A Workload Characterization Methodology for WWW Applications

G. Kotsis[+], *K. Krithivasan*[*], *S.V. Raghavan*[*]
[*]*Department of Computer Science and Engineering, Indian Institute of Technology*
[+]*Institute for Applied Computer Science and Information Systems, University of Vienna*
[*]*Madras 600036 India, phone +91 44 2351365, fax +91 44 2350509, email: [kamala|svr]@iitm.ernet.in*
[+]*A-1080 Vienna Austria, phone +43 1 4086366, fax +43 1 4080450, email: gabi@ani.univie.ac.at*

Abstract
With the World Wide Web (WWW) traffic being the fastest growing portion of load on the internet, describing and characterizing this workload is a central issue for any performance evaluation study. In this paper, we present an approach for generating a profile of requests submitted to a WWW server (GET, POST, ...) which takes explicitly into account the user behavior when surfing the WWW (i.e. navigating through it via a WWW browser). We present Probabilistic Attributed Context Free Grammar (PACFG) as a model for translating from this user oriented view of the workload (namely the conversations made within browser windows) to the methods submitted to the Web servers (respectively to a proxy server). The characterization at this lower level are essential for estimating the traffic on the net and are thus the starting point for evaluations of net traffic.

The model is general enough to cover any form of web activity (e.g. different browsers, different protocols, JAVA applets, ...). The model can either be used to generate workloads which try to mimic the usage of a real systems (e.g. using parameters obtained from measurements on the system under study), but could also be parametrized in order to define worst case scenarios, i.e. capturing the system behavior under heavy load. Both approaches are discussed in the paper.

Keywords
Distributed Systems, Internet Traffic, World Wide Web, Performance evaluation, Workload Characterization, PACFG.

1 INTRODUCTION

When talking about the information highway, one typically associates it with fast and easy access to a large amount of information stored in a distributed way. The World Wide Web (WWW) can thus be visualized as (probably the largest) distributed multimedia hypertext system. From an architectural point of view, the WWW can be seen as a huge client server system built on top of the Internet protocol suite. The demands placed on such a system are manyfold: users require convenient access via a unified easy-to-use interface. WWW browsers try to accomplish this task. Their success is illustrated by Web traffic being the fastest growing portion of traffic on the internet. This leads to the second major demand: maybe the most critical factor in the satisfaction of the user is the speed at which requests for information can be fulfilled. Aspects determining the response time of the servers include network related characteristics (available bandwidth, ...) and server related characteristics (hardware characteristics, cacheing policies, ...). But most of all, the performance is determined by the load, that the servers have to process.

Therefore, a characterization of the load generated by Web-Traffic is a fundamental issue for any further investigation of performance. In this paper, we use a generative technique for modeling the Web traffic. This technique is based on Probabilistic Attributed Context Free Grammars (PACFG) (Fu 1974), which have proven to be a useful way to translate the workload from the application-oriented user's point of view to the resource oriented system level in distributed environments (Raghavan, Vasukiammaiyar & Haring 1995), (S.V. Raghavan 1996).

Previous work in this area (Sedayao 1994), (Arlitt & Williamson 1996), (Brakmo & Peterson 1996), (Cunha, Bestavros & Crovella 1995), (Crovella & Bestavros 1996) has focused on the characterization of internet/WWW traffic mainly at the system's level. Our approach distinguishes from the others in that we try to map the actual user behavior (characterized in terms of the number of browser sessions a user would start and in terms of the number of conversations within a browser) to the traffic characteristics at the lower level. Thus, the analyst can experiment with the effect of changes in user behavior on the systems load. Such studies are crucial e.g. for capacity planning of computer networks.

We will first describe the characteristics of the environment we address in Section 2. In the next section (3) we will briefly introduce the concept of PACFG (illustrated on a simple single-serve example). Finally (Section 4), we will apply PACFG for modeling the traffic at the system level generated by users surfing the WWW. We will restrict our study to traffic coming from conversations of the type http and ftp as supported by the Netscape browser. We will show, how to obtain the relevant estimates for the workload parameters from measurements. We conclude with an outlook on future work.

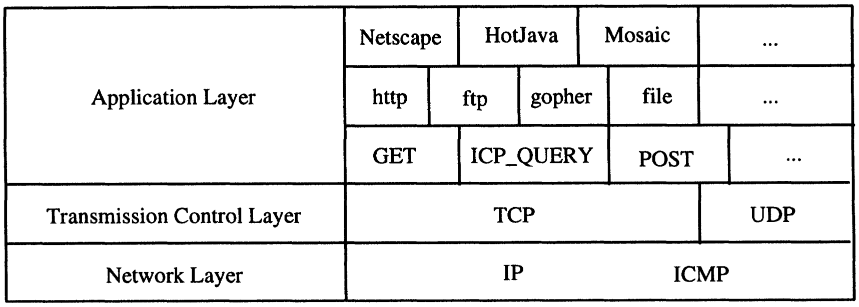

Figure 1 Hierarchical Layers of the Architecture

2 DESCRIPTION OF THE ENVIRONMENT

The general architectural model of WWW-browser applications is shown in Figure 1. At the top level, we have different types of browsers, through which the users will navigate the WWW. Within such browsers, the user has the possibility to access information via giving so called Uniform Resource Locators (URL) which identify both, the server (from where to retrieve the information) and the file, containing the information. The user can either enter such an URL directly, but more typically he or she would just select a link to which an URL would be associated (this is what makes the WWW a hypertext application). We will call such an access a **conversation** between the user's browser (the client) and the host where the information resides (the server). The conversations are based on protocols, typically the HyperText Transfer Protocol (http). But one of the key features in the success of the WWW is its ability to provide a unified access to (nearly) all sort of information available on the internet, thus other services such as ftp or gopher and recently interactivity incorporated in JAVA applets are also supported. Most browsers also include facilities for sending e-mail (via the smtp protocol) or to read newsgroups. Any of these conversations will result in a request which will be transformed into the **methods** of the request which are submitted to the server (GET, POST, ...). All the services described so far are based on the TCP/IP protocols, which form the next lower levels.

To describe the system we will use a six level hierarchy as represented in Figure 2. At the top level, we have several **users** logged onto machines in the network. We will call the time between a login and a logout the session duration. Within a **session**, a user will start web **browsers** (level 2). The time between starting a browser and quitting it is referred to as the life-time of the browser and is represented at the second level by a thick line. As modern computing environments will allow the user to start several browsers and run them in parallel, their lifetimes may have overlaps. The actions that can occur on the browser level are either $\mathbf{start}_i$ or $\mathbf{quit}_i$, where i denotes the type of browser.

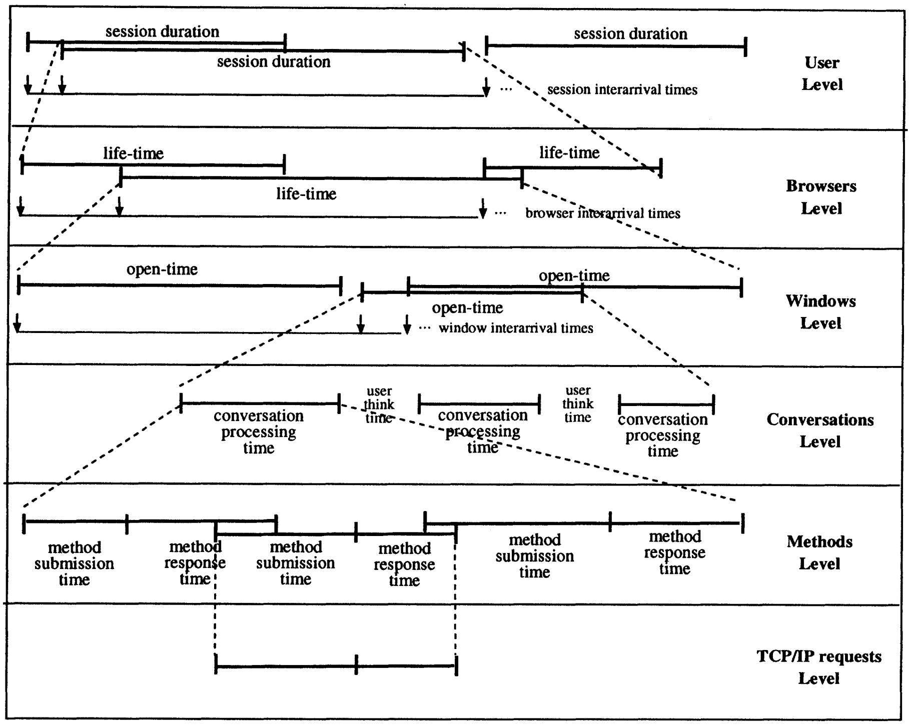

Figure 2 Hierarchical Levels in Web Traffic

During the lifetime of a browser, the user may open several **windows**. As most browsers allow several windows being open simultaneously, the open-times of the window may overlap. The actions at this level are **open** and **close** of a window.

Within each window, the user initiates a **conversation** with a server. These conversations, which are typically initiated by selecting a link on a web page or by entering an URL, consist of requests following a particular protocol (http, ftp, ...). After issuing the request for a conversation the user will wait for the request to be fulfilled and would then spend some time in consuming the delivered result, e.g. reading the page that has been displayed (this time is typically referred to as user think time). Thus we can identify at the conversations level a sequence of active phases (conversations are being processed) and idle phases (user **think times**).

At the methods level, we will characterize the arrivals of the GET requests. When mapping from the conversations level to the methods level, we have to consider, that an ftp conversation will invoke a single GET, while an http conversation will invoke one or more GETs (depending on the contents of the page to be retrieved). Finally, each method (GET) will be mapped onto a TCP/IP requests. Note, that some browsers (e.g. netscape) will allow for more

Level	Action
Netscape	start, quit
Windows	open, close
Conversations	http, ftp
Methods	GET

Table 1 Types of Actions at each Level

than one TCP/IP stream being simultaneously open*, thus the several GETs may be processed simultaneously and requests r_j submitted after request r_i could in fact finish before r_i.

Although the approach that we present can be applied to the general model in a straight forward manner*, we wanted to keep the presentation simple but still being able to highlight the most important issues (i.e. capturing parallelism in user behavior, capturing the representation of a sequence of commands, and showing the mapping from one level to another and the corresponding parameter estimates). Thus, we restrict the modeling to a subset of this environment in that we only consider the load generated from one user working with up to n_B netscape browsers* simultaneously and using http or ftp conversations. We assume a value of n_W denoting the maximum number of windows that can be opened simultaneously within one netscape browser. Finally, we will only characterize the workload to the level of methods. Thus, we are only considering the 4 inner levels in the hierarchy. Table 1 summarizes these four levels and the actions that could occur at each of them. Applying the proposed approach of PACFG for workload modeling in performance evaluation studies has already been demonstrated in previous papers (Raghavan, Vasukiammaiyar & Haring 1993), (S.V. Raghavan & Harring 1994), (Raghavan et al. 1995), (S.V. Raghavan 1996). In this paper, we will therefore sketch only briefly the parameters to be estimated.

3 WORKLOAD CHARACTERIZATION USING PACFG

A Probabilistic Attributed Context Free Grammar (PACFG) is a 3-tuple $G_A = \{G, A, Q\}$ with G as the regular grammar, $G = \{V_N, V_T, P, S\}$. Here, V_N and V_T are a set of non-terminal and terminal symbols, P represents a

*This behavior is conform to the http1.0 specification and might be different in subsequent specifications.
*In fact, adding levels on top or at the bottom is accomplished by extending the set of production rules and estimating the corresponding parameters.
*Throughout the remaining of this paper we restrict our observations to Netscape browsers, currently one of the most widely used browsers.

RuleNo	Rule	Probability
1	$S \to aS$	1/3
2	$S \to bS$	1/3
3	$S \to a$	1/6
4	$S \to b$	1/6

Table 2 A Simple Example of a PCFG

set of production rules, and S is the start symbol.A is a set of attributes and Q is a set of probabilities associated with P.

Each of the non-terminals, which will be expanded to a (sequence) of non-terminals or terminals at the next lower level, has two attributes s and e. They represent the start and end times of the non-terminal respectively. The duration of an operation is described by the difference between the start and end times. Let us consider a hierarchical system with n levels. At each level, the system supports a set of operations each being represented by a non-terminal. Non-terminals in the n^{th} level of the hierarchy expand into a string of non-terminals in the $(n-1)^{th}$ level. Production rules are used for representing this. At level n, the number of classes of operations is represented by K_n. To decide on which of the operations in a lower level to which an operation in a given level expands to, a set of probabilities are used. A non-terminal in level n and of type i can either always go to ϵ, or there are a set of production rules mapping it on to units of the lower level. The distribution functions are essentially used to determine when the expansion of a non-terminal has to be stopped, that is, when the ϵ rule has to be taken. A proposed end time for a non-terminal is generated using the distribution function, and as it expands, when the operations at the lower level reach the end time, the ϵ rule is taken.

Attributes are associated to each non-terminal, denoting the start time and end time of the operation $(start(V_{NT}), end(V_{NT}))$ with $time(V_{NT}) := end(V_{NT}) - start(V_{NT})$. Start and end times (or durations resp.) will depend on the order in which the production rules are applied and are derived from the start/end times of terminals and non-terminals at higher/lower levels. Start times are always inherited (e.g. derived from parents), end times (or durations) will always be synthesized (from children to parents or among siblings), thus guaranteeing an evaluation free of cyclic dependencies. The time attributes of the terminals, the end symbols, which are not expanded further, are defined analogously, but their duration $time(V_T)$ is a parameter to be estimated, thus the end time is given by $end(V_T) := start(V_T) + time(V_T)$.

To control the order in which rules can be applied, a control set can be specified (Salomaa 1973). As an example, let us consider the PCFG given in Table 2.

This grammar could generate for example string *aabb* or string *abab*, both with a probability $(1/3)^3 * (1/6)$. If we define the following control set $(1(2+4))^*$ we would restrict the grammar in that rule 1 always has to be applied first, followed either by rule 2 or 4, this order can be repeated several times. Note, that now the probability of string *aabb* is zero. In addition, the probabilities of the rules has to be defined in order to meet the restrictions in the control set, i.e. rule 1 would have a probability of 1 (will always be applied if possible), rule 3 would have a probability of 0 (will never be applied), and the probabilities of rule 2 and 4 have to sum up to 1 (e.g. $Q_2 = 2/3, Q_3 = 1/3$).

If a rule given in the control set cannot be applied, the rule would simply be omitted. For example, in the control set 134 rule 4 would never be applied.

To illustrate PCFGs in workload modeling, we will model the user behavior as a sequence of user think times and command executions in a single server environment (Raghavan et al. 1993), (S.V. Raghavan & Harring 1994). The commands used by such a user, will, in general be of different types, such as file edit, program compile and program executions. At a lower level, the command executions can be mapped onto the time necessary to submit the request and to the time to process the request. At a higher level, we consider different user sessions from where commands can be executed. Thus, we represent the workload for this single server environment in a three level PCFG*

$$Networkload = \{\{\{B, D, E, G, Z_1, Z_2, Z_3, WL\}, \{a, c, f, h\}, \{P\}, WL\}, \{Q\}\}$$

where

B	is a set of non-terminals to relate evel 3 to level 2. The choice of a production is defined by Q_1
E	is a set of non-terminals to relate level 2 to level 1. The choice of the production is defined by Q_3
D	is a set of non-terminals to have a compact representation for productions at level 2. The choice is defined by Q_2
G	are non-terminals to have a compact representation for productions at level 1. The choice is defined by Q_4
Z_1, Z_2, Z_3	are (simple) non-terminals
WL	is the Start symbol of the networkload
a	is Session interarrival time
c	is User think time
f	is Request generation time
h	is Request service time

and the set of productions $\{P\}$ and probabilities $\{Q\}$ is given by

*For simplicity of presentation, we will not consider attributes.

PR I	$WL \rightarrow aB_i \mid aB_iZ_1$	$Q_1[i]$
PR II	$Z_1 \rightarrow aB_iZ_1 \mid \epsilon$	$Q_1[i]$
PR III	$B_i \rightarrow cD_{ij} \mid c\ D_{ij}Z_2$	$Q_2[i,j]$
PR IV	$Z_2 \rightarrow cD_{ij}Z_2 \mid \epsilon$	$Q_2[i,j]$
PR V	$D_{ij} \rightarrow E_j$	$Q_3[i,j]$
PR VI	$E_j \rightarrow fG_{jk} \mid fG_{jk}Z_3$	$Q_4[j,k]$
PR VII	$Z_3 \rightarrow fG_{jk}Z_3 \mid \epsilon$	$Q_4[j,k]$
PR VIII	$G_{jk} \rightarrow h_k$	$Q_5[i,j]$
PR IX	$h_k \rightarrow$ Constant.	

where $1 \leq i \leq$ No_of_session_groups, $1 \leq j \leq$ No_of_command_classes, and $1 \leq k \leq$ No_of_request_types.

The production rules PR I & PR II generate the session level networkload sequence. The number of production rules designated by PR I is equal to the number of session groups. Each production is associated with the probability of occurrence of a session of group i as given by $Q_1[i]$. The production rules PR III & PR IV generate the networkload sequence at the command level and the production rules PR V, PR VI & PR VII generate the networkload sequence at the request level. The matrix Q_2 gives the percentage of commands of class j issued during the sessions of group i. Similarly, the elements of the matrix $Q4$ denote the percentage of the requests of type k generated during the commands of class j. The matrices Q_3 and Q_5 are the unity matrices. Q_3 implies that the requests generated during a command execution are independent of the session in which the command was issued and Q_5 implies that the request service time is independent of the class of command during which the request was generated. This PCFG is a generative representation of the networkload sequence at any level of the three level hierarchy. The model parameters to be estimated are a, c, f, h, Q_1, Q_2 and Q_4.

4 PACFG FOR MODELING WWW TRAFFIC

4.1 Representation in a PACFG

We can define the workload model as a PACFG with

$$\begin{aligned} WWWLoad &= \{\{\{WL, B_i^k, W_{ij}, W'_{ij}, C_{ij}, H_{ij}, F_{ij}, G\}, \\ &\quad \{t, s, r, \beta, \omega, \epsilon\}, \{P\}, WL\}, \{A\}, \{Q\}\} \end{aligned}$$

where

WL	is the Start symbol of the networkload
B_i^k	is a set of non-terminals denoting the different browsers
W_{ij}, W'_{ij}	is a set of non-terminals denoting the different windows in browser i
C_{ij}	is a set of non-terminals denoting the conversations made from window j in Browser i
H_{ij}	is a set of non-terminals denoting the http conversations from window j in Browser i
F_{ij}	is a set of non-terminals denoting the ftp conversations from window j in Browser i
G	is a non-terminal denoting the method GET
t	is the user think time
s	is the time to submit a request
r	is the elapsed time for fulfilling a request
β	is the time for starting a browser
ω	is the time for opening a window
ϵ	is a marker indicating the closing of an operation

The set of production rules $\{P\}$ with the associated probability set $\{Q\}$ and the attributes $\{A\}$ is given in Table 3. For each non-terminal, the calculation of start times is given. In addition, either the end time or the duration is given.

In the grammar, i is an index denoting the number of browsers $(1 \leq i \leq n_B)$, j denotes the number of windows per browser $(1 \leq j \leq n_W)$. β and ω are markers to denote the start of a browser and the opening of a window, ϵ denotes any termination. λ_{B_i} is the interarrival time, if there are already i started browsers and λ_{W_j} is the interarrival time, if there are already i windows open with $\lambda_{B_0} = \lambda_{W_0} = 0$.

The production rules are numbered in order to identify them in the control set. An index i in a production rule denotes the number of the browser, an index j denotes the window in the browser. Nonterminals B_i^k are used for expressing the different expansion possibilities within browser i, non-terminals W_{ij} and W'_{ij} denote the different expansions for window j in browser i.

Rule r_0 spawns the set of all possible actions at the top level (opening up to n_B browsers). We assume, that the observation of the workload starts at time T, thus $start(WL) := T$. The observation period ends with quitting the last browser, thus $end(WL) := max_i\{end(B_i^0\}$.

Rules $r_{i(n_W+1)}$ to $r_{i(2 \cdot n_W)}$ control the opening of new windows (starting a browser will automatically open a new window), while rules r_{i0} to r_{in_W} terminate the activities in a browser. Probabilities determine whether to expand a browser by starting windows or to quit a browser. Start and end times are calculated as given in the table (according to these definitions, the duration of the life-time of browsers may overlap).

Rules r_{ij1} to r_{ij4} define the behavior within a window as a sequence of think times and conversations (including the possibility to quit the window). The probabilities are used to decide which of the possible activities within

Rule No.	Production Rule, Probabilities, Attributes
r_0	$WL \to \beta B_1^0 \beta B_2^0 \dots \beta B_{n_B}^0$ $start(WL) := T;\ end(WL) := \max_i\{end(B_i)\}$ $start(B_i^0) := \begin{cases} time(\beta) + start(WL) & \text{for } i = 1 \\ time(\beta) + start(B_{i-1}) + \lambda_{B_{i-1}} & \text{else} \end{cases}$
r_{ik} for $k = 0..n_W$ r_{ik} for $k = (n_W + 1)..2 \cdot n_W$	$B_i^k \to \epsilon$ $B_i^{(k-n_W-1)} \to \omega W_{i(k-n_W)} B_i^{(k-n_W)}$ $Q_{ij}(p_{ij}^{quit}, p_{ij}^{expand})$ $start(W_{i(k-n_W)}) := start(B_i^{(k-n_W-1)}) + time(\omega)$ $start(B_i^{(k-n_W)}) := start(W_{i(k-n_W)}) + \lambda_{W_{(k-n_W)}} + time(\omega)$ $end(B_i^k) := \begin{cases} start(B_i^k) + time(\epsilon) & r_{ik} \text{ for } k = 0..n_W \\ \max_k\{end(W_{ik})\} & r_{ik} \text{ for } k = (n_W+1)..2 \cdot n_W \end{cases}$
$r_{ij[1\|3]}$	$W_{ij} \to tW'_{ij} \mid \epsilon$ $start(W'_{ij}) := start(W_{ij}) + time(t)$ $time(W_{ij}) := \begin{cases} time(W'_{ij}) + time(t) & r_{ij1} \\ time(\epsilon) & r_{ij3} \end{cases}$ $start(t) := start(W_{ij});\ end(t) := start(t) + time(t)$
$r_{ij[2\|4]}$	$W'_{ij} \to C_{ij} W_{ij} \mid \epsilon$ $start(C_{ij}) := start(W'_{ij})$ $start(W_{ij}) := end(C_{ij})$ $time(W'_{ij}) := \begin{cases} time(C_{ij}) + time(W_{ij}) & r_{ij2} \\ time(\epsilon) & r_{ij4} \end{cases}$
$r_{ij[5\|6]}$	$C_{ij} \to H_{ij} \mid F_{ij}$ $end(C_{ij}) := \begin{cases} end(H_{ij}) & r_{ij5} \\ end(F_{ij}) & r_{ij6} \end{cases}$ $start(H_{ij}) := start(C_{ij})$ $start(F_{ij}) := start(C_{ij})$ $Q_{ij}(p_{ij1}^{think}, p_{ij3}^{close}, p_{ij5}^{expandH}, p_{ij6}^{expandF})$
$r_{ij[7\|8]}$	$H_{ij} \to GH_{ij} \mid \epsilon$ $Q_{ij}(p_{ij7}^{next}, p_{ij8}^{done})$ $start(H_{ij.r}) := start(H_{ij.l}) + \lambda_G$ $end(H_{ij.r}) := \max end(G)$ $end(H_{ij.l}) := \begin{cases} end(H_{ij.r} & r_{ij7} \\ end(\epsilon) & r_{ij8} \end{cases}$
r_{ij9}	$F_{ij} \to G$ $time(F_{ij}) := time(G)$ $start(G) := start(H_{ij}) \mid start(F_{ij})$
r'	$G \to sr$ $time(G) := time(s) + time(r)$

Table 3 Production Rules, Probabilities, and Attributes for the WWW load model

a window should occur. The time of a window is calculated by aggregating the time of all activities within it (think times and conversation processing times). The think time is a parameter to be estimated, the time to process a conversation is either the time of the http or the ftp request.

In rules r_{ij5} and r_{ij6} we distinguish between an http and an ftp conversation, while rules r_{ij7}, r_{ij8}, and r_{ij9} defines behavior of an http resp. an ftp conversation. Probabilities are used for controlling the length of the GET sequence in an http conversation. The end time for an FTP is equal to the end time of the GET, while the end time of an http is determined by the largest end time of one of the GETS.

Finally, rule r' maps the actual timings of submitting and responding to a GET method, and the time of GET is simple determined by adding the time for submit and respond.

Whenever an operation terminates, the appropriate termination time is added.

In order to characterize the user behavior adequately, we have to consider the following restrictions to the grammar using a control set:

$$\begin{aligned}
C &= r_0 S && (1)\\
S &= (\sum_i [\sum_{j=n_W+1}^{2\cdot n_W} r_{ij} + S_i] + \sum_i \sum_{j=n_W+1}^{2\cdot n_W} r_{ij1} + S_{ij} + r_{ij3} + r_{ij4})^* && (2)\\
S_i &= (\sum_{j=0}^{n_B} r_{ij}) \prod_{j=n_W+1}^{2\cdot n_W} (r_{ij3} + r_{ij4}) && (3)\\
S_{ij} &= (r_{ij1}[r_{ij4} + r_{ij2}(r_{ij5}(r_{ij7}r')^* r_{ij8} + r_{ij6}r_{ij9}r')])^* r_{ij3} && (4)
\end{aligned}$$

The control set guarantees, that the grammar always starts with applying rule $r0$ (constraint 1). The behavior in a window is controlled by constraint 3 while the sequence of think and conversation processing is controlled in constraint 4. Constraint 2 guarantees, that when quitting a browser, all corresponding windows will be closed. By definition, the closing of a window will terminate the sequence of think/conversation phases.

The attributes of the Nonterminals representing the start and end times guarantee, that browsers and windows may run in parallel, but their start times are in increasing time stamp order. The activities in a window are strictly sequential, i.e. a sequence of think times and processing times with increasing time stamps. The start times of GETs belonging to a http conversation are in increasing time stamp order and may have overlapping durations.

Any sentential form generated by the grammar following the rules of the control set will give the following information: The non-terminals B will denote what browsers are currently open, the activities within each browser will be embraced by β and b resp. ϵ, if the browser has been quitted. The non-terminals W would give the number of active windows and the activities within a window will be embraced by ω and W resp. ϵ. The conversations in each window would be represented by the sequence of GETs generated from the ftp or http requests as well. Thus, we have an exhaustive representation of the "state" of the system.

For a quantitative description of the load, we have to associate values to the variables in the PACFG. In particular, we have to estimate the probabilities in $\{Q\}$ and a set of timing parameters, which will be discussed in the next section.

4.2 Parameter Estimates

(a) Characterizing Production Rule Probabilities

The probabilities for selecting production rules include:

p_{ij}^{expand}	the probability of starting browser i, $j = n_W + 1$, which would also start window $i1$.
p_{ij}^{expand}	the probability of opening windows $j - n_W$, $j = n_W + 2..2 \cdot n_W$
p_{ij}^{quit}	the probability of quitting browser i after window j, $j = 0..n_W$
p_{ij1}^{think}	the probability of thinking in window i of browser j
p_{ij3}^{close}	the probability of closing window i of browser j
$p_{ij5}^{expandH}$	the probability of expanding into an http conversation
$p_{ij6}^{expandF}$	the probability of expanding into an ftp conversation
p_{ij7}^{next}	the probability of issuing another GET
p_{ij8}^{end}	the probability of terminating the GETs in an HTTP

The probabilities can be obtained from logging the user behavior during browsing the www. If the source code of the Web browser is available, the source code can be instrumented for obtaining this information. Otherwise, a monitoring program has to be written collecting the user actions on the client side. Note, that the information in server or proxy log files is insufficient, as it would provide no information about the different instances of browsers and windows. From these measurements, relative frequencies can be obtained, which have to be "normalized" in order to meet the restrictions defined in the control set. It has to be guaranteed, that all probabilities of rules appearing in an OR term in a production rule will sum up to one.

The number of GETs within an http connection depends on the contents of the page to be retrieved. To estimate this parameter, either statistics about typical pages can be collected, characterizing the number of objects (text, images, ...) they contain. Alternatively, this information can be obtained by analyzing the log file of a proxy server, which can be configured to show for each GET the corresponding reference to the URL initially retrieved in the http conversation. The probability for next and end have to be chosen in order to generate GET sequences of characteristic length.

(b) Characterizing User Oriented Time Parameters

From the set of terminals, the following parameters are user oriented, i.e. depend on the particular behavior and preferences of the users*:

λ_{B_i}	interarrival time between browsers $i-1, i$
λ_{W_j}	interarrival time between windows $j-1, j$
$time(t)$	the user think time

The interarrival times at the browser level determine the time at which users will start a new browser. The arrivals will be bursty according to the time of the day. Furtheron, studies of the user behavior have shown, that typically users will navigate the web only on some days per week. For a characterization, all these aspects have to be taken into account, leading to the conclusion, that empirical distributions might fit best to characterize this behavior. The analyst might also consider to experiment with stress cases in setting the interarrival time rather low, resulting in a large number of browsers being simultaneously open. Similar considerations hold for the interarrival times at the windows level.

The user think time represents the time, the user needs to process the requested information. This timing information is difficult to capture and can vary from a few seconds (just having a glimpse at the page) to up to an hour (reading the contents of a text page). Thus, we would again propose to characterize this time by a histogram representing an empirical distribution of user think times. This empirical distribution can be derived from measurements of user sessions (logging the actions of the user on a particular machine). Studying the correlation between the contents of a page (given for example by the amount of text, that appears on a page) and the time a user needs to consume this information might lead to further insights in obtaining meaningful estimates. Observations made in the field of Human Computer Interaction will be a further source of helpful information, see e.g. (Catledge & Pitkow 1994).

As the user behavior will typically differ depending on the user's individual preferences, it seems reasonable to introduce classes of users with similar behavior. Such an approach, which is common practice for modeling of conventional workloads, has for example been followed in (Yan, Jacobsen, Garcia-Molina & Dayal 1996), but here the objective was to study user behavior from an HCI point of view.

*If a full study of all levels is conducted, also the interarrival time between user sessions has to be considered.

Example No.	Sentential form
1	$\beta, \epsilon, \beta, \epsilon$
2	$\beta, \omega, t, sr, \epsilon, \beta, \epsilon$
3	$\beta, \omega, t, sr, t, sr, sr, W_{1,1}, \epsilon, \omega, W_{1,2}, \epsilon, \beta, B_2^0$

Table 4 Possible Strings Generated by the Grammar

(c) Characterizing System Oriented Time Parameters

The following terminals belong to the group of system oriented parameters, i.e. their duration will depend on the performance of the system/network:

λ_G	interarrival time between two GETs
$time(s)$	the time to submit a GET
$time(r)$	the time to respond to the GET
$time(\epsilon)$	the time to terminate an activity
$time(\beta)$	the time to start a browser
$time(\omega)$	the time to open a window

The timings related to GETs - and to TCP/IP requests if all levels are considered - have been studied in various papers (Almeida, Bestavros, Crovella & de Oliveira 1996), (Arlitt & Williamson 1995), (Braun & Claffy 1994). As most Web servers and proxy servers provide access log files, these data can serve as a basis for the evaluation. In addition, network sniffer programs can be used to obtain further information.

Finally, the timings for opening/starting and terminating of browsers and windows have to be estimated. These timings depend on the performance of the client machine. Typically, these values will not vary much for a particular machine type and their order of magnitude is typically small compared to other timings. Assuming these values to be constant seems to be reasonable, e.g. an average of a set of measurements on the real system can be used.

5 EXAMPLE

The sentential form, which is the sting of terminals and non-terminals currently produced by the grammar, gives a representation of all the necessary details for capturing the state of the system at any instance in time (number of open browsers, number of active windows, ...).

Table 4 shows some possible strings. In all examples, we have assumed, that the number of browser a user may open is equal to 2 ($n_B = 2$), thus the first string generated by the grammar applying r_0 is $\beta, B_1^0, \beta, B_2^0$.

Example one is a rather pathological one, with no activity at all, i.e. B_1^0 and B_2^0 are both expanded to ϵ applying rules $r_{i,0}$ for $i = 1, 2$. In example two, a browser and a window are opened and after a thinking period (t) a request is performed, then the session is terminated. Example three shows a string, which is still in a phase of expansion, which can be seen by the presence of the non-terminal symbols W_1^0 and B_2^0. In this example, the user has opened two windows in the first browser, while the second browser has not been started yet. In the first window there is already some activity, which will continue by expanding non-terminal $W_{1,1}$. The activities in window two have not started yet, as $W_{1,2}$ has not been expanded.

The sentential forms generated by the grammar could be used for example as the input to a discrete event simulator of server behavior to obtain estimates on the load on the server. The terminals generated by the grammar indicate the arrival of events, the associated time stamps give their respective arrival times.

According to the desired number of users, one load generator per user would produce the corresponding string according to the particular user behavior (captured in the user-oriented parameters). A preprocessor would merge all the strings into a single arrival stream for the simulator.

Another application example would be the simulation of the cacheing behavior of a proxy server. In that case, the grammar has to be extended by adding another type of attribute, namely the identifier of the requested object (page, image, file, ...). The sentential form would then give a profile of the arrival requests for particular objects, which could again feed a simulator of cacheing behavior.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a hierarchical, generative approach for workload modeling of WWW-applications. We have identified 6 hierarchical layers, bridging the gap between the user level (characterized in terms of sessions) to the level of TCP/IP requests, i.e. the actual load imposed on the network. Our approach differs from previous attempts to WWW load characterization in that we consider both, the actual physical characteristics of the system as well as the application oriented view. Thus, the analyst is able to investigate both, changes in user behavior as well as the effects of changes in the system characteristics.

By using PACFGs for modeling, we are able to increase the expressiveness of the grammar while the control set will ensure, that the generated workloads truly represent the user and system behavior (e.g. closing a browser will terminate all activities within). The sentential forms give a representation of all the necessary details for capturing the state of the system at any instance in time (number of open browsers, number of active windows, ...), while the timing attributes provide all information for further evaluation studies.

Future work will focus on a detailed comparison of the expressiveness and representativeness of the proposed approach in contrast to simpler models of WWW traffic characterization.

REFERENCES

Almeida, V., Bestavros, A., Crovella, M. E. & de Oliveira, A. (1996), Characterizing reference locality in the www, Technical Report TR-96-11, Department of Computer Science, Boston University, USA.

Arlitt, M. & Williamson, C. (1995), A synthetic workload model for internet mosaic traffic, Technical Report TR-95-08, University of Saskatchewan, Canada.

Arlitt, M. & Williamson, C. (1996), Web server workload characterization: The search for invariants, *in* 'Proceedings of ACM SIGMETRICS'96', Saskatchewan, Canada.

Brakmo, L. & Peterson, L. (1996), Experiences with network simulation, *in* 'Proceedings of ACM SIGMETRICS'96', Saskatchewan, Canada.

Braun, H.-W. & Claffy, K. (1994), Web traffic characterization: An assessment of the impact of caching documents from ncsa's web server, *in* 'Proceedings of Second International WWW Conference', Chicago, IL, USA. `http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/claffy`.

Catledge, L. D. & Pitkow, J. E. (1994), Characterizing browsing strategies in the world-wide web, *in* 'Proceedings of Third International WWW Conference'.

Crovella, M. & Bestavros, A. (1996), Self similarity in world wide web traffic: Evidence and causes, *in* 'Proceedings of ACM SIGMETRICS'96', Saskatchewan, Canada.

Cunha, C. R., Bestavros, A. & Crovella, M. E. (1995), Characteristics for www client-based traces, Technical Report BU-CS-95-010, Computer Science Department, Boston University, USA.

Fu, K. (1974), *Syntactic Methods in Pattern Recognition"*, Academic Press.

Raghavan, S., Vasukiammaiyar, D. & Haring, G. (1993), Generative models for networkload in a single server environment, Technical Report CS-TR-3166, UMIACS-TR-93-112, University of Maryland, College Park.

Raghavan, S., Vasukiammaiyar, D. & Haring, G. (1995), 'Hierarchical approach to building generative networkload models', *Computer Networks and ISDN Systems* **27**(1), 1193–1206. (Reprint available).

Salomaa, A. (1973), *Formal Languages*, Academic Press.

Sedayao, J. (1994), Mosaic will kill my network!, *in* 'Proceedings of Second International WWW Conference', Chicago, IL, USA. `http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/sedayao/`.

S.V. Raghavan, B.Prabhakaran, S. K. T. (1996), 'PACFG Based Synchronization Model for Multimedia Presentation', *IEEE Journal on Selected Areas in Communications* **14**(1).

S.V. Raghavan, D. V. A. & Harring, G. (1994), Generative networkload models for a single server environment, *in* 'Proceedings of ACM SIGMETRICS'94', Nashville, Tennessee, pp. 118–127.

Yan, T. W., Jacobsen, M., Garcia-Molina, H. & Dayal, U. (1996), From user access patterns to dynamic hypertext linking, *in* 'Proceedings of Fifth International WWW Conference', Paris, France. `http://www5conf.inria.fr/fich_html/papers/P8/`.

7 BIOGRAPHIES

Gabriele Kotsis was born in 1967 in Vienna, Austria. She received her masters degree (1991) and her PhD (1995) from the University of Vienna. Since December 1991 she is working at the Institute of Applied Computer Science and Information Systems, University of Vienna as a researcher and teacher. Her research interests include performance modeling of parallel and distributed systems, CAPSE (Computer Aided Parallel Software Engineering), visualization of parallel program behavior and performance, interconnection networks, and workflow systems and simulation.

Kamala Krithivasan received her B.Sc. (1967), her M.Sc. (1969) and her PhD (1974) from the University of Madras. She is currently a full professor at the Indian Institute of Technology, Madras. Her research interest focuses on automata theory.

S.V.Raghavan was born in 1951 in Madras, India. He received a B.Sc. in Physics, University of Madras 1971, a D.M.I.T. in Electronics, Madras Institute of Technology, 1974, an M.E. in Automation, Indian Institute of Science, 1976, and a Ph.D. in Computer Science, Indian Institute of Technology, 1986. He is currently a full professor and the head of the Department of Computer Science and Engineering at the Indian Institute of Technology, Madras. Prof. Raghavan has written over 80 papers in national and international journals and conferences relating to computer systems workload modeling, highspeed networks and protocols.