Exploiting Clustering Techniques for Web User-session Inference [*]

A. Bianco, G. Mardente, M. Mellia, M. Munafò, L. Muscariello

Dipartimento di Elettronica, Politecnico di Torino Corso Duca degli Abruzzi, 24 – I-10129 Torino, Italy

E-mail: {`last name`}`@mail.tlc.polito.it`

Abstract

We focus on the definition and identification 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, which 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 discuss pros and cons of this approach, and we define a methodology to be applied to real traffic traces. The proposed methodology is evaluated on artificially generated traces to show its benefits against traditional threshold based approaches. We then analyze the characteristics of user sessions extracted from real traces, studying the statistical properties of the identified sessions.

Keywords: Network Measurements

1 Introduction

Study of telecommunication networks was often based on traffic measurements, out of which traffic models and performance estimates are built. While the measuring and modeling processes proved to be reasonably simple in traditional, circuit-switched telephone networks, they seems to be much harder in packet-switched data networks. In particular, on a data network like the Internet, the layered structure of the TCP/IP protocol suite requires the analysis of traffic at least at the IP (packet), TCP/UDP (flow[1]), and Application/User–behavior (session) layers. While lot of attention has been traditionally dedicated to the characterization of the packet and flow levels (see for example [1, 2, 3, 4, 5, 6]), few are the studies on the properties at the session/user layer [1, 7]. This is due to the difficulties in the definition of "session" itself [8], which depends on the application used: applications such as TELNET or SSH tend to generate a single TCP connection per user session, whereas application layer protocols such as HTTP, IMAP/SMTP and X11 usually generate multiple connections per user session. Also, the generally accepted conjecture that such sessions would follow a Poisson arrival process (see [9] for example) might have reduced the interest in the session process analysis.

The paper goals are, first, to correctly identify, and, second, to determine statistical properties of Web user-sessions, i.e., set of TCP connections created by a given user while surfing the Web in a given time frame, by analyzing traces of measured data. We concentrate on the identification of client Web user-sessions generated by a single host, being WWW 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.

The identification of user-session plays an important role both in Internet traffic characterization and in the proper dimensioning of network resources. Unfortunately, the identification of active and silent periods is not trivial. Traditional approaches [1, 7] 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

[*]This work was supported by the FP6 European Network of Excellence "EuroNGI".

[1]In this paper we interchangeably use the term "TCP connection" and "TCP flow".

statistical behavior, threshold based mechanisms are highly error prone in session identification.

Clustering techniques [10] are used in many areas to find groups of similar variables/objects, to analyze a large data set, and in particular they allow to partition the 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. Then, we tested the proposed methodology using artificially generated traces to assess the error performance of the proposed technique and to compare it with traditional threshold based mechanisms. Finally, the defined algorithms were 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.

The remainder of the paper is organized as follows: in Sec. 2 we revise some basics of clustering, whereas in Sec. 3 we describe the methodology adopted to analyze the measured data set, including the details on the specific use of clustering techniques. To obtain a statistically significant model, we need to ensure that the proposed methodology is able to correctly identify a set of TCP connections belonging to the same user session. Thus, the proposed methodology is tested in Sec. 4 against artificial data sets based on a simple statistical model of user behavior. Finally, in Sec. 5 we analyze the statistical properties of Web user-sessions as identified by the clustering technique methodology. Sec. 6 summarizes the paper.

2 Clustering Techniques

In this section we briefly describe two main classes of 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 [10].

Let us consider a metric space X, which we refer to as a sampling space, and a set of samples $A = \{x_1, \dots, x_N |\ x_i \in X\}$ which have to be clustered into K subsets: we wish to find a partition $\mathcal{C} = \{C_1, \dots, C_K\}$, 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

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

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

Now we briefly introduce two techniques, known in the data mining and in the descriptive multivariate statistic environments as "hierarchical agglomerative" (named hierarchical in the remainder of the paper for simplicity) and "partitional" clustering. For the sake of completeness, another hierarchical clustering methodology exists, named "Hierarchical Divisive Clustering", but it is not considered in the paper.

2.1 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 $C = A = \{x_1, \dots, x_N\}$, and $N_c = 1$. This procedure defines a merging sequence based on minimum distance between clusters. At each step $i = 1, \dots, N$, 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. Fig. 1 sketches a hierarchical clustering evolution starting from 6 initial samples, which are merged step by step into clusters.

It is clear how time consuming this approach can be, especially when the data set is very large, given the need of starting with an initial number of clusters equal to $N_c = |A|$, i.e., the number of samples in the data set. For this

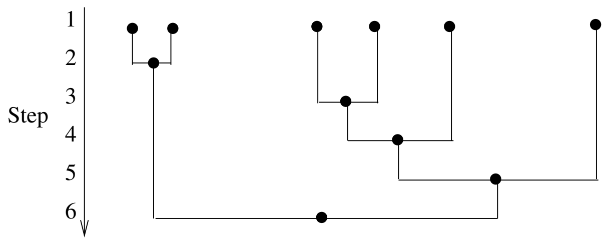


Figure 1. Hierarchical agglomerative clustering scheme

reason, non-hierarchical approaches, named partitional, are often used, since they show better scalability.

2.2 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 is modified through an iterative process until "steady state" is reached, e.g., no significant variation in the cluster is obtained by more iterations. 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 sample 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. Clearly, there is no guarantee that the final solution is always the same when varying the initial state.

3 Using Clustering Techniques on Measured Data Set

In this section we describe the methodology we developed for the identification of Web user-sessions, including the choices made in the cluster analysis. The description refers to the analysis of data collected through measurement procedures; similarly, the same process was used to analyze artificial traffic, when trying to determine the procedure ability to correctly identify user sessions. 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.

3.1 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[11], 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: there is a single point of access to the GARR/B-TEN network and, through it, to the public Internet. The link speed is 28Mbps. 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 [12, 13].

Since 2001, several traces have been regularly collected. Among the available data we selected two different time periods:

- OCT.02: from 23/10/2002 to 31/10/2002
- APR.04: from 29/4/2004 to 6/5/2004

Both periods comprise 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. In Sec.5 we mainly report the results referring to the latter period, being the differences among the two dataset almost negligible.

Bidirectional packet level traces have been collected using *tcpdump* [14], and then later processed by *Tstat* [13] 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. Among all the statistical analysis performed, Tstat produce a TCP level log file, which logs all the TCP connection observed. A TCP flow starts when the first SYN segment from the client is observed, and is terminated either when the tear-down sequence of segments is observed (either the FIN/ACK or

RST messages), or when no segment is observed for an amount of time larger then 15 minutes[2]. Only TCP connections whose three-way-handshake was successfully completed are tracked; thus misbehaving connections and activities (e.g., port-scanning) do not affect the dataset. 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).

3.2 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, we noticed that 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 $X = \mathbb{R}$. We tried to include in the space definition also the ending time of a TCP connection, but we found that this can lead to misleading results, due the presence of very long data transfers which may span over long period of time. Similarly, considering IP destination addresses is not helpful, given that during a user session several servers may be contacted with very different IP addresses (e.g., advertisement pictures may be retrieved from a different server under a completely different administrative domain, or Content Delivery Network service may force the information retrieval from different servers).

Before running the clustering algorithm on this variable, traces are preprocessed according to the following rules. 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 among the about 7000 IP addresses that appear in the traces, 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, i.e. a trace containing only data with a given IP source address, 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 avoiding to be confused by too long silence periods.

After the metric space definition, we need to select a cluster analysis method. Recall that hierarchical agglomerative cluster analysis methods proceed by creating clusters through a cluster merging procedure: this method is easy to implement, but it does not scale well with the number of samples, which in our case is fairly large (during OCT.02, $N = |A(\mathrm{IP})|$ ranges up to 57000 samples, while during APR.04, $N = |A(\mathrm{IP})|$ ranges up to 26000 samples). On the other hand, partitional methods, which are relatively efficient, require an a priori knowledge of the number of clusters K. Moreover, in partitional methods the placement of the centroid may have a great impact on both the quality of the clustering, and the speed of convergence. To take the advantages and to avoid the drawbacks of both 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

[2]Given the possibility that a tear-down sequence of a flow under analysis is never observed, a timer is needed to avoid memory starvation. We selected a quite large value for the timer, according to the findings in [15].

the final number of clusters N_c

3. a partitional algorithm is used to obtain a fine definition of the N_c clusters.

3.2.1 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 Sec. 4). 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 $g=0$.

3.2.2 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 iterations, the process ends.

We need to define the clustering quality indicator function $\gamma^{(i)}$ that allows us to select the best clustering among those determined in the iterative process. Indeed, at each step, we must evaluate the quality of the clustering to decide if the optimal number of clusters has been found. 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)

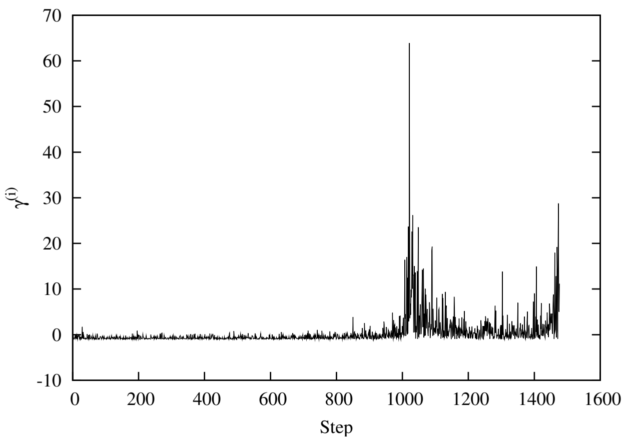


Figure 2. A sample plot of the quality indicator function $\gamma^{(i)}$. Exponential flow inter-arrival scenario.

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:

$$N_c = N - \left(\operatorname*{argmax}_i \left(\gamma^{(i)} - \gamma^{(i-1)}\right) - 1\right)$$

which corresponds to the sharpest increase in $\gamma^{(i)}$ and is thus considered the best one for partitioning the sample set. A typical behavior of the evolution of the $\gamma^{(i)}$ function is reported in Fig. 2, where the sharpest increase is clearly visible. The plot refers to an artificial trace obtained as described in Sec. 4, and shows that for about 1000 steps the aggregation of the two closest clusters is clearly beneficial in terms of clustering quality. Then, the aggregation process joins two clusters which are too far apart, forcing a sudden increase in $d_{min}^{(i)}$ from step $i-1$ to step i and therefore in $\gamma^{(i)}$. When $\gamma^{(i)}$ reaches the maximum, the joining procedure is forcing the artificial aggregation of two distinct clusters. Other errors are induced later in the iterative aggregation process: although clearly visible, they have a minor impact on the quality indicator function.

3.2.3 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 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 cluster 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 [16], which efficiently implements many clustering algorithms.

4 Performance Analysis: Artificial Traffic

In this section, we discuss the correctness properties of the proposed approach and compare it with respect to threshold based mechanisms.

Let us consider a simple artificial trace in which a *single* user generates sessions according to an ON/OFF process. Denote the ON and OFF period average durations as T_{on} and T_{off}. During each ON period of a session, a random number of TCP flows is generated, with average inter-arrival time denoted by T_{arr}. The session ON and OFF periods are random variables, exponentially distributed, with average $T_{on} = 20s$, whereas T_{off} ranges between $30s$ and $2000s$. For what concern flow arrivals, we set the average inter-arrival time $T_{arr} = 1s$ and we consider both the exponential and Pareto distributions. Indeed, considering the exponential distribution yields to a classic Poisson process, with no possible control on the variance of the flow inter-arrival process ($\sigma^2 =$ ‘). On the contrary, considering Pareto distribution allows us to control the variance of the process. Recall that the Pareto distribution $f(x)$ is characterized by two parameters a and b; the probability density function is:

$$f(x) = \frac{ab^a}{x^{a+1}} \quad x > b$$

while the average μ and variance σ^2 are respectively:

$$\begin{aligned} \mu &= \frac{ab}{a-1} \\ \sigma^2 &= \frac{ab^2}{(a-1)^2(a-2)} \end{aligned}$$

The Pareto distribution is known to be an heavy tailed distribution, and depending on the value of a may have finite or infinite moments of order n. In particular, by selecting $a = 2$ we obtain a distribution with finite mean, but infinite variance. This choice yields to a larger probability that the inter-arrival of flows is comparable with the inter-arrival of sessions, therefore creating a stiffer scenario. To obtain the average value $T_{arr} = 1s$ we set $a = 2$ and $b = 0.5$.

We also examined Weibull distributed flow inter-arrivals to consider the case in which connection inter-arrival process exhibits a variance smaller than 1, but we do not report results for this distribution, since they are similar to those shown.

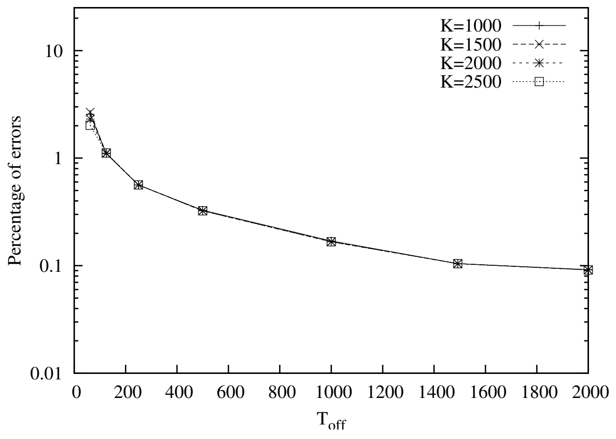


Figure 3. Sensitivity to the initial number of clusters K. Exponential flow inter-arrivals.

As performance metric, we use the percentage of misidentified sessions. Two types of errors are possible: (i) to erroneously separate a session in two or more clusters or (ii) to merge two or more distinct sessions. The percentage of errors is then defined as 100 times the total number of errors observed divided by the total number of flow arrivals. All curves are averaged considering 50 different runs, each comprising 500 sessions (an average of 10000 flow arrivals per run).

4.1 Parameter sensitivity

We start by evaluating the impact of the parameters given as input to the clustering methodology, namely the initial number of clusters K in the first phase, and the values of the percentile g during the hierarchical clustering. Considering the exponential flow inter-arrival scenario, in Fig. 3 we show that the error probability is practically independent from the value assumed for the initial number of clusters, which therefore is not a critical parameter (but it must be sufficiently larger than the supposed number of sessions). From now on we set $K = 1000$.

Fig. 4 shows instead the influence of the parameter g that determines the value of the percentile used to represent a cluster in the cluster-to-cluster distance. We report the single linkage method (which take the two extreme values in the sample distribution as representative, i.e., $g = 0$), the centroid method (which uses the average of the sample distribution) and curves for different values of g when using percentiles. Observe that the best performance is obtained with the single linkage method. The centroid and the 45-

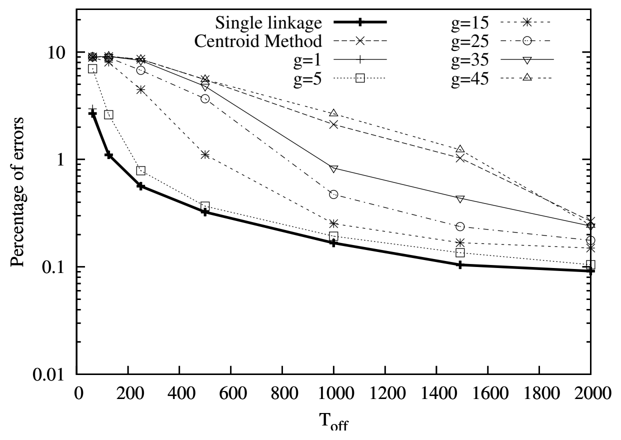


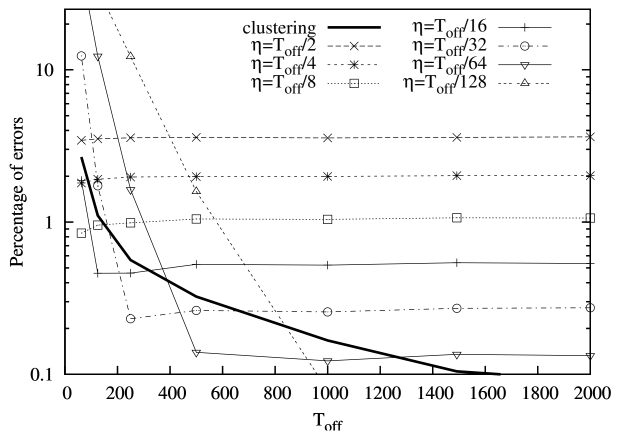
Figure 4. Sensitivity of the percentage of errors to the percentile g. Exponential flow inter-arrivals.

th percentile methods exhibit the worst performance, and performance of percentile methods shows that small values of g are preferable. Indeed, in all the observed scenarios (i.e., different distributions) we got the best performance by using the single linkage method, which will therefore be the choice made in the remainder of the paper. This solves also the issue of setting a proper value for g, which again allows us to run the clustering scheme without being dependent, in terms of performance, by any external parameter.

4.2 Percentage of misidentified sessions

We now consider the percentage of misidentifies session to assess the quality of the results and to compare them with traditional threshold based approach. Results are reported in Fig. 5, where we show, as a function of the average OFF period T_{off}, the percentage of errors obtained using the proposed clustering scheme (three steps, initial number of cluster K set to 1000, single linkage hierarchical clustering) and the classical threshold schemes for variable values of the threshold η. Exponential flow inter-arrivals are reported in the top plot, Pareto flow inter-arrivals in the bottom plot; clustering scheme performance is shown with a solid black line.

For all schemes, performance obviously improves as T_{off} increases, since long silent period among user sessions is easily detected with respect to short silent period among flow arrivals within an ON period. Threshold based mechanisms may perform better, provided that the proper threshold value is chosen. However, if the threshold is not correctly selected, errors are much higher than those obtained when using the clustering scheme. In general, large



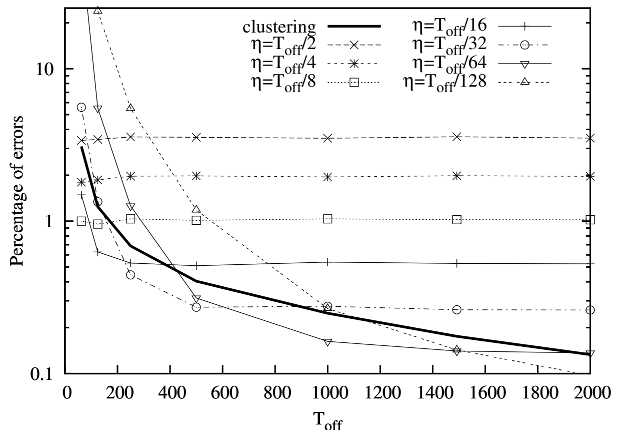


Figure 5. Percentage of errors. Exponential and Pareto flow inter-arrivals in top and bottom plot respectively.

values of η exhibit a higher percentage of errors, due to the probability of erroneously merging two subsequent different sessions. On the contrary, a sharp increase in error probability occurs for small values of η when T_{off} goes below a given value, i.e., when T_{off} becomes closer to T_{arr}.

In general, the clustering scheme shows a percentage of errors never larger than 2%, and it is less sensitive to variations of T_{off}. Moreover, it does not require to set any parameter like the threshold value that can dramatically affect the error probability. Notice also that, when considering the Pareto distribution of flow inter-arrivals, no big differences are noticeable. Only an increase in the percentage of errors when using the threshold based approach for small values of T_{off} is visible. This is due to the increase in the probability that a flow inter-arrival is quite large, thus becoming comparable with the OFF duration. The clustering approach is less affected by such events.

4.3 Mean inter-arrival estimation errors

To gather the accuracy of the methodology in the estimation of the system parameters, Fig. 6 reports the percentage of errors when determining $E[T_{arr}]$, the mean connection inter-arrival time, and $E[T_{off}]$, the mean session OFF period duration as obtained after running either clustering algorithm (lines), or threshold based approach with different values of η (lines with points). Also in this case we consider the exponential scenario, in which T_{arr} is fixed to $1s$, and $T_{off} = 62.5s$. Errors are averaged over 5 runs each comprising 500 sessions.

It is immediate to notice that the clustering approach is very accurate in the estimation of both T_{off} and T_{arr}, exhibiting a relative error of about -1.4% in the estimation of T_{off}, and 4.4% in the estimation of T_{arr}. On the contrary, the threshold based approach is very sensitive to the choice of η. In particular, in the case $\eta < 5$, a larger number of session are identified, therefore under-estimating both T_{arr} and T_{off}, while for $\eta > 5$, an over-estimation of system parameters is due to the large number of erroneous merging of sessions.

In summary, the clustering approach proposed in this paper is less error prone than threshold based approaches during the identification of Web user-sessions, and does not require any ad hoc parameter settings which largely affects the performance of simple threshold-based approaches.

Results show that threshold based mechanisms are very sensitive to the choice of η. This makes their use very questionable, especially when considering real traffic datasets, as a bad choice of η may lead to large errors. We therefore avoid to report results using traditional threshold method and only adopt the novel technique we devised so far.

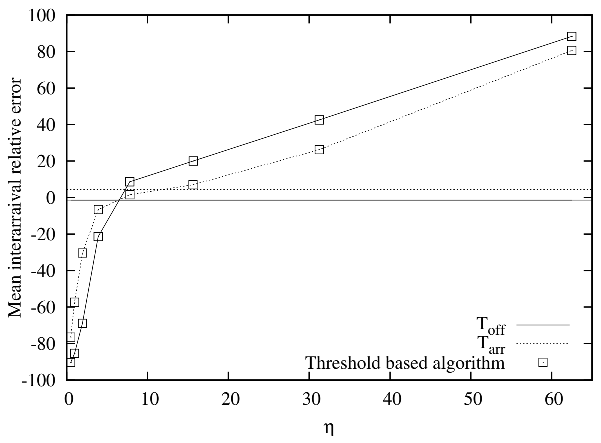


Figure 6. Percentage of errors on the estimation of the mean OFF session time (solid lines) and flow inter-arrival time (dotted lines). Clustering (no points) and threshold (square points) algorithms for different values of the threshold η for classic approach.

5 Web user-session characterization

Fig.7 shows the probability density function for the duration of the sessions identified during APR.04. The OCT.02 data show similar characteristics and are not reported. 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 tail.

The two different distributions shown in Fig. 7 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. 3, for a given session/cluster C, we can define

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

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

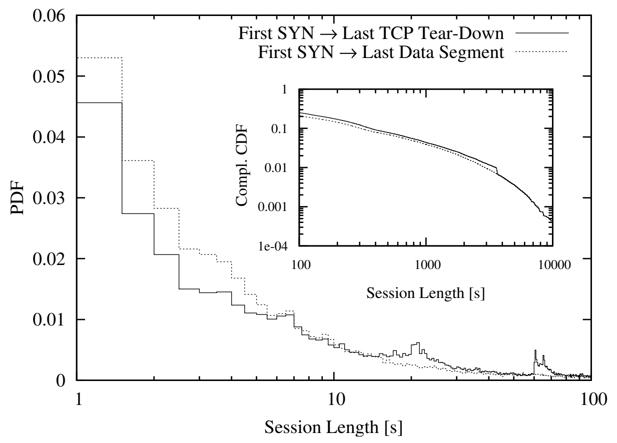


Figure 7. PDF (and Complementary CDF in the inset) of the session length.

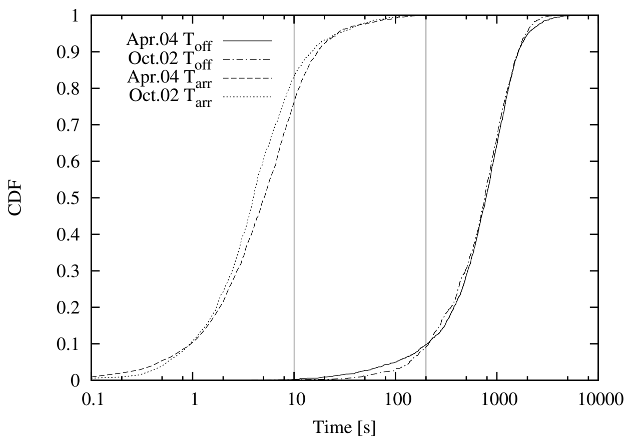


Figure 8. CDF of the mean user session inter-arrival and mean users flow inter-arrival.

(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.

Fig. 8 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. Both OCT.02 and APR.04 dataset are considered, to show how similar are the results obtained during both time frame during which the session analysis was performed. 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.

6 Conclusions

We propose an application of clustering techniques to a large set of real Internet traffic traces to identify Web user-sessions. We compared our methodology with the classical threshold based methods. The effectiveness and robustness of our clustering method has been assessed applying it to an artificial data set, showing its ability in the identification of user-sessions without requiring the a priori definition of threshold values.

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

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] 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.

[2] M.E. Crovella, A.Bestavros, “Self Similarity in World Wide Web Traffic: Evidence and Possible Causes,” *IEEE/ACM Transactions on Networking*, vol.5, no.6, pp.835–846, Dec. 1997.

[3] R.Caceres, P.Danzig, S.Jamin, and D.Mitzel, “Characteristics of Wide-Area TCP/IP Conversations,” *ACM SIGCOMM ’91*, Aug. 1991.

[4] P.Danzig, S.Jamin, "tcplib: A library of TCP Internetwork Traffic Characteristics," *USC Technical report*, 1991.

[5] P.Danzig, S.Jamin, R.Caceres, D.Mitzel, D.Mestrin, "An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations," *Internetworking: Research and Experience*, vol.3, no.1, pp.1–26, 1992.

[6] V.Paxons, "Empirically Derived Analytic Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, vol.2, pp.316–336, August 1994.

[7] 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.

[8] W.Willinger, V.Paxson, M.S.Taqqu, "Self-similarity and Heavy Tails: Structural Modeling of Network Traffic," *In A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, R.Adler, R.Feldman, M.S.Taqqu editors, Birkhauser, 1998.

[9] T.Bonald, A.Proutière, G.Régnié, J.W.Roberts, "Insensitivity Results in Statistical Bandwidth Sharing," ITC 17, Salvador, Brazil, December 2001.

[10] D.J.Hand, H.Mannila, P.Smyth, *Principles of Data Mining*, MIT Press, 2001.

[11] "The GARR Network topology," *http://www.garr.it/mappagarr/garr-b-mappagarr.shtml*, 2005

[12] M.Mellia, A.Carpani, R.Lo Cigno, "Measuring IP and TCP behavior on Edge Nodes," *IEEE Globecom 2002*, Taipei, TW, November 2002.

[13] M.Mellia, R.Lo Cigno, F.Neri, "Tstat web page," http://tstat.tlc.polito.it/, 2001.

[14] S.McCanne, C.Leres, and V.Jacobson, "Tcpdump," http://www.tcpdump.org, 2001.

[15] G.Iannaccone, C.Diot, I.Graham, and N.McKeown, "Monitoring very high speed links," *ACM Internet Measurement Workshop, San Francisco*, November 2001.

[16] P.Dalgaard, Introductory Statistics with R, *Springer*, 2002.