CHAPTER 2

RELATED WORK

2.1 Introduction

Internet Services are complex systems; hence the challenges in designing these services, ways to model the services, workloads, and metrics to evaluate performance are to be well understood. For that purpose, this chapter elaborates the literature relevant to the work. The terms Internet Service and Web application essentially mean the same as far as this work is concerned. However both these are retained here for contextual convenience.

Several researchers have developed models for deriving the resource requirements of hosted applications.[37] presents SEDA architecture which uses event queues tied to different stage of request processing and facilitate dynamic resource re-deployment ensuring efficient system utilization(memory and CPU usage). [21] describes a simple admission control mechanism based on bounding the length of the Web server request queue. This work analyzes various settings for the queue abatement and discard thresholds, but does not specify how these thresholds should be set to meet any given performance target. [8] demonstrates the feasibility and benefits of overbooking resources in shared hosting platforms. The work proposes techniques to overbook (i.e. under provision) resources in a controlled fashion based on application resource needs. It employs a kernel-based profiling mechanism to empirically monitor an application's resource usage and proposes techniques to derive QoS requirements from this observed behavior. It shows that with more bursty resource needs of an application the benefits derived from resource overbooking are also high. [6] discusses sophisticated queuing models based on networks of queues to capture multi-tier applications. The model provides predictable allocation of CPU based on the specified resource requirements. End- to-end request profiling is done by the *capsule* dedicated to that tier. [7] presents an approach to reduce the policing overhead. The key idea behind this technique is to periodically *pre-compute* the fraction of arriving requests that should be admitted in each class and then simply enforce these limits without conducting any additional per-request tests.

Findings

Many of the services that we use daily, for example banking, have transformed from traditional customer services into Internet Services. With computing becoming pervasive, people increasingly rely on public computers to do business over the Internet thus making it a preferred environment for a multitude of E-Services like E-Commerce, E-Banking, etc. Security for these applications is an important enabler. Internet Services experience huge variations in service load, with bursts coinciding with the times that the service has the most value. As services that contain sensitive data are moved to Internet, strong authentication is required to provide a high enough level of security and privacy.

Specifically the focus of this work is to provide a generic black-box model based framework that achieves robust performance on a wide range of Internet Services subject to variation in load using a '*Tokens-Basket*' data structure over and above the server's service architecture. It conditions overload at different application processing stages using application aware admission control strategies. The model captures concurrency limits of Internet Services which is important due to its impact on the performance of these applications and their clients. The work also discusses how the proposed model can be used to quantify service availability based on drop-rate, latency and response time. Additionally, it suggests an authentication scheme that can contribute for service availability metric.

2.2 Overview of Internet Services

At their core, Internet Services are computer programs that use the Internet as a medium to facilitate communication between the Browser (client) and the Web Server processes. The general architecture of hosting service platforms used for servicing Internet Service requests has already been presented in the previous chapter (Figure 1.1). The 3-tier architecture of the Internet Services has been widely modeled in literature. For study of scalability issues which is in focus here, the model in [53] is apt and has been chosen as the basis for analysis. A brief description of the same follows:

The hardware platforms, operating systems, and middleware that Internet Services use make a server system. Upon receipt of a request, the service computes a response based on the request's contents, and sends the response back over the network. Each process – on the client as well as the server- executes on the respective dedicated hardware.

When a HTTP request is sent via the Browser, it is first received by a Web Server. After parsing the request, the Web Server decides on the subsequent processing steps. Static-content such as HTML, images, and client-scripts (like Java Script) does not need further processing at the server-end; if requested, the same is sent in an HTTP response back to the client from the Web Server [12, 14, 27]. If further processing is required, as in the case of dynamic-content (PHP, ASP.NET, Java EE, etc.,), the Web Server forwards the request to the Application Server or 'Language Runtime' (JVM,VirtualBox, etc.,). After the processing and generation of the dynamic content, the Application Server reverts to the Web Server which in turn replies back to the Client. For data retrieval or modifications, the Application Server executes queries at the backend-Database Server, processes the received query results, and finally sends the reply back to the client through the Web Server.

Findings:

The popularity of the Web and its advantages as a client-server platform has led to countless Web applications.

2.3 Challenges in designing Internet services

The challenges in scalability of Internet Services are four fold:

a. *Popularity of Web and Increased number of users*: Increase in the number of users – scaling – results in complexity; the same with its associated dimensions is depicted in Figure 2.1.

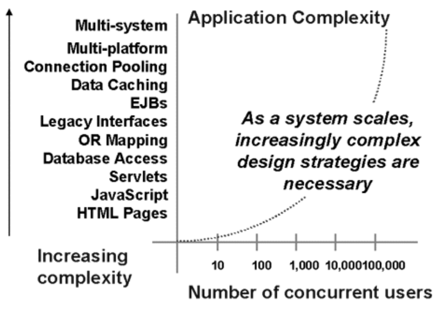

Figure 2.1: Complexity in Design of Internet Services (Large Scale == Complex)
[*Source: How i-Commerce Design Issues & Solutions, A3T, GemStone Professional Services, White Paper Series*]

b. *Lack of organized integral framework*: The current Internet has evolved from an academic network to a broad commercial platform. The emerging demands for security, mobility, content distribution, etc. are hard to be met by incremental changes through ad-hoc patches. New clean-slate architecture designs based on new design principles are expected to address these challenges [22].

c. *Gap between programming paradigms and underlying hardware*: The increase in processor-memory speed gap leads to increase in cache misses and leads to degraded server performance [52].

d. *Platform Features*: Resource virtualization in existing operating systems presents a number of challenges for scalability and load conditioning by restricting participation of applications in resource management decisions and failing to indicate the scarcity of the resources. The support for handling load spikes depends upon threads/processes supported by the operating system [55].

Findings:

The increase in value of Internet Services, and the reliance of users on them for day-to-day activities, automatically implies an increased level of expectations. Users expect 24 X 7 service availability and reliability with acceptable performance (quality of delivery).

2.4 Performance Evaluation

Performance evaluation provides metrics (throughput, resource-utilization, response time, etc.) to discover the system's bottlenecks, which altogether aid in finding performance-related problems in the design [18, 39]. Measurement, simulation modeling, and analytical modeling are the three main approaches used in studying of a system's performance [9]. *Measurement* is a direct assessment of the actual or representative system under varying workloads, providing the most accurate results. *Simulation* and *analytical* modeling are performance modeling approaches. *Performance* modeling encapsulates the performance characteristics of a system that are available from its design in a performance model. In s*imulation* modeling, a software representation of the system and its interactions are developed, and statistics collected from the model software's execution provide the performance metrics [26]. *Analytical* modeling uses a system with variables representing the system entities, relating them through equations, and studying them through analysis, and simulation. An alternative is modeling as a *black-box*, where the arrival pattern and request processing are modeled as stochastic processes, defining the service time of request as a random variable with an associated distribution [13].

Findings:

Simulation models provide high accuracy; however, the time required to develop and simulate the models can be considerable for large systems with need of highly accurate predictions [9, 26]. Including this, the manageability of these models is challenging, as modification made to the system design need to be incorporated to the model, which involves software code modifications; but this is non-trivial.

With *Measurement* based modeling the setup cost is very high. *Analytical* models are based on collection of logs from server and using mathematical models to study the collected log information and predict the performance in future. Although this type of modeling can be used for predicting the bottlenecks at various resources, these are not suitable to model the behavior of users (randomness in the behavior of users) which forms the most important part of Web application modeling. *Simulation* modeling is based on usage of simulators to work with virtual workload and user characteristics. The difficulty in using simulators lies in customizing the profiles to be collected and tuning input parameters according to application requirement using the language of the simulator. The *black-box* approach allows reaching a greater scalability in the performance evaluation by means of simulation.

2.5 Modeling Web applications

Common factors affecting the system performance include the application characteristics, workload properties, user behavior, system management policies, and available resources in the hosting platform model. The work presented here facilitates one to model Web applications based on all of the above factors.

2.5.1 Application Characteristics

Finite state machines (FSMs) provide a convenient way to model Web application characteristics [3]. Theoretically, Web applications can be completely modeled with FSMs as the Web pages have the data, if data is there then they have the states (called as stage here) and if states are there, they are represented using the FSMs. By modeling Web applications with FSMs, it becomes possible to create models that present deterministic responses for unanticipated actions in any condition or situation. In general a Web application can be modeled using the following four steps:

1. The Web application is partitioned into clusters.
2. Logical Web pages/cluster/Web application component are defined.
3. FSMs are built for each cluster.
4. An Application FSM is built to represent the entire Web application.

The general term 'cluster' is used to refer to collections of software modules and Web pages that implement a logical, user level, function. The first step partitions the Web application into clusters. At the highest level of abstraction, clusters should be abstractions that implement functions that can be identified by users. At a lower level of abstraction, clusters should be collections of cohesive software modules and Web pages that work together to implement a portion of a user level function. At the lowest level, clusters may be individual Web pages associated with software modules that represent single major functions. Such repeated 'finer graining' should get clearly reflected in the model as well to facilitate study at each level. Key to modeling Web applications as FSMs is to be able to generate FSMs that are both useful and reasonable in size. FSMs for a variety of Internet Services like online voting, E-Commerce, online bidding, and surfing are shown in Figures 2.2a, 2.2b, 2.2c, and 2.2d respectively

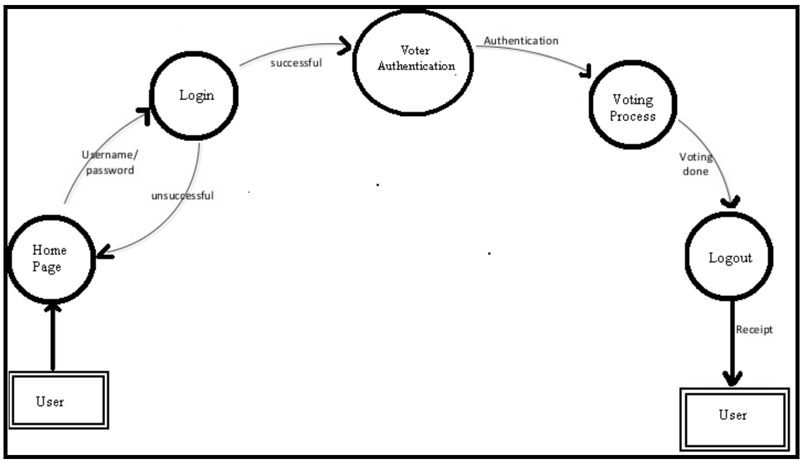

Figure 2.2a: FSM for an Online Voting System

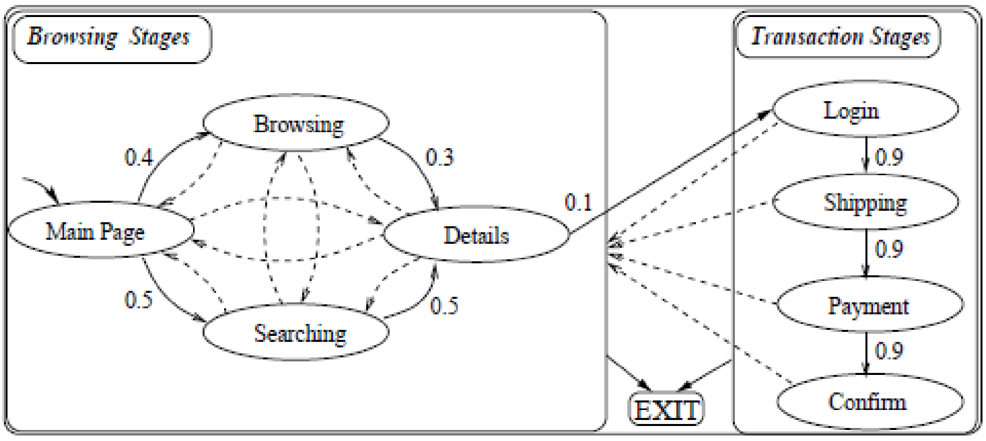

Figure 2.2b: FSM for an E-Commerce System

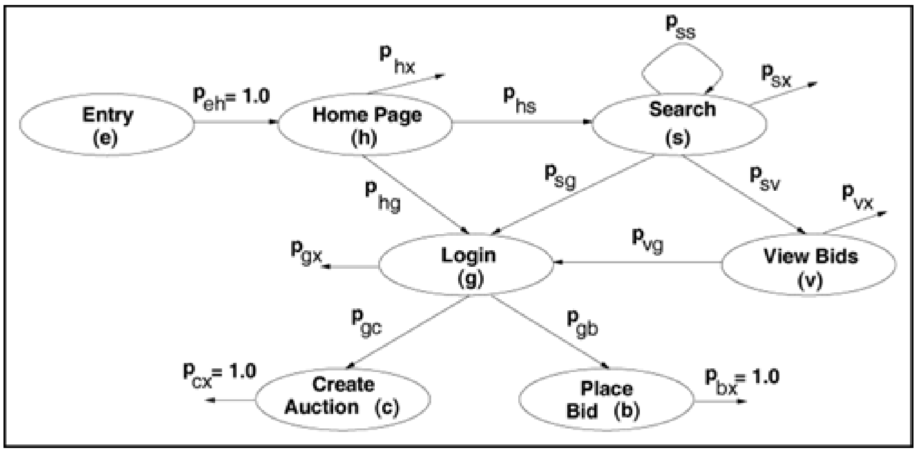

Figure 2.2c: FSM for an Online Bidding System

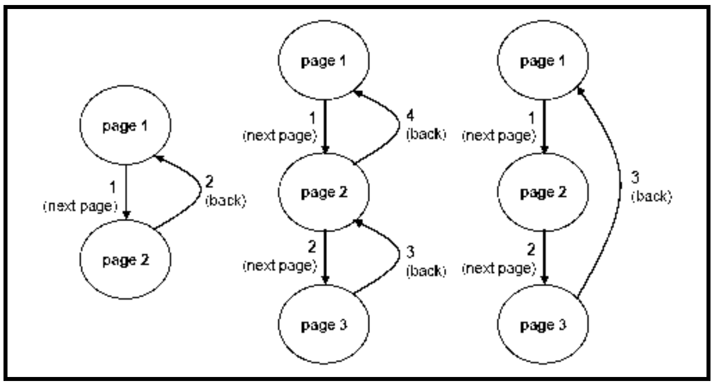

Figure 2.2d: FSM for Internet Surfing System

Findings:

A close study of the FSMs reveals the following patterns in state transitions:

Pipeline: Set of states arranged in a serial form as shown in the Figure 2.3. This helps in achieving parallelism. This approach also allows varied admission control decisions to be made based on the resource needs of each particular state.

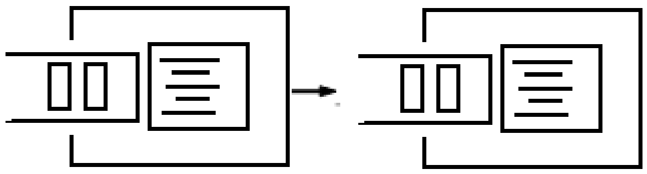

Figure 2.3: Pipeline Pattern in FSM

Partition: In partition pattern the states are arranged in a fashion as shown in Figure 2.4.

i. Do task at State1.
ii. If some specified condition holds, do task State2.
iii. Otherwise, do task State3.

The approach allows varied admission control decisions to be made, for differentiating service quality of the offered service.

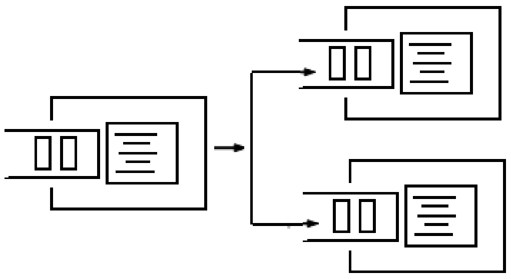

Figure 2.4: Partition Pattern in FSM

Combined: The *Combine* pattern showed in the Figure 2.5 form steps to combine two states into a single state. It is the inverse of, the *Pipeline* and *Partition* states, and is used to aggregate the processing of separate states. The pattern aids in reduction of code complexity.

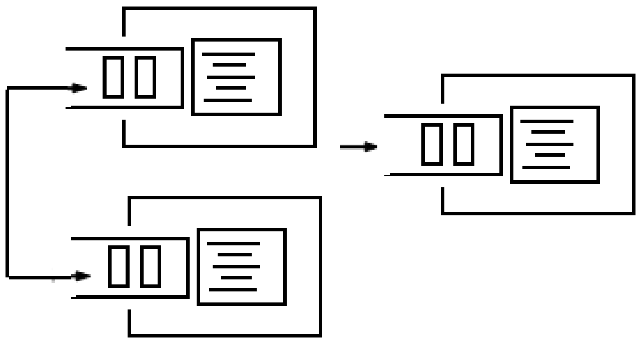

Figure 2.5: Combine Pattern in FSM

Wrap: Used to tie dedicated resources to a state. The alternative is to have a system wide common resource pool.

2.5.2 Workload Characteristics

Workload characterization has a significant impact on performance evaluation [56]. Understanding the nature of the workload and its intrinsic features can help to interpret performance measurements and simulation results better. Web Server workload characterization can be performed at different levels as shown summarized in Table 2.1.

Table 2.1: Workload Characterization Levels

Levels	**Description**
Request Level	Request arrival process, file type, file popularity, file size, and other aggregated workload features are characterized
Function Level	The functions provided at a Web site are analyzed
Resource Level	The usage of the system resources is analyzed
Session Level	The client sessions are identified and characterized

Most published studies on workload characterization for Web Servers are at the request level. Some more recent studies on Web workloads, in particular, on workloads to E-Commerce servers [41], provide more updated characterization at request, function, resource, and session levels. Workload modeling quantifies Web application portal-load, which consists of user request arrivals and their resource requirements. User request arrivals can be recorded from Web Server logs and resource requirements can be obtained by profiling the live system and gathering data. Using them, both arrival and resource usage can be characterized using probability distributions [25] to create the workload model.

The basic inputs to analytical workload characterization models are parameters that describe the request patterns, their inter arrival times, service demands, and execution mix. There are two main approaches to generate a stream of requests (workload) namely:

- TRACE –BASED: Workload is collected from data recorded in a Web Server log file. It may not be representative in general.
- DISTRIBUTION-DRIVEN: Workload characteristics are given by Probability Distribution Functions (PDFs). The resulting sequence has a synthetic nature but can be easily modified by changing distributions or their parameters. Distributions are mainly characterized by special parameters that govern the spread out and the shape. Models for network traffic need to consider the known burstiness, flash crowds, and the length of processing. Ideally it has to follow a distribution that matches with real time logs collected from servers.

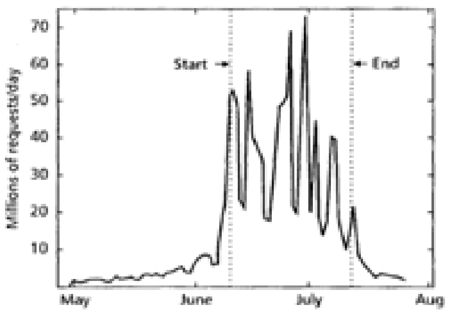

Figure 2.6a: Daily Traffic Volume of 1998 World Cup Website
[*source: Martin Arlitt,Tai Jin Internet Systems and Applications Laboratory HP Laboratories Palo AltoHPL-1999-35(R.1) September, 1999*]

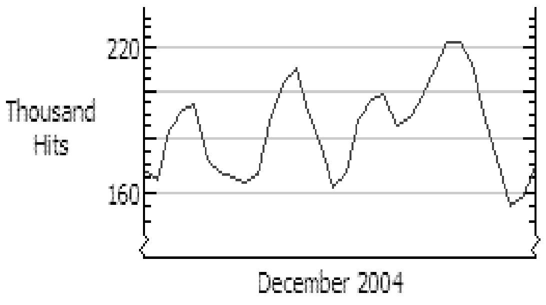

Figure 2.6b: Web Traffic at Wikipedia in December 2004
[*source: https://en.wikipedia.org/wiki/Web_traffic*]

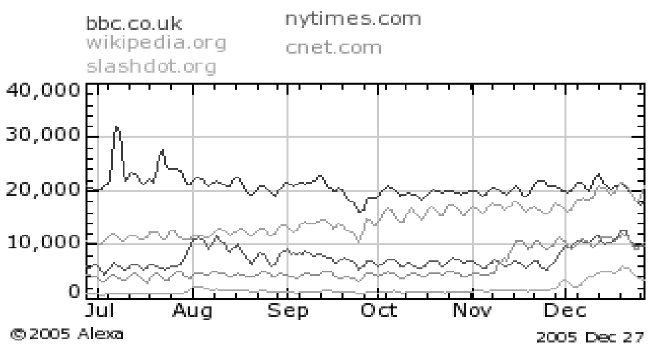

Figure 2.6c: Traffic in Different Sites on 27 December 2005
[*source: https://en.wikipedia.org/wiki/Alexa_Internet*]

2.5.2.1 Peak Load

The workload shown in Figures (2.6a, 2.6b, and 2.6c) depicts the randomness in the nature of the arrival pattern at the server. It also brings out the fact that at times of significant events, the server experiences peak load (workload that are several times greater than the normal workload) but not much of change in average workload before or after the event. Further, heavy workload may induce server thrashing and service unavailability. In E-Commerce applications, such server behavior could translate to sizable revenue losses. For instance, [55] estimates that between 10% and 25% of E-Commerce transactions are aborted because of slow response times, which translates to about 1.9 billion dollars in lost revenue. It is possible that the surge of malicious traffic electively blends in with legitimate or normal traffic, making it difficult to be weeded out causing starvation for the unaware clients. The annual infrastructure security report compiled by Arbor Networks [36] shows that DoS and Bots top the chart as the primary security concerns at 46% and 31% respectively out of 55 requests.

Findings:

A study of traffic patterns of a wide category of Internet Services brings out the following properties:

a. Normally each pattern can be modelled as a stationary process.
b. The process can be characterized in terms of a mean value, a variance, and a distribution.
c. At times sites are subject to peak loads which can be treated as ‘impulse load’ superposed on steady load.

The above are well brought out through the representative patterns in Figures 2.6a, 2.6b, and 2.6c. Filtering these malicious requests at an initial stage would enable better service availabilities. [56] presents a comprehensive workload characterization study of Internet Web Servers. It discusses different characteristics like document type, access patterns, size of file, remote requests, etc., Although there have been many changes in the web technologies in terms of protocols, scripting languages etc many of these still hold true today.

2.5.3 User Behavior

A deeper understanding of the adoption rates, purposes, and characteristics of Internet users in different services offers multifold benefits; these can be: developing better digital infrastructures, maximizing profits in E-Business etc. User visits to a Web site are represented by the browsing trajectories categorized as a session [14, 15, 54]. A session is characterized by a sequence of pages visited by user and the time spent by the user. If sessions are not explicitly given they must be reconstructed from the Web logs. User behavior patterns are obtained using clustering techniques and represented as User Behavior Modeling Graph (UBMG) in graphical form or in matrix notation [14, 54]. UBMG is a state transition graph, where states denote results of service requests (Web pages / user functions), and transitions denote possible service invocations. Transitions in UBMG are governed by probabilities $p_{i,j}$ of moving from state i to state j. Web workloads consisting of several different UBMG session structures can approximate any given sequence of user requests (Web request log) as ‘close’ as desired, by appropriately choosing the model parameters (i.e., the number of UBMGs and their transition probabilities). The greater the number of UBMGs in the workload model, the closer such an approximation can be made. In Web workloads consisting of several UBMGs, each one of them represents a typical navigational pattern exhibited by the service users. [24] proposes a data mining method that allows online businesses to cluster their frequent transactions and item sets. This methodology can be used to identify user behavior in online shopping sites.

Findings:

A close study of user behavior model reveals the following:

1. The Web user browsing behaviour can be described by three kinds of data: the Web structure, the Web content, and the Web user session.

2. Profiling processing for Web user's usage can be described from two different points of view depending on, whether the profile is pre-established or is created on the run. The collected profile of the user can be used to derive various metrics that could enhance the business performance (like visit–to–buy ratio, number of customers who left the site after adding at least one item to the cart in E-Commerce application). In fact this graph based profiling method could be used as recommender systems for enhancing business functionalities and Web site functionalities.

2.5.4 Performance Management Strategies

With Internet Services becoming indispensable both for people and especially for companies this importance is translated into strong requirements in the performance, availability, or reliability of these applications. Recent research has comprehensively explored mechanisms for managing the performance of Internet applications, with special focus on dealing with overload situations and providing QoS guarantees to users. Distinct schemes have been proposed, each of them tackling the problem from a different perspective or focusing on a specific scenario [23].

Findings:

Techniques in this category are mainly request scheduling, admission control, service differentiation, dynamic resource management, service degradation, and almost any combination of them. These techniques basically cover all the actuations that the SP can undertake when an increase in an application demand jeopardizes its performance guarantees, namely allocate additional capacity to the application by assigning idle or under-used resources (dynamic resource management), degrade the performance of admitted requests (if further degradation is allowed by the agreed SLA) in order to temporarily increase the effective capacity, or turn away excess requests (admission control), while preferentially admitting more important requests (service differentiation) and giving preferred service to them (request scheduling).

2.5.4.1 Metrics for Performance Evaluation

QoS is a critical element for achieving the business goals of an SP, for the acceptance of a service by the SR. The metric measurements are mainly used to describe the service quality capabilities or requirements of an SP or SR. SLAs encapsulate QoS characteristics and functional service properties that address the following dimensions: performance, efficiency, reliability, recoverability, permissibility, and availability as an important set of quality factors for the majority of Internet Services [28]. Considering the role of business processes in transforming inputs to outputs, factors such as throughput, timeliness, and execution costs are considered as quality factors of performance. The model to measure performance must capture the peculiarities deriving from the intended use of the service; to facilitate this, the non-applicable attributes and their categories are to be removed and focus to be laid on specialized attributes.

Findings:

The major metrics used for evaluating Web applications as discussed in [28] are given in Table 2.2.

2.5.4.2 Service Level Agreements

The SLA is a part of a service contract where a service is formally defined. Particular aspects of the service - scope, quality, responsibilities - are agreed between the SP and the SR. The Service Level Objective (SLO) is described with defined SLA parameters. The SLA parameters include QoS, service level priority, parameter weights, and other high-level parameters obtained by computing existing SLA parameters to evaluate the overall operation quality, such as the parameter of service availability. The SLA violation process sets timer/content and means of the penalty to be paid by SP to the SR in case the SP fails to provide the expected service level. This violation process is supposed to ensure the fairness, legality, and effectiveness of SLA negotiation and is very important in the negotiation process. It also helps to build fair rewards and punishment mechanism and protects user’s benefits as well. The task of SLA decomposition is to find the mapping of overall service level goals to the state of each individual component involved in providing the service (e.g., resource requirement and configuration).

Findings:

Given high level goals, SLA decomposition translates these goals into bounds on low level system metrics such that the high level goals are met.

Table 2.2: Service Quality Attributes for Internet Services

Service Quality Attribute	**Description**
Response Time	Time taken for processing a single request(includes latency, wait)
Processing Time	Actual processing time for a request
Latency	Time lag between arrival of request and commencement of service
Earliest Start time	Guarantee expected by SR to keep latency lower
Latest Finish time	Guarantee expected by SR to keep Response time lower
Jitter	Measure of internal consistency of the timeliness parameters
Throughput	Number of requests served per unit of time.
Availability	Ratio of the Number of requests accepted for service to the total number of request load over time
Accessibility	Parameter provided by SR to determine service quality
Scalability	Ability to add new functionalities to service
Stability	Ability of system to resume normal functioning after deviating from normal
Resource Efficiency	Indicates the usage of system resources
Data Maturity/age	Historic data that facilitates in determining service Quality
Believability	Measure of deviations from those specified by SR

2.6 Web Authentication

User authentication is one of the fundamental procedures to ensure secure communications and share system resources over an insecure public network channel. A simple and efficient authentication mechanism is required for securing the network system in the real environment. The design of authentication methods raises crucial questions on how to solve conflicts between security and usability goals that are at opposite ends of a "see-saw".

2.6.1 One Time Password (OTP)

A one-time password (OTP) is a password which is usable and valid for only one login requirement/transaction. The number of shortcomings associated with static passwords is avoided by OTPs. The hackers will not able to perform replay attacks because of the following reasons:

i. The captured old OTP will be no longer valid once it already used to login to user account.
ii. It is only valid for a short time.

The pseudo randomness forms the basis for OTP generation algorithms and this is much needed or otherwise people can easily predict the next OTPs by analyzing the previous ones. OTPs are difficult for most of the people to memorize so they require more advance technology to get this done.

Findings:

User authentication is a very important security mechanism for many network applications. Traditional “static” password authentication techniques are widely used due to their convenience. However, they often suffer from eavesdropping, replaying, and guessing attacks. Moreover, many malicious programs like Trojan Horse may steal passwords from the victim’s computer, no matter what applications or protocols these passwords are used for. OTP authentication, in which every password is used only once, provides stronger user authentication via “dynamic” passwords than traditional static password authentication. Once an OTP is used, it will be no longer valid even if it is eavesdropped, replayed, guessed, or stolen.

2.6.2 Short Message Service (SMS)–One Time Password (SMS OTP)

SMS based OTP were introduced to counter phishing and other attacks against authentication and authorization of Internet Services. In these scenarios, SMS OTPs are mostly used as an additional factor in a multi-factor authentication system. Users are required to enter an OTP after logging in with a user name and password, or the OTP is required to authorize a transaction [43, 67, 68, 69, 70, 71, 72, 73]. The prime example of SMS OTP is the mobile Transaction Authorization Number (mobile TAN or mTAN) that is used to authorize transactions for online banking services. OTPs are quite popular as an additional authorization or authentication factor in Web-based services. These passwords can be utilized to authenticate a user, i.e., the user needs a valid OTP to prove his identity to log into a Web application or to access the company’s private network [43, 67, 68, 69, 70]. SMS OTPs are also used for account verification, e.g., Google App Engine [71] and Google Mail [72].

Findings:

The SMS-OTP mode requires a one-time registration. The OTP generated is unique and can only be used once upon each login. All service providers that are on GSM network and provide SMS services support the SMS-OTP.

2.6.2.1 Problems with SMS–OTP

The main problems with the SMS-OTP design are under overloaded situations; these are:

- Delay in delivery of SMS.
- Low Coverage Areas.
- Downtime with SMS Gateway.
- Non-availability of service for roaming user.
- High Cost for roaming user.

Findings:

Inability to receive SMS-OTP may be due to various reasons like–the mobile service providers' message centre’s experience congestion at that point in time, one may be out of the mobile service providers' network coverage area, one’s SMS inbox is full etc.

2.6.3 Authentication using Quick Response Code (QR-Code)

In 2002 [29] suggested the usage of camera-based devices as an alternative but more secured authentication method for critical transactions with un-trusted computers. With the explosive growth in the amount of camera-equipped smart phones around us mobile based authentication [29] may become a popular authentication method in the near future. QR-Code (a two-dimensional barcode) was introduced by the Japanese company Denso-Wave in 1994. Its error correction capability facilitates data restoration even under conditions when substantial parts of the code are damaged. Modern cellular phones are natively equipped with the QR-Code decoding software. Fortunately, for camera phones that are not equipped with QR-Code readers, *Quick-Mark and i-nigma* are free tools that are available for many manufactured models and devices to decode QR-Codes free of cost. Depending on the data recognized and the nature of the application a variety of operations can follow its decoding stage. There are many advantages to use the QR-Code [29] in mobile phones such as omni-direction readability and error correction capability.

Findings:

Recently much research has been done that focus on application of QR-Code and advancement of the technology for providing better user experience. Usage of QR-Code does not require extra hardware and offers increased convenience for the user. All that is required is a mobile phone with a camera which most Internet users already have.

2.7 Summary of Findings

Web applications are becoming increasingly complex and mission critical. To help manage this complexity, they need to be modeled. The result of a modeling process should be models representing the relevant aspects of the system in a simplified and ideally comprehensible manner. FSM is a widely adopted method for modeling Web applications. Models for Web applications need to consider the user behavior characteristics, workload characteristics, metrics of interest to business, underlying hardware, software, network characteristics, and dependence on external services for assessing the performance.

2.8 Conclusions

Prior to building a model to evaluate Web applications it is important to understand the features to be modeled. Hence, an insight on the overview of Web applications has been carried out in this chapter; ways of modeling Web applications, and the factors to be considered in modeling Web applications have been clearly brought out with relevant references. The thesis presents a model that evaluates the performance of an Internet Service which can aid SP in capacity planning to improve service availability considering factors affecting the system performance like application characteristics, workload properties, user behavior, and resource allocation policies, and available resources in the hosting platform. The following chapters are devoted to the development of the model and its use to study and fine-tune Web applications.