CHAPTER 1

INTRODUCTION

1.1 Preamble

Today the Internet has become a platform to perform searches, transact business, and connect with friends and colleagues. With growing Internet user population, which is expected to be around 3 billion in 2015[49] and the growing number of services like: social networking (Facebook, hi5.com), online shopping websites (Amazon, Flipkart), blog pages (Blogger, Word Press), online news websites (CNN, NDTV, IBNLive), Internet search engines (Google, Bing), alternatives to desktop software applications (Google Docs, Slide Rocket) etc., offered through the Internet, the future will likely see more of business Web presence in the form of Web applications. Today, global enterprises are increasingly dependent on Internet-based applications for E-Commerce, supply chain management, human resources, and financial accounting; many individuals consider e-mail, chat, and social-networking and Web access to be indispensable lifelines. This growing number of users and their dependence upon Internet Services underscores the importance of the ability of the service in terms of its responsiveness, availability, scalability, and ability to handle large loads.

Internet Services are subjected to huge variations in demand, with bursts coinciding with the times that the service has the most value. Apart from these flash crowds, sites are also subject to denial-of-service (DoS) attacks that can knock a service out of commission. Such attacks are becoming increasingly sophisticated, often being launched simultaneously from thousands of sites across the Internet [57]. DoS attacks have had a major impact on the performance of sites such as Buy.com, Yahoo!, and whitehouse.gov [36, 61]. Such attacks have to be identified and filtered out without hampering the genuine requests or their progress.

Popular sites are constantly adding new functionality to stay competitive, a trend which is supported by the increasing ease of Internet Service development using simplified Web scripting languages such as PHP [62] and Java Servlets [63]. It is excessively time-consuming to carefully engineer and fine-tune each new service component to deal with real-world loads. A meaningful approach to this problem is through a generic service platform that manages load in a manner that is cleanly
separated from the service-specific logic; this gives services the ability to participate in load-management decisions. The high complexity and cost of maintaining a hosting platform infrastructure has resulted in a growing trend among businesses and institutions to have their applications hosted on platforms managed by another party. Such a hosting platform is a Web hosting provider (Service Provider – SP); an SP sells space on its servers to Website owners (Service Requestor – SR). The SP provides a full-time, high-bandwidth connection to the Internet, so that visitors to the Website can access the sites easily. SPs enable entrepreneurs and emerging organizations to focus on their business rather than technology. They are typically expected to provide performance guarantees to the hosted applications (such as guarantees on response time or throughput) in return for revenue [1, 17, 19, 23]. Service Level Agreements (SLAs) between the SP and the SR incorporate the scope of the Web Service, the performance guarantees to be offered, their quality, and modes of their quantification.

A survey from 2006 revealed that 62% Web users were willing to wait only 6 seconds or less for a single page load before they leave the Web site [64]. A more recent research indicated that this performance expectation has become more demanding as 83% Web users expected a Web page to load in 3 seconds or less [65]. In addition, this research also found that 79% of online shoppers who visited an underperforming Web site were not likely to buy from that site [59]. Obviously, performance guarantees for Web applications have become business critical. Performance metrics of an Internet Service are influenced by many factors. Several papers have investigated various configurable parameters, and the way in which they affect the performance of Web-based software systems [20, 27, 28, 33, 34, 42, 51]. Statistical methods and hypothesis tests [32] are being used to retrieve factors influencing the performance. The performance metrics can be predicted at early stages with the help of a properly designed performance model. All these point to a clear need for the crafting and use of a performance model.

Depending on the resource requirements of the applications and the strictness of the performance or resource guarantees they require, a platform may employ a dedicated or a shared model for hosting them. Designing a hosting platform is made challenging by the characteristics of Internet applications and their workloads (Complex multi-tier software architecture and Internet workload characteristics like:
multi-time-scale variations, extreme overloads, session based, and multiple session classes) [44, 48, 52, 56]. To meet its goal of maximizing revenue given the above challenges, a hosting platform needs to carefully multiplex its resources among the hosted applications. The performance model should have the flexibility to accommodate the variety in software architectures and Internet workload characteristics.

Performance modeling is an important part of the research area of Web servers. Without a correct model of a Web server it is difficult to give an accurate prediction of performance metrics. This is the basis of Web server capacity planning, where models are used to predict performance in different settings [9]. An important component of Web performance is the architectural design of the Web server software. The biggest challenge in designing an efficient Web server lies in efficiently supporting thousands of connections concurrently. Different Web server architectures therefore vary mainly in how they implement concurrency [9]. The earliest Web servers were all process based [9, 23], i.e., they maintain a separate Operating System (OS) process for each request. The process driven approach simplifies code development, since all responsibility for handling concurrency is relayed to the OS. However, a major drawback of processes is that they are heavy-weight, and creating and deleting them and context-switching between them is expensive. An alternative approach is to use threads instead of processes [23]. Threads are more light-weight since they allow sharing of the same address space. While sharing the address space reduces the context switching cost, it has the disadvantage that a poorly programmed thread can crash the entire server. In the above approaches the Web server software is implemented in user space. Some Web servers such as Tux [5, 30, 31, 38] are entirely implemented in kernel space thereby avoiding overheads associated with system calls and copying of data between user and kernel space. The downside of the in-kernel implementation is that it is extremely sensitive to programming errors, since a crash of the Web server code translates to a crash of the kernel and therefore the whole machine. The basic need here is to provide concurrency whether the service is process-based or thread-based.
1.2 Motivation

The above overview of Internet Services leads us to the following inferences which constitute the motivation for the present work:

1. The key attributes expected of an Internet Service are its responsiveness, availability, scalability, and ability to handle large loads.
2. DoS attacks – despite their sophistication – have to be identified and filtered out without hampering the genuine requests or their progress.
3. SLAs between the SP and the SR incorporate the scope of the Web Service, the performance guarantees to be offered, their quality, and modes of their quantification. The dynamic nature of present day services should be reflected in SLA formulation and implementation.
4. The course of exposed resources along with the level of privacy, value of the information they hold, and the increase in their usage, has led to the escalation in the number of security threats and violation attempts that existing systems do not appear robust enough to address peak loads.
5. Whether the service is process-based or thread-based, the basic purpose is to provide concurrency.
6. Perhaps one of the most compelling problems of the modern Internet is the lack of a comprehensive model that can aid in performance evaluation, unifying approaches to deal with service concurrency, security, and availability while preserving ease of implementation.
7. Any model crafted and used to analyse an Internet Service should have the flexibility to accommodate the variety in software architectures and Internet workload characteristics.

1.3 Research Objectives

This research work focuses on providing a comprehensive black-box model that can be used to evaluate the performance of a variety of Web applications. The research objectives are stated as follows:

1. To present a coherent and inclusive way to model Web applications with details and abstractions which are appropriate for SPs in terms of capacity planning and SRs in terms quality assessment.
2. To demonstrate the utility of the proposed model for the purpose of performance prediction and capacity planning with varying workload and service differentiation strategies.

3. To demonstrate the ability of the model to capture the category based service differentiation, improved service availability, and maximize user satisfaction for a real time Web application.

4. To introduce the problems that exist in the currently used authentication schemes and workloads that particularly affects the availability of the Internet Services and subsequently derives an appropriate solution for the same.

The approach leverages models of a service’s application-level performance in evaluating the Quality of Service (QoS) of an Internet Service. The proposed model represents the relevant aspects of the system in a simplified and ideally comprehensible manner. The model presented and discussed in this work does not address network characteristics. It allows SPs to query and understand what performance would be like if a particular action was taken. For example, the model answers the following queries:

- What happens to performance if a new functionality is added to the service under the current configuration?
- What if the allocation policy for resource assignment was changed?
- What configurations yield high performance?

The evaluation is based on the implementation, assessment, and analysis (based on collected logs and statistical techniques). The logs collected by the model allow SR to study the behavior patterns of online customers on their site and improve business strategies. It also serves as a guideline for SRs to understand the measures of the service quality that have been offered by the hosting platform services. It aids SP to evaluate the measures of commitments in service quality that could be agreed. These derived measures from the model could enable them in formalizing the SLA. Further the model shields application programmers from many of the details of scheduling and resource management.
1.4 Research Methodology

Several approaches like Measurement, Simulation, Analytical, and Black-box modeling are used in Internet Service performance evaluation [52]. In this work, the relevant aspects of the system to be modeled are analyzed first, followed by study of Web application characteristics. After having gained insight into the workings of Internet Services, a preliminary working model (‘BBAM-I’ – Black Box Analytical Model Version I) was implemented to understand the nuances of the implementation and make a case for a sophisticated one. The behavior of this prototype in terms of offering service availability was carefully and systematically examined for various workloads.

The preliminary model was enhanced (‘BBAM-II’– Black Box Analytical Model Version II) to improve service availability with the addition of a QUEUE (‘Tokens-Basket’) data-structure over and above the server architecture. A detailed analysis on the effect of this added data structure on concurrency (number of simultaneous users accessing the service), with varied number of tokens was conducted. The results of the preliminary working model were compared with the results of enhanced version to understand the contribution of the enhanced model in terms of improved service availability. The capability of the enhanced version to serve additional user’s requests with and without sacrifice in service quality was investigated. With detailed simulation it has been demonstrated that the enhanced version has been potentially able to provide improved service availability without performance degradation by utilizing the gap between execution time and response time effectively and any further improvement was possible only with an acceptable reduction in the QoS.

The next step was to explore the possibility of deriving limits for various quality factors that determine the performance of Web applications. Our experimentation with different parameter sets, confirmed that a basis to quantify the guarantees to be made for various parameters in SLA can be derived using the prototypes.

Subsequently, the model was tested for its suitability to accommodate real time Internet Services, with E-Commerce as a representative Web application for
study. It was verified that the prototype is versatile enough to accommodate new classes of E-Customers, logs collected were sufficient enough to evaluate performance metrics, and were able to do informed scheduling and resource management decisions, such as reordering, filtering, or aggregation of requests as well as several other characteristic operating scenarios.

Two aspects that could affect the service availability were identified at the end of the study of the modeling the E-Commerce application:

1. Possible delay in system throughput, increased response time at the home page due to malicious requests in the workload.
2. Possible service unavailability (time out of sessions/delayed response time) at the payment stage because of the dependency on an external mobile network service (Short Message Service – One Time Password–SMS-OTP) [60] for authentication.

The model supports integration of a pre-filtering algorithm to condition the workload (removing malicious requests) before their acceptance into the system to avoid (1) and uses a Quick Response-Code (QR-Code) based OTP scheme capable of overcoming the delay caused by the prevailing SMS-OTP scheme (2).

1.5 Hosting Platform Model

A dedicated hosting service is a type of Internet hosting in which the client leases an entire server for its own use. This is more flexible than shared hosting, as organizations have full control over the server(s). In this thesis, we assume a dedicated hosting model. However the developments and the associated discussions are equally valid for the sharing hosting case as well. In case of shared hosting independent capsules are required for capturing and monitoring the performance of each application. Figure1.1 depicts the architecture of the platform. The main components include:

Nucleus: Responsible for resource allocation and management.
Capsule: Responsible for monitoring the performance (collecting logs).
Sentry: Responsible for admitting sessions.
Web applications (3-tier) hosted in the platform [53, 3].
An SLA is a formal written agreement between the SP and the SRs. It defines the parameters of service, which the SRs expect and the SP guarantees to deliver. Since the hosted information and data may be highly confidential requestors demand that a very high degree of security be delivered. Additionally, usability is an important area of concern, particularly when high availability of service is demanded. The proposed model assumes all the 3 tiers of the web application to be present on the same node and hence uses only 1 capsule. If the tiers are distributed on different nodes then 3 capsules need to be used.

![Diagram of Hosting Platform]

1.6 Dissertation Roadmap

Employing black-box simulation to model Web applications and study their performance, the thesis focuses on analyzing the reasons for service unavailability of Internet Services and proposes ways of improving the same. The rest of the thesis is structured as follows:

In Chapter 2 the challenges faced by the current Internet Services, and the literature related to modeling Web applications are presented. The aspects analyzed include workload characteristics, user behavior and representation of Web
applications. This laid groundwork for developing a preliminary model for performance evaluation of Web applications. A Finite State Machine (FSM) has been used to simulate processing in a server. A sequential set of FSM states were taken in this scheme to accommodate the service architecture. Behaviors of users of Web applications have been modeled using User Behavior Model Graphs (UBMG) [14, 54] which represent the probabilities of transition from one state of FSM to another.

With the service architecture ‘thread-per-connection’ in use by APACHE as basis, Chapter 3 develops a comprehensive model (‘BBAM-I’) to represent Web applications. The model is structured as a sequence of stages with equal execution times. At every time slot all active client requests as well as the new requests are queued conforming to a Priority List (PL) representing the desired service differentiation policy. The threads are allocated to the requests in the queue; left over requests – if any– are denied service (dropped requests). User requests under normal conditions conform to a Gaussian distribution pattern with specified \( \{ \mu, \sigma^2 \} \) set. Further they progress conforming to the UBMG.

Extensive simulation over the full meaningful \( \{ \mu, \sigma^2 \} \) range was carried out. With each set the simulation duration was long enough to ensure that each possible request values occurred sufficient number of times. The simulation study effectively brought out the following:

- The mean arrival rate should be smaller than the system capacity representing the number of threads for the service to be effective.
- The maximum \( \sigma^2 \) value decides the difference between the above two. For a given value of the difference the drop rate and the deficiency in system utilization increase with \( \sigma^2 \).

The simulation was continued with an impulse load superposed on the regular stationary random load – representing (possibly) the peak load at its extreme. It confirmed that the system accommodates peak loads if the mean arrival rate is sufficiently smaller than the system capacity; the price paid is an increase in response time.

The simplest sequential set of stages considered here directly represents a pipelined process at the highest level of abstraction. Each stage can be further split
and represented as sub-stages of execution at the next finer level of granularity. This can be continued down to the finest level of granularity desired.

The model does not call for any detailed decision making for thread allocation. There can be time slots when threads are free and those when requests are dropped though both do not happen simultaneously. The simulation results with the model showed its suitability for Services where there is not much of deviation in average workload and those like video streaming (i.e., it is suitable for Internet Services that are yet to become popular and those which require high quality of service guarantees for accepted connections rather than serving many concurrent users). The service becomes unavailable as soon as system capacity exhausts. Additional increase in average load is managed by rejecting the requests.

The generic model ‘BBAM-I’ can represent any architecture with any combination of sequential/parallel stages of execution. After showing this Chapter 4 presents an enhanced version of the scheme (‘BBAM-II’) that uses a variation of ‘thread-per-request’ model used by servers. In ‘BBAM-II’ client requests can be executed concurrently until the number of simultaneous requests exceeds the number of threads in the pool. At this point, additional requests (a number decided in advance) are queued in a ‘Tokens-Basket’ until a free thread becomes available. This is used to limit idling of threads and reduce denial of requests (provide enhanced service availability). The requests are assigned tokens and serviced conforming to a predetermined service differentiation strategy which is decided by the algorithm/procedure defined by the ‘Priority List’ (PL). Thread assignment is done afresh for each request in each time slot for each stage separately. Complexity in assigning threads here is more than that of ‘BBAM-I’. The number of tokens in the ‘Tokens-Basket’ is chosen such that it is greater than the number of threads in the thread pool. If used correctly, bounded ‘Tokens-Basket’ is an effective means to represent service availability in an Internet Service. The important aspects of the enhanced operational model are:

1. Inclusion of a ‘Tokens-Basket’ structure over and above the existing server architecture enables the server to accommodate requests (enhance server availability) that would have been otherwise rejected as in the case of ‘BBAM-I’.
2. The introduction of ‘Tokens-Basket’ achieves two purpose:

- Serving additional requests without degraded quality as long as the execution time is lower than the expected response time. (Free threads available in thread pool at different timeslots can exploit this to serve additional requests without sacrifice of response time).

- Serving additional requests with a slight delay in response time. The acceptable limit set by the SR on response time can be used to decide on how many of such additions can be accepted.

3. The PL Rule that is incorporated to schedule the accepted requests enables the SR realize the mission of offering differentiated services to their visitors/customers.

4. Making the ‘Tokens-Basket’ explicit allows applications to make informed scheduling and resource management decisions, such as reordering, filtering, or aggregation of requests.

5. The increase in number of tokens poses a corresponding delay in service time. This is obvious from the fact that it is possible for a set of requests to be kept waiting with a token assigned to them due to unavailability of threads in that timeslot. This otherwise means that the introduction of the ‘Tokens-Basket’ has reduced the service rate (computed as the ratio of the number of requests completed in that time slot to the total number of requests residing in the system) at each time slot whenever the number of requests in the system exceeds the system capacity.

6. Deployment is made easy, shielding application programmers from many of the details of scheduling and resource management.

7. The model supports Internet Services to analyze the request stream and to adapt behaviour to changing load conditions. For example, the system should be able to prioritize and filter requests to support degraded service under heavy load.

Chapter 5 focuses on the utility of the model in terms of support for strong profiling. It has been shown that the collected logs and derived performance metrics allow detailed assessment of the business performance (like studying behaviour patterns of
online customers on their site and improving business strategies) for SR and capacity planning (like indicating resource bottlenecks and under utilization) for SP; it also aids them to find if and when SLA re-negotiation is required.

In Chapter 6, ‘BBAM-I’ and ‘BBAM-II’ are explored for their suitability to accommodate the real time requirements of Web applications. The requirements of an E-Commerce application were used to evaluate user satisfaction in terms of service availability and response time. A detailed experimentation with different workload mix and varied request scheduling policies carried out, clearly brought out the flexibility of the model to accommodate service specific requirements. It also helped in identifying two reasons other than request rejection that may affect the availability of the service namely malicious requests accepted in the ‘Tokens-Basket’ and dependency on external SP’s network for request completion.

Chapter 7 explores ways of improving service availability without degrading QoS. A filter component has been added to the architecture along with an authentication scheme that minimizes the delay in the response time by reducing dependency on Mobile Network Operator for SMS-OTP. Incorporation of these leads to better service availability particularly at times of overloads on the server.

Chapter 8 presents the conclusions and suggestions for future work based on the thesis.

1.7 Highlights of the thesis

A model – ‘BBAM-I’ – which represents the full web service comprising its architecture, its workload – steady as well as peak conditions, and its performance in terms of tangible and quantifiable items and metrics has been developed. The need for accommodating service differentiation has been accomplished through scheduling using the concept of a PL introduced here. The fact that a variety of service differentiation algorithms can be accommodated in the PL is its novelty. Different user categorization possibilities and service differentiation approaches can be implemented using the PL.

Service quality attributes that can be quantified are measurable or computable in terms of measured ones. The important ones in the latter set – namely request drop
rate, response time, availability, and user frustration have been quantified in terms of measurables. Their utility to assess and quantify service performance has been brought out.

The versatility of the model to represent varied architectures – pipelined, sequential, and their combinations – has been brought out. Basic model has been subsequently enhanced to ‘BBAM-II’ to accommodate queuing of requests at any desired stage. A novel concept of a ‘Tokens-Basket’ has been introduced for this. The size of the ‘Tokens-Basket’ relative to system capacity is to be selected based on the spread of user request size.

The model can be enhanced to represent each processing stage as a sub sequence of smaller stages. Such finer granularity for representation can be continued as much as desired.

A simulation scheme to study the performance of web services represented by the model has been developed. Through detailed simulation on a simple Web service the procedure to collect QoS characteristics and compute specified performance metrics has been put in place. The simulation scheme was also adapted to simulate user behavior transitions randomly and realistically.

The utility of the model to relate SP’s capacity to SR’s attributes quantifiably has been emphatically brought out through the simulation. The need to scale up the token basket size to match system capacity and have the provision to accommodate the spread of user request size has been clearly established through simulation. Representing peak loads as impulse loads riding on the stationary load, the changes in the system performance were brought out through simulation.

The external factors that may affect the response time have been explored; discusses the techniques that can minimize the delay in response time without sacrifice of the QoS (i.e., without service degradation). The thesis also presents a scheme that performs a comprehensive multi-factor authentication process based on QR-Code based OTP, satisfying the important requirements including friendliness, resistance to various kinds of sophisticated attacks, and stolen credentials that is particularly useful at times of peak loads on servers.