Management and Production Engineering Review
Volume 1 • Number 4 • December 2010 • pp. 67–76

INTEGRATING A KEY CUSTOMER-ORIENTED STRATEGY INTO THE B2C E-COMMERCE SERVICE

Grażyna Suchacka[1], Leszek Borzemski[2]

[1] *Opole University of Technology, Institute of Automatic Control and Computer Science*
[2] *Wrocław University of Technology, Institute of Informatics*

Corresponding author:
Grażyna Suchacka
Opole University of Technology
Institute of Automatic Control and Computer Science
Sosnkowskiego 31, 45-272 Opole, Poland
phone: +48 77 4006212
e-mail: g.suchacka@po.opole.pl

Received: 15 October 2010
Accepted: 20 November 2010

Abstract
The paper brings up the problem of broadening customer relationship management (CRM) in an e-commerce company to an automated mechanism of request service in a Web server system hosting a Business-to-Consumer (B2C) Web site. We propose applying Recency-Frequency-Monetary value (RFM) analysis to automatically compute customer values in a Web store, and using these values in a new admission control and scheduling algorithm for a Web server system. The proposed method and algorithm are discussed, and simulation results of its efficacy compared to the First-In-First-Out (FIFO) scheduling are presented.

Keywords
customer value, e-commerce, Web server, admission control, scheduling, QoS, Quality of Service.

Introduction

In recent years it has become evident for many companies that a key factor for their success is the adoption of a customer-oriented business strategy. Such approach is especially important in the highly competitive Business-to-Consumer (B2C) electronic commerce environment, where Internet users can easily compare available products and their prices in various Web stores, and switch between them with only several mouse clicks. In such a situation, maintaining strong relationships with e-customers, based on their long-term loyalty and a reliable company image, can considerably contribute to the e-business profitability [1].

The adoption of a customer-focused strategy is referred to as customer relationship management (CRM). CRM relates to acquiring, analyzing and using knowledge of customers to automate and synchronize business processes in many areas, such as sale, customer service, technical support, and marketing. Modern technologies, including dedicated CRM systems, are used to support this strategy. Effective customer relationship management can bring many benefits to a company [2], including increased customer retention and loyalty, higher customer profitability, creation of value for the customer, as well as customization and higher quality of products and services.

Defining and measuring customer value

CRM pursues maintaining long-term relationships with profitable customers. To identify them, a company should choose and apply a method for gathering information on each individual customer's value.

Customer value is often referred to as customer profitability, and in the long-term perspective is also called Life Time Value (LTV) or Customer Lifetime Value (CLV). Generally, customer value may be de-

fined either from a supplier-oriented point of view as the customer's economic value to the company, or from a demand-oriented point of view as the company's or its products' value to the customer [3]. Within the scope of this paper, we consider customer value in the first meaning.

As yet, there has not been one commonly accepted model of customer value. According to the most common definition of LTV, it is the sum of the revenues gained from a company's customer over the lifetime of his/her transactions after the deduction of the total cost of attracting selling, and servicing the customer, taking into account the time value of money [2]. Other classic definitions of LTV specify additional factors or groups of factors affecting customer value, e.g. retention rate and different kinds of revenue and costs [3]:

- *Retention rate* refers to the probability that a customer remains loyal to the company and keeps yielding expected revenue and costs within a fixed period of time. The retention rate can be established with the help of such determinants of loyalty as customer satisfaction, variety-seeking behavior, attractiveness of alternatives, and switching barriers.
- *Revenue* can be classified into four subcategories: "autonomous" revenue, up selling revenue, cross selling revenue, and contribution margins. "Autonomous" revenue means the revenue which is not directly influenced by the company or is only influenced by standard marketing measures like advertising. Up selling revenue is obtained by the additional selling of the same products as a consequence of increased purchase frequency and a price effect. Cross selling revenue is obtained by the selling of complementary products which have not been bought by the customer before. Contribution margins result from new customers who were won through referral activities of the customer.
- *Costs* can be divided into costs of the customer acquisition, marketing costs, and sales costs. Costs of the customer acquisition are calculated depending on the acquisition procedure used (e.g. direct marketing). Marketing costs include costs of customer retention and development (e.g. costs for sending catalogues or personalized greeting cards), as well as recovery costs. Sales costs include the production costs of the goods sold to the customer and costs of serving the customer.

It is often emphasized that customer value should include not only economic aspects, but also aspects difficult to express with financial measures which are generated as a result of a successful customer-company relationship [4]. Examples include references provided by prestigious customers, customers' recommendations of the company leading to attracting new customers, learning from customers (e.g. through information system implementation), and innovations (e.g. research collaboration and early product testing).

Furthermore, for some industries customer value may be affected by geographical, demographic, or social data, such as the customer's gender, age, education, the structure of the family, etc.

Taking into consideration all relevant aspects of customer value is not a trivial task for a company and has rarely been applied in practice yet. In the meantime, a very effective and practical way of computing customer value is *Recency-Frequency-Monetary* value (RFM) analysis, based on the customer's behavioral data observed throughout the entire life of his/her transactions with the company. These data concern three aspects of the customer's buying behavior [5, 6]: recency, frequency and the monetary value of the customer's past purchases.

1. *Recency* means the time interval from the customer's last purchase until now. The smaller the recency value, the higher the probability of making another purchase.
2. *Frequency* means the total number of times the customer has made a purchase. A bigger value of frequency indicates a bigger customer loyalty.
3. *Monetary* value means the total amount of money spent by the customer. The bigger the monetary value, the more valued customer is for the company.

After performing a customer database segmentation according to RFM analysis, each customer is described with recency, frequency and monetary codes. From a simple or weighted combination of these codes, a single *Customer Value Score* (CVS), can be derived for each customer. In different types of business individual three components may be of different importance for CVS, so the choice of the corresponding weights should be made by loyalty consultants or people who have a really deep insight into the company's business.

Sometimes, when a company offers a variety of products with different margins, RFM analysis is broaden to RFM-P analysis, where P means a Product category or Profitability [4]. Another extension of RFM analysis is based on a more sophisticated use of recency, called the Velocity Index (VI) [7]. The VI defines the recency relative to the average lag between transactions of a customer. It involves adding a statistical distribution around the velocity index in the formula for CVS and calculates the probability

that a particular transaction stream will continue.

To sum up this section, identifying key factors affecting customer value and defining it are basic and essential steps towards effective customer relationship management in a company. Traditional ways of using customer values concern marketing strategies. We argue that in the case of a B2C activity, the effort a company has made to apply a CRM strategy can be thwarted by First-In-First-Out (FIFO) scheduling policy, which is commonly used to process customers' requests in a Web server system hosting a B2C Web site. That is why we propose broadening the use of customer value to an automated process of request service in a retail Web store and integrating it into a novel admission control and scheduling algorithm, implemented in the Web server system instead of FIFO algorithm.

The need for customer differentiation in retail Web stores

We consider a retail Web store, where Internet users are customers browsing and buying goods through a B2C Web site. The site consists of many Web pages, each of which can be assigned to one of typical "business" operations, such as browsing and searching goods, adding them to a virtual shopping cart, paying etc. During a single visit to the site a user accesses many Web pages one after another, so he/she performs a sequence of operations, which makes up a *user session*.

Communication between a user and a Web site within a session is realized according to the idea of the client-server network processing and is based on HyperText Transfer Protocol (HTTP). The client is a user's Internet browser which sends HTTP requests through the Internet to the Web server. The server generates replies to arriving requests and sends them back to the client.

Web servers for e-commerce sites are usually complex systems, where request processing is realized at three logical software layers: Web server, application server, and database server layer (they are so-called multi-layer applications).

The first layer consists of one or more Web servers which are in charge of admitting incoming HTTP requests and starting further appropriate site processing functions to serve them. Web servers handle requests for static Web objects by fetching requested files from disk or cache, while request for dynamic Web objects are forwarded to a back-end dynamic content generator in the second layer.

The second layer, supported by application servers, realizes business logic-oriented application tasks. It is responsible for generating dynamic content through execution of the corresponding code online, typically accessing a database in further software layer. It assembles and formats the resulting content into an HTML page and passes it back to the Web server layer.

The third layer, supported by database servers, is devoted to database functions when required.

Web, application and database servers together constitute a *Web server system*. Software at individual logical layers may run on separate machines, on clusters of machines, or on a single machine. In this research, we consider the situation when application and database functions are combined together in background server(s) that is acceptable in many practical developments (Fig. 1).

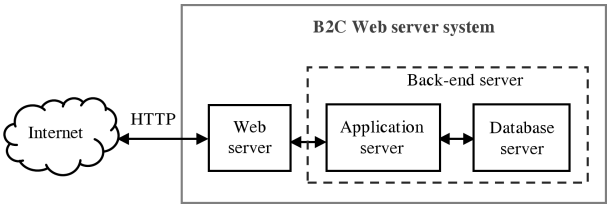

Fig. 1. Multi-layer configuration of a B2C Web site.

For each single Web page request issued by the user, the Internet browser automatically generates multiple separate HTTP requests and submits them to the Web server: the first hit is for an HTML document and the following hits are for Web objects embedded in it. When the browser receives all page's objects, the page is completed and displayed in a browser window.

Many users may interact with a B2C Web site simultaneously and they do not see one another. A Web server system processes many HTTP requests concurrently, and handles a queue of incoming requests according to FIFO policy without any differentiation. Usually the system capacity is sufficient to effectively handle incoming requests. However, it is not uncommon situation when a volume of traffic coming to the server suddenly exceeds its capacity and the system becomes overloaded. Such situations may be observed e.g. in retail Web stores especially during very busy days before Christmas every year.

The server overload results in long request response times and dropped requests. Consequently, e-customers experience very long page response times or do not get a requested Web page at all. Many results showed that such low quality of service (QoS) has very negative consequences for e-business [8–10]. Low efficiency of a Web site leads to a weak company image and discredits Web site security in the eyes of customers. As a result, customers are less

likely to make a purchase during the current visit and to return to the Web store in the future. What's more, customer perception of the received QoS affects not only the likelihood of going to competitors' Web sites, but also the customer opinion of the company's products and the company itself [8]. It has been also shown that page latency (along with other related factors like shopping enjoyment and trust) highly influences customer loyalty and online purchase intentions [11, 12].

The aforementioned observations motivated us to focus on a QoS-enabled Web server system for a B2C application, and to extend it with the ability to offer differentiated page response times with respect to customer values. Although it is not possible to offer high QoS to all customers under the system overload, one can distinguish the most valuable, loyal customers and give them a precedence of service before less valued ones, including other relevant performance criteria.

Admission control and scheduling algorithms for B2C Web server systems

Improving QoS of a Web server system can be achieved by replacing FIFO scheduling by an admission control and scheduling algorithm.

Admission control is to prevent Web server system overload by rejecting some of incoming HTTP requests when the system load exceeds some threshold. In the case of B2C applications, admission control usually is to support user session integrity. It means that a decision on acceptance or rejection of a request is taken only for requests opening new sessions, while all requests belonging to sessions in progress are accepted [13–16].

Scheduling concerns reordering a queue of requests to server input or queues to server resources, such as the CPU, disk, or network interface. Some approaches make use of Web traffic heterogeneousness implying differences in request service times, e.g. they apply Shortest Job First (SJF) scheduling [17] or Shortest Remaining Processing Time (SRPT) scheduling [18] to shorten mean request response time. Other solutions are based on request prioritization and apply the priority-based scheduling according to various criteria. Common scheduling criteria are to minimize the mean request response time of the Web server system and to maximize the system throughput.

Beside general solutions for QoS-enabled Web server systems there has been a lot of research on specific Web applications, such as streaming audio and video, distance-learning, e-banking, and e-commerce.

In particular, admission control and scheduling algorithms for B2C Web sites have been targeted at maximizing revenue obtained by a business owner. To this end, request priorities have been assigned based on some characteristics of a user session: the session state, corresponding to a kind of a business operation at the site [13, 14, 19–23], probabilities of transitions between the session states [14, 21], the session length, and the financial value of goods in a shopping cart [19]. The admission control algorithm proposed in [16] has considered the aspect of purchase repeatability by using information on the correlation of the request sender's IP address with the fact of making a purchase in the past.

To the best of our knowledge, none of QoS mechanisms for a B2C Web server system have characterized the most valued customers or applied priority scheduling for them. We address this problem in combination with the problem of ensuring high revenue. Our work complements and extends the previous studies mentioned in this section by applying customer values to B2C Web server system control.

A novel approach integrating customer value into a QoS mechanism for a B2C Web server system

Storing customer values on the Web server

Once an e-tailing company calculated their customers' values, this information may be stored in a *customer database* on the Web server. In fact, the customer database has to be periodically updated to reflect the actual information on the customers.

We propose applying RFM analysis to quantify customer values, as described in [24]. Strong arguments for this method are its simplicity, low cost, possibility of the automated gathering of necessary data, and lack of necessity to perform complex computations, what is crucial to real-time Web-based systems. RFM analysis does not require collecting economic, social or demographic information on customers, since data required to calculate recency, frequency and monetary codes can be easily captured by an observation of customer purchasing behavior.

For each customer who has made at least one purchase in the Web store, there is a corresponding record in the customer database. Each record contains the following purchase history data: a date of the customer's last purchase, a counter for the total number of times the customer has made a purchase, and a counter for the total amount of money spent

by the customer. These values are updated after each successive purchase made by a customer. Furthermore, each record contains data resulting from database segmentation according to RFM analysis: recency, frequency and monetary codes, which are updated at each database segmentation. Lastly, the value of a CVS is stored in the customer record and periodically updated at each database segmentation. CVS is calculated as follows:

$$CVS = w_R \times R + w_F \times F + w_M \times M, \qquad (1)$$

where R, F, M are codes of the corresponding behavioral variables for the customer, i.e. recency, frequency and the monetary value, respectively, and w_R, w_F, w_M are weights assigned to the corresponding behavioral variables according to the business strategy of the company.

Model of request service control in a B2C Web server system

We propose extending a Web server system with two functions: the ability to ensure successful interaction for all customers who are likely to make a purchase and the ability to identify the most valued customers to offer the best possible QoS to them. The proposed method of request service is called KARO (*Key customers And Revenue Oriented admission and scheduling*), and the appropriate admission control and scheduling algorithm is called KARO-Rev (*KARO – Revenue-oriented*).

We consider a B2C Web server system organized as a multi-layer application (cf. Fig. 1), which model consists of a single Web server and a single back-end server (Fig. 2). We distinguish two types of requests in the system, HTTP requests and dynamic requests.

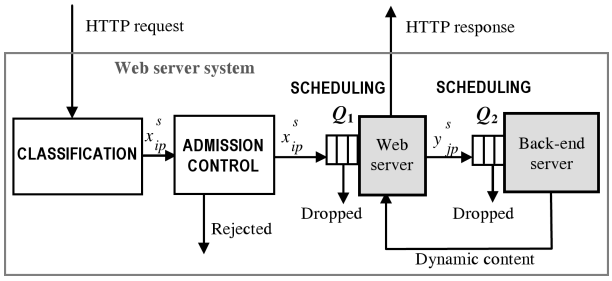

Fig. 2. Model of request service in a Web server system.

HTTP requests come to the system from the outside. Each i-th HTTP request is classified in the context of belonging to the p-th Web page in user session s and it is denoted by x_{ip}^s. It is also characterized by its kind, $k_i \in \{0, 1\}$, where:

$$k_i = \begin{cases} 1 \text{ for an HTML page skeleton} \\ 0 \text{ for an embedded object} \end{cases}. \qquad (2)$$

Introducing admission control means that under the system overload some HTTP requests may be *rejected*, i.e. they will not be processed. Accepted HTTP requests are put into a queue Q_1 in front of the Web server. A request may receive service from the Web server at once or may wait for it for some time, no longer than a predefined threshold T_1. If a request's waiting time exceeds T_1, the request will be *dropped* from the queue.

Dynamic requests are generated by the Web server and sent to the back-end server if a response to an HTTP request has to be created online, i.e. when it requires online computations and a database access. Each dynamic request is indexed with the dynamic request number j ($j \leq i$), the session number s, the page in session number p, and is denoted by y_{jp}^s. All dynamic requests are put into a queue Q_2 in front of the back-end server. The maximum request's waiting time, after which a request will be dropped from the queue, is specified by a predefined threshold T_2.

Values T_1 and T_2 should be determined in relation to the Internet user page latency limit T_u, which reflects maximum page response time, which users are likely to tolerate.

Rejecting or dropping an HTTP request x_{ip}^s, or dropping a dynamic request y_{jp}^s, means that the p-th page in session s will not be completed and displayed in a customer's Web browser. We assume that in such a situation the customer retries to download the Web page. The maximum number of user retries for a page is limited by Γ_u.

Let n be the moment of a new HTTP request arrival to the Web server or a moment of a new dynamic request arrival to the back-end server. Request service control in the Web server system consists of taking decisions at successive moments n concerning: (1) HTTP request admittance or rejection, (2) scheduling of admitted HTTP requests in the queue Q_1, and (3) scheduling of dynamic requests in the queue Q_2. The control decisions are based on business-oriented criteria, connected with potential revenue and customer value.

Furthermore, let m be the moment of HTTP request's rejection, dropping or successful completion in the system, when some output values are observed.

At any moment n some number of users are interacting with a B2C Web site, so there are $S(n)$ active user sessions in the system. Depending on the current system load, each session may be finally successfully completed or aborted.

Def. 1. Session s is considered to be *successfully completed* at the m-th moment, if an HTTP request x_{ip}^s completed at this moment has been the last request of session s and the response time for

page p has not exceeded user page latency limit T_u (such a situation means that all previous pages in session s have also been completed within the page latency limit because session s had not been aborted earlier). Even if a customer had any item in the shopping cart and gave up the interaction for unknown reasons, other than too long page response time, the session is considered to be successfully completed.

Def. 2. Session s is considered to be aborted at the m-th moment if one of the following occurs: (1) a request belonging to the p-th page in session s has been rejected or dropped at this moment and the number of customer's retries for page p had just achieved the threshold Γ_u (such a situation means that page p has not been completed); (2) after completing an HTTP request x_{ip}^s at the m-th moment, the response time of page p exceeds user page latency limit T_u and the number of user's retries for that page had just achieved the threshold Γ_u (in such a situation we assume that the customer gave up the interaction due to poor QoS).

In order to evaluate the importance of active user sessions in terms of business-oriented goals, we propose characterizing each session s, $s = 1, 2, \ldots, S(n)$, by the following vector:

$$\boldsymbol{V}^s = [c^s(n), RFM^s(n), e^s(n), v^s(n), l^s(n)], \quad (3)$$

where the vector's components correspond to the following session attributes:

1. $c^s(n)$ means the *session class*, $c^s(n) \in \{KC, OC\}$. We define two session classes: KC class for key customers, characterized by some purchase histories, and OC class for ordinary customers. We assume that every customer starts the session as an ordinary customer and may be identified as key after logging on.
2. $RFM^s(n)$ means the *session rank*, which reflects the customer value, $RFM^s(n) \in \{0, r_{\min}, \ldots, r_{\max}\}$. For KC class the session rank corresponds to CVS of the customer read from the customer database; thus, the possible session rank values, ranging from r_{min} to r_{max}, depend on the formula for customer value, i.e. (1) in our case. For OC class the session rank is assumed to be zero.
3. $e^s(n)$ means the *session state* corresponding to a type of business operation being performed at the B2C Web site, $e^s(n) \in \{H, L, B, S, D, A, R, P\}$. We define the following session states: H – entry to the Home page, L – Logging on, B – Browsing, S – Searching for products, D – product's Details, A – Adding a product to a shopping cart, R – Registration, and P – making a Payment.
4. $v^s(n)$ means the *value of a shopping cart*, computed as an amount of money (in \$) corresponding to prices of products in the shopping cart.
5. $l^s(n)$ means the *session length*, $l^s(n) = 1, 2, \ldots$, which is the number of Web pages visited in the session so far, including the current page.

In order to support differentiated QoS in the Web server system, we propose introducing dynamic *session priorities* based on sessions' attributes and applying priority-based request admission control and scheduling. Priority-based approach to the request service in a Web server has been successful in guaranteeing differentiated levels of service. Our session priority scheme is based on the method [19].

Let session priorities range from 1 to 4, where the higher the priority value, the higher the priority is. Thus, the value of 4 corresponds to the highest possible priority, while the value of 1 means the lowest possible priority. The priority of session s changes with the session progress depending on its attributes contained in vector $\boldsymbol{V}^s$.

Let $P^s(n) \in \{1, 2, 3, 4\}$ be the priority of session s at the n-th moment. We define two thresholds, T_{Med} and T_{Low} ($T_{Med} < T_{Low}$), which determine three ranges for the session length. The priority of session s at the n-th moment is calculated according to the formula:

$$P^s(n) = \begin{cases} 4 \text{ for } (c^s(n) = KC) \vee [(c^s(n) = OC) \\ \quad \wedge (v^s(n) > 0) \wedge (e^s(n) = P)], \\ 3 \text{ for } [(c^s(n) = OC) \wedge (v^s(n) > 0) \\ \quad \wedge (e^s(n) \neq P)] \vee [(c^s(n) = OC) \\ \quad \wedge (v^s(n) = 0) \wedge (l^s(n) < T_{Med})], \\ 2 \text{ for } (c^s(n) = OC) \wedge (v^s(n) = 0) \\ \quad \wedge (T_{Med} \leq l^s(n) < T_{Low}), \\ 1 \text{ for } (c^s(n) = OC) \wedge (v^s(n) = 0) \\ \quad \wedge (l^s(n) \geq T_{Low}). \end{cases} \quad (4)$$

The priority 4 is reserved for all key customers, as well as for ordinary customers finalizing their purchase transactions. The priority 3 is assigned to all customers arriving to the site (to give a chance of successful interaction to all users and to allow key customers to log into the site), as well as to ordinary customers with some items in their shopping carts. Priorities 2 and 1 are assigned to ordinary customers who have stayed at the site for a long time with empty carts.

A session priority is verified and updated only at arrivals of HTTP requests for new Web pages, i.e. for HTML page skeletons. Thus, all HTTP requests for objects embedded in a page and all corresponding dynamic requests have the same priority.

The session priorities are used in an admission control and scheduling algorithm KARO-Rev.

KARO-Rev algorithm realizing request service control in a B2C Web server system

In this section, we present a heuristic request admission and scheduling algorithm KARO-Rev.

To enforce admission control decisions, we define two thresholds for the system load intensity, I_1 and I_2, where $I_1 < I_2$. When the load exceeds the first admission control threshold, I_1, the system will reject HTTP requests for new Web pages in sessions with priority 1. Above the second threshold, I_2, HTTP requests for new Web pages both in sessions with priority 1 and 2 will be rejected. All requests for embedded objects are always accepted regardless of the session priority, as well as all requests for new Web pages in sessions with priority 3 and 4.

We apply priority scheduling in the queues Q_1 and Q_2, which makes it possible to change the order of request execution at the Web and back-end server, respectively. For each queue, strict priority scheduling is applied between four priorities, i.e. all higher-priority requests are scheduled before the lower-priority ones. KARO-Rev algorithm is oriented towards revenue maximization, and the additional performance criterion is connected with customer values. Therefore, requests with priority 4 are ordered decreasingly according to shopping cart values, and requests from sessions with the same shopping cart values are additionally ordered decreasingly according to session ranks. Requests with priority 3 are ordered decreasingly according to shopping cart values, while requests with priorities 2 and 1 are queued according to FIFO order.

When a new request arrives at the system, the order of requests waiting in the corresponding queue does not change, but a position for a new request is determined in the following way.

Let a^{s_a} be a request a belonging to session s_a, waiting in a queue Q_k, $k \in \{1,2\}$. We define the following sets and subsets of requests waiting in the queue Q_k at the n-th moment, when a new request belonging to session s arrives:

$$Q_k(n) = \{a^{s_a} \in Q_k\}, \tag{5}$$

$$Q_{k\_2}(n) = \{a^{s_a} \in Q_k : P^{s_a}(n) \in \{2,3,4\}\}, \tag{6}$$

$$\begin{aligned} Q_{k\_3}(n) = \{a^{s_a} \in Q_k : (P^{s_a}(n) = 4) \\ \vee[(P^{s_a}(n) = 3) \wedge (v^{s_a}(n) \geq v^s(n))]\}, \end{aligned} \tag{7}$$

$$\begin{aligned} Q_{k\_4}(n) = \{a^{s_a} \in Q_k : (P^{s_a}(n) = 4) \wedge \\ [(v^{s_a}(n) > v^s(n)) \vee [(v^{s_a}(n) = v^s(n)) \\ \wedge (RFM^{s_a}(n) \geq RFM^s(n))]]\}. \end{aligned} \tag{8}$$

Requests in a queue Q_k are ordered and executed according to the request position numbering. The beginning of the queue is determined by number 1 and its end at the n-th moment is determined by number $|Q_k(n)|$, where $|\bullet|$ means cardinality of queue Q_k at the n-th moment. We assume that the length of queue Q_k at this moment does not change.

The position number $z_k(n)$ in a queue Q_k, determined for a new request having arrived at the n-th moment ($k = 1$ for an admitted HTTP request, and $k = 2$ for a dynamic request), is calculated according to the following formula:

$$z_k(n) = \begin{cases} |Q_k(n)| + 1, \text{ if } P^s(n) = 1, \\ |Q_{k\_2}(n)| + 1, \text{ if } P^s(n) = 2, \\ |Q_{k\_3}(n)| + 1, \text{ if } P^s(n) = 3, \\ |Q_{k\_4}(n)| + 1, \text{ if } P^s(n) = 4. \end{cases} \tag{9}$$

A pseudocode of KARO-Rev algorithm, executed at the n-th moment, is presented in Fig. 3.

Algorithm KARO-Rev

if (a new HTTP request x_{ip}^s)
  if (the request concerns an HTML page skeleton)
    update session_s attributes: $c^s(n)$, $RFM^s(n)$, $e^s(n)$, $l^s(n)$, $v^s(n)$;
    update session_s_priority $P^s(n)$;
    //admission control
    if [(threshold I_1 ≤ system_load < threshold I_2)
        and ($P^s(n)$ == 1))
      or ((system_load ≥ threshold I_2)
        and (($P^s(n)$ == 1) **or** ($P^s(n)$ == 2))]
      reject the request;
    else
      accept the request;
    end if
  else //the request concerns an embedded object
    accept the request;
  end if
  //scheduling in queue Q_1
  if (the request has been accepted)
    put the request into the queue Q_1 at the position determined according to (9);
  end if
else if (a new dynamic request y_{jp}^s)
  //scheduling in queue Q_2
  put the request into the queue Q_2 at the position determined according to (9);
end if

Fig. 3. A pseudocode of KARO-Rev algorithm.

Simulation experiments setup

Using a simulation approach, we have evaluated the efficiency of a Web server system under the proposed KARO-Rev algorithm, and compared its performance to the performance of the same system under FIFO scheduling. To this end, we have designed and developed a simulation environment [25],

and used it to carry out experiments for representative B2C-oriented Web workload.

Our simulation tool consists of a session-based workload generator and a B2C Web server system simulator. The simulation program was written in C++ using CSIM19 package [26].

We have designed three workload scenarios, differing in values of two key parameters: the user page latency limit T_u and the percentage of key customer sessions Δ_{KC} in the generated workload. First, *typical scenario* emulates the most common workload conditions, where $T_u = 8$ s and $\Delta_{KC} = 10\%$. Second, *4s scenario* emulates the behavior of customers with a more restrictive page latency limit of 4 s, while $\Delta_{KC} = 10\%$. Third, *40% KC scenario* is to mimic a B2C Web site with a large share of key customer sessions in order to evaluate how KARO-Rev copes with favoring a large number of key customers interacting simultaneously with the site; $T_u = 8$ s.

Due to lack of data from a real online retailer we assumed that three weights assigned to behavioral variables are equal to 3, which gives CVSs ranging from 9 to 45. Such a range is large enough to differentiate between key customers characterized by various customer values, and allows evaluating relative levels of their QoS without going into business-specific details.

Other parameters of KARO-Rev, chosen based on our simulation model and the results of preliminary simulation experiments have been the following: $T_1 = T_2 = T_u = 8$ seconds, $\Gamma_u = 0$, $T_{Med} = 2$, $T_{Low} = 20$, $I_1 = 30$, $I_2 = 80$.

Business-oriented performance metrics

We present some results concerning the Web server system performance in terms of two performance metrics: the percentage of achieved potential revenue and the 90-percentile of page response time for different customer values.

First, *potential revenue* is defined as the total financial value of products in shopping carts of the sessions which had ended with a purchase or had been aborted in the observation window. *Achieved potential revenue* is defined as the total financial value of products in shopping carts of the sessions ended with a purchase. Then, the *percentage of achieved potential revenue* is defined as the percentage of potential revenue which turned into actual revenue. It gives the information on how effectively the system has processed sessions with goods in shopping carts.

QoS is expressed as the 90-percentile of page response time. First, *request response time* is defined as time needed by the system to complete a single HTTP request. Then, *page response time* is defined as time needed by the system to complete a whole Web page, i.e. all HTTP requests for that page. Page response time t_p^s for the p-th page in session s is computed according to the following formula:

$$t_p^s = \sum_{x_{ip}^s \in O_p^s} t_i^s, \tag{10}$$

where t_i^s is request response time provided by the system to the i-th HTTP request belonging to session s, x_{ip}^s is the i-th HTTP request belonging to the p-th page in session s, O_p^s is a set of Web objects making up the p-th page in session s. Page response time is computed only for successfully completed pages.

The *90-percentile of page response time* is defined as P(page response time $\leq Y$) = 90; e.g. if 90-percentile of page response time is equal to 4 s, it means that in 90% of the system observations the page response time is less than 4 s.

Simulation results

Each single simulation experiment was run for a constant session arrival rate, i.e. for a constant number of new user sessions initiated per minute. The system performance was monitored in a 3-hour observation window following a 10-hour preliminary phase (according to CSIM19 simulation time).

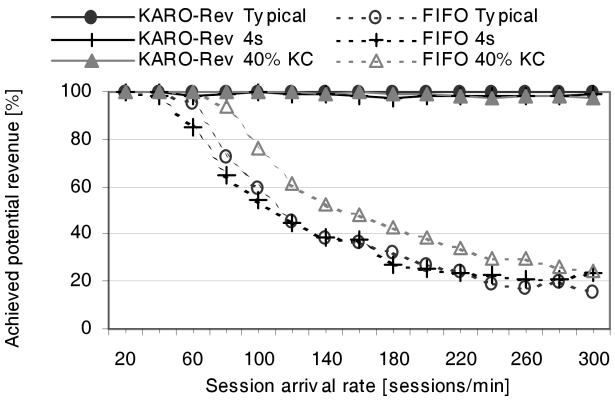

Fig. 4. Percentage of achieved potential revenue.

Figure 4 presents a variation of the percentage of achieved potential revenue as a function of the session arrival rate. As it can be seen, for FIFO scheduling this percentage starts to decrease just above 40 new sessions per minute, which indicates that as the system load increases, more and more sessions with products in shopping carts are aborted and thus, larger potential revenue is lost. On the contrary, for KARO-Rev this percentage is fairly stable regardless of the system load. Even with 40% of *KC* sessions at the site the algorithm has been able to ensure 97.76–

100% of achieved potential revenue throughout the whole load range.

Figure 5 shows a variation of the 90-percentile of page response time as a function of the session arrival rate. In all cases KARO-Rev has been able to achieve the 90-percentile of page response time of less than 2 s. This percentile for FIFO was much higher, up to 6.4–13.7 s for the maximum load, depending on the workload scenario; in many cases it significantly exceeded the 8-seconds threshold that is used as a latency measure acceptable to users.

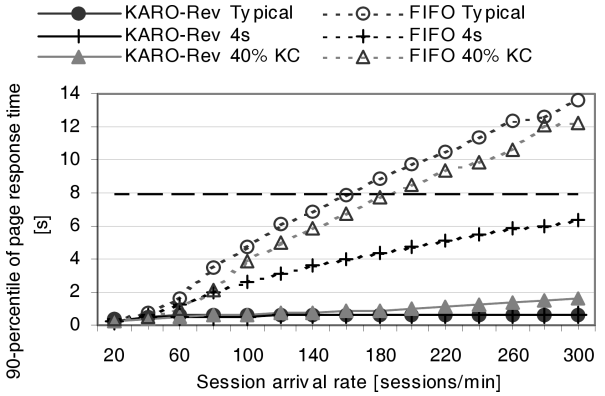

Fig. 5. 90-percentile of page response time for key customers.

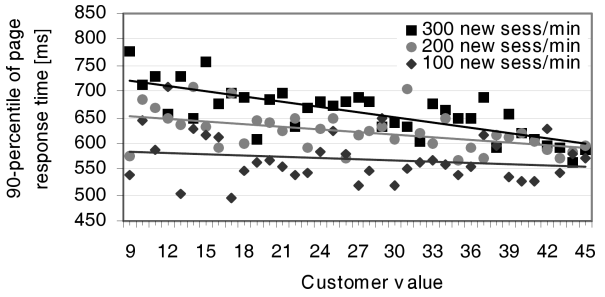

Fig. 6. 90-percentile of page response time for different customer values (*Typical* scenario).

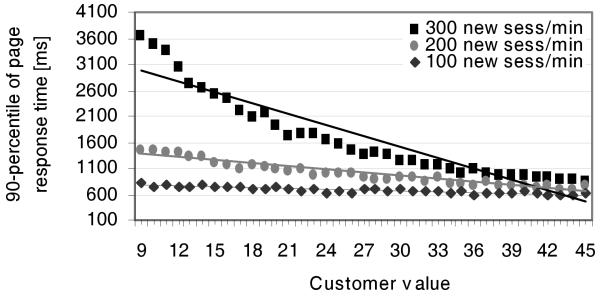

Fig. 7. 90-percentile of page response time for different customer values (*40% KC* scenario).

Figures 6 and 7 present the 90-percentile of page response time for different customer values, i.e. for different session ranks, for *typical* and *40% KC* workload scenarios, respectively. Results are presented for three load levels, i.e. for the session arrival rates of 100, 200, and 300 new sessions per minute. Trend lines in the figures indicate that the system tended to offer lower page response times to higher customer values. The higher the system load and the more *KC* sessions, the greater the QoS differentiation in terms of page response times between customers with different values.

The simulation results have shown the effectiveness of KARO-Rev algorithm both in minimizing potential revenue losses and differentiating QoS levels offered to key customers with respect to their values. Furthermore, other results showed its ability to ensure significant improvements in overall system throughput and page response times for all sessions under overload.

Conclusions

The literature analysis has revealed that a customer-focused strategy may significantly contribute to the e-business profitability and there is a need for the preferential handling of the most valued customers of Web stores. We propose to broaden a customer-oriented strategy to a Web server system hosting a B2C Web site, which is a mediator between the company and customers.

To this end, we have proposed applying RFM analysis to automatically calculate customer values and using them in a novel method of request service control in a Web server system. Simulation results have shown that the proposed KARO-Rev algorithm was successful in providing differentiated QoS levels for customers characterized by different values, and in minimizing amount of potential revenue lost due to the server overload.

Our future work will concern applying other ways of calculating customer values in KARO method, as well as further investigation on using computer technologies to support CRM.

References

[1] Reichheld F.F., Schefter P., "E-Loyalty: Your Secret Weapon on the Web", Harvard Business Review, 78, 4, 105–113, July – August 2000.

[2] Kim S.-Y., Jung T.-S., Suh E.-H., Hwang H.-S., "Customer Segmentation and Strategy Development Based on Customer Lifetime Value: A Case Study", Expert Syst. Appl., 31, 1, 101–107, July 2006.

[3] Bauer H.H., Hammerschmidt M., Braehler M., "The Customer Lifetime Value Concept and its Contribu-

tion to Corporate Valuation", Yearbook of Marketing and Consumer Research, 1, 2003.

[4] Bazarnik J., „Szacowanie wartości klienta", ZN AE w Krakowie, 694, Kraków, 2006.

[5] Hughes A.M., "Making Your Database Pay Off Using Recency, Frequency and Monetary Analysis", Technical Report, Database Marketing Institute, January 2001.

[6] Miglautsch J.R., "Thoughts on RFM Scoring", The Journal of Database Marketing, 8, 1, 67–72, August 2000.

[7] Ferguson R., "Using Loyalty Analytics to Grow from Data Infancy", Colloquy, July 2003.

[8] Bhatti N., Bouch A., Kuchinsky A., "Integrating User-Perceived Quality into Web Server Design", Computer Networks, 33, 1–6, 1–16, June 2000.

[9] Silverstein M., Stanger P., and Abdelmessih N., "Winning the Online Consumer 2.0. Converting Traffic into Profitable Relationship", A report by The Boston Consulting Group, February 2001.

[10] "The Need for Speed II", Zona Market Bulletin, 5, April 2001.

[11] Wang F., Head M., "How Can the Web Help Build Customer Relationships? An Empirical Study on E-tailing", Information and Management, 44, 2, 115–129, March 2007.

[12] Yoo B., Donthu N., "Developing a Scale to Measure the Perceived Quality of an Internet Shopping Site (SITEQUAL)", Quarterly Journal of Electronic Commerce, 2, 1, 31–47, 2001.

[13] Carlström J., Rom R., "Application-aware Admission Control and Scheduling in Web Servers", IEEE INFOCOM, 2, 506–515, June 2002.

[14] Chen H., Mohapatra P., "Overload Control in QoS-aware Web Servers", Computer Networks, 42, 1, 119–133, May 2003.

[15] Kihl M., Widell N., "Admission Control Schemes Guaranteeing Customer QoS in Commercial Web Sites", IFIP NET-CON, 235, 305–316, 2002.

[16] Yue C., Wang H., "Profit-aware Admission Control for Overload Protection in E-commerce Web Sites", IEEE IWQoS'07, 188–193, June 2007.

[17] Cherkasova L., "Scheduling Strategy to Improve Response Time for Web Applications", LNCS, Springer-Verlag, 1401, 305–314, 1998.

[18] Schroeder B., Harchol-Balter M., "Web Servers under Overload: How Scheduling Can Help", ACM TOIT, 6, 1, 20–52, February 2006.

[19] Menascé D.A., Almeida V.A.F., Fonseca R., Mendes M.A., "Business-Oriented Resource Management Policies for E-Commerce Servers", Perform. Eval., 42, 2–3, 223–239, September 2000.

[20] Shaaban Y.A., Hillston J., "Cost-based Admission Control for Internet Commerce QoS Enhancement", Electron. Commerce. Res. Appl., 8, 142–159, 2009.

[21] Singhmar N., Mathur V., Apte V., Manjunath D., "A Combined LIFO-Priority Scheme for Overload Control of E-commerce Web Servers", IISW'04, Lisbon, December 2004.

[22] Totok A., Karamcheti V., "Improving Performance of Internet Services Through Reward-Driven Request Prioritization", 14th IEEE IWQoS, 60–71, June 2006.

[23] Zhou X., Wei J., Xu C.-Z., "Resource Allocation for Session-Based Two-Dimensional Service Differentiation on e-Commerce Servers", IEEE Trans. Parallel Distrib. Syst., 17, 8, 838–850, August 2006.

[24] Borzemski L., Suchacka G., "Discovering and Usage of Customer Knowledge in QoS Mechanism for B2C Web Server Systems", LNAI, Springer, Heidelberg, 6277, 505–514, 2010.

[25] Borzemski L., Suchacka G., "Business-Oriented Admission Control and Request Scheduling for E-Commerce Websites", Cybernetics and Systems, 41, 8, 592–609, November 2010.

[26] CSIM19, Development Toolkit for Simulation and Modeling, http://www.mesquite.com.