Web User Session Characterization via Clustering Techniques

A. Bianco, G. Mardente, M. Mellia, M. Munafò, L. Muscariello
Dipartimento di Elettronica, Politecnico di Torino
{last name}@mail.tlc.polito.it

***Abstract*— We focus on the identification and definition of "Web user-sessions", an aggregation of several TCP connections generated by the same source host on the basis of TCP connection opening time. The identification of a user session is non trivial; traditional approaches rely on threshold based mechanisms, which are very sensitive to the value assumed for the threshold and may be difficult to correctly set. By applying clustering techniques, we define a novel methodology to identify Web user-sessions without requiring an a priori definition of threshold values. We analyze the characteristics of user sessions extracted from real traces, studying the statistical properties of the identified sessions. From the study it emerges that Web user-sessions tend to be Poisson, but correlation may arise during periods of network/hosts anomalous functioning.**

Keywords: Network Measurements

I. Introduction

The identification of user-session plays an important role both in Internet traffic characterization and in the proper dimensioning of network resources. We concentrate on the identification of web sessions generated by a single user, as WWW is the most widely used interactive service. We assume that only a single user runs a browser on each host, a reasonable assumption today given the vast majority of PC based hosts. The informal definition of a user session can be obtained by describing a typical behavior of a user running a Web browser: an activity period, when the user browses the Web, alternates with a silent period over which the user is not active on the Internet. This activity period, named session in this paper, may comprise several TCP connections, opened by the same host toward possibly different servers.

Unfortunately, the identification of active and silent periods is not trivial. Traditional approaches [9], [8] rely on the adoption of a threshold η: TCP connections are aggregated in the same session if the inter-arrival time between two TCP connections is smaller than the threshold value; otherwise, a new session is identified. This approach works well if the threshold value is correctly matched to the average value of connection and session inter-arrival time; however, to know these values in advance is unrealistic in practice. If the threshold value is not correctly matched to user session statistical behavior, threshold based mechanisms are highly error prone in session identification.

Clustering techniques [4] are used in many areas to partition a given data set in "similar subsets", by defining a proper notion of similarity. Typically, several metrics over which a distance measure can be defined are associated to points (named samples in this paper) in the data set; informally, the partitioning process tries to put in the same subset neighboring samples and in different subsets distant samples. The main advantage of using such approach is that there is no need to define a priori any threshold value. Thus, this methodology should be less error prone than simpler threshold based mechanisms.

The contributions of this paper are the following: first, we adapted classical clustering techniques to the described scenario, a non-trivial task that requires a lot of ingenuity to optimize the performance of user session identification algorithms both in terms of speed and precision. In [1], the proposed methodology was tested using artificially generated traces to assess the error performance of the proposed technique and to compare it with traditional threshold based mechanisms, proving the strength of clustering approaches versus threshold based algorithms. In this paper, the defined algorithms are run over real traffic traces, to obtain statistical information on user sessions, such as distributions of i) session duration, ii) amount of data carried over a single session, iii) number of connection within a single session. A study of the inter-arrival times of Web user-sessions is also presented, from which it emerges that Web user-sessions tend to be Poisson, but correlation may arise during network/hosts anomalous functioning.

II. Clustering Techniques

In this section we briefly describe two clustering techniques, which will be used in the next sections as key tools. Clustering is a general framework which can be applied to many different areas to infer some sort of similarity among subsets of data, typically on a large size repository. More details on clustering techniques can be found in [4].

Let us consider a metric space X, which we refer to as a sampling space, and a set of samples $A = \{x_1, \ldots, x_N |\ x_i \in X\}$ which have to be clustered into K subsets: we wish to find a partition $\mathcal{C} = \{C_1, \ldots, C_K\}$, such that $\cup_i C_i = A$ and $C_i \cap C_{j \neq i} = \emptyset$, with K possibly being unknown a priori. The subsets in the partition are named *clusters*. They contain "similar" samples, whereas samples associated to different clusters should be "dissimilar", the similarity being measured via the sample-to-sample and cluster-to-cluster distances. Depending on the dataset, ad hoc distance definition must be provided on the basis of a trial and error procedure. From now on, we assume that: $X = \mathbb{R}^n$, x_i^k represents the

k-th component of sample x_i, the sample distance $d(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_i^k - x_j^k)^2}$ is the classical Euclidean metric; the distance between two clusters C_i,C_j is defined as

$$d(C_i, C_j) = \min_{x \in \mathcal{R}(C_i), y \in \mathcal{R}(C_j)} d(x, y) \quad (1)$$

where $\mathcal{R}(C) \subseteq C$ is a set of selected points representing the whole cluster C.

A. The hierarchical agglomerative approach

At the beginning of the procedure, each sample is associated to a different cluster, i.e., $C_i = \{x_i\}$, thus the number of cluster N_c is equal to $N = |A|$. Then, based on a definition of a cluster-to-cluster distance, the nearest clusters are merged to form a new cluster. Iterating this step, the procedure ends when all samples belong to the same cluster $C = A = \{x_1, \ldots, x_N\}$, and $N_c = 1$. This procedure defines a merging sequence based on minimum distance between clusters. At each step $i = 1, \ldots, N$, the *quality indicator* function $\gamma^{(i)}$ is evaluated. The set A is finally clustered by selecting the number of clusters $N_c = N - (i-1)$ such that $\gamma^{(i)} - \gamma^{(i-1)}$ is maximized. Intuitively, the quality indicator function $\gamma^{(i)}$ measures the distance between the two closest cluster at step i. A sharp increase in the value of $\gamma^{(i)}$ is an indication that the merging procedure is merging two clusters which are too far apart, thus suggesting to adopt the previous partition as the best cluster configuration. Refer to [1] to see a typical behavior of γ.

This approach can be rather time consuming, especially when the data set is very large, given the need of starting with an initial number of clusters equal to $N_c = |A|$.

B. The partitional approach

This technique is used when the number K of final clusters is known. The procedure starts with an initial configuration comprising K clusters, selected according to some criteria; the cluster configuration's procedure is iterated. Cluster C_i is represented by a subset of samples $\mathcal{R}(C_i)$ in Eq. (1). When the cluster representative is the so called *centroid* $\hat{c}_i$, defined as the mean value of the cluster samples, i.e.,

$$\hat{c}_i^k = \frac{1}{|C_i|} \sum_{x \in C_i} x^k \qquad k = 1, \ldots, n$$

the algorithm is named K-means algorithm in the literature.

At the beginning, K clusters are created, with cluster centroids positioned according to a given rule in the measure space, e.g., randomly positioned or partitioning the measured space in $K+1$ equi-spaced areas. Each sample is associated to the closest cluster, according to the distance between the samples and the centroid of each cluster. When all samples are assigned to a cluster, new centroids are computed and the procedure is iterated. This algorithm ends when either a prefixed number of iterations is reached, or the number of samples which are moved to a different cluster is negligible according to a predefined threshold.

III. Using Clustering Techniques on Measured Data Set

We start by giving some details about the dataset of traces that will be analyzed, to define the variables that will be used by the clustering algorithm.

A. Traffic trace description

Traffic traces were collected on the Internet access link of Politecnico di Torino, i.e., between the border router of Politecnico and the access router of GARR/B-TEN[10], the Italian and European Research network. Within the Politecnico campus LAN, there are approximately 7,000 hosts; most of them are clients, but several servers are also regularly accessed from the outside. The backbone of the campus LAN is based on a switched Fast Ethernet infrastructure. It behaves as a stub Internet subnetwork, which is connected to the public Internet via a single 28Mbps link. A strict regulation of the network facilities is imposed by means of a firewall architecture which blocks (most of) the peer-to-peer traffic. Thus, still today the majority of our Internet traffic is built by Web browsing. Details on the measurements setup and traffic characteristics can be obtained from [6], [7].

Since 2001, several traces have been regularly collected. Among the available data, we selected the time period from from 04/29/2004 to 05/06//2004 (named APR.04), which comprises more than a week of continuously traced data. We performed the analysis by considering only the *working* period, i.e., traffic from 8AM to 8PM, Monday to Friday. The APR.04 dataset includes two working days (the 5th and 6th of May) during which terminals in our campus were attacked and infected by the so called "Sasser.B" worm [11]. The worm infection does not affect our measurement campaign, because the spreading of the worm itself is not based on the HTTP protocol. Nonetheless, the drawbacks of the network and host malfunctions and subsequent requirements to download worm removal tools and Operating System patches have quite a large impact on the properties of the user session, as we will show later in this paper.

Bidirectional packet level traces have been collected using *tcpdump* [5], and then later processed by *Tstat* [7] to obtain a TCP level trace. Tstat is an open source tool developed at the Politecnico di Torino for the collection and statistical analysis of TCP/IP traffic. In addition to standard and recognized performance figures, Tstat infers TCP connection status from traces. For the purpose of this paper, we used Tstat to track:

- $f_{id} = (C_{IP}, S_{IP}, C_{TCP}, S_{TCP})$: the 4-tuple identifying the flow, i.e., IP addresses and TCP port numbers of client and server (when the IP protocol field is set to TCP).
- t: the flow start time, identified by the time stamp of the first client SYN message.
- t_d: the time instant in which the last segment carrying data is observed (either from the client or form the server).
- t_e: the flow end time, identified by the time instant in which the tear-down procedure is terminated.

- B_c and B_s: the byte-wise amount of data sent from the client and server respectively (excluding retransmissions).

B. Clustering definition

The first fundamental choice regards the n statistical variables used to define the metric space $X = \mathbb{R}^n$ to be used in the clustering analysis; this implies also to select the metric space that best fits our problem. The typical and easiest approach is to let the clustering algorithm to run over a very large number of statistical variables, typically including the vast majority of available data (in our example, potential statistical variables may be IP source address, TCP destination port, TCP flows starting and ending time, etc).

However, after several trials, an accurate pre-filtering of data available in traces both improves algorithmic speed and provides more accurate results. Recalling that we wish to identify a Web user-session, i.e., a group of TCP connections corresponding to the activity period of a user running a Web browser, we used the opening time t of a TCP connection as the only statistical variable for the clustering process; thus $n = 1$, and $X = \mathbb{R}$.

Before running the clustering algorithm on this variable, traces are preprocessed. We assume that a "user" is identified by its client IP address C_{IP}, and only connections having TCP server port S_{TCP} equal to 80 (HTTP protocol) are considered to be Web connections. To consider only significant IP (users), we selected the most active hosts, i.e., the top 1500 campus LAN IP addresses with respect to the number of generated TCP connections. Each user trace is preprocessed according to the following steps: i) data are partitioned day by day, ii) only working hours of working days are considered to obtain a set of statistically homogeneous samples and iii) opening times of two subsequent flows separated by more than half an hour are a priori considered as two independent data sets. Thus, for a given host IP, the set of samples

$$A(\mathrm{IP}) = \{t \mid C_{IP} = \mathrm{IP}, S_{TCP} = 80, t_i - t_{i+1} < 1800s\}$$

represents the opening times of TCP connections within a given time-frame. The intuition behind this is to allow the clustering algorithm to concentrate only on TCP connections created by a single user.

After the metric space definition, we need to select a cluster analysis method. To take the advantages and to avoid the drawbacks of hierarchical and partitional clustering methods, we use a mix of them. Thus, for each $A(\mathrm{IP})$, the following 3-step algorithm is run to identify sessions relative to a given user:

1) an initial smart clustering is obtained by selecting a number of clusters large enough but smaller than the number of samples of A
2) a hierarchical agglomerative algorithm is used to aggregate the clusters and to obtain a good estimation of the final number of them N_c
3) a partitional algorithm is used to obtain a fine definition of the N_c clusters.

1) Initial clustering selection: We use a partitional method with K clusters, where K is large enough, but significantly smaller than the total number of samples (a study of the impact of K is presented in [1]). To efficiently position the K centroids at the first step, in our uni-dimensional metric space, we evaluate the distance between any of two adjacent samples t_i, t_{i+1}. According to the distance metric $d(t_i, t_{i+1}) = |t_i - t_{i+1}|$, we take the farthest $(K-1)$ couples and determine K intervals. Let $t_{i,inf}, t_{i,sup}$ be the inferior and superior bounds of interval i, the centroid position of each cluster is set to $\hat{c}_i = (t_{i,sup} + t_{i,inf})/2$, and the partitional method is run for up to 1000 iterations: therefore, we define K initial clusters.

At this step, we represent each cluster C with a small subset $\mathcal{R}(C)$ of samples; $|\mathcal{R}(C)| \leq 2$ is enough in our case, since the metric space is $\mathbb{R}$. Possible choices for $\mathcal{R}(C)$ are: (i) the cluster centroid, which gives the name "centroid method" (or K-means) to the procedure; (ii) the g-th and $(100 - g)$-th percentiles, which yields the so-called (iii) "single linkage" procedure when $g = 0$.

2) The hierarchical agglomerative procedure: A hierarchical method is iteratively run using only the representative samples $\{\mathcal{R}(C)\}$ to evaluate the distance between two clusters, and starting with K initial clusters, therefore enormously reducing the number of steps of the hierarchical method. At each iteration, the hierarchical procedure merges the two closest clusters; distances among clusters are recomputed. After $K - 1$ iterations, the process ends.

To decide which is the optimal number of clusters among those determined in the iterative process, we define the clustering quality indicator function $\gamma^{(i)}$. Denote the j-th cluster at step i as $C_j^{(i)}$; at each step, we evaluate the function $\gamma^{(i)}$:

$$\gamma^{(i)} = \frac{d_{min}^{(i)} - \bar{d}_{min}^{(i)}}{\bar{d}_{min}^{(i)}}$$

where

$$d_{min}^{(i)} = \min_{j,k \neq j} d\left(C_j^{(i)}, C_k^{(i)}\right), \quad \bar{d}_{min}^{(i)} = \frac{\sum_{l=1}^{i-1} d_{min}^{(l)}}{i-1}$$

and $d\left(C_j^{(i)}, C_k^{(i)}\right)$ is defined according to Eq.(1)

A sharp increase in the value of $\gamma^{(i)}$ is an indication that the merging procedure is artificially merging two clusters which are too far apart. The optimal number of clusters N_c is the one that correspond to the sharpest increase in $\gamma^{(i)}$ (the first one is chosen in case of multiple occurrences):

$$N_c = N - \left[\operatorname*{argmax}_{i \geq 1} \left(\gamma^{(i)} - \gamma^{(i-1)}\right) - 1 \right]$$

3) Final clustering creation: Finally, a partitional clustering procedure is run over the original dataset which includes all samples, using the optimal number of clusters N_c determined so far, therefore using the same procedures adopted in the first step (either the centroid, g percentile or single linkage methods). Then a fixed number of iterations of the partitional approach are run, to permit a final refinement on the clustering

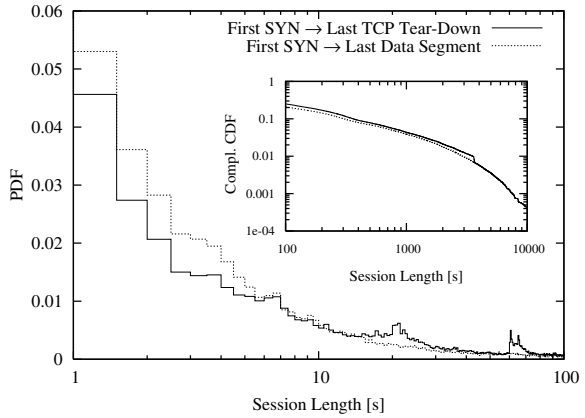

Fig. 1. PDF (and Complementary CDF in the inset) of the session length.

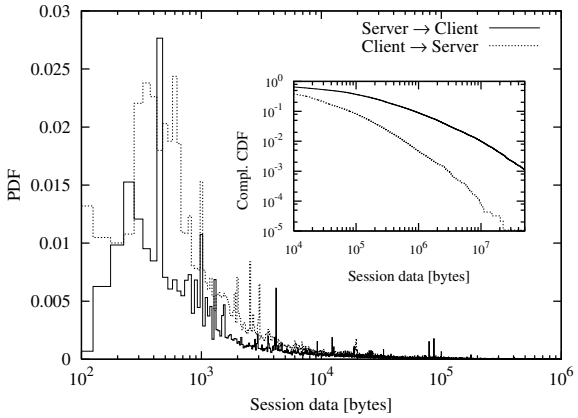

Fig. 2. PDF (and Complementary CDF in the inset) of the client-to-server and server-to-client data sent in each session.

definition. This third phase is not strictly required, given that at the end of the hierarchical procedure a partition is already given. However, it produces clusters of real points instead of centroids (which may not coincide with any data point). In addition, the computational cost of this phase is almost negligible compared to the previous one.

The complete clustering algorithm has been implemented by a number of Perl scripts, and by using the "R-Project for Statistical Computing" tool [2], which efficiently implements many clustering algorithms.

IV. Performance Analysis of Trace Data Set

In this section we summarize the results obtained by running the clustering algorithm for each user separately, according to the trace pre-processing algorithm previously described.

A. Web user-session characterization

In Figs. 1, 2, and 3 the major characteristics of the Web user-session identified during APR.04 are shown. We plot Probability Density Functions (PDFs) using a linear/log scale: since the support is quite large, the PDF is represented only for a limited range of values, and the complementary Cumulative Distribution Function (CDF) is shown using a log/log scale in an inset to highlight the characteristics of the distribution's tail.

Fig.1 shows the probability density function for the duration of the identified sessions. The two different distributions shown in Fig. 1 represent the effect of different definitions for the Web user-session. Considering TCP connections belonging to the same session, we define the session duration as the time between the first SYN segment of the first connection and: (i) for "protocol session", the last segment observed during the last connection tear-down (solid line); (ii) for "user" session, the last segment carrying payload of the last connection (dotted line). Therefore, using the notation introduced in Sec. III, for a given session/cluster C, we can define

$$\Delta T_e = \max_{f_{id} \in C}(t_e(f_{id})) - \min_{f_{id} \in C}(t(f_{id}))$$
$$\Delta T_d = \max_{f_{id} \in C}(t_d(f_{id})) - \min_{f_{id} \in C}(t(f_{id}))$$

in which ΔT_e and ΔT_d are the protocol and user session duration respectively. The first definition is relevant for example when either web server or client resources are considered, since TCP connections must be managed until the tear-down procedure has ended. The second definition on the contrary is relevant to model the user behavior, since users are satisfied when all data have been sent/received.

Clearly, the distribution of the protocol session duration shows longer duration, but also biased peaks at $20s$, $60s$ and $3600s$, corresponding to application layer timers imposed by Web browsers or HTTP servers which trigger connection tear-down procedure after idle periods. For example, due to HTTP protocol settings, servers may wait for a timer to expire (usually set to 20 seconds) before closing the connection; similarly, HTTP 1.1 and Persistent-HTTP 1.0 protocols use an additional timer, usually set to a multiple of 60 seconds. Therefore, for protocol session duration, the bias induced by those timers is evident. The same bias disappears when the session duration is evaluated considering user session duration.

Notice that the session duration distributions have a quite large support, showing a great variability in users behavior. Indeed, there is a percentage of very short sessions (less than few seconds), but also users whose activities last for several hours. Indeed, the tail of the complementary CDF shown in the inset underlines the heavy-tailed distribution of session duration, which can be a possible cause of Long Range Dependence (LRD) at both the connection and packet layers.

In Fig. 2 we have the PDF for the volume of data exchanged during each session in the client-server (dashed lines) and the server-client (solid lines) directions : for a given session/cluster C, $D_c = \sum_{f_{id} \in C}(B_c(f_{id}))$ and $D_s = \sum_{f_{id} \in C}(B_s(f_{id}))$ define the session data volumes. As expected, much less data are transfered from clients to servers and the distribution tail is shorter; the number of sessions transferring more than 10 Mbytes in the server-client direction is not negligible. A peculiar number of peaks are present in the initial part of both PDFs. Investigating further, we discovered that those peaks are due to the identification of sessions which are not generated by

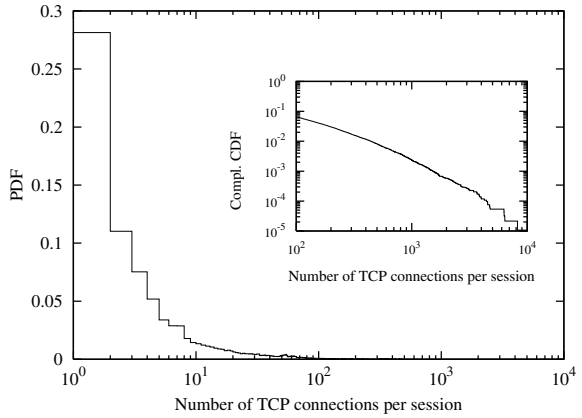

Fig. 3. PDF (and Complementary CDF in the inset) of the number of TCP connection in each session.

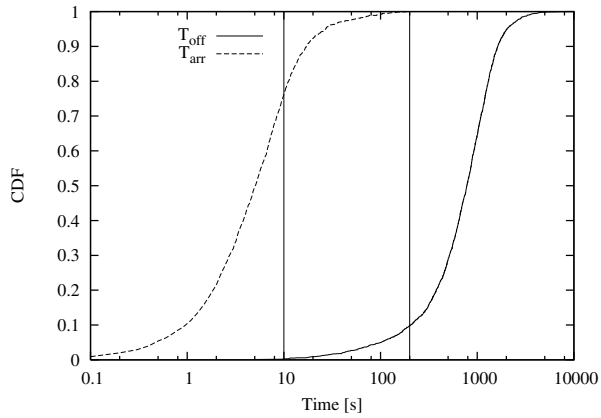

Fig. 4. CDF of the mean user session inter-arrival and mean users flow inter-arrival.

users, but instead by automatic reload procedure imposed by the Web page being browsed. For example, news or trading on line services impose periodic updates of pages which causes the client to automatically reload the pages. If the automatic reload is triggered periodically, the clustering algorithm tends to identify for each connection a separate session, thus causing a bias in the session data distribution.

This is clearly evident also from Fig. 3, which reports the number of TCP connections per session $|C|$. Indeed, more than 25% of sessions count for only one connection. Moreover, most of the identified sessions is built by very few connections (about 50% by 4 connections or less), indicating also that i) the client is usually able to obtain all the required data over few TCP connections, ii) the number of external objects required is limited, and iii) the time spent by the users over one web page is large enough to define each web transaction as a session.

The complementary distribution of the number of TCP connections per session reported in the inset of Fig. 3 shows a linear trend, highlighting that the distribution has an heavy tail. This could be one of the possible causes of LRD properties at the flow level, as already known.

Fig. 4 reports the CDF of the *average* user flow inter-arrival T_{arr} and average user session OFF period T_{off}, i.e., for each user, we evaluated the average T_{arr} and T_{off}, and then derived the corresponding distribution over users. As expected, T_{arr} assumes smaller values than T_{off}, but the two distributions overlap as underlined by the two vertical lines. This shows the variability in user behavior.

Notice that the overlapping of the two distributions would have not appeared if a threshold methodology had been applied, whichever adopted threshold. Moreover the variability of the mean T_{off} and mean T_{arr} makes it very difficult to select an appropriate values for η. This confirms the limits of threshold based approaches and the need of using clustering approaches that automatically adapt to different scenarios.

B. Statistical properties of session arrival process

Finally, statistical properties of the aggregate session inter-arrival times are investigated. We obtained a trace of session arrivals by multiplexing all sessions identified for each IP source address (user) during the same time period, i.e, by considering the Web user-session arrival process to the access router of our institution.

Fig. 5 reports the Q-Q plot of the aggregate session inter-arrival distribution with respect to the best fitted Weibull distribution over the same data set. This distribution has been recognized as a good model for TCP inter-arrival times [3]. The parameters a, b of the Weibull distribution represent the so called "shape" and "scale" parameters. When the shape parameter is set to 1, the Weibull distribution degenerates into an exponential distribution. When it is smaller than 1, the tail of the distribution tends to be heavier, while for values of a larger than 1 the shape of the distribution tends to assume a dumbbell form. The classical maximum likelihood method was used to obtain the best a and b parameters for the fitting procedure.

The upper plot of Fig. 5 refers to a typical measurement day and shows a good matching of the data samples with the indicated Weibull distribution. Being $a = 0.93$, it also shows that the distribution is also very close to an exponential distribution, therefore hinting that the arrival process of Web user-session tends to be Poisson as pointed out by previous studies [8]. The Q-Q plot shows also that the tail of the distribution is in general less heavy than the tail of the fitted Weibull distribution. There is therefore a bias toward small values of session inter-arrivals, with large inter-arrivals which are rare. The Q-Q plot of the same data with respect to the best-fitted exponential distribution showed almost the same behavior.

On the contrary, the lower plot in Fig. 5, which refers to the samples collected on May 5th, 2004, shows a distribution that cannot be fitted by a Weibull distribution. The best fit is obtained by a shape parameters $a = 0.67$ indicating an heavy tailed distribution. Moreover, the best fit Weibull distribution deviates from the measured dataset on large quantiles, showing that the tail of the real distribution is heavier than the one of the best fit distribution. The anomalous behavior is due to

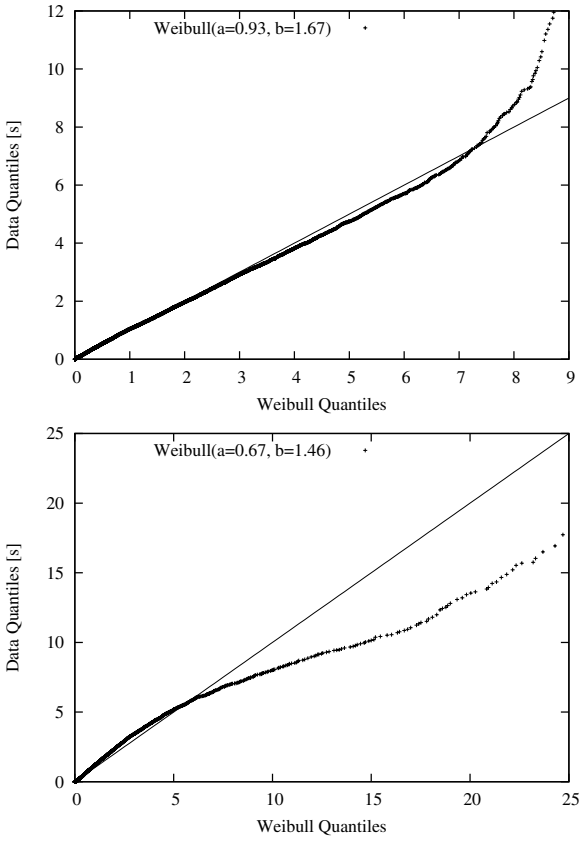

Fig. 5. Fit of user session inter-arrivals to a Weibull distribution: normal day in the top plot and during a worm attack on the bottom plot.

the spreading of the Sasser.B worm [11] in our institution, as previously described, and to the subsequent download of the needed OS patches and Anti-virus updates, which introduced correlation in the arrival process of sessions, being driven by the (large) download time of files.

Fig.6 finally reports the autocorrelation function evaluated on the session inter-arrival obtained during a typical day on top plot, while bottom plot refers to the autocorrelation estimated during the anomalous day during the worm attack. Top plot confirms that Poisson assumption holds for normal day, being the autocorrelation function almost negligible except that in the origin. Similarly, the autocorrelation function among session inter-arrivals can be quite relevant on days during which user activities is driven by external factors, as shown by the bottom plot which refers to the day of the worm infection.

V. Conclusions

We propose an application of clustering techniques to a large set of real Internet traffic traces to identify Web user-sessions. The proposed clustering method has been applied to measurement trace data sets to study the characteristics of Web user-sessions, showing that session arrival process tends to be Poisson distributed. Moreover, the analysis of the identified user-sessions shows a wide range of behavior that cannot be captured by any threshold based method. Finally, we think that

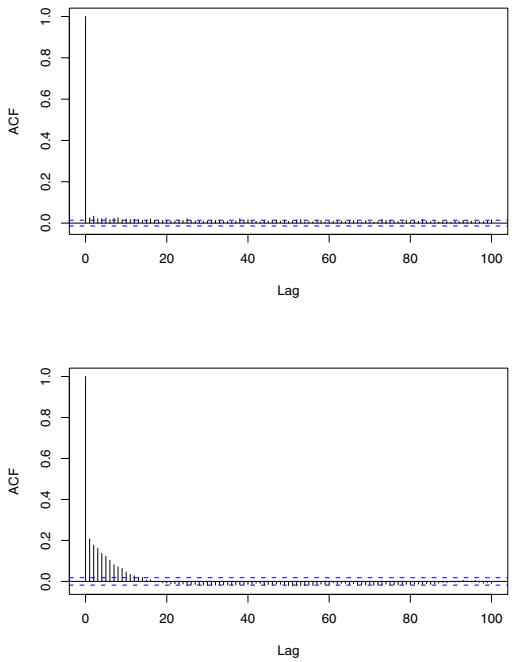

Fig. 6. Auto correlation function of the inter-arrival process of sessions: normal day in the top plot and during a worm attack on the bottom plot.

the clustering algorithms proposed in this paper can be helpful in studying other traffic properties. We intend to apply them to different types of traffic, and to extend the study on the statistical properties of the identified sessions arrival process.

Acknowledgment

We would like to thank people from Ce.S.I.T., the networking facilities of our Campus, who allowed us to access to the real data traces.

References

[1] A. Bianco, G. Mardente, M. Mellia, M. Munafò, L. Muscariello, "Exploiting Clustering Techniques for Web User-session Inference," *IPS-MOME 05*, Warsaw, Poland, March 2005.
[2] P.Dalgaard, Introductory Statistics with R, *Springer*, 2002.
[3] A. Feldmann, Characteristics of TCP connection arrivals. In Self-Similar Network Traffic and Performance Evaluation. Wiley, New York, 2000.
[4] D.J.Hand, H.Mannila, P.Smyth, *Principles of Data Mining*, MIT Press, 2001.
[5] S.McCanne, C.Leres, and V.Jacobson, "Tcpdump," http://www.tcpdump.org, 2001.
[6] M.Mellia, A.Carpani, R.Lo Cigno, "Measuring IP and TCP behavior on Edge Nodes," *IEEE Globecom 2002*, Taipei, TW, November 2002.
[7] M.Mellia, R.Lo Cigno, F.Neri, "Tstat web page," http://tstat.tlc.polito.it/, 2001.
[8] C.Nuzman, I.Saniee, W.Sweldens, A.Weiss, "A compound model for TCP connection arrivals, with applications to LAN and WAN," *Computer Networks, Special Issue on Long-Range Dependent Traffic*, vol.40, no.3, pp.319-337, October 2002.
[9] V.Paxson and S.Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol.3, no.3, pp.226–244, June 1995.
[10] "The GARR Network topology," *http://www.garr.it/mappagarr/garr-b-mappagarr.shtml*, 2005
[11] "What You Should Know About the Sasser Worm," http://www.microsoft.com/security/incident/sasser.mspx, May 04.