Meeting Service Traffic Requirements in SOA

Keerthana Boloor\textsuperscript{1,2}, Bob Callaway\textsuperscript{2}, Marcelo Dias de Amorim\textsuperscript{3}, Adolfo Rodriguez\textsuperscript{2}, and Yannis Viniotis\textsuperscript{1,2}

\textsuperscript{1}NC State University \quad \textsuperscript{2}IBM Websphere \quad \textsuperscript{3}UPMC Univ Paris 06

Abstract—Enterprise networks may host multiple services, located in (potentially physically distinct) servers; an example would be Web server farms using Web 2.0 and/or SOA-based technologies. A service is governed by a client service contract (CSC) that specifies, among other requirements, the rate at which the service may be accessed so that a particular service host is not unduly overwhelmed. We consider the case where the CSC limits the rate to a service provider to no more than $X$ requests per second, with an observation/enforcement time interval of $T$ seconds. In this paper we define, simulate, and analyze CASTS (Credit-based Algorithm for Service Traffic Shaping), a service traffic shaping algorithm to meet the service access requirements (SAR) under the above constraints. Our analysis includes convergence times, responsiveness, and the verification whether the system respects the service contract.

I. INTRODUCTION

Global access to services has been one of the major advances in the information technology domain in the latest years. In this context, service-oriented architectures (SOA) have appeared as the main paradigm to satisfy business goals in terms of flexibility and integration of legacy systems [1], [2], [3]. A way of realizing SOA is through the concept of web services, which rely on the service concept to deliver services on the web using technologies such as eXtensible Markup Language (XML) [4], Web Services Description Language (WSDL) [5], SOAP [6], and Universal Description, Discovery, and Integration (UDDI) [7].

One of the major obstacles preventing wider adoption of Web services is performance [8], [9]. The slow response time for web services is due in part to XML parsing. For this reason, service providers adopt specific hardware (called SOA appliances) that provide accelerated XML parsing [10]. Another role of the SOA appliances, along with enabling security and integrating with legacy services, is to control the flow of documents that are sent to the service hosts. A service is governed by a client service contract (CSC) that specifies, among other things, the rate at which the service may be accessed so that a particular service host is not unduly overwhelmed. Typically, service clients access services from multiple entry points (and, consequently, multiple SOA appliances). The existence of multiple entry points may be dictated by policy (e.g., the presence of multiple security zones), fault tolerance, or performance requirements (e.g., clusters of SOA appliances).

The CSC may define the shaping requirement as follows: “Limit the rate to a service provider to no more than $100$ requests per second, with an observation/enforcement time interval of $10$ seconds.” The main challenge is to enforce the traffic contract by taking local actions at each appliance.

To the best of our knowledge, when the number of entry points (noted $B$) is greater than one, the typical solution is a manual and static allocation (MSA) of the “global” shaping requirement $X$ to the $B$ entry points: each appliance is simply allowed to send “$X/B$ requests per second” within the observation period. If traffic loads at each entry point are fairly static and known in advance, allocations other than the abovementioned one are also possible. In both cases, leaky buckets are used locally to shape traffic.

We propose that SOA appliances rely on their communication capacity to coordinate and better respond to the requirements defined by the service contract. We propose CASTS (Credit-based Algorithm for Service Traffic Shaping), an algorithm that computes the maximum rate at which an appliance can send traffic within some time interval. CASTS relies on a strategic division of the observation period in time slots that serve both as a measurement and enforcement period. The traffic sent by an appliances depends on its activity relatively to the other appliances. The amount of credits an appliance obtains is adjusted at each subperiod, which guarantees the adaptability of the system.

We developed a simulator to evaluate the performance of the proposed scheme. We show that CASTS strictly respects the requirements defined in the service contract. The results comfort the expectation that CASTS outperforms the static approach when the input traffic is unevenly distributed.

As a summary, the contributions of our work are:

1) We identify the need for a measurement-based, dynamic allocation mechanism for the CSC enforcement problem.
2) We propose a reactive algorithm that dynamically adapts to the current state of the system on a per-appliance fashion.
3) We underline the fact that implementation artifacts might have an impact on the efficiency of the system. In particular, because service contracts are ill-defined (in terms of documents per second instead of data units per second), the system might be under-utilized when approximation functions (e.g., flooring) are applied.

II. PROBLEM FORMULATION

First of all, let us describe the network elements of interest in a Web farm. In this paper, we consider the multipoint-to-point case. This means that $B$ SOA appliances send documents to a single service host. Individual appliances must control that output rate of documents (based on some strategy) in order to meet the requirements specified in the contract.

The global picture of the system is illustrated in Fig. 1. The elements composing the network are:
Fig. 1. The general architecture of the system.

- **Clients.** Nodes that are generators of requests. For the purposes of this work, we make abstraction of the individuality of these nodes and assume that these requests are later grouped by the gateway, as indicated in Fig. 1.
- **Gateways.** These are the end-point routers of TCP connections. Gateways are responsible for forwarding requests to the different entry-points (or appliances), eventually performing some sort of load balancing. Our work does not focus on the operation of the gateways. Instead, we assume that requests arrive through the different entry points at some (uncontrolled) rate.
- **Appliances.** These nodes, noted $B_i$ in Fig. 1, are responsible for translating XML documents into the local system language. They are also responsible for controlling the rate at which documents are forwarded to the service host.
- **Service host.** This entity is responsible for effectively handling the requests. It also defines the rules in the service contract that will drive the operation of the appliances.

Let us first provide the necessary background before we introduce our algorithm in Section III. A typical contract specifies two parameters: the maximum number of documents $X$ and the enforcement period $T$. We refer to this specific aspect of the service contract as the service access requirements (SAR), defined as:

$$\textbf{[SAR]}: \text{Limit the rate to a service provider to no more than } X \text{ requests per second, with an observation/enforcement time interval of } T \text{ seconds.}$$

Note that such a contract does not specify any further requirements such as a maximum burst size or a maximum size of the documents. When there is a single entry point, shaping client traffic to meet a CSC is a well-studied problem. Solutions abound in the networking space (e.g., in packet/ATM networks). They are typically variations of a leaky-bucket shaping algorithm [11], implemented typically in hardware. The problem can then be addressed as a traffic shaping problem.

The “service traffic shaping” problem described above is fundamentally different from the well-studied network traffic shaping problem [12], [11], [13]. The main differences are:

1. In the packet/ATM world, traffic shaping has local scope, since traffic is in the form of a single connection. In the service shaping environment, service clients access services from multiple entry points. The existence of multiple entry points may be dictated by policy (e.g., the presence of multiple security zones) or performance requirements (e.g., clusters of appliances); the desired effect is “global” shaping. The challenge is to enforce the traffic contract by taking local actions at each entry point.
2. The resource protected by the shaping function is typically link bandwidth and buffer space, the units of which are precisely defined and measurable. Service Level Agreements are standardized by industrial bodies and CSC contracts are very well-defined. In the enterprise networking world, the resource protected by the shaping function is (distributed) CPU processing power; moreover, CSC contracts are not precisely defined and measurable. For example, with “requests” being defined in units of XML documents, CPU processing time is not known exactly.

Let $x_i$ be the number of documents appliance $i$ is allowed to send to the server (we refer to this value as $i$’s credits). Formally speaking, we must guarantee that the cumulated number of documents sent by all appliances must respect:

$$\sum_{1 \leq i \leq B} x_i \leq X \times T. \quad (1)$$

There are different strategies for guaranteeing the specification described in Eq. 1. The simplest one is to apply a static, homogenous policy by assigning the same rate to each appliance at all times:

$$x_i = \left\lfloor \frac{X \times T}{B} \right\rfloor, \quad \forall i \in [1, \ldots, B]. \quad (2)$$

This approach only provides satisfactory performance when the incoming traffic rates at the appliances are the same. In practice, there is no a priori knowledge on the rates at which the appliances receive requests from the clients. Although the gateway performs some load balancing, this does not guarantee that the rate of documents sent to the output queue will be balanced because of the high variability in the delays required for parsing the documents. We must then assume that the rates at which appliances are ready to send documents do not follow a uniform law.

The consequence of such a heterogeneity on the static allocation scheme is that it is likely that some appliances block traffic while others remain idle, especially when the cumulated number of documents handled by the appliances is in the order of (or higher than) $X \times T$. Given the costs of implementing a web farm and the issues inherent to the provision of web services, it is fundamental to design efficient algorithms that improve the overall utilization of the system.

III. A DISTRIBUTED ALGORITHM FOR SERVICE TRAFFIC SHAPING

We now have enough background to introduce the Credit-based Algorithm for Service Traffic Shaping (CASTS). This
Algorithm 1 CASTS algorithm.

Input: \( k \). 

Output: new count \( x_i(k) \).

\[
\begin{align*}
\text{if} & \quad k = 1 \\
\text{then} & \quad x_i \leftarrow \left\lfloor \frac{X}{B} T \right\rfloor \\
\text{else} & \quad x_i(k) \leftarrow \left\lfloor \frac{X}{B} T \right\rfloor + \left\lfloor \frac{Q_i(k)}{\sum_{n=1}^{B} Q_n(k)} \right\rfloor \\
\text{end if}
\end{align*}
\]

Note that this algorithm guarantees that the SAR is respected at all times, as the total amount of credits assigned to the appliances at each subinterval is always smaller or equal to the remaining allowed number of documents:

\[
\sum_{n=1}^{B} x_i(k) \leq X \times T - R(k), \tag{3}
\]

where \( R(k) \) is the total number of documents sent by all appliances during the previous subperiods.

It is important now to discuss the effects of using the flooring function in the algorithms (the same function is used in the static algorithm). Since the service contract specifies the required rate in documents per second within the observation period, each appliance must approximate the number of credits it will use in the next subinterval by an integer value, which can be done either with a flooring or ceiling function. The ceiling function cannot be used because the service contract might not be respected. Because it approximates the credits by the integer immediately below, credits might be wasted in the system. Obviously, the result is suboptimal. In the worst case, the number of remaining credits to be redistributed is \( B - 1 \) and all the appliances have the same queue size. No documents will be sent, although \( B - 1 \) credits are still remaining. In this paper, we focus on the core functionalities of our algorithm and deliberately do not fix the problem of the flooring function, so that this phenomenon can be observed in our graphs. A version of the algorithm that fixes this problem will be subject of future work.

IV. SIMULATION ANALYSIS

The simulation of the single service multi appliance traffic shaping algorithm is implemented as a single threaded discrete event simulator in C. The simulator is modeled with data and model separation. The requests to the appliances are fashioned as a Poisson process. The traffic is modeled either as uniform or bursty with burstiness varied in different intervals of varying duration. The rate of arrivals of the traffic is also varied to be more than \( X \) and less than \( X \). The simulator also permits changing of number of appliances, the threshold \( X \), the number of sub intervals \( K \) and the duration of measurement interval.

Unless otherwise specified, we set in the analysis \( T = 1 \) second. For the purposes of this paper, it does have any influence on the results, as we assume that the time required for the appliances to exchange their status and compute the new amount of credits is \( \ll \frac{T}{K} \). We consider that the amount of documents sent during the observation period \( T \) is \( X = 128 \). Also otherwise specified, the input traffic follows a Poisson process with average \( Y_{in} \). Allowed count is the number of remaining credits, i.e., the number of requests that can be serviced within \( T \). Actual count is the total number of requests services so far.

The first analysis we performed was about the influence of the amount of input traffic on the behavior of the algorithm. We consider a uniform input traffic \( Y_{in} \) under two cases: low rate and high rate. The results are presented in Fig. 3. In both cases, the number of subintervals per observation period is
$K = 20$. In Fig. 3(a), we investigate how the algorithm reacts to an input traffic whose average $Y_{in}$ is equal to the maximum number of documents $X$ defined in the contract. Note that the algorithm automatically reacts to the variations in the number of available credits: most of the time the allowed count plus the actual count is equal or almost equal to $X = 128$. The variations that we observe in the curves are due to either the flooring effect or to the actual number of documents produced ($Y_{in}$ is the average of a Poisson process; in some cases, the number of documents might be $< 128$). In Fig. 3(b), we investigate the behavior of the algorithm under stress, i.e., when $Y_{in} \gg X$ (we set $Y_{in} = 1,000$ documents/s). Note that there is some warmup period which is equivalent to one observation period. This is strictly due to the flooring effect.

The main goal of the shaping algorithm is to enforce the constraints specified in the service contract. Nevertheless, the contract does not specify how good the algorithm must perform. CASTS has been designed both to respect the contract and to achieve good performance. We define performance as the ratio between the number of documents generated and the number of documents sent to the server. The optimal algorithm would lead to $R(K) = \min[X \times T; Y_{in} \times T]$, where $Y_{in} \times T$ is the total number of documents generated within the observation period. Fig. 4 shows the results for CASTS (after five observation periods, i.e., after warmup). Note that the algorithm performs optimally when $Y_{in} \leq X$ and leads to good results for the other values. The main reason it does not achieve the optimum value is the flooring effect.

As explained in Section III, the goal of defining $K$ is to give more chances to the algorithm to react to changes in the behavior of the input conditions. We now evaluate the results for other values of $K$ (1 and 40) and present the results in Fig. 5. It is important to underline that for $K = 1$, the algorithm is equivalent to the static case. For the parameters of the scenario used in our simulations, the algorithm achieves a reasonable responsive behavior when $K = 40$. For higher values, we did not observe any improvement.

Studying the impact of $K$ on the behavior of the algorithm is of major importance, as the higher the number of subintervals, the higher the control overhead. The number of control messages exchanged is linearly proportional to both the number of appliances and the number of subintervals, $O(BK)$. Depending on the length of the observation period and on the size of the documents, the control overhead might compromise the efficiency of the system.

We now perform the same tests but using bursty traffic. The size of the burst is 10,000 packets in average, and follows a Poisson process. The graphs depicted in Figs. 6(a) and 6(b) show the results for, respectively, $K = 1$ and $K = 40$.

We now evaluate the impact of multiple bursts within the observation period. Fig. 7 shows the behavior of our algorithm when $Y_{in} = 128$ documents/s.

We finally evaluate the system when the requests are evenly
distributed among the appliances. This means that all SOA appliances have approximately the same number of documents in the queue (variations occur because the input traffic is not necessarily an integer multiple of $B$). The average input traffic is $Y_{(t)} = 1,000$ requests/s. Except for the first observation period, where we can see the consequences of the flooring function, the system behaves as expected, i.e., as the static algorithm.

V. CONCLUSION

In this paper, we proposed and analyzed CASTS, a reactive credit-based algorithm for the service traffic shaping problem in service-oriented architectures. The main feature of CASTS is to divide the observation periods in subperiods so that the SOA appliances have more opportunities to adapt their output rates to the current input conditions. Such a strategy is opposed to the static one where credits are evenly assigned to appliances in the beginning of the observation period.

We developed a simulator to evaluate the performance of CASTS and our results showed that CASTS meets the requirements defined in the service contract while providing fast responsiveness to changes in the characteristics of the input traffic.

During our experiments, we could identify a problem related to approximations in the algorithms. We refer to this problem as the flooring effect. Such an effect is a consequence of the illness in the description of the service requirements. Our results showed that when the input traffic is greater than the value specified in the contract, the flooring effect has a negative impact on CASTS during the first observation periods. We are currently working on a variation of CASTS that fixes this problem at the cost of some extra overhead.

REFERENCES