POLITECNICO DI TORINO
Repository ISTITUZIONALE

User Patience and the Web: a hands-on investigation

Original
User Patience and the Web: a hands-on investigation / Rossi, D.; Casetti, CLAUDIO ETTORE; Mellia, Marco. - STAMPA. - (2003), pp. 4163-4168. (Intervento presentato al convegno IEEE GLOBECOM 2003 nel novembre 2003) [10.1109/GLOCOM.2003.1259011].

Availability:
This version is available at: 11583/1410424 since:

Publisher:
IEEE

Published
DOI:10.1109/GLOCOM.2003.1259011

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

08 May 2024

User Patience and the Web: a hands-on investigation

D. Rossi, M. Mellia, C. Casetti
CERCOM - Dipartimento di Elettronica
Politecnico di Torino, Torino, Italy
E-mail: {rossi, casetti, mellia}@mail.tlc.polito.it

***Abstract*— We present a study of web user behavior when network performance decreases causing the increase of page transfer times. Real traffic measurements are analyzed to infer whether worsening network conditions translate into greater impatience by the user, which translates in early interruption of TCP connections. Several parameters are studied, to gather their impact on the interruption probability upon web transfers: times of day, file size, throughput and time elapsed since the beginning of the download. Results presented try to paint a picture of the complex interactions between user perception of the Web and network-level events.**

I. Introduction and motivation

Web browsing is one of the most popular activities on the Internet, so it is not surprising that network traffic largely consists of interactive HTTP connections. Web users, at a rather unconscious level, usually define their browsing experience through the page *latency* (or *response time*), defined as the time between the user request for a specific web page and the complete transfer of every object in the web page.

With the improvement in server and router technology, the availability of high-speed network access and larger capacity pipes, the web browsing experience is currently improving. However, congestion may still arise, causing the TCP congestion control to kick in and leading to higher page latencies. In such cases, users can become impatient, as testified by the popularization of the *World Wide Wait* acronym [1]. The user behavior radically changes, the current transfer is aborted, and maybe a new one is started right away, e.g., hitting the 'stop-reload' buttons in Web browsers.

This behavior can affect the network performance, since the network does some effort to transfer information which might turn out to be useless. Furthermore, resources devoted to aborted connections are unnecessarily taken away from other connections.

In this paper, we do not focus on the causes that affect the web browsing performance, but, rather, on the measurement of the impact of the user behavior when dealing with poorly performing web transfers. Using almost two months of real traffic analysis, we study the effect of early transfer interruptions on TCP connections, and the correlation between connection parameters (such as throughput, file size, etc.) and the probability of early transfer interruption.

This work was supported by the Italian Ministry for University and Scientific Research under the project TANGO.

The rest of this paper is organized as follows: Section II defines and validates the interruption measuring criterion; Section III analyzes the interruption of real traffic traces, reporting the most interesting gathered results; conclusive considerations are the object of Section IV.

II. Interrupted Flows: a definition

With *interruption event* we indicate the early termination of an ongoing Web transfer by the *client*, before the server ends sending data.

From the browser perspective, such an event can be generated by several interactions between the user and the application: aborting the transfer by pressing the stop button, leaving the page being downloaded by following a link or a bookmark, or closing the application.

From the TCP perspective, the events described above cause the early termination of all TCP connections[1] that are being used to transfer the web page objects. While it is impossible to distinguish among them, they can all be identified by looking at the evolution of the connection itself, as detailed in the following section. Though it would seem natural to consider the interruption as a "session" metric rather than a "flow" metric, session aggregation is extremely difficult and critical [2]. Therefore, due also to the hazy definition of "Web session", we will restrict our attention to individual TCP flows, attempting to infer the end of ongoing TCP connections, rather than the termination of ongoing Web sessions.

Our results were obtained running a TCP–level logger, called `Tstat` [4], [5] and developed by the Network Research Group at Politecnico di Torino. `Tstat` rebuilds TCP connection status by looking at trace of packets, tracking the connection set-up, evolution and tear-down. It passively analyzes the packet trace which contains both incoming and outgoing packets (so that both data and acknowledgment segments are present). As output, `Tstat` produces a TCP-level trace, logging several connection parameters for each analyzed flow. The results presented in this paper refer to almost two months of real traffic analysis performed on our campus access link (during the months of October and November 2002). Within our network there are more than 7000 hosts, mostly clients, but there are also several servers regularly accessed from the outside of our institution. A total of more than 2.2

[1] In this paper we interchangeably use the terms *connection* and *flow*.

 0-7803-7974-8/03/$17.00 © 2003 IEEE

millions TCP flows have been logged and analyzed, more than 88% of them being HTTP connections, i.e., server port equal to 80, upon which we restrict our analysis.

A. Methodology

In order to define a heuristic criterion discriminating between interrupted and completed TCP flows, we first inspected several packet-level traces corresponding to either artificially interrupted or regularly terminated Web transfers. We considered the most common operating systems and web browsers: Windows 9x, Me, 2k, Xp and Linux 2.2.x, 2.4.x were checked, in combination with MSIE 4.x, 5.x, 6.x, Netscape 4.7x, 6.x or Mozilla 1.x.

Figure 1 sketches the evolution of a single TCP connection used in interrupted (right) versus completed (left) HTTP transaction. In the latter case, after the connection set-up, the client performs a GET request, which causes DATA to be transmitted by the server. If persistent connections are used, several GET-DATA phases can follow. At the end, the connection tear-down is usually observed from the server side through FIN or reset (RST) messages. Conversely, user-interrupted transfers cause the client to abruptly signal the server the TCP connection interruption. The actual chain of events depends on the OS used by clients, i.e., Microsoft clients immediately send an RST segment, while Netscape/Mozilla clients gently close the connection by sending a FIN message first. From then on, the client replies with RST segments upon the reception of server segments that were in flight when the interruption happened (indicated by thicker arrows in the figure). In all cases, any user interruption action generates an event which is asynchronous with respect to the self-clocked TCP window mechanism.
In Figure 1, several time instants are also identified:

- T_{FS} and T_{FE} identifying the time of the TCP Flow Start and End, respectively;
- T_{CS} and T_{CE} identifying the time of the client request Start and End, corresponding to the first and last segment carrying data from the client side;
- T_{SS} and T_{SE} identifying the time of the server reply Start and End, corresponding to the first and last segment carrying data from the server side.

Timestamps are recorded by `Tstat`, which passively analyzes traffic in between the client and server hosts (its location being represented by the vertical dotted line in the figure); therefore, the time reference is neither that of the client nor of the server[2].

B. Interruption Criterion

From the single flow traffic analysis, we can define a heuristic discriminating among client-interrupted and completed connections. We preliminarily introduce a necessary condition to the interruption flow property, which we call *eligibility*, derived from the observation of Figure 1. TCP connections in which the *server* sent DATA but did not send a FIN (or RST) segment and the *client* sent an RST

[2]In the measurement setup we used, the packet monitor is close to the client (or server) side, and therefore the reference error is small, since the delay introduced by our campus LAN is small compared to the RTT.

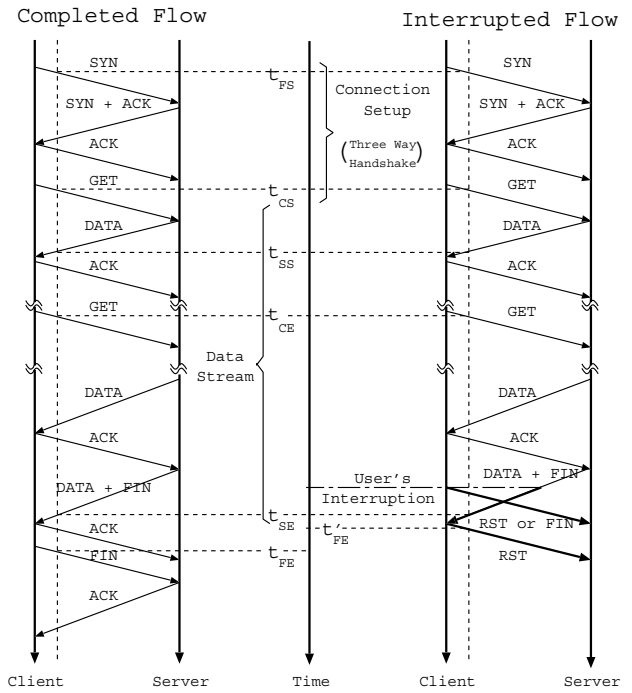

Fig. 1. Completed and Interrupted TCP Flow

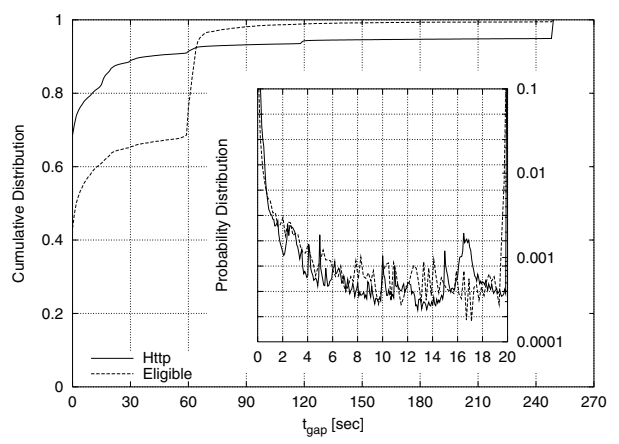

Fig. 2. t_{gap} Probability and Cumulative Distribution

segment are said to be eligible. Thus:

$$\textbf{Eligible} := \neg(\text{FIN}_S \vee \text{RST}_S) \wedge \text{DATA}_S \wedge \text{RST}_C \quad (1)$$

where the index (S or C) refers to the sender of the segment. The client FIN asynchronously sent by Netscape/Mozilla browsers can be neglected, because RSTs are sent anyway upon the reception of the following incoming server packets.

However, this criterion by itself is not sufficient to distinguish among interrupted and completed connections. Indeed, there are a number of cases in which we can still observe an RST segment from clients before the connection tear-down by servers. In particular, due to HTTP protocol settings [3], servers may wait for a timer to expire (usually set to 15 seconds after the last data segment has been sent) before

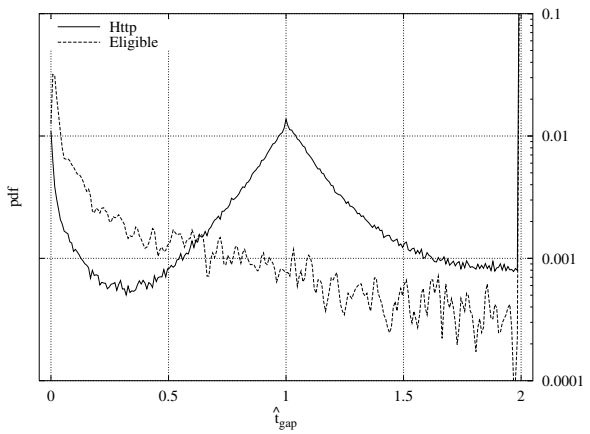

Fig. 3. Normalized $\hat{t}_{gap}$ Probability Distribution, $\alpha = 1,\ \beta = 0$

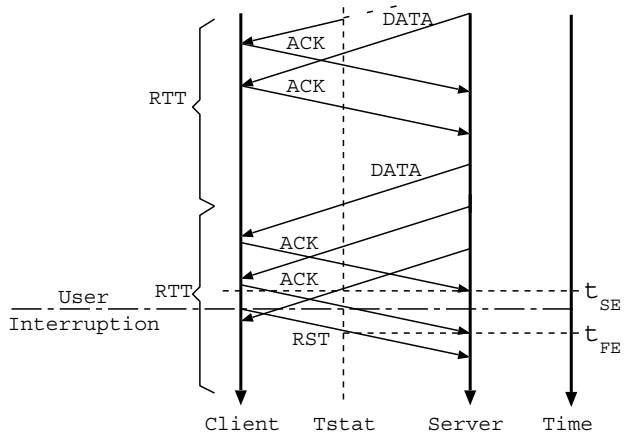

Fig. 4. Temporal Gap Reduction

closing the connection; moreover, HTTP 1.1 and Persistent-HTTP 1.0 protocols use a longer timer, set to a multiple of 60 seconds. Connections abruptly closed during this *idle* time would be classified as interrupted, even if the data transfers were already completed.

To gauge this, let us define t_{gap} as the time elapsed between the last data segment from the server and the actual flow end, i.e. $t_{gap} = t_{FE} - t_{SE}$. In Figure 2 we plot both the pdf (in the inset) and the CDF of t_{gap} for all HTTP connections (solid line) and the eligible ones (dotted lines). As can be observed, the majority of connections are closed within few seconds after the reception of the last data segment. The server-timer expiration is reflected by the pdf peak after 15s, which is clearly absent for the eligible flow class. But the presence of a timer at the client side, triggered about 60s after the last segment is received, causes the client to send an RST segment before the server connection tear-down, as shown by the CDF plot for eligible flows.

Unfortunately, all flows terminated by the timer expiration match the eligibility criterion: we need an additional *time* constraint in order to uniquely distinguish the interrupted flows from the subset of the eligible ones. Recalling that user interruptions are asynchronous with respect to TCP self-clocking based on the RTT, we expect that t_{gap} of an interrupted flow is roughly independent from TCP timings and upper-bounded by a function of the flow measured RTT. Let us define the normalized $\hat{t}_{gap}$ as

$$\hat{t}_{gap} = t_{gap}/(\alpha \cdot \mu_{\text{RTT}} + \beta \cdot \sigma_{\text{RTT}}) \tag{2}$$

where μ_{RTT} and σ_{RTT} are the average and standard deviation of the connection RTT respectively[3]. Figure 3 plots the $\hat{t}_{gap}$ pdf for both the eligible and non-eligible flows when $\alpha = 1$ and $\beta = 0$. For non-eligible flows, the pdf shows that $\hat{t}_{gap}$ can be either:

- close to 0 when the server FIN is piggybacked by the last server data segments and the client has already closed its half-connection or closes its half-open connection by the means of an RST segment;
- roughly 1 RTT when the server FIN is piggybacked by the last server data segments, and the client sends a FIN-ACK segment, causing the last server-side ACK segment to be received 1 RTT later by the server;
- much larger than 1 RTT for connections which remain open and are then closed by an application timer expiration.

Instead, considering eligible flows, we observe that t_{gap} is no longer correlated with the RTT. Moreover, we would expect that, in this case, the asynchronous interruption events uniformly distributed among one RTT. This is almost confirmed by Figure 3, except that the pdf exhibits a peak close to 0. This is explained considering the impact of the TCP window size: the transmission of several packets within the same window, and therefore during the same RTT, shifts the t_{SE} measurement point, reducing the t_{gap} toward smaller values than the RTT, as sketched in Figure 4.

Therefore, from the former observations, we define the flow *interruption* criterion as:

$$\textbf{Interrupted} := \textbf{Eligible} \wedge (\hat{t}_{gap} \leq 1) \tag{3}$$

As a further validation of the criterion, we plot in Figure 5 the CDF of the server data size transmitted on a connection of both complete and interrupted flows. Looking at the inset reporting a zoom of the CDF curve, it can be noted that the interrupted flows size is essentially a multiple of the maximum segment size (which is usually set to the corresponding Ethernet MTU of 1500 bytes). Indeed, for normal connections, the data size carried by flows is independent from the segmentation imposed by TCP. This further confirms that in the former case not all the server packets reached the client before the interruption happened.

In order to test the t_{gap} sensitivity to the interruption heuristic, we analyzed the interruption probability, i.e., the

[3]The μ_{RTT} and σ_{RTT} estimation used by `Tstat` is the same as the one TCP sender uses. The lack of accuracy of the algorithm, the variability of RTT itself and the few samples per flow make this measurement not accurate, affecting the $\hat{t}_{gap}$ distribution.

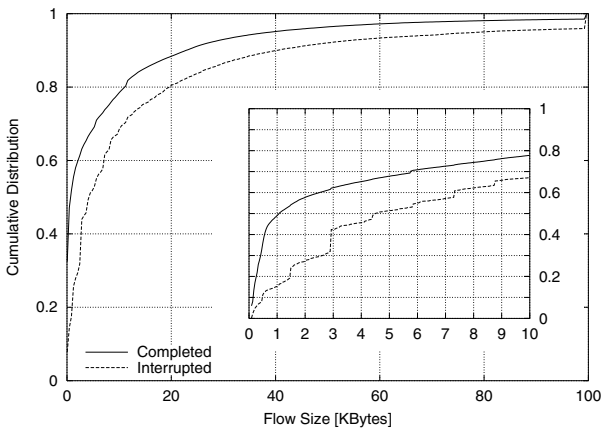

Fig. 5. Interrupted vs Completed Flows Size CDF

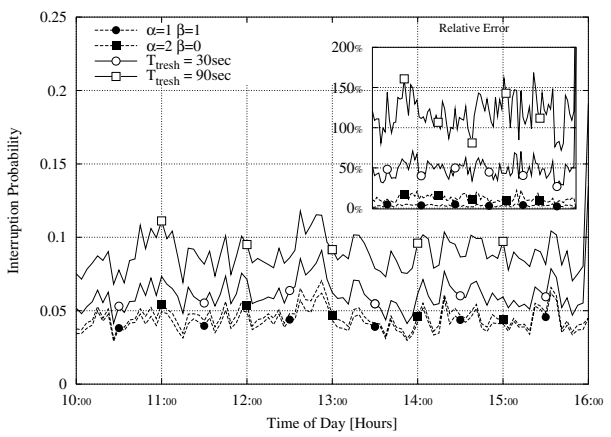

Fig. 6. Sensitiveness of interruption ratio to α, β

ratio of the interrupted connection number versus the totally traced connections, both for different values of α and β and with respect to a simplified interruption criterion that uses a fixed threshold (i.e., $t_{gap} < T_{thresh}$).

Results are plotted in Figure 6, adding in the inset the relative error percentage to the curve $(\alpha, \beta) = (1, 0)$, as a function of the time of day considering 10 min observation window. It can be seen that different (α, β) values do not largely affect the RTT-dependent results (the error is within few percentage points). On the contrary, a fixed-threshold approach deeply alters the interruption ratio, compromising the criterion validity. For example, when T_{thresh} includes the client 60-second timer of persistent connections, the error grows to over 100%: recalling the results of Figure 2, this would qualify almost all eligible flows as interrupted. Therefore we can confirm that the interruption criterion we defined so far is affected by a relative error which is however small enough to be neglected.

III. Results

In this section we study how the interruption probability is affected by the most relevant connection properties, such as the flow size, throughput and completion time. Also, we

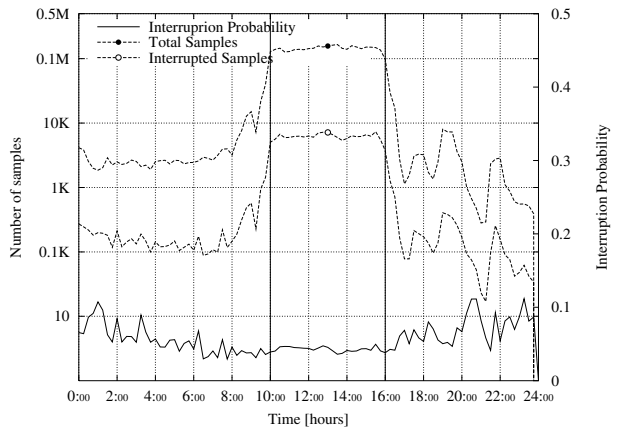

Fig. 7. Interrupted vs Completed vs Flows Amount and Ratio

TABLE I

Three Most Active server and client statistics: total flows ϕ_i and interrupted θ_i

	rank	ϕ_i	$\frac{\phi_i}{\sum_i \phi_i}$	θ_i	$\frac{\theta_i}{\phi_i}$
Internal server	1	186400	23.63%	15969	8.02%
	2	131907	16.72%	10024	7.02%
	3	86189	10.93%	7320	8.02%
External server	1	29300	2.85%	539	1.83%
	2	25637	2.49%	659	2.57%
	3	18448	1.99%	231	1.25%

discriminate flows as *client* or *server* (respectively when the server is external or internal to our LAN) and as *mice* or *elephants* (depending on whether their size is shorter than or longer than 100 KB).

Figure 7 plots the number of interrupted versus totally traced flow (left y-axis scale), together with their ratio (right y-axis), as a function of the time of day. Client flows only are considered[4]. As expected, the total number of tracked flows is higher during working hours, and the same happens to interrupted flows, leading to an almost constant interruption probability.

Given this behavior, in the following we will restrict our analysis to the 10:00–16:00 interval, where we consider both the traffic and the interruption ratio to be stationary. It must be pointed out that our campus is mainly a client network toward external servers, i.e., only the 8% of the tracked connections have servers inside our campus LAN. Therefore, to both have a statistically meaningful data set and to compare the client versus server results on approximatively the same number of connections, we used traces with different temporal extension. The *client* traces refer to the work week from Monday 7 to Friday 12 November 2002 from 10:00 to 16:00, where we observed 5848 unique clients contacting ~18000 unique external server for a total of more than 10^6 flows. Instead, *servers* data refer to a two-month-long trace (Monday to Friday, October to November 2002, 10:00–16:00), where

[4]Server flows yielded the same behavior.

51016 unique external clients contacted 118 unique internal servers generating ~$0.8 \cdot 10^6$ connections. For the same reasons, the *elephants* data-set refers to the same period of the server trace.

Considering the selected dataset, the average percentage of interrupted flow of all logged servers is 9.18%, while for all logged clients is 4.20%. This shows that a significant percentage of TCP flows are interrupted: this quantity was measured on our campus network, which offers a generally good browsing experience, therefore we expect this ratio to be much higher in worse-performing scenarios.

Table I details the interruption statistics for the three most contacted *internal* and *external* servers. ϕ_i and θ_i represent, respectively, the total and the interrupted number of observed flows. Apart from noticing that the number of *external* contacted servers is higher and therefore the traffic is more spread than the *internal* servers, it is worth to notice that the interruption probability of the three most contacted internal servers is roughly the same for each server (~8%). Considering the external servers statistics, the interruption ratio is smaller (from 1.25% to 2.57%), and also smaller than the average interruption probability which is larger than 4%. This suggests that the three most contacted servers offer a good browsing experience to our clients.

In order to better understand the motivations that drive user impatience, in the following subsections we inspect how the interruption probability varies when conditioned to different parameters x. In particular, we define as $\mathcal{P}_{|x}$ the ratio of the interrupted connection number over the total connection number, conditioned to a general x parameter, thus $\mathcal{P}_{|x} =$ P{flow is Interrupted $|\, x$ }. Intuitively, when $\mathcal{P}_{|x}$ is constant over any x value interval, this means that the interruption is not correlated with the parameter x.

A. Impact of the User Throughput

Let the *average user throughput* be the amount of the data transferred by the server over the time elapsed between the connection setup and the last server data packet: referring to Figure 1, we may write[5]:

$$\text{Throughput} = \sum \text{DATA}_S/(t_{SE} - t_{FS})$$

Figure 8 reports $\mathcal{P}_{|Throughput}$ as well as the number of total and interrupted flow samples, for both server (on the top) and client (on the bottom) flows. The number of samples can be read on the left y-axis, while the corresponding probability can be read on the right y-axis

It can be noted that, in the server case, $\mathcal{P}_{|Throughput}$ slightly decreases when the user transfer rate increases, while a general increase in the $\mathcal{P}_{|Throughput}$ is observed by client connections, which is quite counterintuitive. However, this is explained considering mice and elephant flows. Indeed, in the mice case, the interrupted flows throughput is ~1.3 times higher than for

[5]This performance parameter does not include the time elapsed during connection tear-down since it does not affect the user perception of transfer time.

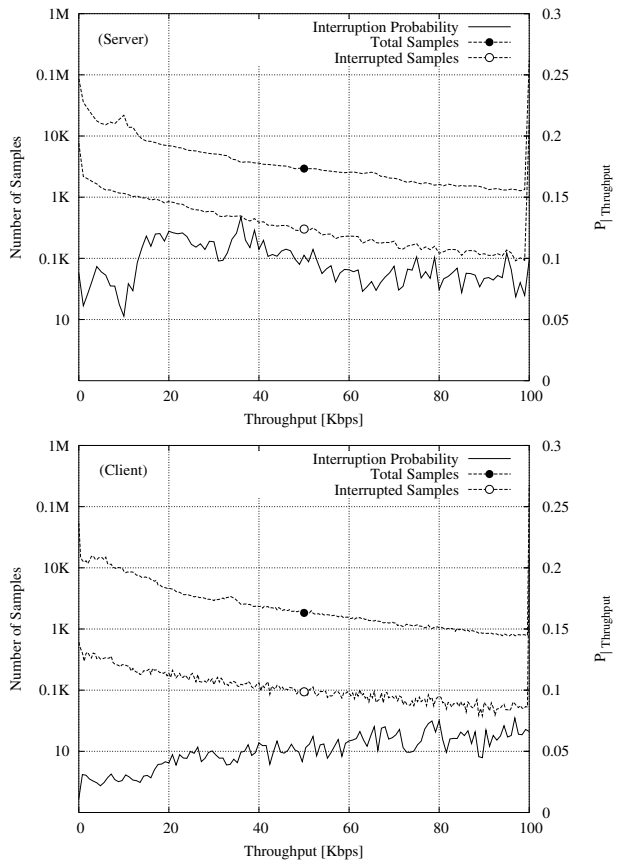

Fig. 8. $\mathcal{P}_{|Throughput}$: server on the top, client on the bottom

completed flows. This suggests that the early termination is due to a link-follow behavior (i.e., the user clicking on a link to reach a new page). On the contrary, interrupted elephant flows have a throughput 1.5 times smaller than the one of completed flows, confirming the intuition that a smaller throughput leads to higher interruption probability.

B. Impact of Flow Size

In Figure 9 the interruption probability is conditioned to the flow size[6], i.e., $\mathcal{P}_{|Size}$. Considering client flows (on the bottom), we observe that there is a peak of short transfers that are aborted: this is due to the interruption of parallel TCP connections opened by a single HTTP session. In the server case (top plot), the $\mathcal{P}_{|Size}$ is higher, on average, than the previous case. In both cases, against expectations, users do not tend to wait longer when transferring longer flows, as the increasing interruption probability suggests.

C. Completion and Interruption Times

Figure 10 shows the dependence of completed and interrupted server flows on the time elapsed since flow start until its end, i.e., $\mathcal{P}_{|time}$. It can be gathered from the figure that users mainly abort the transfer in the first 20 seconds: during this

[6]In the case of interrupted connections, the size has to be interpreted as the amount of data transferred until the interruption occurred.

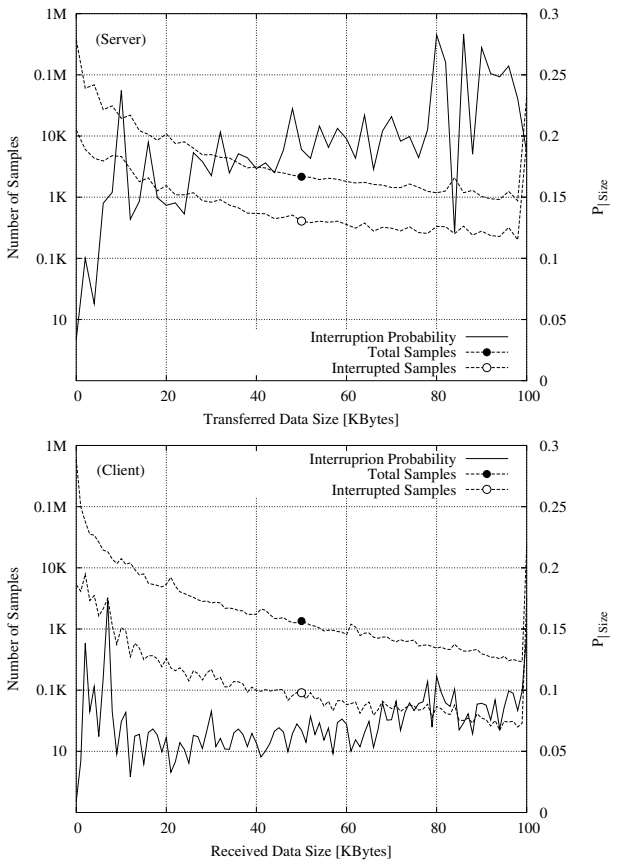

Fig. 9. $\mathcal{P}_{|Size}$: server on the top, client on the bottom

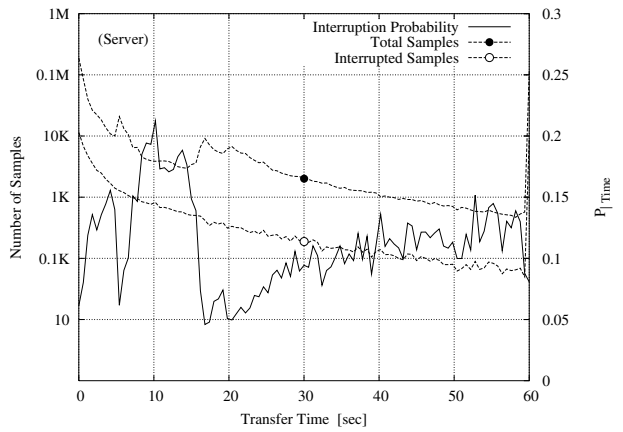

Fig. 10. $\mathcal{P}_{|time}$: server case only

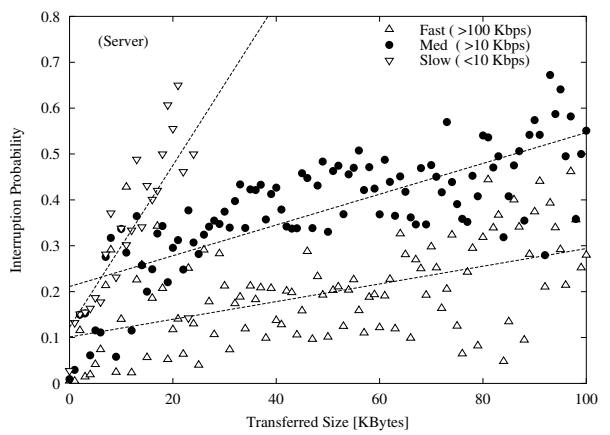

Fig. 11. $\mathcal{P}_{|Size|Throughput}$: server case only

time, users take the most 'critical' decisions, while, after that time, they tend to wait longer before interrupting the transfer. The slow rise in the interruption ratio after the 20 seconds mark, though, shows that users are still willing to interrupt the transfer if they think it takes too much time.

Finally, Figure 11 considers server flows within the 0-20s interval only. The $\mathcal{P}_{|Size}$ probability is further conditioned to different classes of users according to their throughput, i.e.,$\mathcal{P}_{|Size|Throughput}$. Three throughput classes are considered: Fast ($>$ 100Kbps), Slow ($<$ 10Kbps) and Medium speed (between 10Kbps and 100Kbps). Looking at the figure, it can be noticed that the three different classes suffer very different interruption probability: higher for slow flows, and much smaller for fast flows. Linear interpolation of data (dotted lines) is used to highlight this trend. Indeed, slow connections massively increase the interruption probability, while faster connections are likely to be left alone. This shows that the throughput is indeed one of the main performance indexes that drives the interruption probability.

IV. Conclusions

The research presented in this paper inspected a phenomenon intrinsically rooted in the current use of the Internet, caused by user impatience at waiting too long for web downloads to complete. We defined a methodology to infer TCP flows interruption, and presented an extended set of results gathered from real traffic analysis. Several parameters have been considered, showing that the interruption probability is affected mainly by the user-perceived throughput. The presented interruption metric could be profitably used in defining the user satisfaction of Web performance, as well as to derive traffic models that include the early interruption of connections.

References

[1] R. Khare and I. Jacobs, `http://www.w3.org/Protocols/NL-PerfNote.html`

[2] M. Molina et al., *Web Traffic Modeling Exploiting TCP Connections' Temporal Clustering through HTML-REDUCE*, IEEE Network, May 2000

[3] R. Fielding et al., *Hypertext Transfer Protocol HTTP/1.1*, RFC2616, June 1999

[4] `Tstat's Homepage, http://tstat.tlc.polito.it/`

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