KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 227–235

SOME FORMAL APPROACHES FOR DYNAMIC LIFE SESSION MANAGEMENT

F. BOIAN[(1)], D. BUFNEA[(2)], A. VANCEA[(3)], A. STERCA[(4)], D. COJOCAR[(5)], AND R. BOIAN[(6)]

Abstract. At this moment, the lifetime of a Web session is rigidly established at the level of the Web application and certain implicit constant values are suggested for this duration in the same stiffly manner by most of the Web technologies. This paper introduces some formal models for determining, establishing and dynamic maintaining the lifetime of a HTTP session. To achieve these goals we took into account also the personalized type of work every user does and the particular way in which he or she interacts with the Web application/server. In the following, three different formal approaches will be presented regarding the lifetime management of a Web session.

1. Introduction

If Tim Berners-Lee would have realized the impact and the ampleness which the HTTP protocol would have in the widespread and development of the Internet network at the beginning of '90s (but also the huge number of problems that this protocol had to face) then probably the specifications of this protocol would have been quite different. Today, using the Internet and the Web has acieved a social aspect, being a usual part of the everyday life of many. If at the beginning, the Web and the HTTP protocol were designed for presenting and exchanging documentats, manuals or scientific content information between researchers and academics, once the commercial Internet emerged and spread towards ordinary people, the actual requirements and challenges that this protocol and the HTML language has to face have changed. As the popularity and requirements of web applications grow higher, the HTTP design faces several problems. A set of such problems are related to the stateless dialog model using the HTTP protocol between a client-user and the Web server.

To overcome the drawbacks of the stateless communication model between the client and the server, the web application use the concept of HTTP working session.

2000 *Mathematics Subject Classification.* 90C59, 47N30.
Key words and phrases. dynamic HTTP session management.

In its most simple form, an HTTP session may be defined as the set of all the connections issued by a certain client to the Web server involved in solving the same problem at a given moment. The server is responsible for identifying at any given moment to which user a pending request belongs to and for sending back content (dynamically, in most of the cases) accordingly. The returned content varies depending on the requested action and the served user. The most popular mechanisms used to identify the session corresponding to a new request are:

- session id tagging of the requested URLs;
- session id tagging of request's HTTP headers (using cookies);
- SSL based (HTTPS) session management.

While these mechanisms solve the request/session matching succesfully, they do not offer any means for controlling the life-time of the session. The inability to determine the end of a session's life-time comes primarily from the stateless design of the HTTP protocol.

2. Previous work

Most Web technologies apply a rigid and inflexible vision (naive in a certain way) when establishing the lifetime of a session, namely to resort to a fix amount of time (we will call it T_{fixed}), usually 15-30 minutes long. If the user does not issue a request (does not interact with the application/Web server during an T_{fixed} time interval), he is considered "idle" and the working session is invalidated. In other words, from a stateful model perspective, the connection through which the client is served is closed. Although most web technologies allow on the server-side modification of the session expiration time, they do not offer any management or information relative to how this duration can be efficiently changed. Such a mechanism is much more necessary, because most users do not "officially" close the session (they either simply "forget" to click the logout button and either close the browser or leave the web application open for a long time).

Boian presents in [BB06a] an efficient model for calcualting the session lifetime. The mechanism proposed in [BB06a] is based on requiring explicit feedback from the client regarding its status. The time intervals at which this feedback is required are usually much smaller than the session's lifetime, so a fine time granularity is possible for session lifetime calculation. The feedback request intervals remain however rigid, fixed at a constant value s. Therefore, in [BB06a], Boian identifies two types of interactions between the client and the Web server as below (see figure 1):

- Business type actions initiated by the direct interaction between the user and the Web application (GET requests, POST or GET submits);
- *Web-ping* type actions which consist of periodically accessing (at fixed time intervals of s seconds) an URL u by the client, this invocation notifying the server relative to actively maintaining the working session from the client's part.

Client's notification regarding the necessity of accomplishing a *Web-ping* type action is done by using HTTP headers, inserted in the answering pages dynamically generated by the Web server, by means of a field which has the following form: <meta http-equiv="refreh" content="s;url=u">

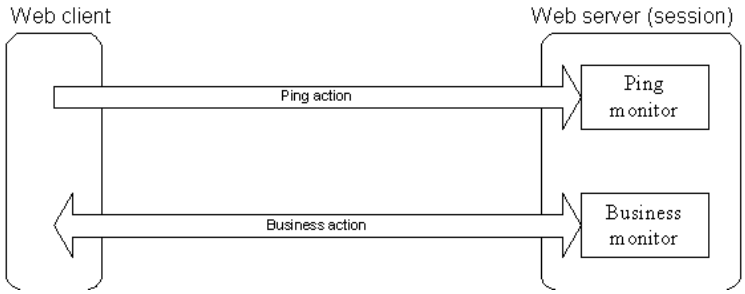

FIGURE 1. Web application architecture with *Web-ping* requests

For the *Web-ping* type actions not to affect the session's lifetime, when receiving such an action by the Web server/application, the T_{fixed} value is diminished by the time elapsed from the moment of the last performed action until the actual moment. For the server to decide if the client (browser) is inactive, it waits maximum r successive *Web-pings*, which follow one after the other without any business action between them. The r value is heuristically established, such that $r \times s << T_{fixed}$.

3. CONTRIBUTIONS

There is a significant number of situations in which the client - Web application interaction assumes frequently repeating different actions initiated by the user - one can identify a certain pattern in the user's behavior. Examples of such activities are: reading the e-mails using a Web-based client, browsing the pages inside a forum, assigning student's grades using the AMS portal [BB06b] etc. In such cases, based on identifying the user's type and time of answer, one can anticipate the user's behavior at the level of the Web server and determine in a more rigorous way a more flexible lifetime (depending on the user/application). The flexibility of the proposed model will be much more significant as the client is explicitly informed about the necessity of offering a feedback in a certain time interval, if it does not directly interact with the Web application.

Anticipating the user's behavior can be done either at the level of the Web application, or at a lower level inside of the application server which hosts that particular application. The application server may export this information to other Web applications by means of an API which the application can decide to

use it or not. The level of reusability, portability and the impact of the proposed mechanism are thus enhanced.

The present paper introduces some formal models for the dynamic management of the session's lifetime and for determining (also as a dynamic value) the time intervals at which the client must notify the Web server/application about its presence. In the exposed approaches we will take into account also the personalized type of work every user has and the particular way in which he interacts with the application.

4. Formal models

We consider until the time moment n, the time intervals elapsed between two user's consecutive interactions (business actions) with the Web application (such an interaction assumes a simple GET request, a submit action either by the GET method either by the POST one). We consider the length of these intervals to be $t_1, t_2, \ldots, t_n$, where t_i is the length of the time interval from the time moment i (the time elapsed between the business actions $i-1$ and i).

4.1. **Statistical approaches.** In [BB06a] the s value was established as an application constant. In the following, after each business action we will compute the maximum probable interval of time in which a business action should normally occur (we will try to anticipate the client's next business action). This value depends on two factors:

- the time moments of client's business actions from the start of the session up to the present time;
- the succession speed of the latest business actions.

Having as input data the random variable $T = (t_1, t_2, \ldots, t_n)$, our goal is to estimate the value t_{n+1}. We will use the following statistics elements [GS97]: mean, standard deviation and linear regression.

The estimation technique is illustrated in figure 2. The figure plots the eleven time intervals between the twelve requests posted by a user to a web application. The linear regression and mean of the elevent points are displayed in continuous lines. Teh dashed lines are parallel to the mean and linear regression lines, but are higher by three standrd deviations.

Let $m_n = \frac{1}{n} \times \sum_{i=1}^{n} t_i$ and $\sigma_n = \sqrt{\frac{1}{n-1} \times \sum_{i=1}^{n} (t_i - m_n)^2}$ be the mean, respectively the standard deviation of random variable T. We also denote by $y = a_n \times X + b_n$ the regression line where $\sum_{i=1}^{n} (a_n \times i + b_n - t_i)^2 \rightarrow$ minimum. This line is the best least squares fitting line for points t_i.

Without going into too many details, through simple calculus, we obtain the formulas for computing m_n, σ_n, a_n and b_n as a function of m_{n-1}, σ_{n-1}, a_{n-1}, b_{n-1} and t_n.

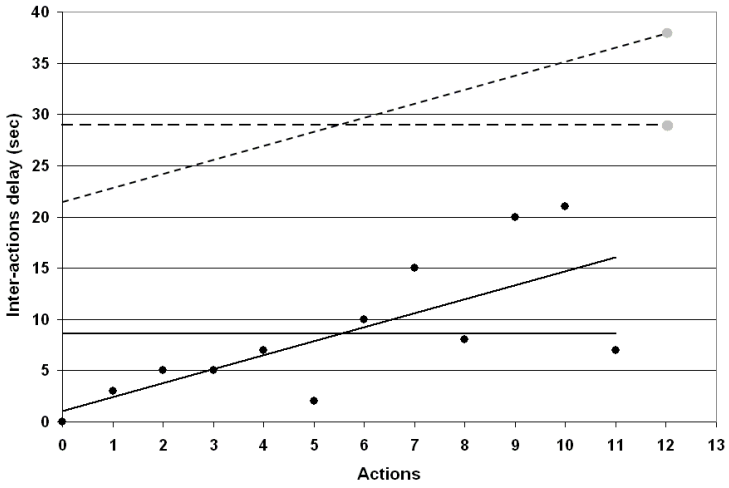

FIGURE 2. Statistical estimation of the latest moment when the next request will arrive

$$M1_n = \sum_{i=1}^{n} t_i = M1_{n-1} + t_n \tag{1}$$

$$M2_n = \sum_{i=1}^{n} t_i^2 = M2_{n-1} + t_n^2 \tag{2}$$

$$MI_n = \sum_{i=1}^{n} i \times t_i = MI_{n-1} + n \times t_n \tag{3}$$

$$m_n = \frac{1}{n} M1_n \tag{4}$$

$$\sigma_n = \sqrt{\frac{1}{n-1}(M2_n - 2m_n \times M1_n + n \times m_n^2)} \tag{5}$$

The simple regresion line $y = a_n x + b_n$ has the following coefficients:

$$a_n = \frac{6[2MI_n - (n+1)M1_n]}{n(n+1)(n-1)} \tag{6}$$

$$b_n = \frac{2M1 - a_n^n(n+1)}{2n} \tag{7}$$

The standard deviation $\overline{\sigma_n}$ of the Z, $z_i = a_i + b - t_i$, random variable is:

$$\overline{\sigma_n} = \frac{\sqrt{\frac{1}{n-1}(\frac{a_n^2 n(n+1)(2n+1)}{6} + nb_n^2 + M2_n + n \times \overline{m_n}^2 + }}{\overline{a_n b_n n(n+1) - 2a_n MI_n - a_n \overline{m_n} n(n+1) - 2b_n M1_n - 2nb_n \overline{m_n} + }}$$
$$\overline{2\overline{m_n} M1_n)}$$

where:

$$\overline{m_n} = \frac{a_n(n+1)}{2} + b_n + \frac{M1_n}{n} \tag{8}$$

The well known three σ rule from statistics says: "the next value of a random variable will be in the interval $[m - 3\sigma, m + 3\sigma]$ with a probability of over 99%". According to this rule, we can expect that $t_{n+1} < m_n + 3\sigma$.

Also, according to a well known heuristic principle taken from operating systems theory [BV07], if at some point in time a certain resource location (memory location, disk location etc.) is being accessed, then the next access will be in the vicinity of the previous one with a very high probability.

In our context, this principle translates to: "if the frequency of the latest business actions' occurrence is high, then it is expected that the frequency of occurrence of the next business actions remains high in the following period". The example from figure 2 depicts such a situation. If this frequency was low, it is expected to remain low. Due to these reasons, we intend to consider the regression line which gives the slope of business actions occurrences frequency. According to this observation we can expect that $t_{n+1} < a_n \times (n+1) + b_n + 3\overline{\sigma_n}$, where $\overline{\sigma_n}$ is the data's standard deviation from the regression line.

Combining the above two principles, we will express t_{n+1} as:

$$t_{n+1} = max(m_n + 3\sigma, a_n \times (n+1) + b_n + 3\overline{\sigma_n})$$

4.2. **A transport level approach for dynamic life session management.** A different approach for the same goal can be considered. It implies the use of a model similar to the one used in the TCP protocol for estimating the nominal value of the round-trip time.

Having at time n an estimation, E_n, for the time that must pass until the user will interact with the web application again, we approximate the length of the time interval between time moment n and time moment $n+1$ as:

$$E_{n+1} = \alpha \times E_n + (1 - \alpha) \times t_n$$

where the interaction time estimation at moment E_i has a higher weight in anticipating the interaction time at moment E_{n+1} than the length of the time interval between time moment $n-1$ and n ($\alpha > 0.5$). This approach eliminates the noise which occurs in estimating the client's interaction time with the web application/server and, on the long run, for a relatively constant behavior of the client makes the estimation value converge to the real value of the interaction time. For an optimal implementation of the above formula, in practice, the value $\alpha = 0.875$ will be used (the two multiply operations will reduce to a bit shift and a substract operation).

For a best fit of this estimated time moment, we use an approximation error interval around the length of the time interval which is computed above. Thus, we approximate the time moment t_{n+1} as:

$$t_{n+1} = E_{n+1} + l_{n+1}$$

where l_{n+1} is a value which increases proportionally to the variation of client's interaction time values (time moment t_{n+1} will be approximated in a bigger time interval):

$$l_{n+1} = \beta \times l_n + (1-\beta) \times (t_n - t_{n-1})$$

where $\beta = 0.875$.

4.3. **The use of the t_{n+1} estimation.** The formal models presented above can be implemented at application level in a web application server to dynamically track the life time of a session depending on the custom behavior of each client. Moreover, once the next interaction time with the web application/server is estimated, the web server can notify the client through an HTTP header (cookie - [FG99]) about the necessity of an explicit feedback (keep-alive message) from him. This feedback can be performed either by a direct action of the user or through a *Web-ping* mechanism like the one described in [BB06a].

In the absence of such a feedback, at time moment $n+1$, in the most conservative approach, the web server/application can decide to invalidate/close the user session even if the life time of such a session is set to a value of $T_{fixed} > t_{n+1}$. A more flexible approach, identical to the one proposed in [BB06a], implies the cancellation of a user session only after $r \times t_{n+1}$ seconds if the web application/server receives no message (action) from the client in this time interval. This approach allows also the cancellation of a user session if the client (browser) does not act.

For example, if x ping messages are received and no business message, then one can consider cancelling the user session for inactivity reasons. The values r and x has to be have heuristic values (we will address this in future papers).

Figure 3 presents real values time evolution vs. estimated ones time evolution.

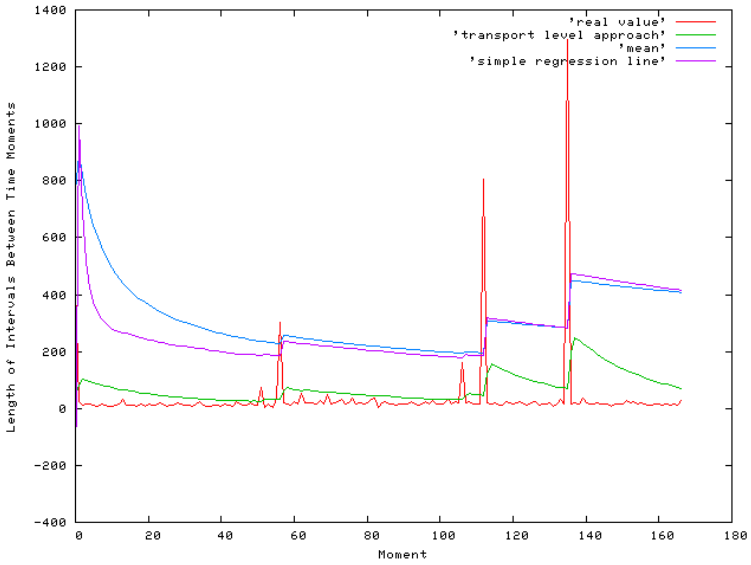

FIGURE 3. Session lifetime estimation results

5. Conclusion and future work

Our paper introduces two formal models for dynamically tracking the life time of an HTTP session, models intended to replace the rigid way of setting a session's life time to a constant value. Also, it proves that a couple of concepts and mechanisms applied to some layers of the TCP/IP stack can be adapted and used at other layers of the stack (in our case, a transport layer mechanism was used at the application layer).

As future work, we intend to implement the proposed models in a web application which is used frequently, e.g. [BB06b]. Also, we want to create an API that extends the functionality of an application server and offers the suggested functionality to all interested applications (migrate the business logic of the proposed models from the web application to the application server layer).

References

[BB06a] F.M. Boian, D. Bufnea, A. Vancea, A. Sterca, D. Cojocar, R. Boian, *A Model for Efficient Session Object Management in Web Applications* in Proceedings of the Symposium "Colocviul Academic Clujean de Informatică" Cluj-Napoca, 2006, pp. 131-136;

[BB06b] F.M. Boian, R. Boian, A. Vancea, *AMS: An Assignment Management System for Professors and Students* in Proceedings of the Symposium "Colocviul Academic Clujean de Informatică" Cluj-Napoca, 2006, pp. 137-142;

[FG99] R. Fielding, R. J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, June 1999;

[WWW1] *Improved Session Tracking*, http://www.mojavelinux.com/blog/archives/2006/09/improved_session_tracking/, September 2006;

[PA00] V. Paxson, M. Allman, *Computing TCP's Retransmission Timer*, IETF RFC 2988, November 2000;

[GS97] Charles M. Grinstead, J. Laurie Snell, *Introduction to Probability*, American Mathematical Society, July 1997;

[BV07] F. Boian, Al. Vancea, D. Bufnea, C. Cobârzan, A. Sterca, D. Cojocar, *Sisteme de operare*, Editura Risoprint 2006, ISBN 973-751-220-0, 978-973-751-220-8.

(1) Babeş-Bolyai University
E-mail address: florin@cs.ubbcluj.ro

(2) Babeş-Bolyai University
E-mail address: bufny@cs.ubbcluj.ro

(3) Babeş-Bolyai University
E-mail address: vancea@cs.ubbcluj.ro

(4) Babeş-Bolyai University
E-mail address: forest@cs.ubbcluj.ro

(5) Babeş-Bolyai University
E-mail address: dan@cs.ubbcluj.ro

(6) Babeş-Bolyai University
E-mail address: rares@cs.ubbcluj.ro