Network Monitoring with Nprobe

Andrew W. Moore, Rolf Neugebauer, James Hall, Ian Pratt

Abstract— **This paper presents an architecture for monitoring 10 Gbps networks, drawing on experience from a current 1 Gbps implementation. The architecture performs full line-rate capture and implements on-line analysis and compression to record interesting data without loss. The use of the monitoring and analysis system is demonstrated along with the presentation of several initial findings made possible with the tool.**

Keywords— **Passive network monitoring, full line-rate capture, multi-protocol analysis**

I. Introduction

A RECENT National Academy of Sciences report on research horizons in networking identified the measurement of network infrastructure as a grand-challenge of critical importance for the computing community in general and networking community in particular [1]. A crucial step toward realising this goal is the construction of tools capable of full line-rate capture and analysis within high-speed networks.

Due to the ongoing development of new applications (e.g., file-sharing) and the continual growth in new users, past measurements of the Internet fail to provide significant insight into its current and future usage patterns, nor do they provide insight into the usage patterns of other new network infrastructures such as the Grid. Comprehensive measurements and historical records either do not exist or what exists is restricted in availability, detail or relevance [1], [2]. A common feature among these limitations is a lack of a suitable measurement infrastructure.

Yet while current historical measurement-records offer only cursory assistance, the task of monitoring and measuring network infrastructures such as the Internet has also become crucial for maintaining a network that is able to support the increasing variety of services, users and size that is predicted and desired. Moreover, monitoring tools allow the study of such phenomena as long-term infrastructure and user trends as well as helping to explain short-term disruption. Alongside providing an input to such studies, any monitoring and analysis mechanism must be able to scale with the Internet as it grows in size, capacity and use. The design and development of a high-performance monitoring platform with sufficient scalability and flexibility to accommodate current and future services is a significant research challenge.

A Moore and R Neugebauer are with the Marconi Research Laboratory, Cambridge. E-mail: {andrew.w.moore,rolf.neugebauer}@marconi.com J. Hall and I. Pratt are with the University of Cambridge Computer Laboratory. E-mail: {james.hall,ian.pratt}@cl.cam.ac.uk .

We postulate that it is only through empirical analysis of monitored traffic that an accurate understanding of a complex system, such as the Internet, may be formulated. Further, we assert that full line-rate data capture is the only realistic way to enable the multitude of different analysis needed to understand the Internet.

In this paper we describe a network monitoring architecture currently operating at 1 Gbps. In Section II, the current full line-rate monitoring architecture along with our approach for working with 10 Gbps networks is described. Using the 1 Gbps prototype, we demonstrate a characterisation of web traffic and the results of this characterisation are presented in Section III.

A. Context

Measurement techniques such as those used in the Internet can be loosely categorised into active and passive types. Each technique has its place, its advantages and its disadvantages. Examples of active measurement network monitoring, includes that commonly undertaken using programs like *ping* and *traceroute*.

Passive monitoring has the advantage of being non-invasive — there is not need to modify end-systems or introduce additional traffic into the network. Passive monitoring is able to observe both: a more representative sample as compared with end-system instrumentation, and to observe subtle effects, relationships between implementations of end-systems, and the subtle relationship between different protocol layers from application through transport and to the network.

Passive network monitoring has seen a large number of previous approaches. An evolution of approaches has been spawned by the growth in network speeds. Examples of monitoring include packetfilter [3], libpcap [4], OC3MON [5] as well as the more recent work of DAG [6]. New projects in this area are also getting underway such as the European SCAMPI [7]. Either using or improving upon these approaches have been fundamental to various projects under the auspice of CAIDA [8], as well as research projects such as WAMA [9].

While passive network monitoring is not a new tech-

nique, performing full line-rate capture and analysis of all data is less common. The full line-rate capture/analysis of all data on high-speed wide area networks provides for a wide-range of known and potential applications.

Importantly, the ability to perform multi-layer, multi-protocol analysis enables the relationship between application-layers such as HTTP/HTML, and their network layers (TCP/IP) to be established. This means that researchers can ask questions such as "what is the level of usage of persistent connections?", "how browser-specific is the use of parallel pipelining of HTTP requests?", or "how do application-level QoS metrics such as full-page download times vary depending on network load?". Such questions can only easily be asked of a multi-protocol/multi-layer system as the one we describe here.

In addition to offering a new approach, by performing capture at full line-rate such a versatile system can perform a number of tasks traditionally performed by separate systems.

Three examples of the use of full-line rate capture, each at quite different timescales include: duplicating the functionality of the DAG systems [6]. The DAG systems, as commonly used [10], [11], [12] capture the header of packets with high precision. A second example is how capture at full-rate can be used to replicate the task currently fulfilled by analysis of HTTP proxy logs, such analysis have provided justification for significant changes in the HTTP protocol in the past [13].

While, a final example, is at the largest time-scale, capture at full-rate could provide the basis of a better, or at least alternative, route analysis system. Such capture would provide insight into the relationship between the timing and transmission of this data, its availability to peer routers and, with access to actual packet-behaviour, the relationship this shares with global routing behaviour.

Of course passive monitoring does have its disadvantages too, for the context of this paper a significant disadvantage is having to interface with state-of-the-art network interfaces that are operating at 10 Gbps. Such high speeds require a careful approach to the architecture of the monitoring system and to the fire-hydrant of data that can be generated.

As indicated in the introduction, the work described by this paper uses a network monitoring and analysis system that is capable of full line-rate data capture.

II. The Nprobe architecture

In addition to network speed, a monitoring architecture must address other issues. The physical structure of the Internet typically leads to an approach that requires a large number of deployed monitoring systems. In addition to network speed there are also interesting problems relating to the control of these distributed monitoring systems, and the management of the mass of data they collect. We acknowledge these complexities however we do not address them in this paper.

Our approach to the need for a large number of deployed systems is to base such systems upon commodity (mass-produced) hardware. This contrasts with studies and systems that use specialised equipment (e.g., [11], [12]). We consider such an approach the only feasibly option as the creation of a widely-distributed monitoring and analysis system is financially untenable using any significant number of purpose-built alternatives.

The use of commodity machines as the basis of network monitoring is not unique. What makes the commodity-machine approach so demanding is that, without the use of speciality hardware, current systems are unable to keep pace with fully utilised network links. The approach we use is to apply a combination of innovative techniques to high-speed monitoring and real-time data reduction in order to manage the potentially overwhelming quantity of data [14]. In particular we use flow-based workload splitting between individual machines in a 'monitoring cluster', and between individual CPUs on those machines. We minimise buffer management overheads through use of a modified OS kernel. We achieve a significant reduction in the data to be written to disk through real-time compression and extraction of the detailed features of interest.

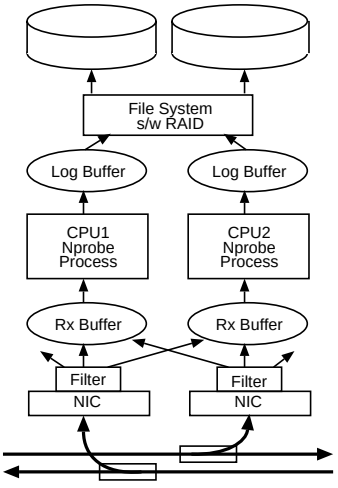

Fig. 1. Major Elements of the Nprobe Architecture

A. Current Prototype

Figure 1 illustrates the major elements of the current Nprobe prototype. The current implementation runs over the GNU/Linux operating system, with modifications made to the kernel networking and memory management

code to support the efficient passing of network buffers to and from user-space. Any network interface supported by Linux can be used with Nprobe, though in some cases we have chosen to modify the device driver (and even the interface firmware) to improve performance or add instrumentation.

Current deployed systems are dual Pentium PIII 500MHz machines, with storage provided by a software RAID array of high-capacity IDE disks. This system performs full line-rate capture/analysis on 1 Gbps (full-duplex) Ethernet networks. We presented the underlying architecture along with some results gained using this architecture in [14]. Salient points of this architecture are described below.

Importantly, the approach we use applies a degree of real-time processing of network, transport and application data to enable compression in both the spatial and temporal domains. This architecture must balance the overheads of CPU processing with the quantity of data that must be written to disk, and these in turn are further balanced against the degree of on-line and off-line processing.

The architecture of this prototype uses a modular approach to traffic processing. The traffic processing modules consist of IP processing module, TCP processing and analysis module, and application layer processing module(s).

The IP module performs check-sums and maintains basic network-usage statistics. The TCP module associates arriving packets by connection and extracts sufficient data from each header to perform later modelling of the connections' dynamics. Missing and duplicated segments and retransmissions are identified and the TCP byte stream reassembled on-the-fly before being presented to higher level protocol modules.

For the results of this paper, the application layer process consists of HTTP processing and analysis followed by HTML parsing — all these activities occur in real-time. The off-line phase of the analysis extracts the pertinent data and passes it to analytic modules specific to the study being carried out.

Once in the application processing module, HTTP headers are parsed and the information extracted is used to follow persistence negotiations and if necessary to identify request/response boundaries on persistent connections. The HTTP data is then parsed by an HTML module to extract the links they contain. In this way information on the browsing patterns, as well as inter-object referencing, etc. are differentiated and can later be used to aggregate the multiple transactions used to fetch a complete page. A similar module exists to track FTP flows, along with UDP-based modules that extract data from DNS exchanges or streaming media flows.

Our current Nprobe installations are attached to 1 Gbps Ethernet access networks, where they have been able to comfortably keep pace with the link traffic present. The resources used in performing the analysis are highly dependent on the traffic mix: Performing a full parse of a HTML document to extract links and image references requires considerably more resources than dealing with a JPEG object, for which we simply calculate and record an object fingerprint. For the traffic mix present on the links we currently observe, Nprobe achieves a typical data reduction of 12:1 when operating at a high level of recorded detail. This makes the task of streaming the information to disk considerably less challenging, though we are careful to streamline the task.

B. Future Directions

Currently, the Nprobe implementation is capable of 1 Gbps full line-rate capture, but our next objective is 10 Gbps. We have considered several approaches and contact with members the DAG [15] team reveal common approaches to a number of the problems faced.

Clearly, the monitoring machine cluster approach developed for 1 Gbps networks is going to be an essential at 10 Gbps. We are evaluating two approaches to distributing the load across the cluster: One relies on using a distribution amplifier to feed the network link to multiple specially built 10 Gbps Ethernet PCI-X NICs located in different machines. The NICs would perform on-board flow-based hash filtering, just as we do for 1 Gbps today. The alternative is to develop a simple FPGA-based board to perform time stamping and flow-based demultiplexing of packets to 10 or more of our current 1 Gbps Nprobe systems.

We currently favour the latter approach for reasons of cost, and believe that the approach is viable for the kinds of network link we wish to monitor, where there are likely to be many thousands of flows rather than a few large ones. Further work and experimentation is required to determine how much buffering we will need in front of each 1 Gbps port, and how many ports will be required to reduce loss to an acceptably small level.

In the next section the current implementation is demonstrated using a characterisation of web traffic.

III. Demonstration of Web Characterisation

The characterisation of web users provides raw data for the creation of web load generators. Web load generators are an important part of the testing and comparison of web servers [16], [17]. Given that web traffic typically accounts for a substantial percentage of the total Internet traffic, the

resulting web load generators can also provide highly effective network load generators [18].

For the web server provider, the browser builder and the network provider, the characteristics of the current web use provides a critical input to assist in the prediction of performance and provision for the changing utilisation of services.

In this section we present results gained using the prototype 1 Gbps Nprobe system described in the previous section. The results presented here are for the analysis of data collected over one period in January 2002. The monitoring point was placed at the connection between the University of Cambridge and the wider Internet.

Our original objective was to provide an update to the characterisation at the foundation of the SURGE work [16], however we found that HTTP protocol usage has changed so significantly that simply repeating their methodology with new data would no longer be appropriate. We are in the process of creating an updated model, but here, we present a few key observations.

A. Background

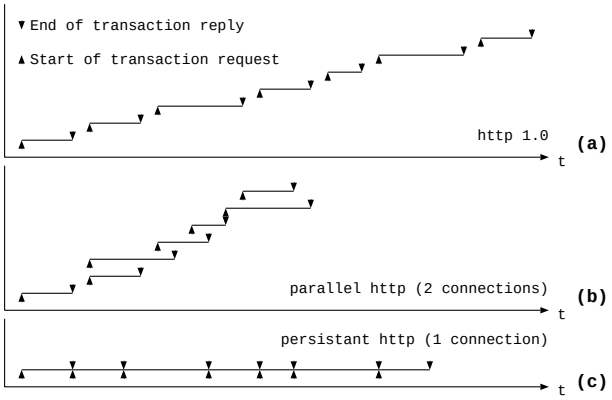

Fig. 2. Protocol behaviour in web transactions

Page downloads using HTTP/1.0 [19] suffer from the problem that each constituent object must be downloaded using a new TCP connection. This is illustrated in Figure 2(a). Each object download is thus subject to TCP connection latency and commences under slow start conditions. Additional load is placed upon the network, and the server has to service and maintain a large number of individual connections.

Browser implementations would actively overcome this limitation by opening a number, often 2, simultaneous connections to the server. Thus objects could be retrieved in parallel. Figure 2(b) illustrates this. Further improvements to overcome the shortcomings in HTTP/1.0 were suggested by Mogul and Padmanabhan [13]. Of particular relevance were *persistent* HTTP connections (P-HTTP), and *pipelining* of multiple object requests over a single connection. Experimental data supported their recommendations and showed a significant reduction in overall latency for retrievals. The use of persistent connections is illustrated in Figure 2(c).

A following paper by Mogul [20] demonstrated the reduction in latency that can be achieved by using P-HTTP and pipelining. Extensive simulation also demonstrated the reduction in use of server resources under TCP TIME-WAIT states of various durations.

HTTP/1.1 [21] published in January 1997 incorporated persistent HTTP connections and pipelining, together with a negotiation protocol allowing for persistence negotiations between HTTP/1.0 and HTTP/1.1 clients and servers.

It becomes clear that the simpler HTTP/1.0 model whereupon every object is retrieved on its own TCP connection, the model used in generators such as SURGE [16], can no longer accurately characterise web activity.

B. Object Sizes

While the mechanics of the HTTP protocol have changed, ultimately it still retrieves web objects; one per request. This section presents our findings of the change in the distribution of object sizes and we compare our results with those reported by Barford & Crovella [16].

Barford & Crovella publish an approximation of requested object sizes based upon the analysis of results gained using an instrumented web browser. This instrumented browser was deployed across a subset of users at an academic site for parts of 1994 and 1995 [22].

They characterise the requested object size with the Pareto generating function $p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$, where k=1000 and α=1.0. The cumulative distribution of this generating function is illustrated in Figure 3 as a comparison with recent results.

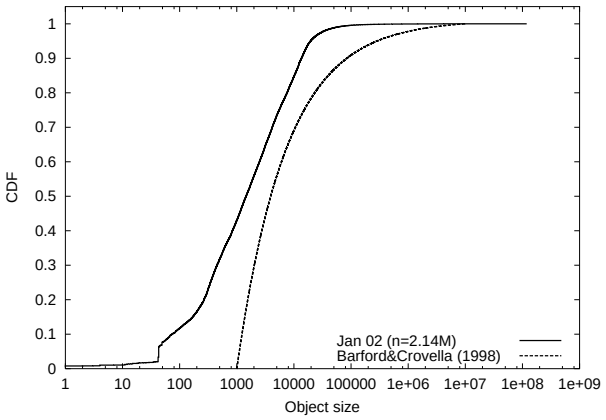

Fig. 3. Cumulative Distribution of Object Sizes (log scale for x axis)

Figure 3 illustrates both how the distribution of object size as returned as part of a HTTP response has reduced by near an order of magnitude between the results of [16] and those recently analysed.

Table I provides the mean response header size and the mean object size for the trace taken in January'02. We also examined the difference between the object size and the response size. The object size is carried by the HTTP protocol such as an image or a page of text. The response size is carried by the TCP protocol: the object plus the HTTP header. Our results indicated that the two distributions were near identical, in shape, differing in that the cumulated distribution of response sizes was approximately 272 bytes larger than the object distribution.

Table I also provides the mean values of request headers and request payloads for all types of requests. A mean size of the request header is then provided for GET requests only. We would assert that these values will provide a valuable comparison with the historical record indicating the growing size of requests. Additionally, other request types, such as POST, clearly have a higher mean size which would cause the difference in average size between the values for all request headers and GET request header only.

TABLE I
MEAN REQUEST, RESPONSE AND OBJECT SIZES (IN BYTES).

	Mean Size
Response header	272
Object (payload)	3,841
Request header	521
Request payload	81
GET requests only	505

C. HTTP connection types

The characterisation of object sizes could also be done by another system (e.g., the analysis of proxy logs). However, the Nprobe system can perform multi-protocol analysis, revealing the interaction between application and transport layer. An example of this analysis is the different use of TCP flows by HTTP requests.

As noted earlier HTTP/1.1 allows both persistent connections and object pipelining. Table II reveals the percentage of total connections that were persistence-capable (HTTP/1.1 vs. HTTP/1.0), and the percentage of total connections that were actually persistent.

These figures are also given for transactions. A transaction can include successfully transferring an object, attempting to retrieve a non-existent object, and finding that an object has not changed since last it was retrieved. The figures for all transactions are given in Table II: the percentage of total transactions moved over connections that were persistence-capable, and transactions moved over persistent connections.

The results of Table II clearly illustrate that the use of persistent transactions is low (10.4%).

TABLE II
PROTOCOL USE: PER-TRANSACTION AND PER-CONNECTION

	Trans's	Conn's
HTTP/1.1	33.3 %	28.6 %
persistent	24.4 %	10.4 %
total	n≈3.68M	n≈1.78M

Figure 4 further illustrates these results by showing the probability distribution of the number of transactions made over persistent connections. This figure illustrates that about 40% of persistent-capable connections transfer just a single object, while less than 5% of persistent connections transfer 5 or more objects.

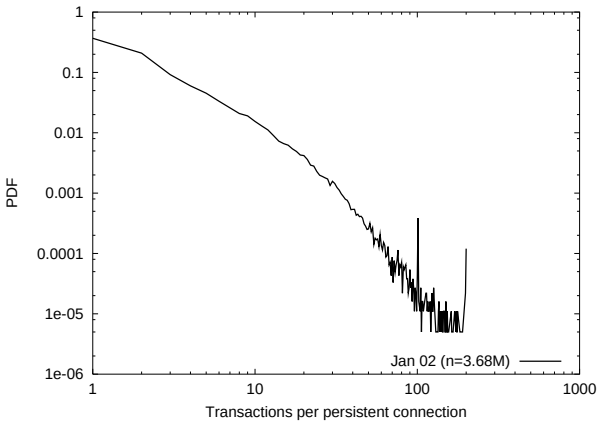

Fig. 4. Probability Distribution of Transactions per Persistent Connection (log scale for x and y axis)

Our analysis revealed that only a minuscule proportion of persistent connections employ pipelining. These were found to be produced by 'bleeding-edge' Mozilla and Opera browsers, whose users had manually enabled this feature. Surprisingly, it appears that a significant number of servers become confused by pipelined requests, either issuing a TCP RST, or simply ignoring the pipelined request. The latter is particularly unfortunate, causing the browser to timeout.

D. Conclusion

With the results of this section we have illustrated operation and characterisation of a 1 Gbps network flow using

the Nprobe system. This characterisation has illustrated results obtained by our system that replace those traditionally gained by other approaches, and our results illustrate the effectiveness of our approach in providing new characterisations.

The results of this section are a demonstration, they are neither meant to be comprehensive or representative, however they provide an illustration of the kinds of application-layer analysis that can be performed by Nprobe.

It is worth noting that the whole task of off-line processing of the data was constructed in 10's of lines of Python code. At collection, the Nprobe system has performed analysis on-the-fly thus the off-line phase of the analysis need only extract the pertinent data and pass it to analytic modules specific to the study being carried out. This flexible interface to the monitoring/analysis may be regarded as Nprobe's greatest asset.

IV. Summary

In this paper we have presented the current and future developments of Nprobe, our network monitoring and analysis system based upon full line-rate capture and analysis.

We have illustrated an approach valid for the next step in high-speed networks, such as those at 10 Gbps.

We have provided results that illustrate the possibilities of a multi-protocol analysis, such as is provided by our prototype. Further to presenting results gained by our prototype, we illustrate the importance of such results in maintaining the validity of both traffic generators and generators of load for specific applications.

Thanks

Our thanks to Ian Graham for his valuable conversations.

References

[1] David A. Patterson, David D. Clark, Anna Karlin, Jim Kurose, Edward D. Lazowska, David Liddle, Derek McAuley, Vern Paxson, Stefan Savage, and Ellen W. Zegura, *Looking Over the Fence at Networks: A Neighbor's View of Networking Research*, Computer Science and Telecommunications Board, National Academy of Sciences, Washington, DC, 2001.

[2] K.C. Claffy, "Internet Measurement: Myths About Internet Data," in *Proceedings of the 24th North American Network Operators' Group (NANOG24)*, Feb. 2002.

[3] Jeffrey C. Mogul, "Efficient use of workstations for passive monitoring of local area networks," in *Proceedings of ACM SIGCOMM'90*, Philidelphia, PA, Sept. 1990.

[4] "tcpdump/libpcap," 2001, http://www.tcpdump.org/.

[5] J. Apisdorf, K. Claffy, K. Thompson, and Rick Wilder., "Oc3mon: flexible, affordable, high performance statistics collection," .

[6] I. D. Graham, S. Donnelly, J. Martens S. Martin, and J. Cleary, "Nonintrusive and accurate measurement of unidirectional delay and delay variation on the internet," in *Proceedings of the INET'98 Conference*, July 1998.

[7] Lieden University (in collaboration), *SCAMPI, a Scalable Monitoring Platform for the Internet*, Mar. 2002, http://www.ist-scampi.org, http://www.liacs.nl/ herbertb/projects/scampi/.

[8] San Diego Supercomputing Center (SDSC), University of California at San Diego (UCSD), *CAIDA, the Cooperative Association for Internet Data Analysis*, Dec. 2001, http://www.caida.org.

[9] P. Barford and M. Crovella, "Measuring Web Performance in the Wide Area," *Performance Evaluation Review, Special Issue on Network Traffic Measurement and Workload Characterization*, Aug. 1999.

[10] J. Micheel, Ian Graham, and N. Brownlee, "The auckland data set: an access link observed," in *Proceedings 14th ITC Specialist Seminar*, Barcelona, April 2000.

[11] Nevil Brownlee, K. C. Claffy, Margaret Murray, and Evi Nemeth, "Methodology for passive analysis of a commodity internet link," in *Proceedings of A workshop on Passive and Active Measurements*, Amsterdam, Apr. 2001.

[12] G. Iannaccone, C. Diot, I. Graham, and N. McKeown, "Monitoring very high speed links," in *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, Nov. 2001.

[13] Venkata N. Padmanabhan and Jeffrey C. Mogul, "Improving HTTP latency," *Computer Networks and ISDN Systems*, vol. 28, no. 1–2, pp. 25–35, 1995.

[14] James Hall, Ian Pratt, and Ian Leslie, "Non-intrusive estimation of web server delays," in *LCN 2001 The 26th Annual IEEE Conference on Local Computer Networks (LCN)*, Tampa, FL, March 2001.

[15] Waikato Applied Network Dynamics, University of Waikato Computer Science Department, *DAG*, Dec. 2001, http://dag.cs.waikato.ac.nz/.

[16] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Perform ance Evaluation," in *Proceedings of ACM SIGMETRICS'98*, Madison, WI, 1998.

[17] The Standard Performance Evaluation Corporation, "The SPECweb99 benchmark," 1999, http://www.spec.org/osg/web99/.

[18] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability an d the impact of control," in *Proceedings of ACM SIGCOMM'99*, Cambridge, MA, Aug. 1999.

[19] T. Berners-Lee, R. Fielding, and H. Frystyk, "RFC 1945: Hypertext Transfer Protocol — HTTP/1.0," May 1996, Status: INFORMATIONAL.

[20] Jeffrey C. Mogul, "The case for persistent-connection HTTP," in *SIGCOMM*, 1995, pp. 299–313.

[21] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "RFC 2068: Hypertext Transfer Protocol — HTTP/1.1," Jan. 1997, Status: PROPOSED STANDARD.

[22] C. R. Cunha, A. Bestavros, and M. E. Crovella, "Characteristics of WWW Client-based Traces," Tech. Rep. BU-CS-95-010, Computer Science Department, Boston University, July 1995.