragoza
javierlm@unizar.es

Isabel García-Muñoz
University of Zaragoza
igarciam@unizar.es

Julián Fernández-Navajas
GTC-I3A-University of Zaragoza
navajas@unizar.es

Abstract

The traditional way to cope with the increasing number of users in distributed information services is to use server replication, but additional and complementary strategies have been proposed in the last few years. In this paper, we evaluate those server selection strategies in which, by means of a packet burst sent from the client side, some QoS parameters such as delay, available bandwidth and packet loss can be estimated. In particular, our study is focused on the estimation of available bandwidth, proposing a two-step algorithm to measure it in the path to each server. Ethernet, ISDN and ADSL technologies have been used, and different scenarios and packet conditions have been tested in order to identify the limitations and design keys of bandwidth estimation methods.

1. Introduction

The increasing popularity of distributed information services in Internet causes continuous problems of scalability in those networks that support them. Actually, when the number of users of such services grows, the quality of service (QoS) decreases. The mentioned problems include: excessive load of certain servers caused by the great number of users that connect to them asking for a file or document, the bandwidth waste that entails sending the same document to several clients that share the same path to the server, or excessive delay in distributing a file when low-bandwidth connections have to be used.

One of the most extended solutions to alleviate these problems consists on replicating the servers and distributing them geographically, so that the clients will be also distributed among the servers [1]. But this solution creates a new problem: how to decide what server a client has to connect to?

The easiest solution is the so-called *static server selection* [2], in which clients are forced to connect to the same server based on fixed criteria. As it does not consider that some characteristics of the network, such as the available bandwidth for example, are variable, *dynamic server selection* [3] was thought as the possibility of not only selecting the server from which to get a file, but also having the possibility to connect to a different server if the quality of the connection with the current one degrades.

At the time of developing a solution for the dynamic selection problem, different alternatives have been proposed [4,5,6] but we consider the *client-side* ones as the most interesting from the point of view of a final user. These kinds of solutions propose that the entity which demands the service (the client) has the responsibility to find out the optimal server. This is achieved by sending certain traffic to each possible candidate and waiting for the responses. Different kinds of traffic have been used for this purpose. In [7], the authors compare six different techniques and consider one based on *tcping* as the best one. Other studies [8] use HTTP HEAD control traffic and measure its request latency. The use of *ping* and the measure of the RTT (Round Trip Time) [9,10] is also well known. A good revision of the *client-side* methods and tools can be found in [11].

In this article we try to identify the main limitations and design keys of *client-side* bandwidth estimators. To carry out our study, we have developed a two-step algorithm that could help *client-side* server selectors, by means of characterizing the path to each server in terms of available bandwidth, before taking a decision. Our method takes advantage of some existing solutions, specially those based on ICMP traffic, but includes some improvements such as the alternative of using UDP packets instead of traditional *pings* and a second step in the algorithm in order to estimate the available bandwidth. We will first compare the traditional solution based on *pings* with our UDP-based and then, despite our method is able to estimate other QoS parameters, we will centre our study in available bandwidth estimation.

The rest of the article is structured as follows. In section 2 we explain the algorithms that we have implemented and tested. A description of the scenarios used for the tests is included in section 3. In sections 4 and 5, we show the results obtained in the tested scenarios and we extract conclusions in section 6.

2. Methodology

A server selection tool should be able to evaluate from which server information could be downloaded in less time. Therefore, the aim of the methods presented in this section is to characterize the path from the server to the client in which they are executed. For this purpose, two alternatives have been studied:

- *ICMP based:* The traditional method for estimating link capacity is based on sending bursts of ICMP request (*ping*) packets to each server and waiting for the reception of their corresponding ICMP reply packets.
- *UDP based:* Due to the tendency of not allowing the incoming ICMP traffic at the servers for security reasons, we propose a second method based on sending bursts of UDP packets to a port that does not accept this type of traffic. Thus, the server generates *port-unreachable* ICMP packets towards the client that are used to make the corresponding estimations of the downlink properties.

In both cases, we use the ICMP traffic received from the server to estimate the delay, the packet loss, the bandwidth of the *bottleneck* and the available bandwidth. A server selection methods should take a decision based on these four parameters after associating different weights to them.

1) *RTT* can be estimated by calculating the average of the RTT for each packet sent in the burst. As this parameter is affected by both the uplink and downlink, it gives us information about the delay that the devices introduce in the path and the degree of server load.

2) The estimation of the *percentage of packet loss* can be calculated as the percentage of lost packets in the burst.

3) Given a path between a client and a server that includes several links $L_1,\ldots,L_n$ with capacity $B_1,\ldots,B_n$, the *bottleneck bandwidth*, which is estimated in the first step of the algorithm presented below, could be defined as [12]:

$$bottleneck\ bandwidth = min\ (B_1, B_2, \ldots B_n) \qquad (1)$$

4) Given a link L_i with capacity B_i and traffic load C_i, the *available bandwidth* in the link is defined as:

$$available\ bandwidth_i = B_i - C_i \qquad (2)$$

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 (Figure 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* (Figure 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.

$$bottleneck\ bandwidth = ((n-1) \times packet\_size)/(t_n - t_1) \qquad (3)$$

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 effect, 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 [9] show that using five packets by burst allows reaching a trade-off between good bandwidth estimation and little bandwidth required for the estimation method. In section 4 we have assumed this value and we have carried out tests for different packet sizes, obtaining conclusions about the suitable size for the packets in the burst.

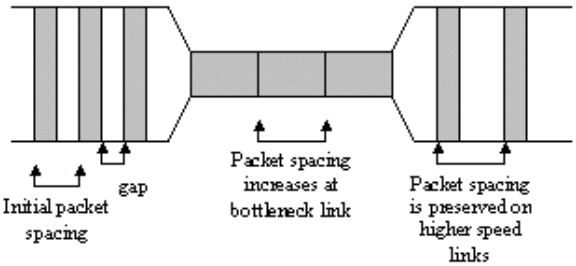

Figure 1. First step: packet burst through a bottleneck link

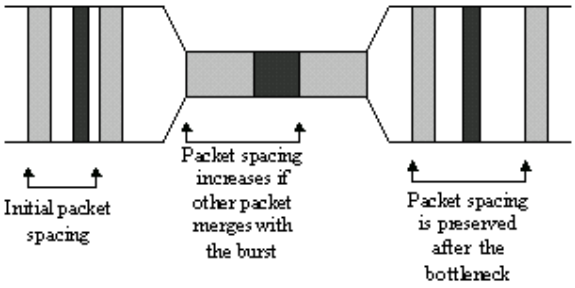

Figure 2. Packet burst sent in the second step

3. Test scenarios

In order to test the performance of our two-step algorithm, we study two scenarios. We have decided to test technologies commonly used to access Internet, such as ISDN (Integrated Services Digital Network) and ADSL (Asymmetric Digital Subscriber Line).

3.1. Scenario 1

As shown in Figure 3, this scenario is composed of three Linux-based computers. The client is responsible for sending the test burst towards the server, running the *client-side* algorithm. The server generates ICMP packets in response to the received burst, depending on the kind of traffic received: ICMP *echo reply* for *pings* or ICMP *port unreachable* for the UDP packets. We implement the *bottleneck* link as a 64kbps ISDN link between the router and the server.

Some of the tests are carried out with competing traffic. This traffic is sent from the server to the router. The reason is two-fold: we are interested in the characterization of the critical link, the *bottleneck*, and we consider that the presence of competing traffic in the way from the server to the client is the most interesting case, as it emulates a server sending information to clients that share links.

In this scenario, we have first studied how UDP and TCP traffic compete. Then, we have tested the influence of UDP and TCP competing traffic in our algorithm results.

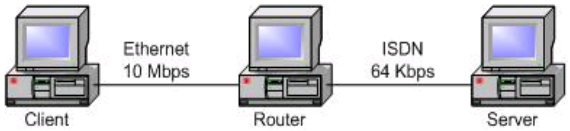

Figure 3. Scenario 1

3.2. Scenario 2

The interest of this scenario, shown in Figure 4, is two-fold. First, unlike the previous scenario, the path is asymmetric in terms of bandwidth. The ADSL connection used in this scenario offers 128kbps in the uplink (the way from the client to the server) and 256kbps in the downlink (the way from the server to the client). This is a particular case where a problem affecting *client-side* estimators appears: they are unable to detect if the *bottleneck* is in the way from the client to the server or in the opposite way. Secondly, we discuss the influence of the packet size in bandwidth estimations in the case of transmitting over ATM (Asynchronous Transfer Mode), which is the ADSL underlying technology.

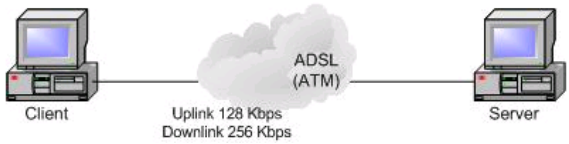

Figure 4. Scenario 2

4. Scenario 1: Results

By understanding how UDP and TCP traffic compete, we could understand the influence of competing traffic in the traffic we inject in the network to take our measurements and vice-versa. In this section we centre our study on link capacity and available bandwidth estimation.

4.1. How UDP and TCP traffic compete

In order to run this test, the server starts to transmit UDP packets at different rates to the client, that try to compete with a *ftp* (TCP) transmission running from the server to the computer which is acting as a router. We evaluate the ISDN link by means of *tcpdump*.

The bandwidth distribution in the case of sending 240-Byte UDP packets is shown in Figure 5. As we can see, in absence of UDP traffic, the *ftp* transmission is obtaining all the capacity of the link. As we increase the UDP packet rate, this kind of traffic obtains more bandwidth. We have verified that, when UDP and TCP traffic compete, TCP obtains the bandwidth that UDP is not using. The reason is that TCP implements flow control but UDP does not. Thus, TCP is able to adapt its transmission rate to the variations detected in the available bandwidth.

We have also tested the case in which two UDP transmissions compete or two TCP transmissions compete. The results reveal that the bandwidth sharing is proportional to the product (*packet_size x packet_rate)* obtaining 50% of the link capacity when their rates are the same.

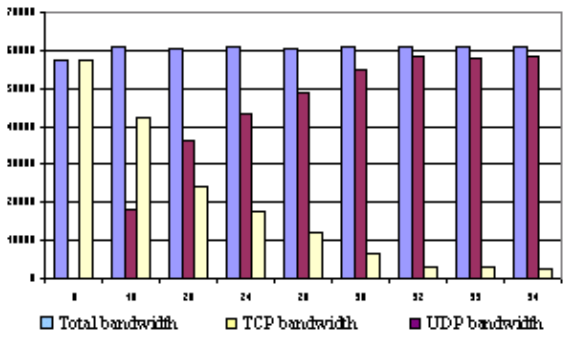

Figure 5. UDP competing with TCP traffic

4.2. Results of the algorithm in absence of competing traffic

In absence of other traffic, we have first compared the capacity of both our *UDP-based* method and the *ICMP-based* to estimate the *bottleneck* capacity (the ISDN link in this scenario), that is, we test the first step of our proposed *two-step* algorithm. We have studied the effect of varying the size of the packets that we send in the burst.

The values shown in the graphs are obtained as the mean of the estimation of several bursts with the same characteristics. Each burst is composed of five packets with the same size. As explained in section 2, the client sends the burst to the server and waits for the reply to each packet in the burst. Then, it can calculate the capacity of the link. Depending on the algorithm used, the packets sent in the burst are ICMP or UDP. In both cases, we have tested the following packet sizes: 60 Bytes, 120 Bytes, 240 Bytes and 296Bytes. As we are testing the first step of our algorithm, we send the packets with the minimum packet spacing (maximum packet rate).

In Figure 6 we compare the bandwidth estimated by the algorithms after the first step with the bandwidth obtained by sending only UDP traffic at different packet rates. We have verified that, despite the packets are sent at a higher rate from the sender, our ISDN pc-cards can process no more than 32 pps (packets per second), that is, the minimum packet spacing we can obtain is around 31ms in the ISDN interfaces. As shown in Figure 7, if we send 60-Byte packets, the maximum bandwidth occupied in the ISDN link is less than ¼ of the link capacity (64kbps). Thus, when we use 60-Byte packets in the test burst, we obtain an estimation around 16kbps. We can also see that the maximum precision could be obtained by using large packets with a size not higher than the MTU (Maximum Transmission Unit). The optimum value would be 250 Byte because the test burst occupies all the bandwidth during less time, being less intrusive.

The MTU depends on the underlying technology and could be defined as *the largest packet size that can be sent in a packet-based network without being segmented.* The default MTU in ISDN connections is 296 Bytes.

As expected in absence of competing traffic, the second step of our algorithm estimates an available bandwidth similar to the link capacity. Results for our UDP algorithm are shown in Table 1.

In summary, the results show that *UDP-based* and *ICMP-based* estimations are similar. Moreover, using 250-Byte packets is the best choice because represents a trade-off between the accuracy of the estimation and the length of the packets in the burst.

Table 1. Bottleneck capacity and available bandwidth estimations

Packet size	*60 Bytes*	*120 Bytes*	*240 Bytes*	*296 Bytes*
Bottleneck capacity (bps)	15750.60	31446.39	54055.77	55301.61
Available bandwidth(bps)	15340.94	29521.40	55458.70	57103.50

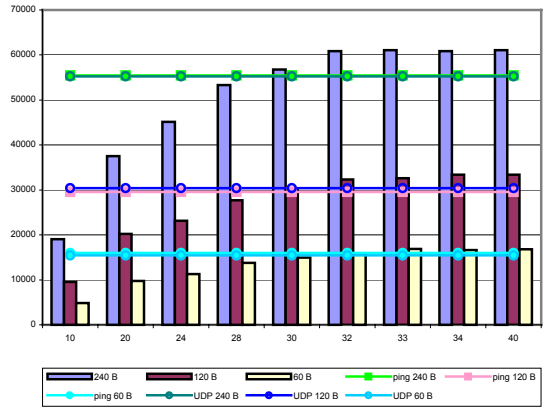

Figure 6. Bandwidth estimation using ICMP or UDP packets in the burst in absence of competing traffic vs. sending UDP traffic only

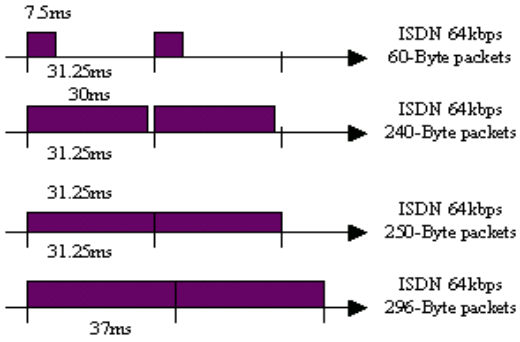

Figure 7. Importance of packet size in bandwidth estimations

4.3. Influence of competing traffic

When we introduce traffic in the ISDN link, it has to compete with the burst we send. In this subsection, we take advantage of the results just obtained in order to centre in identifying other keys and problems. Thus, we use 250-Byte UDP packets in the test burst.

We first run an *ftp* (TCP) transmission from the server to the computer that is acting as a router. This traffic uses all the capacity in the ISDN link from the server to the client. When the client is receiving the ICMP *port-unreachable* responses to the UDP packets of the test burst, we detect a decrease in the amount of bandwidth assigned to the *ftp* connection. The reason is that, as was extracted from Figure 5 in the case of UDP, ICMP does not implement flow control but TCP does. This implies that the ICMP *port-unreachable* packets in the burst are not affected by existing TCP traffic, but ICMP traffic affects TCP transmissions. Results in Table 2 are explained in Figure 8: in the first step of the algorithm, we send the test packets as close as possible, being difficult for the TCP packets to merge with the burst. In the second step, we send the test packets at the rate estimated in the first step. As the new packet rate is lower, some TCP packets can merge with the burst. T1 and T2 are the value of the expression (t_n-t_1) used in (3), for the first and the second step of the algorithm respectively.

Table 2. Estimations in presence of TCP traffic

Packet size	*250 Bytes*
1st Step: Bottleneck capacity (bps)	60338.54
2nd Step:Available bandwidth (bps)	40626.16

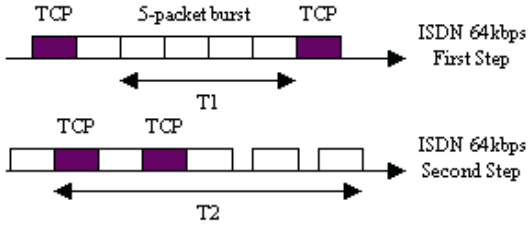

Figure 8. IDSN bandwidth sharing in presence of TCP competing traffic

Then, in presence of TCP competing traffic, we can estimate the *bottleneck* bandwidth but the available bandwidth estimation is not valid because, from the results in subsection *4.1*, we know that the bandwidth available for a desired transmission depends on the kind of traffic. Therefore, if the traffic sent from the server to the client use UDP, it can use more than the estimated bandwidth (the competing TCP traffic will vary its transmission rate). On the other hand, if the desired traffic uses TCP, the bandwidth sharing will be proportional to the product (*packet_size x packet_rate)* of those TCP transmissions.

If our burst has to compete with UDP traffic, which does not implement flow control either, our estimations become more affected as the UDP packet rate is increased. We must remember that in those situations the bandwidth sharing is proportional to the product (*packet_size x packet_rate)* and that our ISDN pc-cards cannot process more than 32 pps.

Table 3. Estimations in presence of UDP traffic

UDP competing packets	*60 Bytes 10 pps (4800 bps)*	*60 Bytes 20 pps (9600 bps)*	*250 Bytes 10 pps (20000 bps)*	*250 Bytes 20 pps (40000 bps)*
1st Step: Bc (bps)	56047.33	44944.35	44832.36	35134.40
2nd Step: Ab (bps)	52875.63	38002.10	43645.65	25412.71

Then, in presence of UDP competing traffic, we can obtain a good estimation for available bandwidth if the size of the competing traffic and burst packets are similar but in any case, an acceptable estimation is obtained. Anyway, the bottleneck bandwidth estimation is not valid because the traffic in our burst is also UDP and it is difficult to avoid packet mixing.

5. Scenario 2: Results

In this section we identify the main problems that a client-side estimator encounters if an asymmetric path or a technology that uses fix-size frames exist.

5.1. The asymmetric path problem

The detection of a *bottleneck* in the path is relatively easy, but knowing if it affects the traffic going to the server or coming from it is not. We consider this as a limitation of *client-side* methods when the path between client and server is asymmetric. This is the case of ADSL in which, due to communications between clients and servers are usually asymmetric (the server sends the information requested by the client, who usually only have to send back some control traffic to maintain the communication), the uplink offers less bandwidth than the downlink. In such situations *client-side* methods could detect the uplink as a *bottleneck*, in spite of the capacity of the downlink is higher, resulting in a wrong estimation of the available bandwidth.

5.2. Influence of technology in the results

In this scenario ATM is the underlying technology. As the transmitted frames have a fixed size of 53-Byte

(The header is 5 Bytes long and the remaining 48 Bytes can be filled with data), the size of the packets we generate in the test burst is important. If a frame is not completely filled with data, the bandwidth estimation will not be as good as desired.

In Figure 9 we show the bandwidth estimations versus the size of the test packets. Depending on this size, one or more ATM cells are filled with data.

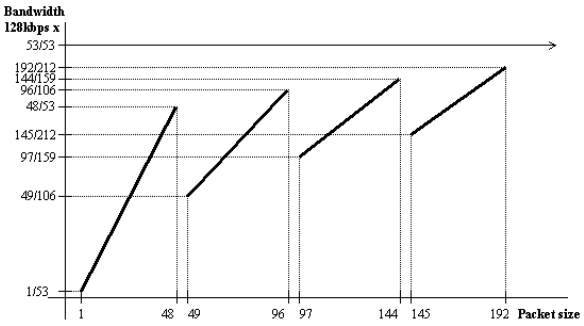

Figure 9. Bandwidth estimation vs. Packet size

6. Conclusions

We have proposed a complementary solution for helping client-side server selection methods to estimate available bandwidth in the path to each server and we have identified the main keys and limitations of those client-side methods.

The main limitations we have identified are:

- Estimations depend on the size of the packets sent in the test burst. A good choice of the size for those packets depends on technology particularities.
- Estimations depend on the type of competing traffic as long as on the size of its packets.
- Client-side estimators are not reliable when tested in asymmetric-path connections.

These limitations allow us to define the main keys in order to design a client-side bandwidth estimator:

- It should adapt the type of the packets in the test burst to the type of both the competing traffic and the traffic the client wants to discharge from the server.
- It should adapt the size of the packets in the test burst to the MTU of the *bottleneck* link.
- The higher the number of packets included in the burst and the larger their size, the more accurate the estimation but the more intrusive the method.

Further work is being done in order to develop a complete tool, capable of adapting the measurement method to the kind of traffic desired to download (UDP, TCP, HTTP) and able to dynamically switch from one server to a better one when the QoS of the communication decreases significantly.

7. Acknowledgement

This work has been possible thanks to the CICYT projects (TIC2001-2481 and TIC2002-04495-C02), financed by FEDER and the Spanish Ministry of Science and Technology, and the project (FISG03/117) financed by FIS.

8. References

[1] J.D. Guyton, M.F. Schwartz, “Locating nearby copies of replicated internet servers”, Proceedings of SIGCOMM’95, August 1995.

[2] M.Conti, E.Gregory, F.Panzieri, “Load Distribution among Replicated Web Servers: A QoS-Based Approach”, WISP 1999, ACM Press, Atlanta (GA), 1999.

[3] Robert L. Carter, Mark E. Crovella, “Server Selection using Dynamic Path Characterization in Wide-Area Networks”, INFOCOM’97, 1997.

[4] Ellen W. Zegura, Mostafa H. Ammar, Zongming Fei, Samrat Bhattacharjee. “Application-layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service”. IEEE/ACM TRANSACTIONS ON NETWORKING, vol. 8, n. 4, August 2000.

[5] D. Andersen, T. Yang, V. Holmedahl, and O. H. Ibarra. “SWEB: Towards a scalable World Wide Web server on multicomputers”, in Proc. of the 10th Int’t. Parallel Processing Symp. (IPPS’96), Apr. 1996.

[6] E.D. Katz, M. Butler, and R. McGrath, “A scalable http server; the NCSA prototype”, Computer Networks and ISDN Systems, vol.27, pp.155-164, 1994.

[7] Sandra G. Dykes, Kay A. Robbins, Clinton L. Jeffery. “An Empirical Evaluation of Client-side Server Selection Algorithms”. IEEE INFOCOM, vol. 3 , PP.1361-1370, March 2000.

[8] Marco Bernardo, “A simulation Analysis of Dynamic server Selection Algorithms for Replicated Web Services”, MASCOTS 2001, 371-378, August 2001.

[9] Mark Crovella, Robert Carter, “Dynamic Server Selection in the Internet”, Proceedings of the 3rd. IEEE HPCS '95, 1995.

[10] Allen B. Downey, “Using pathchar to estimate Internet link characteristics”, Proceedings. of SIGCOMM’99.

[11] K. Lai, M. Baker, “Nettimer: A tool for Measuring Bottleneck Link Bandwidth”, Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 123-134, April 2001.

[12] Ningning Hu, Steenkiste, P, “Evaluation and characterization of available bandwidth probing techniques”, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, vol.21, N.6, pp. 879-894 August 2003.

[13] R. Carter and M. Crovella, “Measuring Bottleneck Link Speed in Packet-Switched Networks”, Boston Univ., Comput. Sci. Dept., Tech. Rep., Mar. 1996.