[Not for Distribution or Attribution: For Review Purposes Only]

A Two-layer Characterization of Front-End E-commerce Traffic

Krishna Kant
Server Architecture Lab
Intel Corporation, Beaverton, OR
krishna.kant@intel.com

M. Venkatachalam
Dept of Electrical and Computer Engineering
University of Texas, Austin
mvenkata@cs.utexas.edu

Abstract

In this paper we provide a systematic two-layer characterization of e-commerce front-end traffic, where the request arrival process characterization is done at the first layer and user behavior characterization (transactional analysis) is done at the next layer. The contributions made by this work are threefold: a) At the first layer, we demonstrate the presence of unmistakable non-stationarity in the request arrival process to the e-commerce server and provide a novel technique to analyze such non-stationary and long-range dependent traffic, b) At the second layer, we characterize the user behavior by a generic embedded Markov chain model and the embedded requests by a distribution function, and c) With the two-layer characterization, we devise a technique for efficient, synthetic, realistic e-commerce traffic generation in the laboratory. In this characterization we separately examine data from both Business-to-Business (B2B) and Business-to-Consumer (B2C) environments and point to some key differences between them.

Keywords: e-commerce server, time-series, self-similarity, nonstationarity, transactional characterization, Markov chain, Embedded requests, traffic generation.

1 Introduction

With E-commerce taking a central place in today's economy, the traffic in the e-commerce environment has become an important issue. It has been shown that Internet traffic is long-range

dependent. Here, we empirically show the non-stationarity of e-commerce traffic over the time-scales that are relevant to long-range dependence. This complicates matters since long-range dependence analysis assumes stationarity. Further, the user behavior is significantly more complex in the e-commerce environment. This leads us to a systematic two-layer characterization of e-commerce traffic, with the first layer characterizing the request arrival process and the second layer characterizing the user behavior in terms of the base and embedded requests. Apart from the characterization itself, the goal is to enable realistic, synthetic traffic generation in a scalable way, in the lab.

The request arrival process characterization in the web environment is a well studied problem. [2] has shown that the web traffic exhibits long-range dependence. Further, web browsing traffic has been shown to be stationary over the time-scales of interest for long-range dependence. But, in the e-commerce environment, we uncover the unmistakable presence of non-stationarity, with the non-stationarity time scale being of the order of 10-15 minutes. Unfortunately, such time-scales are comparable to user actions and session lengths and hence are very relevant for the characterization of long-range dependence. It is well known that non-stationarity could be mistaken for long-range dependence. This raises questions as to the accuracy of the H parameter obtained and calls for a characterization of non-stationarity so that realistic synthetic e-commerce traffic generation is made possible. To go about this problem, we conjecture that the non-stationarity is merely due to the nonstationary marginal distribution of the request arrival process, whereas the inherent correlational properties of the arrivals is stationary. With this conjecture, we use *Abry-Veitch* ("AV") wavelet estimator to estimate H parameter, which is known to be robust to nonstationary marginal distribution of the request arrival process. Once the long-range dependence is accurately characterized, the "Residual Time Series" could be examined for non-stationarity characterization, which we discuss in section 3.

At the user behavior layer, e-commerce traffic is markedly different from that of normal web-traffic, as the user behavior in the web environment is typically browsing and maybe a little searching, whereas in the e-commerce environment, there are a gamut of things that a user does from browsing the hotlist, to searching the product database to ordering products to checking status of past orders and so forth. Thus the transactional analysis in the e-commerce environment is far more complicated as compared to the web environment and hence an analysis framework for the user behavior is necessary (transactional analysis and user behavior will be used interchangeably henceforth). We propose a generic embedded Markov chain model for the user behavior with each

state representing a particular user request/action. With each base request, there tend to be several embedded requests, which are characterized by a random variable and its associated marginal distribution function. Although a complete e-commerce traffic characterization must include characterization of middleware and backend traffic as well, in this paper we concentrate on only the front-end.

In examining this data, we found a number of striking differences between business to consumer (B2C) and business to business (B2B) environments themselves. As usual, individual e-commerce sites vary considerably in their characteristics; therefore, it is possible that the differences noted here do not apply to all sites; however, we believe that these differences are quite representative.

1. B2C sites are characterized by a rather low percentage of purchase transactions as compared to B2B sites. Thus, a detailed representation of various stages in the purchase (e.g., order placement, order status checking, order changes, payment, etc.) may not be necessary, and a simpler model can be used than for a B2B environment.

2. The fraction of secure transactions (typically used for payments) is rather small in B2C environments. In contrast, B2B environments often make all transactions secure in order to disallow any information gathering about the transactions.

3. In both B2C and B2B environments, the probabilities of product search and picking product from hot list are small (as compared with sequential browsing); however, B2C environments appear to have more searches vs. B2B environments that tend to pick products from the hot list more often.

4. As expected, traffic busy periods for B2C and B2B environments are very different. In particular, B2C busy periods are almost complementary to those for B2B, e.g., higher during weekends, lunch hour and evening.

5. B2B systems are apparently designed for much higher robustness than B2C systems, with a lot of background "healthcheck" transactions between the systems.

The rest of the paper is organized as follows: in section 2, we deal with the characterization of the request arrival process, wherein we contrast the arrival processes in the B2B and B2C environments with the web environment and confirm the non-stationarity of e-commerce traffic.

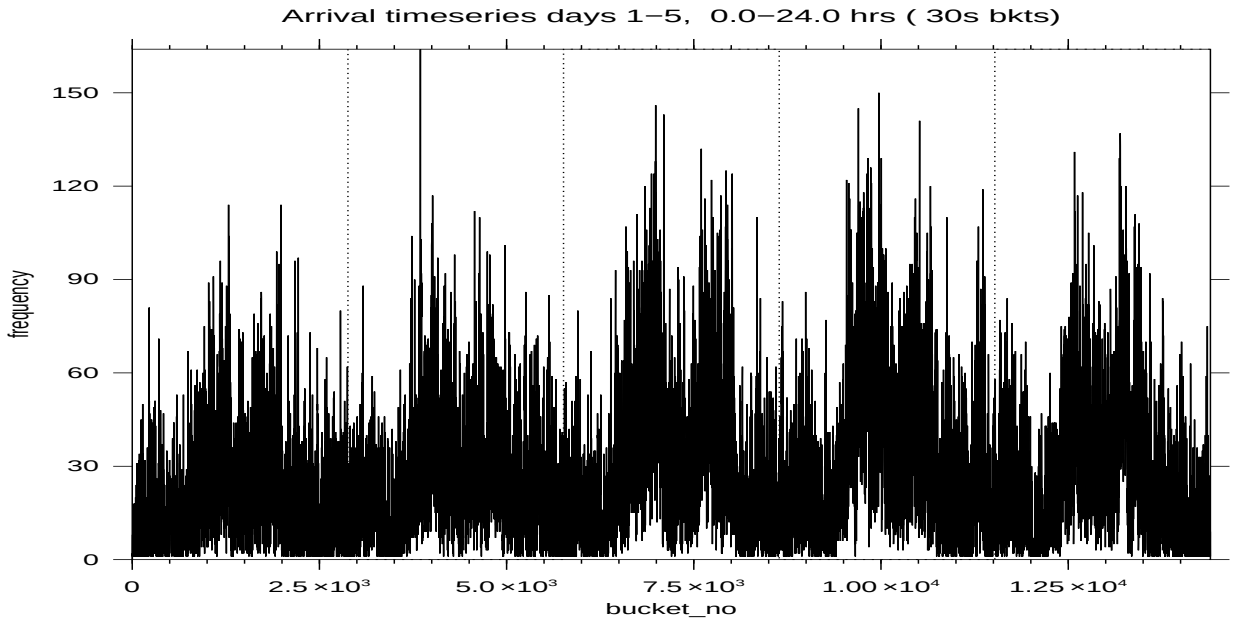

Figure 1: Five day worth of B2B traffic, time-scale = 30s

In section 3 we present a brief overview of the analysis of non-stationarity for the B2B and B2C environments. In section 4, we provide a transactional analysis in the B2B/B2C environments while ignoring the embedded requests which are considered in section 5. Section 6 deals with synthetic traffic generation issues and we conclude in section 7, stating some of the possible future directions of research.

2 Overall Traffic Characteristics

2.1 B2B environment

We examined the HTTP logs for several consecutive weeks of B2B traffic. A business environment is expected to show very low traffic during weekends, therefore, only Monday to Friday data was used for each week. Figure 1 shows the arrival process for 24 hour periods for 5 successive days (dotted lines separate day boundaries). It is seen that the traffic can vary significantly from week to week; in fact, the traffic for the first week is much higher than for all other weeks. This means that one cannot simply take the traffic for an arbitrary week and use it for server engineering. A similar situation exists with respect to days within a week. In all cases, the server errors were found

to be negligible, which indicates that the server didn't experience any periods of overload. (In a mission critical environment such as B2B, it is perhaps expected that the site will have plenty of extra capacity in order to make overloads very rare.) The client errors were also found to be very low. In this environment, a "client" refers to the "transactor node" through which all transactions must pass; thus, low client errors are not surprising.

The request traffic must be examined at several time-scales in order to get a true picture of its nature. In particular, while daily/weekly traffic is relevant for getting an idea of average traffic, traffic over much smaller intervals is more relevant from capacity planning perspective. As a first step, we need to know if there is a well-defined "busy period" during a day, and how stable this busy period is. The first observation from Figure 1 is that the daily profile varies quite a bit within the week. The busiest period is typically between 7:00 AM to 6:00 PM but with some sharp drops during lunch hour. In particular, consistent with typical behavior in a business environment, one could identify a "morning busy period" during about 7:30–11:30 AM and an afternoon busy period 1:30–5:30 PM. However, there appears to be lack of stability in the start/end points of the busy period and the behavior during the busy period (see Figure 2 which gives an expanded view of busy-period traffic). This is markedly different from the behavior that we have seen repeatedly in a pure web-browsing environment where stationarity is a reasonable assumption over the busy period. One crude way of looking at this is to examine the extent of variation in the traffic profile during the busy period when viewed at a rather large time scale. Figure 3 shows this for B2B traffic at a time scale of 900 secs. (The overall mean has been normalized to 1.0 in order to better highlight the variations.) It can be seen that even at this large time-scale, the traffic varies over a rather large range and does not show any regular pattern from day to day (the plotted data is for daily busy periods over 5 weekdays during the first week). This suggests potential nonstationarity in the traffic, which we shall examine in detail later.

2.2 B2C environment

Figure 4 plots the time series for four days worth of data in the B2C environment for a bucket size of 30s. This data has been obtained from a popular e-commerce site. The non repeatability of daily traffic profile is quite obvious from this figure. The figure shows some typical characteristics of week-day B2C traffic. As expected, the day time traffic is rather low. The traffic picks up in the afternoon and is usually at its peak in evening hours. The busiest period is around 7:00 pm

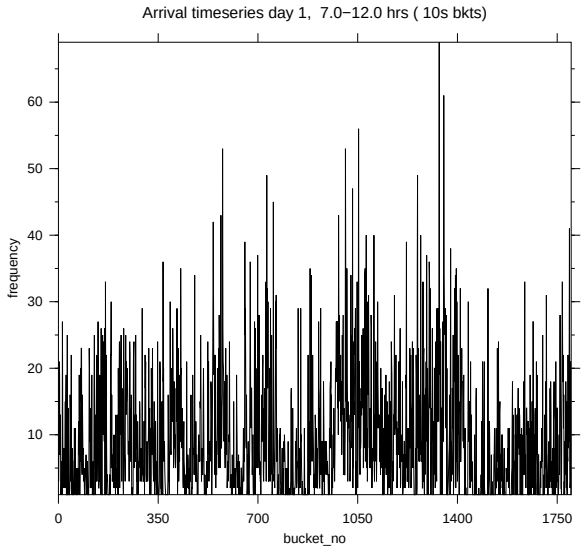

Figure 2: B2B busy-period traffic for day 1, time-scale = 10s

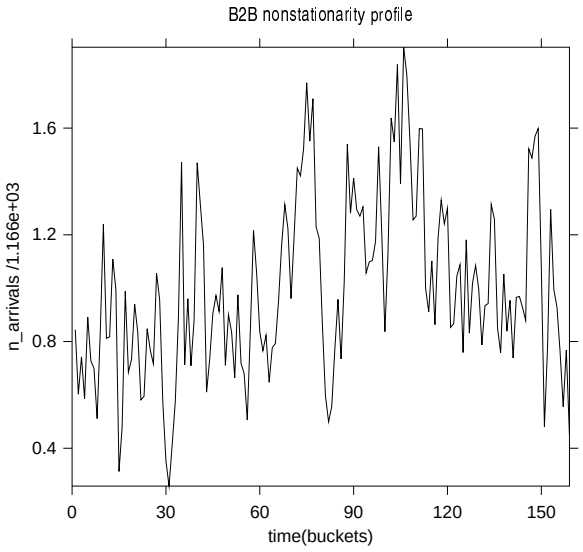

Figure 3: B2B busy-period nonstationarity profile

to 11:00 pm; however, a close examination of these periods over successive days shows a lack of stationarity and stability in the traffic pattern. The lack of stability is more clearly seen from Figures 5 which show 7-11 pm traffic on day 3. The lack of stability in a B2C environment is often caused by special sales, a phenomenon that usually does not occur in B2B environment. Also, the given data doesn't really support the identification of busiest period over 7–11 pm; it appears that depending on which day we consider, the onset and duration of the busy period could change very considerably. Part of the reason for this may be accesses from different time zones, which tends to make the concept of "busy-period" rather fuzzy. In fact, for a mega-site serving many time-zones (such as amazon.com, which is not what we analyzed), there may be no discernible busy periods. In such cases, the analysis may need to consider traffic over 10 hours or more each day.

From this and other data that we have examined, it appears that e-commerce traffic cannot be considered to be reasonably stationary. This observation is primarily based on visual characteristics of the time-series since a formal analysis of stationarity for a potentially long-range dependent arrival process is an open problem. It is possible that the traffic is stationary over small durations (e.g., a few minutes). Unfortunately, the time resolution of the HTTP logs is only 1 second which means that at least an hour's worth of traffic is necessary to do any credible statistical analysis. Moreover, since the time-scale for users actions could be a few minutes or longer, the characteriza-

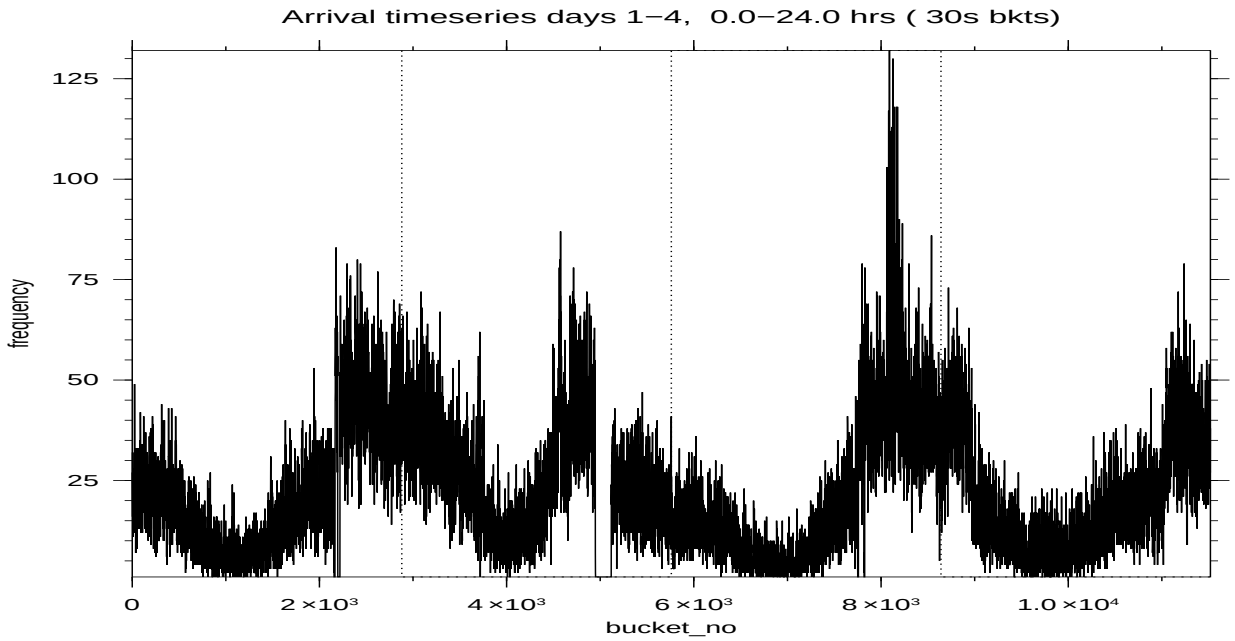

Figure 4: B2C time series for a four day period, time-scale = 30s

tion of long-range dependence also needs to consider periods of about an hour or longer.

3 Modeling Nonstationarity

In order to study the properties of e-commerce traffic, we make the following two assumptions and attempt to verify them using the data:

1. Scaling properties of the traffic are stationary, i.e., the nonstationarity is merely a result of nonstationary marginal distribution of the arrival process.

2. The nonstationarity enters into the picture at a specific time-scale, henceforth denoted as NST (nonstationarity time-scale). That is, the traffic can be considered approximately stationary at time-scales smaller than NST.

As usual, we shall concentrate only on the second-order properties of the traffic. Thus, we are only concerned with weak-sense stationarity properties. This, coupled with assumption (1) above implies that we only need to ensure that the first two moments of the arrival counting process are independent of the time origin [4].

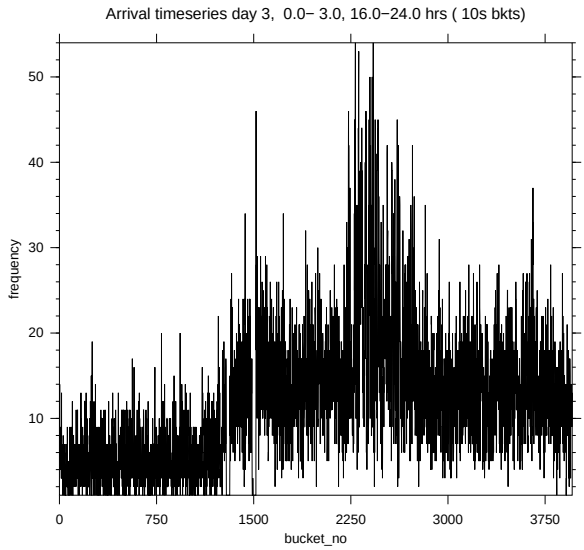

Figure 5: B2C time series during day3 busy period, time-scale = 10s

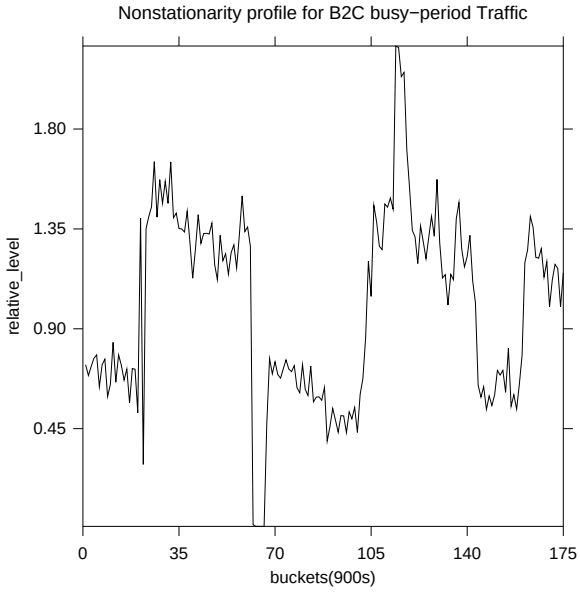

Figure 6: B2C busy-period nonstationarity profile

Intuitively, assumption (1) amounts to saying that the fundamental nature of user behavior (i.e., heavy-tailed active and inactive periods) does not change over time; only the time between successive requests may change over time. Such a view could explain our observation that the busy-period traffic appears stationary in a web-browsing environment, but not in an e-commerce environment. The traffic in an e-commerce environment is strongly influenced by sales promotions and other time-sensitive events. Scaling properties of such a model have been studied in [11] which shows that the wavelet based estimator of H parameter (or "AV" estimator) works very well in such an environment. We also note here that this paper is only concerned with traffic properties during the typical "busy period" of the e-commerce sites, rather than the entire day or week. In [10], the authors examine the diurnal variation of the H parameter, but such a characterization is not pursued here.

In reality, nonstationarity may arise at several time-scales over the course of the period of interest. However, it suffices to determine the smallest such time scale, which we denote as NST. Thus, assumptions (1) and (2) essentially imply that the non-stationarity can be described by a "level-shift" process operating at time-scale NST. Let Δ denote the duration of a bucket over which the original arrival process $\{X_i, i = 1, 2, \ldots\}$ is defined. Then, NST= $\Delta \times J_0$ for a suitably chosen

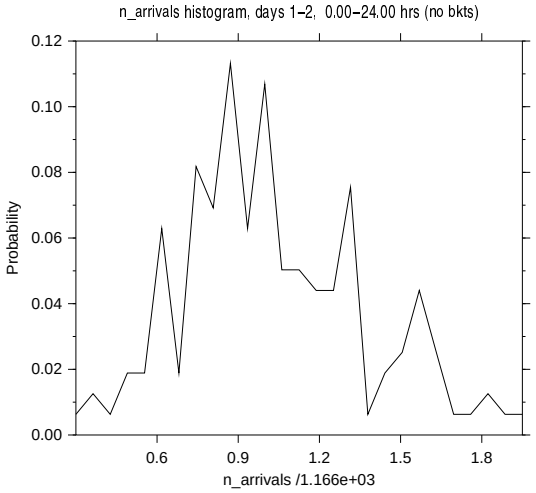

Figure 7: Distribution of Z during busy period for B2B

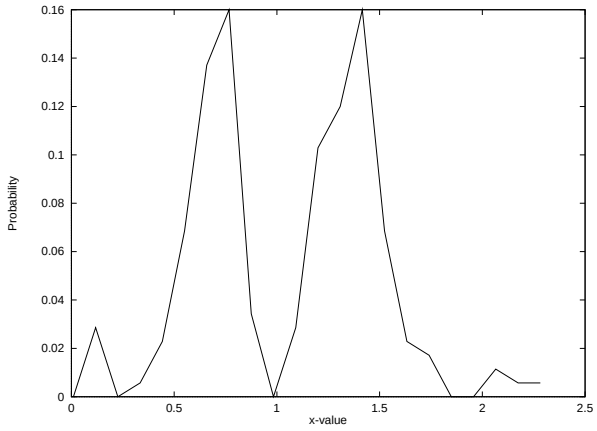

Figure 8: Distribution of Z during busy period for B2C

J_0 and we shall refer both τ and J_0 as the non-stationarity time-scale. In general, it is difficult to determine J_0 very precisely; therefore, we shall mostly assume a dyadic scale where the time-scale values J are chosen from a range of values and the most suitable value J_0 is chosen from among them. The algorithm for doing this can be cast as follows:

1. Based on the visual inspection of the original arrival time series $\{X_i, i = 1, 2, \ldots\}$, choose the smallest value of J at which nonstationarity may appear to come into play.

2. Aggregate the time-series over $K = J$ buckets, i.e., find $Y_n = \frac{1}{K}\sum_{i=(n-1)K+1}^{nK} X_i$, $n = 1, 2, \ldots$. We call the resulting process $\{Y_n\}$ as the *nonstationarity profile* of the traffic.

3. Find the residual process $\{R_i = X_i - Y_n, i = (n-1)K + 1 \ldots nK\}$. If $\{Y_n\}$ truly represents nonstationary component of traffic intensity, the process $\{R_i\}$ must be stationary.

4. Check if $\{R_i\}$ is approximately stationary and has the same scaling behavior as the original process $\{X_i\}$. If so, we have the desired value J_0; otherwise, choose another J and go to step 2.

5. Once J_0 is determined, take the corresponding nonstationarity profile $\{Y_n\}$ and characterize it as a random level shift process.

The details of this process are reported in [7] and are omitted here. The net result from this analysis for our B2B traffic is that NST= 900 sec is optimum from the perspective of both the

stationarity of the residual time-series as well as the scaling properties of the traffic.

Note that we can view $\{Y_n\}$ as the finite realization of a level-shift process Y. As defined, the sequence $\{Y_n\}$ gives average number of arrivals over successive intervals, which means that the expected value of Y is the overall expected number of arrivals per interval. Let us define a new random variable $Z = Y/E[Y]$, and the corresponding sequence $\{Z_n = Y_n/\overline{Y}\}$ where $\overline{Y}$ denotes the sample mean of the sequence $\{Y_n\}$. From a traffic characterization, the RV Z is more meaningful as it can be interpreted as the multiplicative factor to the overall arrival rate. Figure 3 introduced earlier actually shows the time-series $\{Z_n\}$ by considering only the busy period of each day. Figure 7 shows the corresponding probability distribution function. If necessary, one may attempt to fit an analytic distribution through this empirical distribution.

In the B2C case, the nonstationarity profile $\{Z_n\}$ was presented earlier in Figure 6 (for NST= 900 sec). The distribution of Z is shown in Figure 8. The bimodal nature of the distribution on either side of the mean of 1.0 is interesting and perhaps points to lack of stability in the traffic.

4 Transactional Analysis

Beyond the arrival process, the most important traffic characteristics relates to the sequence of transactions issued by a client during its server interaction session. This is particularly important in the e-commerce scenario as opposed to the web environment, due to the complexity and number of transaction types possible here. Transactional analysis involves two important issues – identification of user sessions and indentification of transaction types. In e-commerce sites, the users are usually identified by means of a Globally Unique ID (GUID) or a session ID (SESSIONID) for a particular user/session. We encountered both the types of identification. In the B2B case, SESSIONID was the mode of identification. In case of B2C, GUID was used. If SESSIONIDs are used, then a simple SESSIONID comparison would yield user sessions. If GUID is used, there is a further issue of temporal separation of sessions by same user, since the GUID is independent of the time of access. This can be tackled by using a threshold on the session time. In our results, we use a threshold of 30 mins for the session time. We found that the time between successive transactions from the same user exceeds 30 minutes only 3% of the time, however, this mass stretches into a very long tail. Thus, a threshold of even 2 hours would have left out 0.5-1.0% of the mass.

4.1 Identification of Transaction Types

A detailed characterization of transactions proved difficult because the details of the applications running on the site are generally unavailable. In fact, e-commerce servers often run multiple applications simultaneously. Because of the inability to distinguish between various applications, we consider them as a single canonical application. The only way to tell what a given request is all about is to examine the server-side scripts used to build the requested web-page for the client. The sites we examined used Microsoft's active server pages (ASPs) for this purpose. Since the ASPs invoked are recorded in the HTTP log, it is possible to do a characterization based on those. We note here that a real e-commerce site might use thousands of ASP scripts, but most of them are simple variants of a base ASP. Although the number of base ASPs may run into 100s, top 10-20 ASPs account for more than 99% of the work. In most cases, actual ASP code is not available, and one has to depend on indirect ways of finding their functionality (including site browsing and looking at the HTML code).

In identifying the important transaction types, we determined the frequency of usage of all the ASP scripts invoked over several week's worth of HTTP logs. These were arranged in decreasing order of frequency, and all ASP's that collectively amounted to less than 0.1% of the total ASP samples were simply ignored, whereas many others were grouped together because they performed very similar functions. This still leaves more scripts than what may be really necessary for an effective characterization of the traffic. Thus, the next step is to identify and group the ASPs further driven by their traffic impact.

4.2 Generic User Behavior Model

We classify and group the server side scripts by their abstract functionality. For instance, all scripts that involve the use of the backend database server can be classified into *one state*. All the requests to search server can be classified as another state, and so on. Since the ultimate aim of the classification is to study its impact on server performance, such a classification is particularly attractive. In general, we propose the following classification for any generic E-commerce site.

- **Static:** This class refers to static web-pages (excluding the embedded requests), some of which might be wrapped within ASP scripts. In traditional architectures, these accesses involve streaming of data from the memory to the network interface (NIC), usually with a lot

of CPU intervention. Emerging architectures may also do a direct streaming of large pages from the disk to the NIC.

- **Dynamic:** This state represents all the requests for dynamic pages that do not involve accessing the search server or the backend. A typical example would be looking at hotlists or simply constructing the web-page from logos, images, advertisements, text, audio, etc. This process is typically very CPU intensive.

- **Search:** This state represents all the requests that result in a significant search activity (e.g., search for web-pages with given keywords, or search for a product in a backend database). Since such searches are typically run on a separate server (search server or backend database), the impact on the front-end server is typically in terms of handling a large amount of data.

- **Backend:** This state represents all the requests to the backend database (other than the searches). When a significant non-search database activity is involved, this state can be further decomposed into distinct states such as shopping_cart, product_order, order_status, etc.

- **Home:** This is a hypothetical state from which all the users start and end their sessions.

Even though the above classification looks very reasonable, it runs into trouble if applied literally. Web servers running Microsoft's Internet Information Server(IIS) often tend to encapsulate every web page inside an Active Server Page (ASP). ASPs can contain plain HTML along with clearly marked visual basic scripts that generate HTML. The ASP interpreter executes the script part to generate the HTML code, but leaves the existing HTML alone. Consequently, it is possible to simply rename an HTML file as an ASP file, and it would work just fine (it will simply be scanned for scripting commands by the ASP interpreter, but no other effort will be expended on it.) Literally speaking, such encapsulated HTML would be classified as "dynamic", but it is really entirely static. An even more confusing, but frequent situation occurs in product description ASP scripts. Such an script takes one parameter, which is simply the unique id for the desired product. Using the id, the script retrieves the static page describing the product. In this situation, it appears that the ASP script dynamically generates a different page each time, but in reality it may do little additional work than retrieving the static page directly. We have classified such transactions as "static" in the data shown in this paper.

Often, the product description pages are stored as records in the product catalog database. In this case, the script may actually have to go to the backend to obtain the page. However, backend access would be needed only if the page is not already cached in the front-end due to an earlier access. This is one instance where classifying transactions by merely the names of the scripts breaks down. The correct approach is to classify the transaction as backend if the backend access is actually involved, and as static/dynamic otherwise. If such a classification is not possible, one could perhaps take the following approach: (a) assume that all product pages come from the cache and thus the transactions can be described as static or dynamic, and (b) determine the intensity of backend transactions based on overall access characteristics to the backend. In this approach, a one to one correspondence between front-end transactions and back-end accesses is lost.

Proper handling of security depends on whether all transactions or only certain sensitive ones are secure. Some examples of the former are most B2B e-commerce sites, and banking/trading related B2C sites. These sites authenticate the user upon sign-up and then use HTTPS for all transactions of the session. In contrast, most e-tailer sites use HTTPS only for the sensitive transactions such as payment, account information, etc. If all states are secure, no special treatment is needed for security. If only some of the backend transactions are secure, the "backend" state can be either split into secure and non-secure components, or we can retain just one state and simply specify the percentage of secure transactions. The only disadvantage of the latter choice is that any sequencing between secure and non-secure transactions is not represented. We believe that this loss of precision is unimportant in most cases, and pick the second approach. In fact, we generalize this approach in that every state can have certain percentage of secure transactions, and thereby cover the 100% secure environments as well.

Given the states, a user session can be represented by transitions between these states. Each user starts and ends the session in the home state. During the time that a user is in a given state, it may issue one or more transactions relevant to that state. The transitions between states could, in general, depend on more history than just the current state. In fact, in some cases, we found significant second order effects (i.e., a transition's dependence on last two states visited). Thus, in general, user sessions can be described by a kth order semi-Markov chain (with an appropriately chosen k).

Finally, we mention that a single state can be split into several states, as and when necessary. A good example of this phenomenon would be the "backend" state in the B2B environment. Since

B2B environments are usually very database intensive, it probably makes sense to split the backend state into several states each catering to a particular kind of database transaction.

4.3 Model for B2C environment

A typical characteristic of the e-tailer B2C environment that we analyzed is that most of the traffic concerns "window shopping" (searching products, retrieving production information, comparing prices, etc.) and very little had to do with actual product purchase. In fact, most of the entries made to the shopping cart are done merely as a "bookmark" and are often discarded eventually. Thus a single "backend" state is quite adequate, with the payment transaction part of it considered as secure. Assuming a first order embedded Markov chain model, Table 1 shows the transition probabilities.

The first column in Table 2 shows the probabilities of various states using the embedded first-order Markov chain model. It is seen that although 29% of the visits are spent in the backend state, nearly all of the accesses concern accessing product description pages, rather than going through the purchasing steps. The other two columns in Table 2 show the mean and standard deviation of the residence time in milliseconds in each state. It is seen that residence times are highly variable in most cases (a standard deviation of 6-8 times that of the mean). Note that since embedded requests are ignored in this analysis, the residence time in a state includes the time to request and load embedded pages as well.

We also considered a second order embedded Markov chain model in order to capture any significant second order transitions. We came up with three significant second order transitions that were an order of magnitude higher in probability than than the rest. The significant transitions were: static to dynamic to static; dynamic to static to dynamic; dynamic to search to dynamic. Given the close relationship between dynamic and static pages, a substantial chance of alternating sequences of them is not surprising.

4.4 Model for the B2B environment

Unlike the B2C scenario, the B2B scenario involves a whole lot of actual purchasing and other back-end activities. Therefore, we split the backend state into request changes, order placements, pricing enquiries and order status, due to the heavy volume of database transactions. The functionality of these operations is discussed below:

state	static	dynamic	search	backend	home
static	0.3413	0.0616	0.0165	0.1962	0.3844
dynamic	0.0887	0.1842	0.0157	0.4696	0.2419
search	0.0440	0.0236	0.4440	0.1916	0.2967
backend	0.0985	0.0729	0.0416	0.3838	0.4031
home	0.1391	0.4131	0.0710	0.3768	0.0000

Table 1: State transition probabilities in the B2C scenario

name	no	state-prob	mean	stddev
static	1	0.120	102.3	309.7
dynamic	2	0.191	43.8	220.6
search	3	0.051	89.1	459.2
backend	4	0.293	96.5	348.9
home	5	0.345	1.0	0.0

Table 2: Markov chain state properties for the B2C environment

1. **price_avail:** Checks the price and availability of the products.
2. **order_prod:** Handles placement of an order.
3. **order_status:** Checks the status of an order.
4. **change_req:** Makes changes to an existing order.

The dynamic state primarily consisted of just one script - hot_list, which examines the current "hot-list" and selects products from them. The static pages consisted of most of the other ASP scripts (which, as stated earlier, simply retrieved static pages based on one or two parameters). Also, all user interaction with the server occurred via HTTPS where a secure connection is established at the beginning of the user session. Table 3 shows the transition probabilities for this environment. One immediate conclusion from this matrix is that all states are "sticky", i.e., the Markov chain stays in the same state with a very high probability. The data extracted for each session shows that the client issues a large number of page views for each core operation before moving on to the next step. Table 4 shows the state probabilities and statistics of residence times. It is clear that purchasing forms a very substantial part of B2B activity and thus a finer grain characterization of backed activities is essential. As with B2C, the state residence times have a high variability and thus cannot be reasonably modeled as exponential. Again, since embedded requests are ignored in

current state	change request	dynamic page	order product	order status	product avail	search prod	static page	home state
change_req	0.9215	0.0047	0.0101	0.0276	0.0085	0.0003	0.0123	0.0150
dynamic_page	0.0077	0.8816	0.0152	0.0388	0.0123	0.0108	0.0130	0.0206
order_prod	0.0054	0.0044	0.9151	0.0313	0.0107	0.0047	0.0119	0.0166
order_status	0.0080	0.0063	0.0186	0.9180	0.0163	0.0018	0.0116	0.0194
price_avail	0.0053	0.0045	0.0141	0.0332	0.8964	0.0008	0.0139	0.0318
search_prod	0.0032	0.0363	0.0548	0.0430	0.0098	0.8256	0.0159	0.0114
static_page	0.0126	0.0083	0.0241	0.0551	0.0283	0.0019	0.8568	0.0129
home_state	0.0033	0.0031	0.0071	0.0276	0.0146	0.0001	0.0154	0.9287

Table 3: Transition matrix for the first order Markov chain in the B2B environment

state_name	no	residence-prob	mean	stddev
change_req	1	0.057	590.0	1810.7
dynamic_page	2	0.050	482.3	1753.6
order_prod	3	0.124	448.1	1524.1
order_status	4	0.230	750.0	2525.7
price_avail	5	0.127	814.4	2735.4
search_prod	6	0.020	195.3	841.4
static_page	7	0.116	239.9	1266.3
home _state	8	0.275	110.2	1145.6

Table 4: Markov chain state properties for the B2B environment

this analysis, the residence time in a state includes the time to request and load embedded pages as well. We also looked at second order dependencies in this environment and surprisingly we couldn't find any that was significantly higher than the rest. It is not clear if this is just accidental, or an inherent property of B2B sites.

In addition to the regular client (or user) transactions, our B2B environment had a number of "system" transactions that are routinely sent to the server by the transactor for health-check. We found that the system transactions amount to about 21% of the total transactions, and thus have a substantial influence on performance. However, since system transactions are very much site specific, we ignore them in this paper.

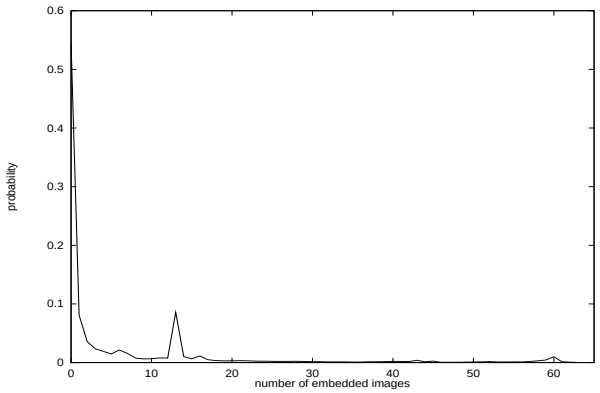

Figure 9: Embedded image probability distribution for B2C scenario

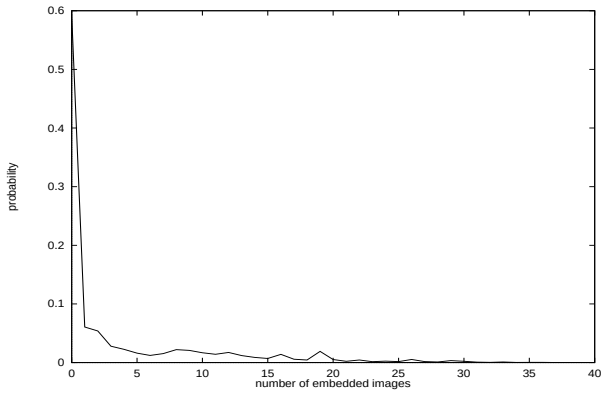

Figure 10: Embedded image distribution function for B2B scenario

Environment	95 percentile	mean	stddev
B2C	37	6.22	12.71
B2B	19	3.58	6.58

Table 5: Statistics for the embedded images

5 Modeling Embedded requests

The HTML code for a web page often contains references to other objects (e.g., images, javascript for animation, etc.) which are not an integral part of the "base page" and are instead loaded via explicit requests from client's browser when it interprets HTML code. These are known as *embedded requests* and they naturally follow shortly after the base web page is received. The number of embedded requests per base page has been steadily increasing and could range from a few to a few tens. Embedded requests are not important from a transactional analysis point of view, since they can be considered as an integral part of the base transaction. However, we still need to be able to generate embedded requests when generating e-commerce/web traffic artificially. In this section, we look at how to model embedded requests in the e-commerce environment and thereby use it in synthetic traffic generation.

Figures 9 and 10 show the probability mass function of embedded image requests in the B2C and the B2B environments respectively. Apart from image embeddings, the other embedding that we came across was HTML embedding. But, we do not present the results of HTML embedding as they were few and far apart.

Table 5 shows the statistics for the embedded images in the B2C and B2B scenarios. Firstly,

we observe that the number of embedded images per base request is fairly variable as the standard deviation is around twice the mean value. Secondly, we see that the B2C environment has more embedded images on an average as compared to the B2B environment, which intuitively makes sense, as B2B environments are designed for mainly serious business transactions, with little fancy stuff and much fewer advertisements.

We note that the B2C site whose data has been used throughout this paper used separate servers to serve embedded requests. Unfortunately, logs from these auxiliary servers were not available. Therefore, the B2C distribution embedded object distribution was actually obtained from yet another B2C site.

6 Synthetic Traffic Generation

So far, we have accomplished a systematic two-layer approach to modeling e-commerce traffic. As mentioned earlier, one of the main goals of this characterization is to enable synthetic traffic generation for server performance analysis.

Most web traffic generators attempt to emulate the behavior of an individual user in terms of time between successive clicks, selection of links on a web page, etc. Such an approach necessarily implies dedicating an O/S process or thread per user. For testing large internet servers in a lab environment, this approach become unscalable since it could require tens of thousands of processes/threads. Also, while emulating individual users allows a detailed control of user behavior, the global properties of the traffic hitting the server become much more difficult to control. For example, if we wish to ensure certain kind of scaling properties for the overall traffic (e.g., asymptotic self-similarity and/or multifractal behavior at intermediate time-scales [6]), adjusting the parameters of individual users for this becomes very difficult. Because of this, we have chosen the opposite approach of trying to generate the aggregate traffic directly. In this approach, a single process/thread can handle multiple interactions (i.e., requests or responses) and thus the total number of processes/threads needed is much smaller. Also, the global properties of the traffic arriving at the server (e.g., the H parameter of the self-similar traffic) can be controlled directly. The general approach in this case is to first generate a trace of successive arrival times and then distribute this trace among N client machines, so that client i is responsible for generating requests $i, i+N, i+2N, \ldots$ at their given times.

Based on this central principle, we propose a three step approach to synthetic traffic generation for the E-commerce environment. At the first step, we use $M/G/\infty$ model [9, 6] for generating long range dependent arrival time series, which represents the aggregated arrivals. In the second step, we add non-stationarity to it by means of the nonstationarity model. In the third step, we incorporate the transactional analysis into the generated non-stationary, long-range dependent request arrival process by splitting the aggregated stream into individual streams, marking the base requests and generating the embedded requests for each base request. We have developed a traffic generator, for this purpose, as reported in [3].

6.1 The $M/G/\infty$ Model

We use the $M/G/\infty$ traffic model for generating long-range dependent request arrivals [9, 6]. This model corresponds to Poisson arrivals of "user sessions" with a heavy-tailed distribution of session length and a constant rate of packet generation (representing the workload) within a session. More specifically, a $M/G/\infty$ model is defined using a $M/G/\infty$ queue operating in discrete time. Thus, all arrivals accumulate until the beginning of next "slot" and then allowed to come to the $M/G/\infty$ queue. Similarly, the service time is also expressed in the units of "slots", whose duration is henceforth represented by the symbol Δ. The number of customers in the $M/G/\infty$ queue at the end of each "slot" is interpreted as the number of arrivals to the actual system. That is, in the $M/G/\infty$ model, the arrival process is characterized by the number of arrivals over successive intervals of length one "slot". The arrivals within a slot could subsequently be assigned individual arrival instants — we do so by using a uniform distribution of arrival instants within a slot. This results in an exponential inter-arrival time within a slot.

The $M/G/\infty$ model was chosen because of a number of advantages including a simple interpretation of the model, easy control over the correlation structure of the arrivals (which allows generation of asymptotically self-similar traffic), and relative ease of introducing other properties (such as multifractal behavior) at smaller time scales. As such, the $M/G/\infty$ model is driven by a stationary Poisson process and thus generates stationary traffic beyond the initial warm-up period.

6.2 Addition of Non-stationarity Properties

Non-stationarity can be introduced in this model in many ways. A simple approach is to drive the $M/G/\infty$ system with non-stationary Poisson process. The main difficulty with this approach

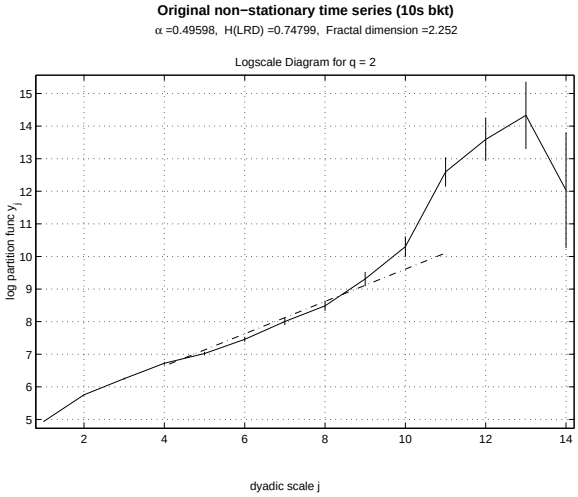

Figure 11: Scaling properties of the original full-day traffic

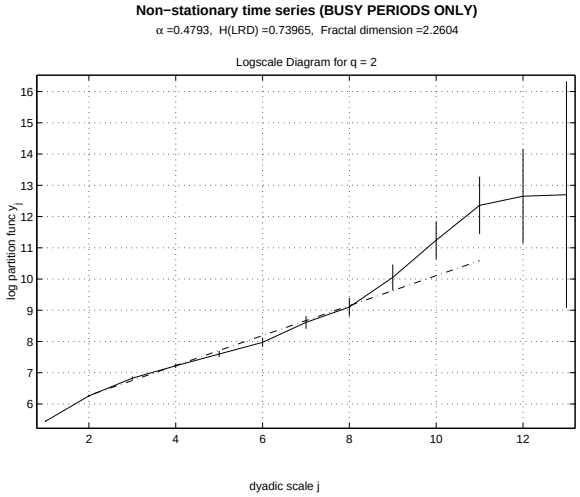

Figure 12: Scaling properties of the original busy-period traffic

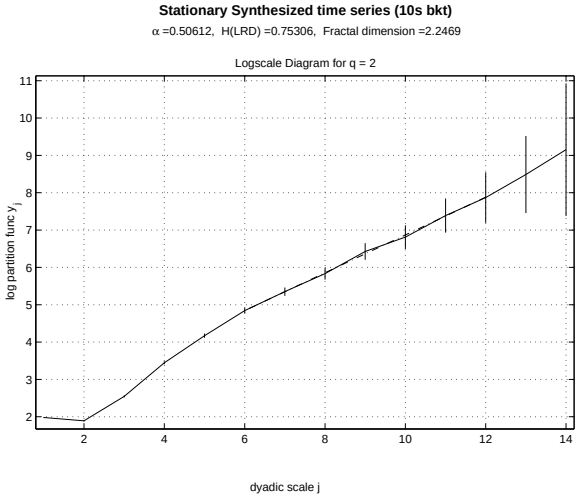

Figure 13: Scaling properties of synthesized stationary traffic

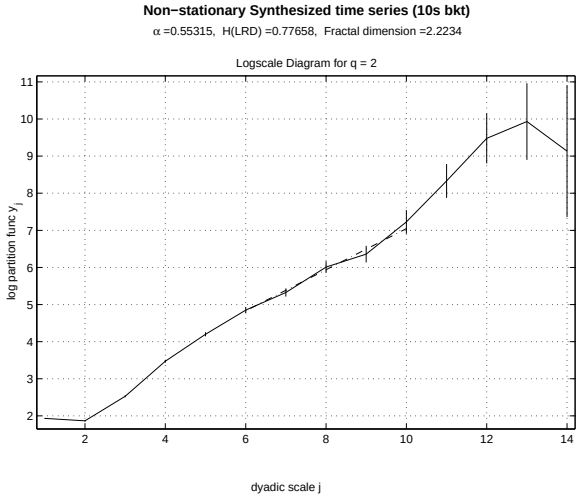

Figure 14: Scaling properties of synthesized non-stationary traffic

is the lack of simple relationship between the non-stationarity profile of the input Poisson process and that of the resulting $M/G/\infty$ process. Without such a relationship, it is very difficult to achieve the desired nonstationarity properties in the final $M/G/\infty$ process. A related issue is that currently there is no theoretical analysis of correlational properties of a $M/G/\infty$ system fed by a non-stationary session arrival process. As a result, we shall take the more direct approach of adding non-stationarity at the output end by modulating the time series with the non-stationarity profile.

In particular, we generate a random number Z using the normalized non-stationarity distribution (such as the one in Figure 7 or 8) every NST seconds. By definition, Z is positive and has a mean of 1.0. We assume that NST is an integer multiple of the slot size Δ with NST$= K \times \Delta$. Thus a new value for Z is generated every K slots, and the number of arrivals during a slot is multiplied by this value. The arrival time is then assigned to the resulting number of arrivals within a slot using the uniform distribution. This approach does not affect the correlational properties of the traffic up to the time-scale of NST and introduces non-stationarity precisely. Although it is possible to use the empirical distribution such as in Figures 7 and 8, it is desirable to fit a simple analytic distribution to the empirical distribution and use that instead.

Figures 11 and 12 show the scaling properties of the original nonstationary B2B traffic for the full day and busy period, respectively. The diagrams shown are the "AV" estimators referred to earlier, and they show a log-log plot of the time-scale (x-axis) and the energy metric of the wavelet coefficients at that time-scale. If the plot shows a linear behavior over a set of time-scales consistent with long-range dependence, the corresponding Hurst parameter is estimated by a modified linear regression procedure detailed in [1]. For the synthesis, we concentrate only on the busy-period, where the H parameter value is 0.748. Figure 13 shows the scaling properties of the corresponding synthesized traffic, targeted for $H = 0.748$. The synthesis was done using a $M/G/\infty$ traffic model. We see that the resultant stationary synthetic traffic has $H = 0.753$. It is to be noted that since the $M/G/\infty$ traffic model is asymptotically self-similar in nature, the correct scaling behavior of the traffic should be estimated from the large time-scale region, as we have done. Figure 14 shows the scaling properties of the synthesized non-stationary traffic. Non-stationarity was imposed over the generated stationary time-series by means of level shifts as discussed. The non-stationarity marginal distribution used corresponded to the busy hour distribution of Z, the normalized level shift random variable. We note that although the addition of nonstationarity actually deteriorates the H parameter estimate (from 0.753 to 0.776), the overall scaling behavior of the nonstationary synthesized traffic is much closer to the actual scaling behavior shown in Figure 12.

6.3 Incorporating user behavior

In this step, we split the aggregate arrival stream obtained in the last section into multiple streams, each corresponding to a single concurrent "user" or session. This step is essential since the Markov chain model discussed in section 4 applies only to individual users, rather than to the entire stream.

Once the splitting is done, we need to mark the transaction type using the Markov chain parameters and generate the embedded requests corresponding to the base request.

The simplest approach for splitting is to exploit the fact that the aggregate traffic is typically a superposition of K *independent* user streams. Thus, to reverse the superposition, it suffices to assign successive arrivals to one of the K constituent streams probabilistically. That is, successive arrivals are then marked equiprobably to belong to one of the K user streams. This method obviously preserves the correlation structure of the original stream. The parameter K, which represents the number of "virtual users" can be determined as the product of the desired aggregate mean arrival rate and the inter-request time for a typical individual user.

Having obtained individual user streams, we next need to mark each arrival with the transaction type and other relevant information. For the former, we need to remember the current state in the Markov chain on a per-user basis and simply simulate the state transitions. As noted in section 4, request sizes don't depend much on the state or the method type (get vs. post); therefore, request sizes can be generated at the level of the aggregate traffic. The size of the actual object requested is determined indirectly by considering the distribution of the stored (or dynamically constructed) web-pages and the skewness of the access pattern. For example, it is generally found that the popularity distribution of web pages is Zipf with α close to 1. More details on these aspects are beyond the scope of this paper and may be found in [3].

Finally, with each base request generated this way, we need to generate embedded requests corresponding to the base request. This is done by using the distribution function for the embedded requests.

7 Conclusions and Future Work

In this paper, we presented a two layer characterization of e-commerce traffic both in the B2B and B2C environments. At the lowest layer we examined the request arrival process to the e-commerce front end. Non-stationarity over short time scales was found to be an important difference between the e-commerce and the web environment. We presented a novel technique to analyze non-stationarity and long-range dependence properties of the request arrival time series in such a way as to enable synthetic generation of e-commerce traffic. The analysis yielded a non-stationarity time scale of 15 mins in both B2B and B2C environments.

At the next layer, we looked at characterizing the user sessions in the e-commerce environment. For this purpose, we devised a generic embedded Markov chain model for capturing the individual user behavior. We also addressed the issue of modeling embedded requests. Along the way, we pointed out some of the significant differences between the B2B and B2C environments and showed that despite the differences they can still be characterized together in a coherent manner due to inherent commonalities.

With the systematic two-layer characterization, we finally looked at issues in generating synthetic realistic e-commerce traffic in the lab for server engineering.

Our work so far has not carefully examined the backend transactional characteristics and their relationship to front-end transactions, primarily due to lack of available data. A detailed study in this regard is essential for a good classification methodology, and plan to pursue this further. We also do not have adequate data to characterize user abandonments and retries and its impact on the aggregate traffic.

References

[1] P. Abry, P. Flandrin, M.S. Taqqu, and D. Veitch, "Wavelets for the analysis, estimation, and synthesis of scaling data", report available from `www.serc.mit.edu.au/ darryl`.

[2] M. E. Crovella, A. Bestavros, " Self-similarity in world wide web traffic evidence and possible causes", Proceedings of the ACM SIGMETRICS 96, pages 160-169, Philadelphia, PA, May 1996.

[3] K. Kant, R. Iyer and V. Tewari, "Geist: An e-commerce front-end traffic generator", Internal Report, Sept 2000.

[4] K. Kant, "Introduction to Computer System Performance Evaluation", McGraw Hill, 1992.

[5] K. Kant and Y. Won, "Server Capacity Planning for Web Traffic Workload", IEEE transactions on knowledge and data engineering, Oct 1999. pp731-747.

[6] K. Kant, "On Aggregate Traffic Generation with Multifractal Properties", proceedings of GLOBECOM'99, Rio de Janeiro, Brazil, pp 1179-1183.

[7] K. Kant and M. Venkatachalam, "Modeling traffic non-stationarity in e-commerce servers", submitted to WW10.

[8] W.E. Leland, M.S. Taqqu, W. Willinger and D.V. Wilson, "On the self_similar nature of ethernet traffic", IEEE/ACM trans on networking, Vol 2, No 1, pp 1–15, Fen 1994.

[9] M. Parulekar and A. Makovski, "M/G/∞ input processes: A versatile class of models for network traffic", Proc of IEEE Infocom 97, April 1997.

[10] M. Roughan and D. Veitch, "A study of the daily variation in the self-similarity of real data traffic", SERC technical report, 1998.

[11] M. Roughan and D. Veitch, "Measuring long-range dependence under changing traffic conditions", proceedings of INFOCOM'99, pp1513-1521.

[12] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson, "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN traffic at the source layer", Proc of SIGCOMM 95, pp100-113.