# Traffic Prediction Algorithm for Speculative Network Processors

Jürgen Foag, Thomas Wild [a]

[a] Institute for Integrated Circuits, Technical. University of Munich, 80290 Munich, Bavaria, Germany
{Juergen.Foag, Thomas.Wild}@ei.tum.de

To support real-time applications, for instance Voice-over-IP, network infrastructure has to provide high bandwidth as well as short end-to-end latency. To cope with both, network processors (NP) are selected. Current NP support transmission rates of OC-48. However, due to sequential packet layer processing, processing delay is not optimized. To solve this problem, speculative packet processing is applied, which requires accurate traffic prediction. In this article, we present a flexible dynamic protocol stack prediction algorithm which is well-suited for accurate traffic prediction and simulations show that it can sustain traffic bursts as well as transitions to changed traffic characteristics.

*Afin d'assurer le support d'applications en temps-réel, comme par exemple la téléphonie IP,les infrastructures réseau doivent fournir à la fois une forte bande-passante, de même qu'une latence minimale. Les processeurs réseau (NP) sont utilisés afin de gérer ces deux nécessités. Les processeurs réseau actuels supportent les taux de transmission OC-48 mais le délai de traitement n'est pas optimisé, notamment à cause du traitement séquenciel de la couche paquet. Afin de résoudre ce probléme, un traitement spéculatif au niveau des paquets est appliqué.Ceci recquiert une prévision précise du traffic. Nous présentons dans cet article un algorithme dynamique prédictif adapté aux prévisions précises de trafic. Les simulations montrent qu'il est capable de prendre en compte à la fois les pics de trafic, mais aussi les changements de caractéristiques du trafic.*

## 1 Introduction

Distributed high performance computing systems possess a strong dependency on the performance of the underlying network infrastructure. By increasing network throughput, the performance bottleneck is shifting from the bandwidth of transmission media to the processing capacity of hosts and transmit systems like routers [1]. The performance of these devices is significantly affected by their architecture [2]. High-speed networking consists not only of the quest for high bandwidth. In the same way, low latency as well as the ability to cope with high bandwidth-x-delay product paths is required [3].

Recently, *network processors (NP)* have been introduced to cope with throughput requirements of OC-48 (2.5 Gbit/s) and higher. They normally comprise multiple micro-programmable RISC processor cores, each realizing a simultaneous multithreading architecture [4]. By means of an efficient software implementation of message-parallelized protocol processing, high performance on NPs can be achieved. In addition, some NP provide dedicated hardware blocks to accelerate calculation intensive networking functionalities.

While these properties made a throughput increase possible, optimization of packet processing delay on a NP was rather uncared. To make real-time applications possible, the delay problems are bypassed: For instance, *content delivery networking (CDN)* shifts centralized real-time application content through the network closer to the user to distributed content engines [5]. Consequently, essential drawback is a high administration effort for the distributed system components.

In [6], as a new concept speculative packet processing was introduced. To evaluate it, an abstract concept verification was done, which illustrated the benefits. The concept comprises two components: *protocol stack prediction* and *speculative data processing*. The latter exploits speculative resolution of network layer dependencies inherent to the packet. To obtain a significant decrease of processing latency, the former has to provide a high prediction accuracy. In [7], the system implementation, which is realized in SystemC , is shown together with results of the associated performance evaluation.

The main contribution of the ongoing work, which is presented within this article, is the presentation of the dynamic prediction algorithm. It is capable of fast self-adaptation to modified traffic characteristics, for example for changes during daytime. With randomized generated traffic, which complies with current traffic distributions, the algorithm is evaluated and showed high accuracy. In addition, the architecture of the prediction instance is explained to insert it in the speculative packet processing device.

The remainder of this paper is organized as follows. Section 2 gives an overview to previous work done in network traffic characterization. The concept, the proposed architecture model and the performance evaluation are summarized in section 3. Section 4 presents the prediction algorithm and discusses the results. Finally, the conclusion and the future work is given in section 5.

## 2 Related Work

Network simulation and characterization of communication networks targets on obtaining network traffic knowledge to dimension more efficiently network and network devices [9] et al. Thus, Internet traffic prediction is a trial to obtain an a priori knowledge of the behavior of future data sent through communication networks [10].

### 2.1 Characteristics

Extensive research has been done to understand the nature of network traffic. One result was the observation of *self-similarity* in various Internet traffic measurements [11]. The expression self-similarity is used for processes which show scale-independent invariant correlational structure. In other word, appearance is unchanged regardless of the scale at which it is viewed. Self-similarity is closely associated with *heavy-tailed distributions*. The expression heavy-tailed results from extremely large values with non-negligible probability [12]. Such distributions decline via a power law with small exponent (less than 2). As a consequence, when $\alpha < 2$, the random variable shows infinite variance. Self-similar processes can show *long-range dependence*. A process with long-range dependence is defined by an autocorrelation function of $s_{xx} \sim x^{-\beta}, x \rightarrow \infty, 0 < \beta < 1$. Some essential observations concerning traffic characteristics are listed in the following. However, their reasons are still not entirely determined. Due to [13], burst sizes of FTP transfers and Telnet inter-arrival times follow a heavy-tail distribution. Examinations in [14] show that transmission and idle times of Ethernet traffic between source and destination pairs are heavy-tailed.

### 2.2 Statistics

To understand the behavior of routing in the Internet, a multitude of measurements have been done [15] [16] et al. In 1997, simulation results were obtained from the MCI backbone [17]: On IP protocols, TCP is the dominant layer 4 protocol of 90 percent of the packets. UDP has second highest level at about 5 to 10 percent. Other protocols make up an almost negligible percentage. More recent results can be taken from the IP Monitoring system of the Sprint Tier-1 IP backbone network. It is deployed at a Point-of-Presence (POP). Summarized, some essential results are the following [18], [19]:

- Local traffic distribution: two of the used POPs, which are located in the east and west, respectively, are dominant and receive about $\frac{2}{3}$ of the complete traffic. This can be explained by a higher population and the termination of international trunks.

- Time of day behavior of large, medium and small POPs: While small POP remain fairly stable during the time of day, medium POPs show a long slow small decline. The most, large POPs decrease rapidly up to 60 percent.

- Elephants-and-mice phenomena: Streams which are aggregation of packets lead to a distribution of a few very high-volume streams (elephants) and many low-volume traffic streams (mice).

- Impact of new applications: On some simulation links, over 60 percent of traffic is generated by applications such as distributed file sharing and streaming media.

- Traffic breakdown by protocols: Over 90 percent of traffic is TCP traffic, even on the links with a significant percentage of streaming media. This can be explained by the use of firewalls which are commonly configured for preference of TCP than UDP for streaming media.

The monitored protocol breakdown of [20] showed a traffic distribution of approximatively 82 - 90 percent of TCP, 8 - 16 percent UDP and 1 - 3 percent ICMP.

## 3 Protocol Stack Processing

The packet processing concept is based on two components: *Protocol Stack Prediction* and *Speculative Packet Processing*. Objective is to overcome processing latency limitations which result from network layer hierarchies.

*Protocol Stack Prediction:*

The decision how to handle and forward a packet is done upon the information which is contained in the packet header. The header of the data-link layer, the network layer and the transport layer is hierarchically arranged. I.e. the type of a higher layer of the ISO/OSI reference model of communication is derived from the header of the lower layer. To circumvent a serial extraction, *protocol stack prediction* predicts the protocol stack of the next packet. The scheme is depicted in Fig. 1.

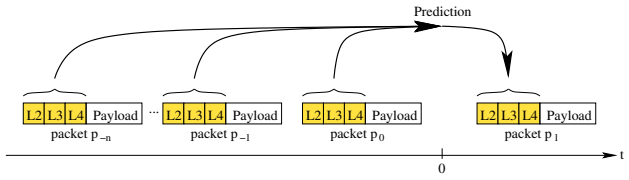


**Figure 1.** Protocol Stack Prediction Scheme

The prediction is based on the history of packets received earlier. It should be noted, that the prediction *not* only delivers protocol layer types of a packet. In a store-and-forward architecture, this information could be gathered quickly from the packet but would not influence significantly processing delay. Furthermore, additional information which is required for control point processing is implicitly provided by the prediction. For instance, costly processing for checksum verification of layer 2 has to be done in front of processing of layer 3 as in Fig. 2.

Starting from the prediction result, packet processing is started speculatively. I.e. the inherent data dependency is temporarily resolved and processing of layer 2, 3 and 4 is

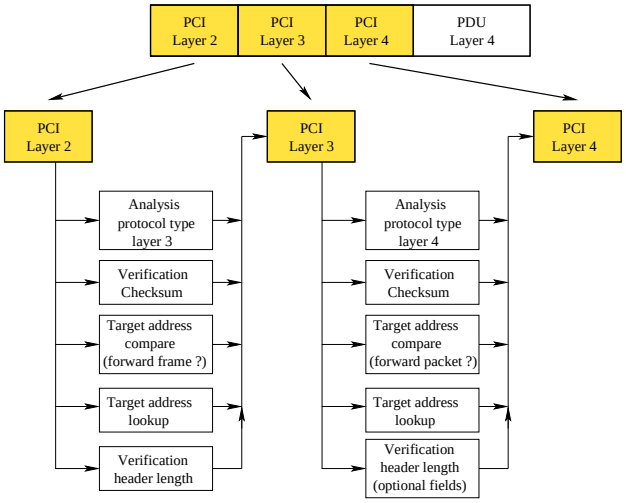


**Figure 2.** Protocol Stack Prediction

started simultaneously. If tasks which concern the identification of the upper layer protocol type are terminated, the prediction is verified. In case of correctness, processing results can be kept. Else, i.e. in case of misprediction, results have to be dropped and task processing has to be restarted. Further details can be taken from [7].

Through the use of both components, mean latency can be reduced. Thus, it is less than the sum of individual processing delays of network layer 2, 3 and, if differentiated services is supported, layer 4. Prerequisite is an accurate prediction.

# 4 Protocol Stack Prediction

## 4.1 Dynamic Algorithm

Static predictor algorithms take advantage of statistical invariance of considered processes. For this purpose, a traffic profile has to be generated before start of operation.

In section 2, traffic statistics were presented that show variable behavior. Reasons are changes of traffic during a day or week, asymmetric traffic volumes, local aspects of operation and other. Thus, a dynamic algorithm seems to be well-suited to achieve better prediction accuracies [18]. However, considering the fact that routers or switches represent aggregation points, and having self-similarity in mind, modeling a prediction algorithm based on stochastic event sequences, i.e. Markov chains, seems no to be precise. Alternatively, an algorithm is proposed that exploits majority of occurring process events.

To distinguish between different protocol stacks which are supported by the system, a stack identifier (SID) is defined. It contains an unique data pattern for each stack. Output of prediction is the SID of the packet which is expected to be received next.

The prediction algorithm is depicted in Figure 3.

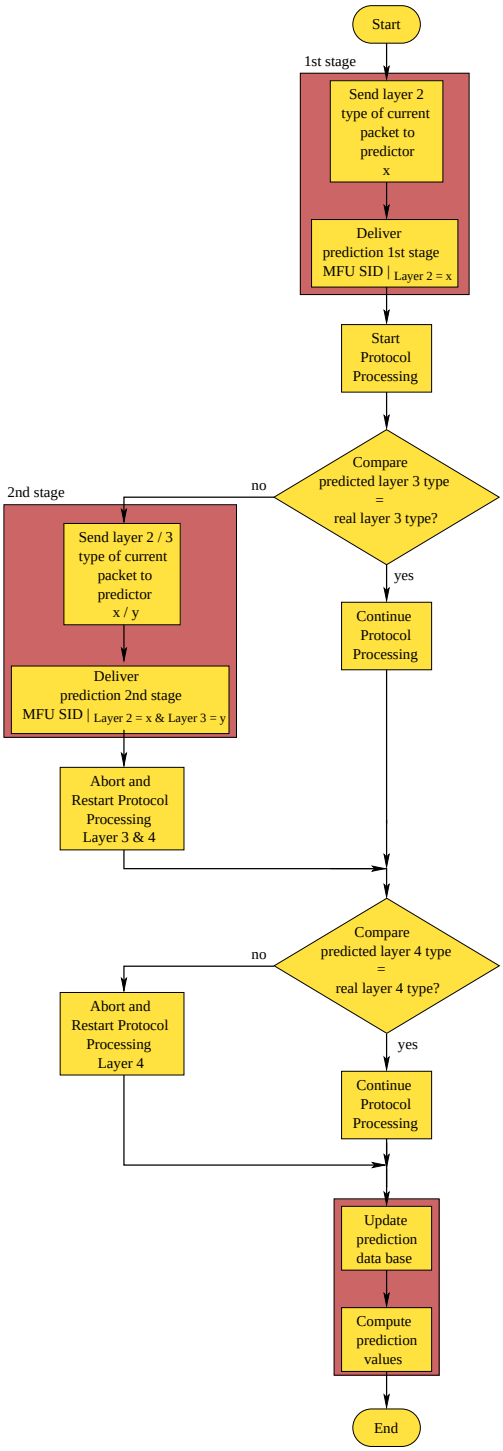


**Figure 3.** Dynamic prediction algorithm

Protocol stack prediction comprises protocol layer 3 and 4. At first, when a packet arrives, the type of protocol layer two is indicated by the precedent data-link layer device. Based on this knowledge, a first prediction value is requested. In the following, packet processing is started. When the type of layer three is derived through analysis of the upper layer type field which is part of layer two, the prediction is verified. In case of correctness, no further prediction is required. Else, a second prediction concerning the type of layer four has to be requested. The request considers then the available type of layer three. Thus, multiple MFU values are provided. They differ for the first prediction stage in the type of layer two and for the second stage in the types of layer two and three.

After analysis of each packet, the prediction values are updated to make a dynamic behavior possible. The SID of the current packet is transferred to the prediction data base. Therein, the frequency of each SID is recorded through summation of receive events. The prediction algorithm takes this packet history information as input to calculate the prediction values. The computation of the prediction values, i.e. the SIDs, follows the policy of *most frequently used (MFU)* [22]. Output is a value that identifies the protocol stack which occurred most frequently in the past. As an extension of the policy, an additional weight factor is appended during prediction computation to each SID [23]. Objective is to prioritize delay sensitive traffic in case of equal stack frequencies.

The flow of computation is illustrated in Figure 4.

When a new SID is transferred to the data base, the associated entry is updated. Depending on the type of protocol layer 2, the MFU values of the referring path have to be recalculated. In the example, two types for layer 2 (Packet-over-SONET (POS), Gigabit-Ethernet (G-E)) and layer 3 (IP version 4, IP version 6) are used. In case of control protocols, e.g. ICMP, protocol processing is done by the separate control plane.

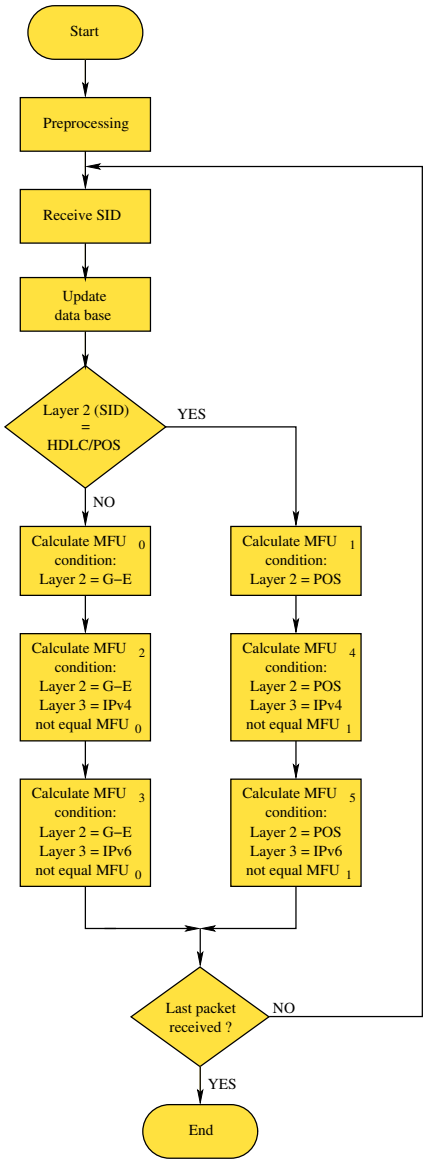


**Figure 4.** MFU computation flow

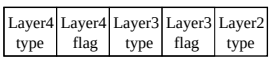


**Figure 5.** Protocol-stack status word (PSSW)

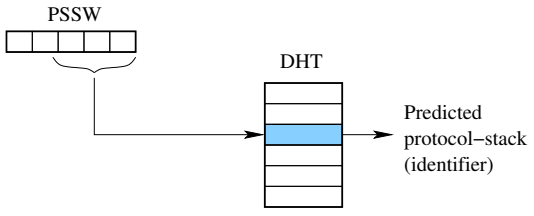


**Figure 6.** Prediction table lookup

### 4.2 Architecture

The progress of protocol layer analysis of a current packet is kept in the Protocol-Stack Status Word (PSSW). To request a prediction value, the PSSW is used to address the prediction values. The format of the PSSW is shown in Fig. 5.

Two flag fields are used to indicate if the type of protocol layer 2 and 3 is already determined. A defined numerical value that characterizes the types of layer 2, 3 and 4 is stored in the corresponding type fields.

The prediction values are stored in a decode history table (DHT) as shown in Fig. 6 [24].

The lines of the DHT correspond to $MFU_i$ with $i \in \{1, ..., 6\}$. The DHT returns the predicted SID stack to proceed packet processing.

### 4.3 Simulation and Discussion

To evaluate the algorithm, various packet protocol breakdowns were generated. Having a software implementation in mind, deterministic test traffic was used to obtain the number of processing cycles necessary for MFU entry update. With regard to the most essential metric of sorting algorithms, the number of compare instructions has been extracted. Based on a SUN-RISC instruction set, it serves for consolidation of the proposed algorithm. By use of a random generator, prediction accuracy as well as the self-adaption on transitions of traffic characteristics have been measured.

The prediction accuracy for various test traffic and identical weight factors is shown in Figure 7.

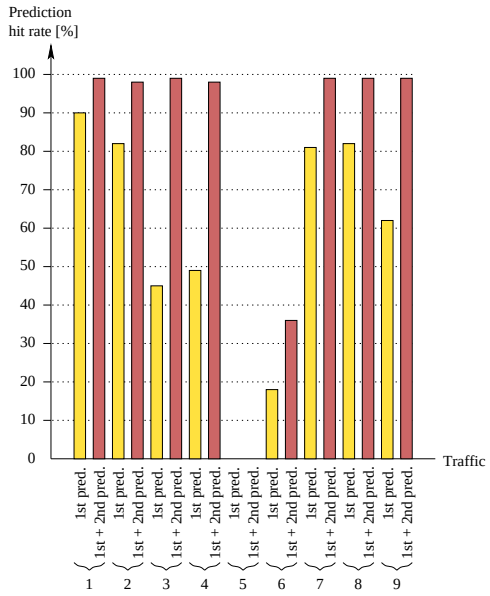


**Figure 7.** Prediction accuracy

Traffic 1 [21] and 2 [18], shown by curly brackets, show hit rates for the first prediction level of 90 % and 82 %, respectively. For both, 90 % and 89 % can be achieved. Other simulation traffic differ in equal protocol breakdown distribution, contain bursts and traffic transitions. For instance, traffic 4 generates 4 different packets 25 % each. Traffic 5, which is generated of a periodical sequence of 8 different packets, is used as unrealistic worst case scenario. Thereby, a regular replacement of the entries in the DHT takes place.

As mentioned above, modified prediction weight factors allow prioritization of certain traffic. Traffic 6 is composed of equal packet distribution for TCP and UDP, IPv4 and IPv6 as well as POS and G-E. Through a weight factor of 4 for UDP traffic versus default weights of 1, the misprediction rate can be reduced from 64 % to 28.8 %.

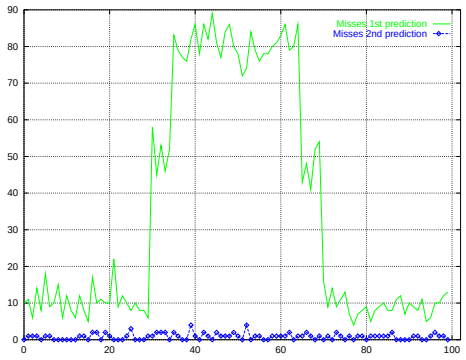


**Figure 8.** Number of mispredictions (traffic R8)

In Figure 8, the prediction convergence of random traffic R8 is presented.

The unit of the x-axis corresponds to 100 packets. R8 is composed of five ranges of different packet distributions. The distribution of the first 3000 packets is: 90 % TCP; 9 % UDP; 1 % ICMP. Two bursts of 500 packet each follow with 50 % TCP; 49 % UDP; 1 % ICMP. Between, the distribution shifts to 19 % TCP; 80 % UDP for 3000 packets. For the last 3000 packets, the distribution is equal to the beginning.

While the misprediction rate is about 10 for the first and last period, the hit rate of the first prediction stage for the meantime is worse at about 80 percent. However, it should be noted, that the misprediction rate of the second stage remains constant at about 1 %. An additional reset facility of the data pool that detects longer periods of continuous occurring mispredictions would compensate the misconduct of the first stage.

### 4.4 Implementation costs

The additional effort which has to spent for speculative packet processing is a prediction data base and the DHT. To obtain a fast memory access, both might be realized as small part of an embedded SRAM. Today, a 4k bit SRAM requires an access delay of about 2.8 ns and has an area of about 0.44 $mm^2$ [25].

The calculation effort for data base update and computation of prediction values has been determined based on a software implementation on a RISC architecture. For traffic 1, mean number of processing cycles is 324.

As in [7], mean processing delay of a packet is 858 cycles. Thus, considering a realization within a RISC core, dynamic prediction update can be done at line-speed for each packet.

Furthermore, the portion of compare instructions necessary for prediction value computation has be measured. Traditional sorting algorithms such as heap sort, bubble sort or quicksort require between $ln(N) \cdot N$ and $N^2$ compare instructions for $N$ sorting elements [26]. In case of N = 16 supported protocol stacks and six MFU values to

calculate, only 26.9 compare instructions were necessary.

## 5 Conclusions

We have presented a dynamic protocol stack prediction algorithm that delivers high accuracy. Its appropriateness for use in speculative network processors has been demonstrated. Due to a small amount of processing cycles required for DHT computation, prediction update at line-speed is feasible. Further work will concentrate on system output queuing algorithms for limitation of delay jitter in case of misprediction.

## References


1. M. Bjoerkman, P. Gunningberg, *Locking Effects in Multiprocessor Implementations of Protocols* ACM Sigcomm, Sep. 1993.
2. D.D. Clark, *Modularity and Efficiency in Protocol Implementation* NIC RFC 817, 1982.
3. J.P.G. Sterbenz, *Protocols for High-Speed Networks: A Brief Retrospective Survey of High-Speed Networking Research* Workshop on Protocols for High Speed Networks, May 2002.
4. S. Lakshmanamurthy, K. Liu, Y. Pun, L. Huston, U. Naik, *Network Processor Performance Analysis Methodology* Intel Technical Journal, Vol. 6, Aug. 2002.
5. T. Vaseeharan, M. Maheswaran, *Towards a Novel Architecture for Wide-Area Data Caching and Replication* Intl. Conference on Internet Computing, Las Vegas, Jun. 2000.
6. J. Foag, N. Pazos, T. Wild, W. Brunnbauer, *Self-adaptive Parallel Processing Architecture for High-Speed Networking* Proc. Int'l. Symposium on High Performance Computing Systems and Applications, Moncton, Jun. 2002.
7. J. Foag, M. Praxenthaler, T. Wild, *Performance Evaluation of a Speculative Network Processor* Proc. Int'l. Symposium on High Performance Computing Systems and Applications, Sherbrooke, May 2003.
8. *http://www.systemc.org*
9. L. Trajkovic, A. Neidhardt, *Traffic characterization and time scales for designing efficient network control policies* Invited paper, Proc. NOLTA'97, Hawaii, Dec. 1997.
10. L. Trajkovic, A. Neidhardt, *Internet traffic prediction* Centre for Systems Science, Simon Fraser University, Vol. 12, Issue 1, Mar. 2000.
11. M.E. Crovella, A. Bestavros, *Self-similarity in world wide web traffic: Evidence and possible causes* IEEE/ACM Transactions on Networking, vol. 6, pp. 835-846, Dec. 1997.
12. L. Guo, M. Crovella, I. Matta, *TCP Congestion Control and Heavy Tails* Technical Report BU-CS-2000-017, Boston University, Computer Science Department, Jul. 2000.
13. V. Paxson, S. Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling* IEEE Transactions on Networking, Jun. 1995.
14. W. Willinger, M. Taqqu, R. Sherman, D. Wilson, *Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level* IEEE/ACM Transactions on Networking, Feb. 1997.
15. V. Paxson, *End-to-End Routing Behavior in the Internet* IEEE/ACM Transactions on Networking, Oct. 1997.
16. k. Claffy, *Internet Measurement and Data Analysis: Topology, Workload, Performance and Routing Statistics* NAE Workshop, 1999.
17. K. Thompson, G. Miller, R. Wilder, *Wide-Area Internet Traffic Patterns and Characteristics* IEEE Network, 11(6), Nov. 1997.
18. C. Fraleigh, S. Moon, C. Diot, B. Lyles, F. Tobagi, *Packet-Level Traffic Measurement from a Tier-1 IP Backbone* Submitted for publication, Sept. 2002.
19. S. Bhattacharyya, C. Diot, J. Jetcheva, N. Taft, *Pop-Level and Access-Link-Level Traffic Dynamics in a Tier-1 POP* ACM Sigcomm Internet Measurement Workshop IMW 2001, San Francisco, Nov. 2001.
20. *http://ipmon.sprintlabs.com* date: 17th march 2002.
21. K. Claffy, G. Miller, K. Thompson, *The nature of the beast: recent traffic measurements from an Internet backbone* Proc. of INET 98.
22. J. Smith, *A Study of Branch Prediction Strategies* Proc. 8th Intern. Symposium of Computer Architecture, May 1981.
23. M. Doshi, U. Pandit, R. Kumbharathi, W. Zhu, J. Tan, *XML-based Data Manipulation* Class CSE 812, Advanced Operating System, University of Illinois at Urbana-Champaign, 2001.
24. J. Pino, B. Singh, D. Culler, *Performance Evaluation of One and Two-Level Dynamic Branch Prediction Schemes over Comparable Hardware Costs* Technical Report, UCB/ERL M94/45, Jun. 1994.
25. T. Sherwood, *Embedded Memories* CSE 291E / EE260C, University of California at San Diego, 2002.
26. R. Sedgewick, *Algorithms* 2nd edition, Addison Wesley, Reading, MA, 1992.