Distributed Server Selection and Admission Control in Replicated Web Systems

N. Bartolini, G. Bongiovanni, S. Silvestri
Department of Computer Science, University of Rome "La Sapienza" - Italy
{bartolini,bongiovanni,simone.silvestri }@di.uniroma1.it

Abstract

This paper addresses the problems of admission control and server selection in a system consisting of several geographically replicated web servers and several access points. We propose a fully distributed solution in which every access point continuously monitors the availability of all server side resources, using a mixture of active and passive measurements. Based on those measures, each access point autonomously applies its decisions to the requests it receives. Admission control is performed prioritizing requests belonging to already admitted sessions, in order to maximize the chance of successfully terminating ongoing sessions. Furthermore, session information is taken into account when performing a probabilistic request redirection and server choice, in order to improve load balancing and mitigate flash crowd effects. Extensive simulations, performed in compliance with industry standards, show that our method exhibits a stable behavior during overloads and improves service quality in terms of both reduced response time and higher successful session termination.

1 Introduction

Internet based applications typically demand high quality of service (QoS), possibly with some guarantees. In order to ensure that the promised QoS is indeed met, these architectures must implement some form of control (defined by specific policies) on the client requests, to avoid excessive application load with consequent degradation of the QoS. The policies are enforced by means of special devices, which we call Access Points (APs). An AP is a special type of router or dispatcher that continuously monitors the state of the server facilities and bases its decisions on such a state. Each request issued by any client does not reach directly a server, but is intercepted by one of the APs. The AP evaluates if, and how, the client request may be forwarded to the server side. A first decision that the AP must make is pertaining to admission control: in the context of this paper, performing access control means deciding either to forward the client request, or to drop it. A second decision is about server selection: if a client request will indeed be forwarded by the AP and there are more than one server that may handle it, the AP must choose the server to which forward the request. We assume full replication of application code and database at each server, although not all servers are required to hold the same content.

This paper studies the problem of admission control and server selection for systems consisting of several geographically replicated web servers and several APs. The web applications we consider offer to users a typical session based service, as is the case of e-commerce sites. In such cases it is usually highly desirable to avoid dropping a session which is near to its successful termination, since in that phase the user will likely buy something, or anyway confirm a final operation giving some revenue to the provider. To this extent, in our proposal we handle differently requests which initiate a session and requests belonging to an already ongoing session. Both the admission and the request redirection policy take into account this differentiation. APs must be able to concurrently adopt QoS policies on the basis of local information only. This scenario is typical of active networks, based on application layer anycast, but our proposals can be applied also to many of the architectures listed above. In such an environment an AP is not aware of the QoS decisions adopted by other APs. This has an important consequence: every decision made by an AP will likely interfere with other AP decisions, possibly generating undesirable effects. As an example, consider this situation: at a given time, several APs identify a server (let's call it A) as the less loaded server of the pool. Consequently, all these AP will forward to A the next requests they receive, thus rapidly overloading A and causing QoS degradation. To alleviate this problem, we adopt a probabilistic method for server selection. Each AP needs to carry on a measurement activity to somehow gather information regarding the behavior of other APs. Measures are also necessary to gather information regarding the so called bystander traffic [4], i.e. traffic that doesn't belong to the web application themselves, but shares common links with the network path going from the APs to the servers. Active or passive measures may be used: the former tend to

Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07)
0-7695-2936-4/07 $20.00 © 2007 IEEE

increase the overall traffic, the latter may give incomplete knowledge to APs (since an AP has no information about a server to which it is not currently forwarding requests). We adopt a dynamically changing mixture of the two types of measures.

These are the main ideas of our contribution:

- adoption of a mixture of active and passive measurements, dynamically driven by application load. Unlike other work, where measurements are based either on active or on passive techniques, our mixture permits to APs to gather enough local information to perform effective decisions, avoiding extra load common in active measure techniques;
- introduction of a probabilistic approach to server selection: purpose of this approach is to alleviate the effect of conflicting AP decisions, improving load balancing and overall response time;
- management of stateful sessions, whose requests are directed to the same server for the entire session: this too contributes to smooth the effects of conflicting AP decisions;
- introduction of session based admission control: we privilege ongoing sessions over new session requests, making existing sessions survive longer and with lower response time. This is obviously of benefit for many types of services, especially for e-commerce and transactional services, because low response time implies low abandon probability while longer sessions are the ones that more likely terminate with a purchase or a transaction.

The implementation of our proposals requires only minor modifications of the server software, and none on client side. In order to evaluate the proposed techniques, we designed a synthetic traffic generator, whose sizing is compliant with the assumptions of industrial standard benchmarks. The think time and session interarrival time are the same of the Rice TPC-W workload generator [17], [11], while the session model is the same as in SPECWEB2005 [16]. We tested the proposed policies on a simulator based on the OPNET modeler software [15]. The experiments show that:

- if concurrent APs apply deterministic techniques for server selection, then conflicting decisions arise that make several APs concurrently redirect traffic to the same servers, causing load imbalances and even flash crowds;
- the introduction of our probabilistic approach to server selection greatly alleviates this effect, improving load balancing and overall response time;
- the management of stateful sessions also contributes to smooth the effects of conflicting decisions;
- session based admission control is indeed effective in permitting to existing sessions to survive longer and with lower response time.

The paper is organized as follows. In Section 2 we introduce the state of the art of admission control and server selection in geographically distributed web systems. In Section 3 we give some details on the measuring framework. In Section 4 we introduce our QoS policies, to be implemented on APs. In Section 5 we describe the experimental setup, while in Section 6 we present an experimental evaluation of the proposed policies in different scenarios. Section 7 concludes the paper with some final remarks.

2 Related Work

The problem of admission control and request redirection for Web servers has been extensively studied in the recent literature. Most of the studies have been conducted on locally distributed architectures [1] with single access points while few works analyze these problems in a geographically distributed scenario, with many servers, several access points, and non dedicated links from request dispatching points to servers, possibly subject to non negligible delays due to bystander traffic. An exhaustive survey of related work is out of the scope of this paper and due to space limitations we just mention the most recent proposals and results.

Many works consider the admission control problem in single server architectures. Both [2] and [5] show how considering the admission of whole sessions rather than individual requests can improve service quality in some circumstances. Both these papers locate the admission control policy at the server level, for this reason the proposed methodology cannot scale to a geographically distributed systems, with replicated content. The authors of [14] analyze the problem of parameter oscillation when performing on/off admission control to web servers. Differently from our proposal, this scheme only considers single web servers, and single access points, which means that the suggested admission control policy may not scale to the scenario considered in this paper. In [12] the authors introduce the problem of QoS policies in a distributed environment with only one AP. A prediction of the service time requirements of an incoming request is considered while making the admission control decision with the purpose of maximizing the provider's income. Requests are scheduled on the basis of an estimate of the service time. The authors of [6] study a multi-tier cluster architecture and propose the use of a proxy server to transparently intercept database requests. Such a proxy maintains several measurement-based estimates on the total capacity of the database, the current database load and the work generated by each query type. On the basis of these measures the proxy performs adaptive admission control and request redirection, achieving overload control and improving response time. In [8] a delay prediction and control scheme based on feedback and feed-forward strategies is evaluated. In [10] the authors present a control-theoretic

approach to provide guaranteed absolute or relative delays between different service classes of a single web server. Unlike our work, the cited proposals [6, 8, 10, 12] cannot be applied to a geographically distributed environment with many replicated servers and several access points.

In [4] a single server architecture is considered, where the possible presence of bystander traffic on the links between the clients and the server is also taken into account by the admission control policy. An application level mechanism is introduced to mitigate flash crowds. Unlike our work, the authors of [4] do not take into account session information when performing admission control. They only addressed informational services, typically composed by static contents. Also, they do not afford the problem of server selection and request redirection as they consider only one target server. The authors of [3] study the problem of request redirection in a geographically distributed architecture with replicated server groups. Their work is based on a simple service model and active probing methods to gather information on the load status of the available servers. A similar problem has been addressed by the authors of [7]. They studied the performance of some redirection mechanisms on the basis of single measurements of the round trip time. In [9] the problem of anycast flow admission control is discussed. The authors present three algorithms for admission control and propose the adoption of a probabilistic redirection mechanism. The cited works [3, 7, 9] do not analyze the problem of session based traffic nor study the possibility of having conflicting decisions among several access points, for this reason their work is not directly comparable to ours.

3 Measuring Framework

APs enable QoS policies to improve the provider's revenue and the user perceived quality, by increasing the probability of successful session termination (of interest for both providers and users) and imposing a constraint on the request round trip time (of interest for the users). They operate admission control and server selection on the basis of locally available information on the quality of service that can be perceived by clients. A measurement activity consisting in a suitable alternation of active and passive techniques is at the basis of these policies. When the incoming traffic provides a sufficient message rate, the access points perform a periodic evaluation of the 95%-ile of the response time. This calculation is executed in a sliding *time window* of T seconds. If the number of samples obtained in a time window with such a passive method is not sufficient to ensure an acceptable confidence level on the estimate of the response time, the active measurement phase is started. In fact, in absence of such information, the access point cannot make any assumption on either link or server load. The active measurement phase stops as soon as a sufficient number of samples is newly available, giving a reliable estimate of the response time. During the passive measurement phase, the access points periodically evaluate the 95%-ile of each server response time, which from now on will be indicated as $RTT_{95}(\cdot)$. During the active measurement phase, each AP periodically sends active probing messages in the form of *status query* (SQ) messages. The probes are sent to all the servers for which the AP does not have a sufficiently dense sample of response time in a recent fraction of the last time window (we use the last $T/2$ seconds in our experimental setup). This way, if the frequency of application messages is too low to guarantee a sufficiently dense set of passive measures, the probing activity is started in advance with respect to the instant of evaluation of the server status. This is motivated by the necessity to ensure continuous data availability, in order to always have a reliable estimate of the response time. We assume that SQ processing time is negligible, so the time elapsed between sending the SQ and receiving the response can be used as a measure of the network delay. These measures constitute a sample space adopted by the APs for the evaluation of 95%-ile of the network delay, to which we will refer with ND_{95}. The application servers also take part in the active measurement phase, periodically evaluating the 95%-ile of the request service times observed in the last T seconds. From now on we will indicate this metric with the variable ST_{95}. The servers reply to SQ requests with a message containing the last calculated value of ST_{95}. In the active measurement phase the frequency of probing messages from an AP to an application server is tuned on the basis of the necessary number of samples to have a reliable estimate of ST_{95} and ND_{95}. In order to have a unique function to describe the estimated response time, in both the active and passive phases of measurements, we define $resp\_time(\cdot)$ as follows:

$$resp\_time(s) = \begin{cases} RTT_{95}(s), \text{ passive phase} \\ ST_{95}(s) + ND_{95}(s), \text{ active ph.} \end{cases} \quad (1)$$

where $s \in S$ is a server in the set of available servers S. The function $resp\_time(s)$ represents the 95%-ile of the user perceived latency with sufficient approximation if we assume that the latency from user to AP is negligible if compared to the latency observed in the links between access points and servers. Purely active measuring mechanisms are not scalable because servers receive a number of probing messages proportional to the number of APs. For large topologies, with an high number of APs, it is easy to see that the processing time of probing messages may be not negligible and may cause performance degradation during overload. With our adaptive method, instead, each AP queries only servers whose response time was not sampled recently enough (in the AP time window).

4 QoS policies

In the considered architecture, the AP intercepts the requests directed to the application servers and perform dis-

tributed QoS policies on the basis of locally available information pertaining to the load of application servers and network links.

4.1 Server selection

In geographically distributed web systems, when an AP intercepts a request, it selects a suitable server. The standard approach to server selection adopted in content delivery networks consists in a deterministic selection of the best replica in the pool, i.e. the server that minimizes/maximizes a given metric. We show that this method is not suitable for multiple AP environments, and propose a probabilistic method whose performance is scarcely influenced by the number of APs in the topology. In the present section we detail both the cited techniques and we assume that no admission control policy is in place. With both the server selection policies, the APs make periodic evaluation of the system status, and update the policy decision every T_{obs} sec (*inter-observation period*).

4.1.1 Deterministic server selection

The deterministic selection policy is commonly used in shared server architectures with only one access point and in content delivery networks. For this reason we use this policy as a benchmark for comparisons with our proposed approach. According to this policy, each AP uses the locally available information to select the best server in the pool. The APs forward all incoming requests to the reputed best server until the server status changes and another replica becomes the best choice. By using the measuring framework described in section 3, each AP chooses the server s^* with the lowest value of $resp\_time(\cdot)$, that is

$$s^* = \arg\min resp\_time(s),\ s \in S, \qquad (2)$$

where $resp\_time(\cdot)$ is the function defined in equation 1. If more than one server satisfies equation 2, the AP performs a random selection among the servers with minimum value of the function $resp\_time(\cdot)$.

4.1.2 Probabilistic server selection

Purpose of this mechanism is to balance the load among servers in a way that, if the same policy is adopted by several APs, conflicting decisions are reduced in comparison with the deterministic approach. The deterministic selection policy introduced in subsection 4.1.1 has two main drawbacks. A first drawback is that if there is more than one unloaded server in the system only the least loaded is selected with consequent scarce resource utilization. A second drawback is that in a multiple AP environment, if servers are shared among different APs, the deterministic selection increases the number of conflicting decisions. A conflicting decision happens when more APs select the same server as best replica, generating an overload and possibly even a flash crowd effect on the chosen server. To avoid conflicting decisions we allow the APs to select a set, $S^* \subseteq S$, of "reasonably" unloaded servers. Incoming requests are then redirected to a server in S^* with a probability which decreases as the server load increases. Notice that the server selection mechanism is separate from the admission control policy, therefore we want to be able to select the subset of suited servers in a way that the presence of at least one server for redirection is always guaranteed. To this extent we set a threshold T_{SS} on the estimated 95%-ile of the response time. Since it is impossible to set an arbitrary value of T_{SS} that guarantees the selection of a non empty subset of servers, we adopted the following heuristics that adapts the response time threshold value to the present server status. Be $R = \{r_1, r_2, \ldots, r_n\}$ the set of the estimated 95%-iles of the response times of the n available servers, and be $\tilde{r} \triangleq \min R$, the minimum value of R. Be T_{SS} an arbitrary threshold on the response time, possibly lower or equal to the user timeout, i.e. the average time the user waits until he abandons the site due to impatience. We define the adaptive threshold θ as follows: $\theta \triangleq k \cdot T_{\mathrm{SS}}$, where $k = \left\lceil \frac{\tilde{r}}{T_{\mathrm{SS}}} \right\rceil$. This way we can define S^* as the subset of servers with estimated 95%-ile of the response time under the threshold θ, that is

$$S^* \triangleq \{s \in S | resp\_time(s) \leq \theta\}.$$

Obviously the set S^* is not empty since at least $\tilde{s}$ exists such that $\tilde{s} \in S$ and $resp\_time(\tilde{s}) = \tilde{r}$. The threshold-based probabilistic server selection mechanism avoids the common drawbacks of the deterministic selection method. Instead of selecting one server, it selects all servers whose estimated response time is under the adaptive threshold θ. Incoming requests are probabilistically distributed among the servers of the set S^* on the basis of a weight assignment that keeps into account the status of the server load. To distribute incoming user requests among servers in S^*, APs follow a weighted probability function $p_\theta(s)$:

$$p_\theta(s) \triangleq [\theta - resp\_time(s)]/L \qquad \forall s \in S^*$$

where $L = \sum_{s \in S^*}(\theta - resp\_time(s))$. Other weighted probability functions have been proposed in previous works, such as in [3, 7] , for a single access point scenario. A comparison with other weighted probability functions is out of the scope of this paper. We just mention that our formulation generates probabilities that are less oriented to the least loaded servers if compared to the formulation proposed in [3, 7]. If there is more than one server in S^* it will receive a fraction of incoming requests proportional to its available capacity. By probabilistically selecting a server in a subset, the server selection policy relieves the effects of conflicting decisions among several APs. In fact even if the same servers were selected by different APs, each of them would receive only a fraction of the incoming traffic of each AP.

4.2 Admission control

The proposed Admission Control (AC) mechanism is based on the measuring activity described in section 3. Every T_{obs} seconds, each AP updates its status information on the basis of the samples gathered in the most recent time window. We use a threshold policy on the estimate of the 95%-ile of the response time. We introduce the threshold T_{AC} so that if the estimated 95%-ile of the response time (calculated with respect to the whole server pool) is over the threshold T_{AC}, the AP rejects new session requests and only accepts requests belonging to already ongoing sessions. On the contrary, if the AP obtains a value of the 95%-ile of the response time that is lower than T_{AC}, new session requests will be accepted. As we show in section 6, simulation results reveal that this policy contributes to lowering the average and the 95%-ile of the request round trip time and to increasing the average session duration thanks to a lower abandon rate.

4.3 Request redirection

Modern web applications often need some state information regarding user sessions. Because of the stateless nature of the HTTP protocol, there must be an application specific management of this information. Many techniques can be adopted to this purpose. Among these, some directly involve the role of clients, such as cookies or hidden input types in forms. Other techniques are preferred for efficiency and security reasons, requiring the creation of objects with session scope on the server application tier. The implementation of such objects depends on the server technology in use. If J2EE is in use they may be objects implementing the `HttpSession` interface or beans with a `session` scope. For simplicity we refer to these objects with the name of *session objects*. Session for which *session objects* are handled will be called *stateful sessions*, and *stateless session* otherwise. When a session object is created, for a given user session, all future requests pertaining to that session must be locked to the same server, otherwise an error will occur unless context transfer management is in place, which is not the case in our reference architecture.

5 Simulation environment

We developed a simulator on the basis of the OPNET modeler software [15] to study the performance of our QoS policies. The main entities of the simulation environment are clients, APs and application servers. We assume that clients are partitioned into geographic areas, and for each area there is a single AP. In this way all clients of the same area send their requests to the same AP, and latency on the links between clients and the AP can be ignored. In the experimental setup we do not introduce bystander traffic because we want to focus on the effects of the policies in presence of several APs adopting concurrent admission control decisions and server selections.

5.1 Client model and traffic generation

According to our model, each client issues a user session, in the shape of a series of correlated requests. We assume that the interarrival time of new sessions, for each AP, follows a negative exponential distribution with average $1/\lambda$, while the interarrival time of requests belonging to the same session is more complex. After the first request, each client issues subsequent requests waiting for the corresponding response (response time), and spending some time analyzing the content of the received response (think time). In order to have a realistic traffic generator, we use the phase model of an industrial standard benchmark: SPECWEB2005 [16]. Unlike its preceding releases, SPECWEB2005 takes into account for the most relevant types of dynamic content (php and jsp included). It models an e-commerce web site selling personal computers, with typical requests such as browse, login, search, customization of selection, add to cart, buy and logout functionalities. Refer to [16] for a detailed description of the state model and of the functionalities of each phase. Client sessions start from the home page and follow the state model according to its transition probabilities, possibly ending in any state if the user voluntary abandons the site before completing the whole session life-cycle, or reaching the last state of the model where an item can be purchased or not. Upon reception of a response, the next request is sent after a certain amount of time, called *User Think Time (UTT)*, that represents the time the user spends analyzing the received web page. Our model of *UTT* is based on TPC-W [17] and on other works in the area of web traffic analysis that justify the assumption of heavy tailed distributions of the user think time [13]. As in the TPC-W model, we assumed a logarithmic distribution of think times. We also assume a lower bound of 1 sec to the think time. Therefore $UTT = max\{-log(r)\mu, 1\}$, where r is uniformly distributed in the interval [0,1] and $\mu = 20sec$. To model a realistic user behavior, we also introduce a timeout on the maximum time the user can wait for a response (client timeout). The interaction of a client with the web application may end for three possible reasons: *a*) session block: the AP denies service to the client's first request (no navigation session is started); *b*) session drop: no response is received within the client timeout and the ongoing session is interrupted; *c*) successful session termination: all responses are received in time and the user voluntary terminates the session.

5.2 Access point model

APs are the key entities of our model since all QoS policies are enforced by them. They adopt a two-way dispatching policy, so APs receive client requests and dispatch them to the servers. They also receive server responses and forward them to the clients. APs enable the adoption of QoS policies for admission control, server selection and request redirection, as extensively detailed in section 4.

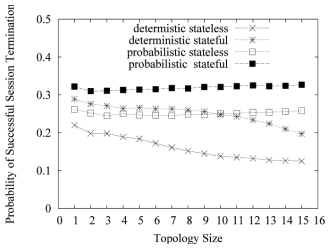

Figure 1. Successful Termination Probabiliy

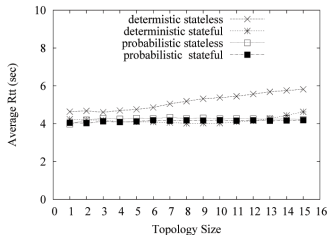

Figure 2. Average Request RTT

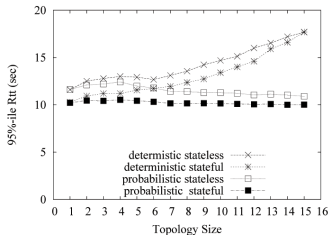

Figure 3. 95%-ile of Request RTT

5.3 Server model

Servers communicate with the clients via the APs. For each received request a thread is created and added to the tail of the waiting queue. Servers use a time sharing, round robin scheduling. Thread processing time depends on the type of request being served. As it is widely accepted, we consider three categories of requests depending on which tiers are involved in the request processing: *pure http requests* involve the http server tier, *servlet requests* involve the application tier while *database requests* reach the third tier for the execution of a query to the database. We give an approximate estimate of the average processing times of the different categories on the basis of the experiments detailed in [6]. In our simulations we assume each session phase requires an exponentially distributed execution time. Execution times are set as follows (waiting time due to preemptive processor scheduling is not considered in these measures): average execution time for pure http requests is 0.001sec, while for servlet request is 0.01sec and for database requests is 1sec.

6 Simulation results

In this section we present two sets of simulations. The first set, detailed in subsection 6.1, shows the performance of the two server selection mechanisms, deterministic and probabilistic, in absence of admission control. This set of simulations points out how the probabilistic method solves the problems of conflicting decisions that characterize the deterministic method. The second set of simulations, detailed in subsection 6.2, studies the effects of the admission control policy, showing that it can be used to improve both the user perceived quality and the provider interests. In our simulations we consider both stateful and stateless navigation sessions, For stateful sessions the session object is created as soon as the user issues a customization request.

6.1 Server selection mechanisms

Concerning the performance of the two selection mechanisms described in 6.1, we show how, augmenting the size of the topology, the classic deterministic selection exhibits decreasing performance while the probabilistic method is not adversely influenced. Before we introduce the simulation results, a short analysis of the simulation scenario must be done. We want to show the benefits of the probabilistic method over the deterministic choice as the number of APs grows. To make a fair comparison among results obtained with an increasing number of APs we must have an invariable load level on servers. To this extent, we have two choices: *a*) to decrease the traffic rate flowing through each access point, so that the load that insists on each server does not change with varying the number of APs, or *b*) to increase the number of servers in order to make it grow proportionally to the number of APs and maintain a constant load on each server. The first solution makes comparisons unfair because by reducing the amount of traffic coming from each AP we also reduce the amount of traffic reaching the target server of a conflicting decision. For this reason we let the number of APs and the number of servers grow proportionally, keeping a constant rate of traffic from each AP. In our experiments we set a number of servers three times higher than the number of APs and scale the topology size increasing the number of APs from 1 to 15, and consequently the number of servers from 3 to 45, while the traffic coming from each AP has a rate of λ sessions/sec. This way, the total load divided by the number of servers remains invariant. Here is a list of the other important simulation parameters used in the simulations: session arrival rate λ = 3 sessions/sec, client timeout t_{out}= 8 sec, probabilistic server selection threshold T_{SS}= 8 sec. The measuring activity works with the following time slots: sliding time window of 60 sec, inter-observation period of 5 sec. With this set of experiments, the performances of the four combinations (stateless and stateful sessions, each with deterministic and probabilistic server selection) are evaluated in terms of probability of successful session termination (*PSST*, figure 1) and RTT (figures 2 and 3). Figure 1 shows that both probabilistic server selection and management of stateful sessions are beneficial in terms of reducing the negative effects of load inbalance due to conflicting decisions made by APs. In fact, stateless sessions with deterministic server selection exhibit the poorest behavior: as the number of APs increases, *PSST* steadily decreases. Stateless sessions with probabilistic server selection show a better behavior, but cannot avoid a progressive decrease in *PSST*. On the contrary, probabilistic server selection for both stateless and stateful sessions proves to be immune from decrease in *PSST* as the number of APs grows, indicating that this technique is indeed effective in

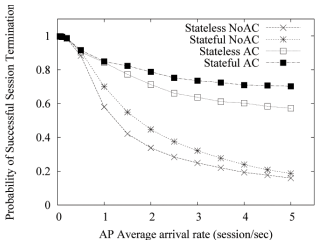

Figure 4. Session Termination Probability

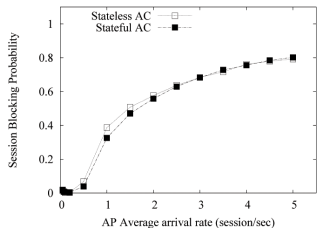

Figure 5. Session Blocking Probability

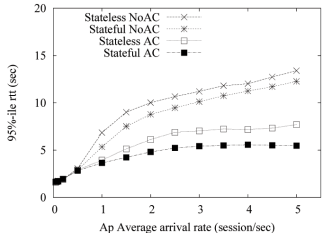

Figure 6. 95%-ile of Request RTT

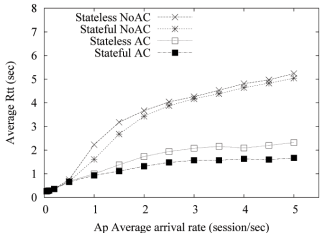

Figure 7. Average Request RTT

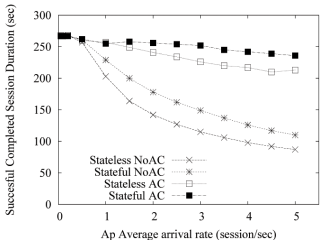

Figure 8. Average Session Duration

avoiding load inbalances. The best performance is obtained by using probabilistic server selection along with stateful sessions: in this case the *PSST* exhibits the highest values. Figure 1 shows another interesting phenomenon. The beneficial effects of probabilistic server selection sum up with those of stateful sessions. This is reasonable, since the two techniques act in different and independent ways: the former helps by distributing the load on a set of servers rather than on a single server, the latter contributes by avoiding redirection of locked sessions. Figures 2 and 3 show that a similar behavior is confirmed by looking at the values of RTT. Again, stateless sessions with deterministic server selection exhibit the worst behavior and produce the highest RTTs, while stateful sessions with probabilistic server selection show the best performance with the lowest values of RTT. In particular, figure 3 shows that deterministic server selection (for both stateless and stateful sessions) causes a progressive increase in the 95%-ile of RTT, while probabilistic server selection (again, for both stateless and stateful sessions) does not. Figure 3 indicates that the beneficial effects of the two techniques sum up together in terms of RTT too: for both deterministic and probabilistic server selection methods, stateful sessions produce lower RTTs than stateless sessions.

6.2 Admission Control

Whatever server selection method is in use, performance can degrade unacceptably in case of overload. To ensure probabilistic guarantees on the user perceived quality and to increase performance parameters of interest to the service provider, we introduce an admission control mechanism. In all these test we use the same topology size and adopt the probabilistic server selection method. We vary the average arrival rate from each AP and study the impact of the admission control policy with stateful and stateless session. The simulation parameters setting are as follows: 8 access points, 24 application servers, admission control threshold T_{AC}=4 sec, client timeout t_{out}=8 sec, probabilistic server selection threshold T_{SS}=8 sec. The measuring activity works with the following time slots: sliding time window of 60 sec and inter-observation period of 5 sec. Let us consider the case without session objects. Figure 4 shows the probability of successful termination of the admitted sessions. Without admission control (*NoAC scenario*) all incoming sessions are admitted to the system. By raising the session arrival rate, too many sessions arrive and the probability of a client timeout increases, with a consequently increased number of interrupted sessions. With the admission control mechanism (*AC scenario*), instead, the probability of successful termination is consistently higher. As we show in figures 6 and 7, without admission control, both the 95%-ile and the average of the request RTT significantly increase with the session arrival rate, worsening the quality of service perceived by the users. In presence of the admission control, instead, the overload effect is smoothed the average and 95%-ile values of the request RTT are much lower. A similar situation is obtained for stateful sessions, with figures that are better than their stateless counterparts. This behavior shows that the beneficial effects of admission control and those of stateful sessions sum up together. In commercial web sites another important aspect is the duration of completed sessions, because longer sessions are the ones that most likely result in purchases. During overload, without an AC mechanism, longer sessions are penalized because they are composed by many requests, so the probability that a request processing incurs in a client timeout increases. Figure 8 shows the average duration of the success-

fully completed sessions for stateful and statlsess sessions with and without AC. Note how, going from the smallest to the largest value of λ, without AC, the average duration drops as musch as 60% for stateful sessions and 70% for stateless session. In a commercial web sites this behavior is strongly undesirable because it can cause profit losses. With the AC mechanism, instead, the session duration is much more stable and the degradation due to overload with or without the creation of persistent session objects is only 12% and 20% respectively.

7 Conclusions

This paper addresses the problem of adopting distributed policies in a geographically distributed server environment with several access points. The reference architecture models several service paradigms such as active content delivery networks, edge computing, anycast based services and other multi-broker services as well. In such an environment every policy decision made by each AP will likely interfere with other AP decisions. According to our proposal each AP can keep into account the behavior of other APs, by means of a measuring framework that alternates active and passive measurements techniques adapting to the application load. We introduce a probabilistic method for server selection, a session aware request redirection technique and a session based admission control. By means of simulations we show that our probabilistic method for server selection, based on estimates of the request round trip time, leads to a better load balancing among servers by reducing the probability of conflicting decisions among different access points that could cause load imbalances and flash crowds. This results in a higher successful termination probability and in a reduced user perceived response times. We also show that additional improvements are achieved by performing session aware request redirection, i.e. redirecting requests belonging to the same session towards the same server. Simulation results also point out that with session based admission control, a reduced number of sessions gain access to the system, but with improved successful termination probability. We show that the expected lifetime of successfully terminated sessions is increased. This last result is of primary importance for web applications implementing e-commerce capabilities.

References

[1] V. Cardellini, E. Casalicchio, and M. Colajanni. The state of the art in locally distributed web server systems. *ACM Computing Surveys*, 34(2):263–311, 2002.

[2] J. Carlstrom and R. Rom. Application aware admission control and scheduling in web servers. *Proc. of the IEEE Conf. on Computer Communications (INFOCOM)*, 2002.

[3] H. Chang, W. Jia, and L. Zhang. Distributed server selection with imprecise state for replicated server group. *Proc. of the IEEE Int. Symp. on Parallel Architectures, Algorithms and Networks (ISPAN)*, 2004.

[4] X. Chen and J. Heidemann. Flash crowd mitigation via adaptive admission control based on application-level observation. *ACM Trans. on Internet Technology*, 5(3):532–569, 2005.

[5] L. Cherkasova and P. Phaal. Session based admission control: a mechanism for peak load management of commercial web sites. *IEEE Trans. on Computers*, 51(6), 2002.

[6] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel. A method for transparent admission control and request scheduling in e-commerce web sites. *Proc. of the ACM World Wide Web Conference(WWW)*, May 2004.

[7] H. B. Hashim and J. A. Manan. An active anycast rtt-based server selection technique. *Proc. of the IEEE Int. Conf. on Networks (ICON)*, 2005.

[8] D. Henriksson, Y. Lu, and T. Abdelzaher. Improved prediction for web server delay control. *Proc. of the IEEE Euromicro Conf. on Real-time systems (ECRTS)*, 2004.

[9] W. Jia, D. Xuan, W. Tu, L. Lin, and W. Zhao. Distributed admission control for anycast flows. *IEEE Trans. on Parallel and Distributed Systems*, 15(8), August 2004.

[10] C. Lu, Y. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son. Feedback control architecture and design methodology for service delay guarantees in web servers. *IEEE Trans. on Parallel and Distributed Systems*, 17(9):1014–1027, 2006.

[11] D. Menasce. Tpc-w: A benchmark for e-commerce. *IEEE Internet Computing*, May/June 2002.

[12] A. Verma and S. Ghosal. On admission control for profit maximization of networked service providers. *Proceedings of ACM World Wide Web (WWW)*, 2003.

[13] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer. Off the beaten tracks: Exploring three aspects of web navigation. *Proc. of ACM World Wide Web (WWW)*, 2006.

[14] Z. Xu and G. v. Bochmann. A probabilistic approach for admission control to web servers. *Proc. of the Int. Symp. on Performance Evaluation of Computer and Telecommuinication Systems (SPECTS)*, July 2004.

[15] Opnet technologies inc. *http://www.opnet.com*.

[16] Specweb2005. *http://www.spec.org/*.

[17] The transaction processing council (tpc). tpc-w. *http://www.tpc.org/tpcw*.

0-7695-2936-4/07 $20.00 © 2007 IEEE