CHAPTER II

THEORIES AND LITERATURE REVIEWS

In this chapter, the adaptive QoS system and incoming request scheduling are briefly revised. Their related concepts and some literature reviews are also presented.

2.1 Adaptive QoS system

An *adaptive application* is one in which the application changes its behaviors according to the perceived constraints in the environment, so as to maintain the semantics (or expected behavior) of the application for the user [8].

Bechler et al. [9] expressed that a common application does not perform any actions to adapt itself to the current availability of resources. If a resource becomes scarce, the quality of the application that uses this resource will be degraded. Compared to common application, the behavior of adaptive application is more intelligent. These applications are characterized by the ability to react to changes in the quantity of resources. Adaptive applications support the user with the best service that is possible under the current circumstances. This behavior is advantageous in the mobile environment with varying network characteristics. Note that adaptation is a passive and thus reactive process. The application tries to support the best *Quality of Service (QoS)* that is possible by utilizing the available resources. Thus, adaptation can deliver an improved QoS support to the user. Changes in the environment do not necessarily cause a degradation of the application QoS support. Even if a degradation of QoS is unavoidable, e.g., as a result

of too limited resources, the level of QoS support is also improved compared to common applications. Additionally, an adaptive application can decide to terminate itself if the monitored application specific parameters that represent the quality of service fall below a predetermined threshold. They proposed the implementation of adaptive and proactive applications supported by their feedback architecture. User of this architecture can interact with the system without the technical knowledge through simple and intuitive user interface.

Suthon et al. [10] presented that the Internet Engineering Task Force (IETF) has proposed several standard service models and mechanisms to meet the demand for QoS. Notably, there is a widely used QoS reference model merging these technologies. This includes the models combining Integrated Services (IntServ)/RSVP in access networks and DiffServ in the backbone networks. The IntServ/RSVP architecture intends to provide end-to-end bandwidth reservations by maintaining per-flow state information along the path from the sender to the receiver. The main disadvantage of IntServ is that it is not scalable to large networks. This scalability problem resulted in the DiffServ approach where QoS is achieved by assigning packets to certain service classes according to QoS level defined in IntServ flows.

Kurose and Ross [11] explained that scalability of RSVP and flexibility of IntServ frameworks also stated as the considerations that lead to DiffServ development. The DiffServ criticisms also presented that in order to provide end-to-end differentiated service, all the ISPs between the end systems must provide differentiated service and cooperate and make settlements in order to offer end customers a truly differentiated service. Without this kind of cooperation, the criticism occurred. Another criticism of DiffServ are complex and cost to police and authenticate to prevent cheating when the users differentiate. New billing system for the differentiate services also needs, most likely by volume rather than with a fixed monthly fee as currently done by most ISPs. Finally,

some researchers feel that even if an end-to-end DiffServ were in place, most of the time, there would be no perceived difference between a basic and a priority service. Indeed, today, the end-to-end delay is usually dominated by access rates and router hops rather than by queuing delays in the routers.

This Section presents the related knowledges that are the fundamental principals of the adaptive QoS system, such as wireless Internet infrastructure, Quality of Service, Service License Agreement, software agent technology.

2.1.1 Wireless Internet Infrastructure

The dramatically changes in using mobile wireless communication (which will refer as 'wireless' throughout this paper) lead to the introduction of new services to the users. The wireless technology is applied in many places, such as offices, sport centers, entertainment complexes, homes, etc. There are many kinds of wireless terminals, such as laptop computers, mobile phones, personal digital assistants (PDAs), etc. Wireless communication technologies; Wireless LAN, Bluetooth, cellular technologies, satellite and more; are widely used as infrastructure. There are many innovations which drive each generation of wireless communication as shown in [12]. Voice services was driving the first-generation (1G) based on analogue technologies. Developing of digital technologies has driven the second-generation (2G) with adding data services. In the third-generation (3G), data services will play more roles with guaranteed quality of service (QoS) and more data capabilities.

Evans and Baughen [13] expressed that the aim of 3G, 'to provide multimedia multirate mobile communications anytime and anywhere', can only be partially met. It will be uneconomic to meet this requirement with cellular mobile radio only. Fourth-generation (4G) must itself be dynamic and adaptable in all aspects of the users' traffic,

air interfaces and terminal types, radio environments, quality-of-services types, and mobility patterns, with built-in intelligence. The vision of next generation network is to provide a seamless IP-based core network connects to the distinctive radio access.

Infrastructure of *wireless Internet* communication can be separated into two parts [1, 2, 3, 4]; wireless network and wireline network. The wireless terminals connect to wireless networks, and an Internet application server connects to wireline network. Both parts are attached together through the Internet system. In wireline network, the network components are connected together with lines and exchanged the information with fixed and reliable relationship. While in the wireless network, wireless terminals connect to base stations through air with limited bandwidth and unreliable. The examples of wireless network problems are range coverage, interference due to spectrum reuse, and capacity limitation [1]. The wireless environment characteristics are long latencies, highly variable delays, and sudden disconnections. All of these characteristics create problems that users have never met in the wireline networks [14].

Internet and Internet Protocol (IP) technologies are the motivators that increase the usage of wireless communication which distinguishes wireless Internet from other mobile/wireless communication systems. Formerly, there was only voice service via cellular phone and short message service. After merging with the new technologies, there are various additional data services, such as web browsing, electronic mail (e-mail), and streaming applications. All of these communication services are transferred to work on wireless terminals with high expectation from the users to have the same service quality as working on desktop computers [13].

Furthermore, Internet application server must serve requests from many wireless terminals simultaneously. Each request works independently. Bottleneck problem occurs when they compete for limited server's resources. Different clients, different requested applications, and diverse environments lead to different requirements that the server

must serve. An adaptive QoS aware application is required to address the above problems under the wireless terminals' constraints [2, 5, 6].

2.1.2 Quality of Service Concept

Quality of service (QoS) is defined in the context of customers/users or service providers. In the context of the customers or the users, it is defined by attributes or criteria which are considered to be essential in the use of the service. In the context of the service providers, QoS is defined by parameters which contribute toward the end-to-end performance of the service, this end-to-end performance reflecting customer's requirements [15].

Gronroos [16] defined quality within the service environment as:

$$Quality = Customers' Expectations - Customers' Perceptions \tag{2.1}$$

In this context, QoS is a differentiation of the perceived performance (P) and the expected performance (E). If the perceived performance more than the expected performance the quality is qualified the user and is dissatisfied if the perceived performance is less than the expected performance. If the perceived performance have nearby value to the expected performance, the fair situation will occur.

Oodan et al. [15] expressed that QoS characteristics represent some aspects, services or resources which can be identified and quantified, of the system QoS. The QoS-management activities drove by user requirements which originate with an application process that wishes to use a service. *QoS management* includes functions to meet the users and applications. These functions are called *QoS mechanisms* which may require many different types of actions to be performed, such as negotiation, admission control and monitoring.

Quality cycle is a component of a framework to facilitate the study of QoS in telecom-

munications in 1994 defined by ETSI (European Telecommunications Standard Institute) [7]. The quality cycle is based on two principal parties: customers and service providers, as shown in Figure 2.1. QoS requirements by the customer are the statement of parameters and the level of quality of a particular service. QoS offered by the service provider is a statement of the level of quality expected to be offered to the customer by the service provider. QoS achieved/delivered by the service provider is a statement of the level of quality provided to the customer. The QoS delivered parameters should be the same as the QoS offered parameters for comparison reason. QoS perceived by the customer is a statement expressing the level of quality which they 'believe' they have experienced. The perceived QoS is usually expressed in terms of degrees of satisfaction and not in the technical terms. QoS problems are occurred when there are gaps in the QoS cycle. These gaps can be classified as an alignment gap, an execution gap, a perception gap, and an value gap.

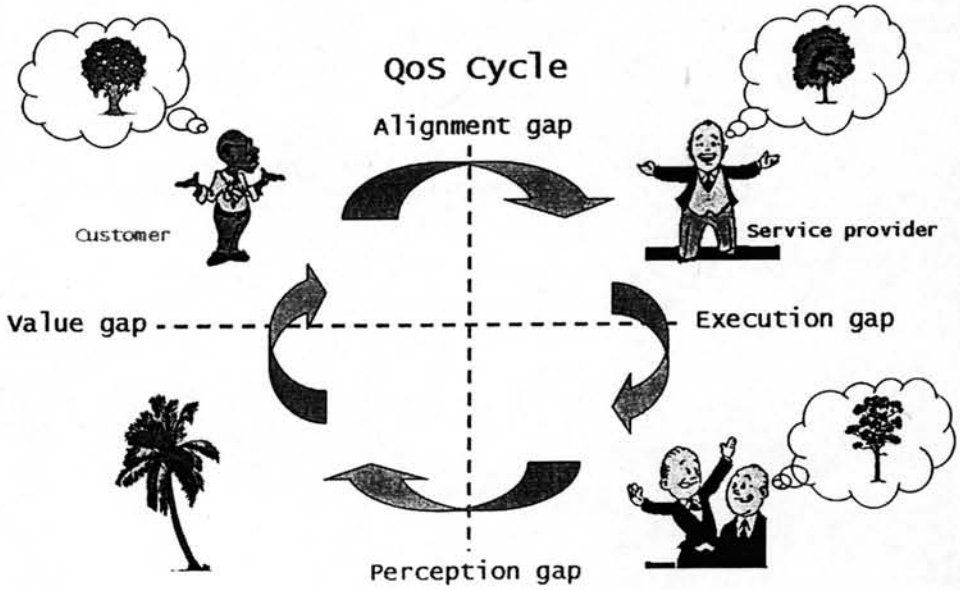

Figure 2.1: Quality of service cycle.

Quality of Service Management

QoS problems are occurred when there are gaps in the QoS cycle as shown in Figure 2.1. If the gap between each step is large, the different values between the expected QoS and the perceptible QoS will be increased according to (2.1). To solve this problem, QoS management functions are required.

Aurrecoechea et al. [17] introduced that QoS mechanisms are selected and configured according to user-supplied QoS specification, resource availability and resource management policy. In *resource management*, QoS mechanisms can be categorized as either static or dynamic in nature. Static resource management, which described as QoS provision, deals with flow establishment and end-to-end QoS renegotiation phases and dynamic resource management, which described as QoS control and management), deals with the media-transfer phase. The distinction between QoS control and QoS management is characterized by the different time scales over which they operate. QoS control operates on a faster time scale than QoS management. The fundamental QoS management mechanisms include the following: QoS monitoring, QoS availability, QoS degradation, QoS maintenance, and QoS scalability (which comprises QoS filtering and QoS adaptation mechanisms).

Chalmers and Sloman [18] summarized that the *QoS management functions* can be categorized into two groups, static and dynamic. The static functions apply to the system at the initiation of an interaction, such as specification, negotiation, admission control, and resource reservation. The related parameters in this group usually unchanged throughout the processes. On the other hand, the parameters that can be changed during the processing time will be served by the dynamic QoS management functions which are applied continuously or as needed during an interaction. Monitoring, policing, maintenance, renegotiation, and adaptation are the functions in this group.

QoS Metrics

The generic *QoS metrics*, which is commonly used when assessing network performance at application level, are described in [19, 20, 21] as the following:

- *End-to-End Delay (Latency)*: a time delay while one waits for something to happen. A widely used measure of network delay is round-trip time (RTT), the time for a packet to make the round trip from a client to a server and back. Miras [21] suggested that this factor has a direct impact on user satisfaction.

- *Delay Jitter* or Delay Variation: the variation of time between one delay and the next delay. The multimedia data cannot be played smoothly under high delay jitter. Therefore, the delay jitter must be controlled within an acceptable value.

- *Throughput*: the rate at which data is sent through the network, usually expressed in bits per second (bps).

- *Acceptable Error Rate*: Multimedia data, such as video information can tolerate a higher error rate, but data files or business transaction information cannot tolerate any error. Thus, to maintain an acceptable error rate for applications is one of the QoS demands criteria.

- *Packet Loss Ratio*: Most multimedia data can tolerate a higher packet loss ratio, but data files, business transaction information or massive real-time data cannot tolerate any packet loss. Therefore, to maintain an acceptable packet loss ratio for applications is also one of the QoS demand criteria. The last two factors, accept error rate and packet loss ratio, can be determined as availability and continuity of service in [21].

- *Denial of Service (DoS)* and *security*: these two factors cause unavailability and untrusted communication services.

Gurijala and Molina [22] focused on the application level QoS metrics. They assigned QoS metrics to the corresponding class of application as shown in Table 2.1. They also suggested high-level measurement methods for each of the QoS metrics given in the Table that end-to-end delay, jitter, packet loss ratio, throughput, response time, accuracy of location, availability can be measured at the user mobiles while delivery success rate and delivery time can be measured at the server. However, they commented that for some of the metrics, the existing tools need to be optimized to operate in mobile environment and for the remaining metrics, a new set of tools must be developed. The following list show high-level measurement method for the QoS metrics:

- End-to-end delay is measured at the destination node. It $t1$ is the time at which an information packet is sent at the source node and $t2$ is the time at which the same packet is received at the destination node then $t2-t1$ give the end-to-end delay for the packet. For this method to give accurate results, the clocks must be synchronized at the source and destination nodes and the information packets must carry the timestamps. RFC2679 [23] gives more detailed measurement methodology. If the clocks cannot be synchronized, then another approach is to measure the round trip delay at the source node and halving it to obtain the one-way end-to-end delay.

- Jitter is measured at the destination node by measuring the variation of end-to-end delay measurement as discussed above. RFC3393 [24] gives more detailed measurement methodology.

- Packet loss ratio is measured at the destination node by measuring the lost packets and total received packets. Each packet consists of a sequence number, which is incremented every time a packet is sent. The missing sequence number at the receiving give the missing packets.

- Throughput is measured at the destination node. If I bytes is the amount of information downloaded in T sec, then the throughput is given by I/T bytes/sec.

- Response time is measured at the destination node. If $t1$ is the time at which a request has been sent and $t2$ is the time at which the corresponding response is received, then $t2 - t1$ gives the response time.

- Delivery success rate is measured at the server. If k is the number of success message deliveries out of N number of messages, then the delivery success rate is given by k/N.

- Deliver time is measured at the server. If $t1$ is the time at which the server received the message and $t2$ is the time at which that message is successfully delivered to the receiver, the $t2 - t1$ gives the delivery time.

- Accuracy of location is measured at the destination node. This measurement requires another reference that provides the standard accurate measurements. The accuracy of low cost and terrestrial method provided by the service provider is compared with more accurate GPS.

- Availability is measured at the destination node. Each mobile stores the number of successful and total attempts to access the service. The ratio of successful attempts to total attemps gives the percentage of availability.

Table 2.1: Classification of applications and their QoS metrics by Gurijala and Molina [22].

Application	Characteristics/examples	QoS metrics
Real-time	Require bounded end-to-end delay and jitter e.g. VoIP, Video streaming application.	End-to-end delay, Jitter, Packet loss ratio
Nonreal-time	Require high bandwidh, less sensitive to end-to-end delay and insensitive to jitter e.g. FTP, E-mail, WWW.	Throughput
Transaction based	Require high secured channels and shorter response times. Do not consume much network bandwidth e.g. all M-commerce based applications.	Response time, Security breach rate
Message based	Require successful delivery and sometimes, in bounded time. Do not consume much network bandwidth e.g. SMS, MMS, updates of news, weather, sports, and financial information, instant messaging.	Delivery success rate, Delivery time
Location based	Information transfer depends on the location of the user e.g. location based.	Location accuracy, Response time

Chen et al.[25] presented QoS metrics for many classes of application. Some QoS metrics samples are shown in Table 2.2.

Table 2.2: Samples of QoS metrics by Chen et al. [25].

Application	Response time expected by users	Delay (ms)	Jitter (ms)	Data rate (bps)	Required bandwidth (bps)	Loss rate	Error rate
Web browsing	2-5 sec.	<400	N/A	<30.5K	<30.5K	Zero	Zero
E-mail	2-5 sec.	Low	N/A	<10K	<10K	Zero	Zero
FTP	2-5 sec.	Med	N/A	High	High	Zero	Zero
Audio broadcasting	2-5 sec.	<150	<100	56-64K	60-80K	<0.1%	<0.1%
Multimedia on Web	N/A	N/A	<150	N/A	28.8-500K	<0.001%	<0.001%
Mono-quality MP3	N/A	N/A	N/A	N/A	32-448K	<0.1%	<0.1%

For broadband service, Wood and Chatterjee [26] introduced some required transmission characteristics as Table 2.3

Table 2.3: Sample of QoS metrics by Wood and Chatterjee [26].

Application	Virtual bandwidth	Tolerable error rate	Acceptable max delay	Tolerable delay jitter	Max burst length
High-quality real-time voice	$\leq$0.20Mbps	$\cong 10^{-3}$	$\approx$300ms	$\leq$30ms	(bytes)
Real-time video	$\geq$4Mbps	$\leq 10^{-6}$	$\approx$100ms	$\approx$5ms	(Kbytes)
Web browsing	$\geq$0.25Mbps	$\leq 10^{-5}$	$\approx$100ms	$\approx$10ms	(Mbytes)
Multipart network games	$\geq$0.1Mbps	$\leq 10^{-5}$	$\approx$50ms	$\approx$5ms	(Kbytes to Mbytes)
Telecommuting	$\geq$1Mbps	$\leq 10^{-4}$	$\approx$1min	$\approx$500ms	(Mbytes)

Brownlee and Looseley [20] suggested that *application-level measurements* are needed for a clear view of overall application performance, which cannot easily be synthesized from lower level data. They may also offer some insights into the performance of the client and server hosts, and of the network link between. Some interesting papers about QoS parameters are in [27, 28].

2.1.3 Service License Agreement (SLA)

Generally, *Service License Agreement (SLA)* is a commitment between a customer and a service provider. Many researchers explained the SLA as follows:

- Fonseca [29]: documented result of a negotiation between a customer/consumer and a provider of a service, and SLA specifies the levels of availability, serviceability, performance, operation, or other attributes of the service.

- Szymczyk [30]: agreement between the customer and the vendor as the services to be provided by the vendor, and the measurable level of those services that the vendor is expected to achieve.

- Jin et al. [31]: analyzed the SLA on web services and expressed that A service level agreement is an agreement regarding the guarantees of a web service. This defined mutual understandings and expectation of a service between the service provider and service customers. Components of an SLA may be consist of: purpose, parties, validity period, scope, restrictions, service-level objectives, penalties, optional services, exclusions, and administration.

SLA may be a narrative document, XML file which is transparent to the users, and so on. An example of a SLA based on application-level metric, i.e. metrics that explicitly express application performance, is presented by Philippe et al. [32] as:

- (Q) Response time for 95% of requests is under 500ms.

- (W) Number of concurrent client sessions is under 256.

Other examples of SLA are also presented in [29, 30].

2.1.4 Software Agent Technology

A *software agent* is a technique that can be assisted computing wireless QoS, so that it can be used in the QoS management. The software agent is a software entity that performs actions for its owner. Software agents are often described as being autonomous, goal-oriented and having social ability to communicate with other agents [33]. *Agent technologies* can be classified into two contexts. In the context of the single-agent system, Local agents and Networked agents can be identified, while in the context of the multi-agent system (MAS), Distributed Artificial Intelligence (DAI) based agents and Mobile

Agents can be distinguished. The agents in MAS may extensively cooperate with each other to achieve individual goals. The main concern of DAI-based multi-agent systems is the coordination of intelligent agents. In this approach, an agent may communicate with the user, system resources and other agents [34]. Sanneck et al. [35] proposed that software components characterized by autonomy (to act on their own), reactiveness (to process external events), proactiveness (to reach goals), cooperative (to efficiently and effectively solve tasks), adaptation (to learn by experience) and mobility (migration to new places).

2.1.5 Literature Reviews on Adaptive QoS System

There are some adaptive QoS system's related works as followed:

- Trzec and Huljenic [36] introduced the structural and behavioral characteristics of multi-agent system (MAS) for QoS management using MESSAGE (Methodology for Engineering Systems of Software Agents) modeling language that extends UML by contributing agent knowledge level concepts and diagrams with notation for viewing them. Such a multiagent system is an environment composed of Intelligent Agents (IAs) that ensure guaranteed QoS offered by multi-service communication networks according to SLAs among users and service providers. This method is unsuitable for multi-domain because every service providers must have SLAs among themselves which depends on the QoS need of individual users of the service providers.

- Bennani and Simoni [37] analyzed the expression of end-to-end requirements. A QoS dynamic management architecture based on DiffServ domains interconnecting units is proposed. The basic generic components is needed to make the network QoS-aware are arranged in both the time scales axis and the execution planes

axis. The requirement classification process uses the fixed information to assign the request into a set of the seven application sets. Disadvantage of this approach is that every domain in this system must be the DiffServ domain which is not suitable for diverse domains in the Internet.

- Al-Ali et al. [38] proposed the QoS management and QoS adaptation which defined in the context of their Grid-QoS Management (G-QoSM) framework. Three service classes: guaranteed service, controlled load service, and best effort service; are covered in this study. The monetary cost of the QoS set for a particular service is presented and the optimization problem is defined with aim to maximize overall monetary profit while maintaining the user's perceptible quality. This method is unsuitable for a general architecture likes the Internet which the services should be fair and free for every users.

- Khan et al. [39] presented Adaptive Multimedia System (AMS) server. AMS is the media server to support multiple concurrent sessions from multiple users. In the system, the QoS problem of individual sessions is determined in order to maximize objective functions which subject to a set of system resource constraints.

- Bechler et al. [9] allowed the user to express his/her preference by simply clicking on a Q(uality)-Button. The process of adaptation is performed without any user interaction. Feedback loops connecting applications, operating system, and network system realize the adaptation. This method is not an effective process because it gets a feedback from the user. User may not feedback the problem, but change to other service providers.

To apply QoS to an adaptive application, many QoS frameworks are proposed for different environments as:

- In [40], a Java-based framework for building QoS controlled multimedia applications is described as a QoS framework for heterogeneous environments. They defined five components in the logical QoS architecture: adaptive application, QoS manager, user interface, QoS mapper, and resource manager. These five components work together to provide the QoS to the users. The proposed adaptive application adapts to a QoS level which would be determined by user preferences and cost constraints. The network bandwidth is categorized as external resource and CPU usage, memory usage and access to the computer's bus are categorized as internal resources.

- Wang et al. [41] explained a framework that brings QoS to end systems by handling incoming requests in accordance with their priorities that are pre-negotiated in between users and application service provider, marking outgoing responses according to some SLAs between users and their ISPs. The basic structure of the framework consists of three main components: 1) queuing part for handling incoming requests, 2) marking part for processing outgoing response, 3) management part for controlling the queuing and marking methods. The queuing part introduces QoS to end systems to classifying and handling requests on the basis of their priorities instead of on the basis of arriving sequence. The marking part integrates the network QoS and end system QoS by marking outgoing responses according to the SLA between the network part and the end system part. The management part determines how to serve requests with respect to their priorities. It controls the policies of classifying, scheduling and marking by sending control messages to queuing part and marking part. Administrators can control the whole framework

through a management console.

- Al-Ali et al. [38] proposed the QoS management and QoS adaptation which defined in the context of their Grid-QoS management (G-QoSM) framework. G-QoSM delivers three QoS levels: 'guarantee' QoS, 'controlled load' QoS, and 'best effort' QoS. A generic adaptation model is outlined based on reserving extra resource capacities to guarantee resources for the 'guaranteed' class of users if there is resource failure or congestion. The dynamic nature of this model allows unused resources to be more effectively utilized. The implementation of the resource reservation and monitoring features, as the underlying tools for the adaptation functions, is also described. An optimization heuristic to optimize resource utilization is proposed, which allows the system to maximize monetary benefits to the Grid service provider; with the basic concept being to enable better resource allocation while satisfying pre-agreed SLAs. The adaptation scheme aims to provide the best possible resource quality within a dynamically changing environment.

2.1.6 The Proposed QoS Model

Comparative of the related works and the proposed solution is presented in Table 2.4. In [36] and [37], the propose systems are based on multi-domain infrastructure. Every domain must have agreements between themselves which depend on the QoS needs of individual users of each domain. In [38], the monetary cost of the QoS set for a particular service is proposed and the optimization problem is defined with aim to maximize overall monetary profit while maintaining the user's perceptible quality. In [39], only multimedia application is considered. The infrastructure of these studies is not suitable for the Internet infrastructure which should be independence, fair and free services, and diverse application services. In [9], users must push their feedback by clicking on a Q-Button

to express their preferences. In [37], requirement classification process used the fixed information to assign to the user's request; seven application sets can be defined.

For [36] and [37], their model is not suitable to the Internet infrastructure. Each network domain in the Internet is independence; agreement between the domains can't be forced. The model in [38] and [39] are interested. The monetary model is applied in [38], but this study points out fair policy. The multimedia application is considered in [39], but this study points out the diverse requirements of application type. User pushes feedback in [9] is not an effective process to get a feedback from the user. User may not feedback the problem, but change to other service providers. In [37], the requirement classification process is not a flexible process, only seven application sets is assigned.

Table 2.4: Comparison between the proposed QoS model and related works

Title	Platform	IDA	Cost-based	Service classes	Related layers	Knowledge
IA for QoS management [36]	General	Yes	No	General	Network & upper	Yes
End-to-End IP QoS [37]	DiffServ Domain	Yes	No	General	Network & upper	None
GQoS-M [38]	Grid Computing	Yes	Yes	General	Network & upper	Yes
AMS [39]	Supports small group of users	None	No	Image, Audio, and Video	Application	None
Q-Button [9]	General	None	No	General	Link & upper	None

continue on the next page

Table 2.4: (*continued*)

Title	Platform	IDA	Cost-based	Service classes	Related layers	Knowledge
The proposed QoS model	Internet	None	No	General	Application	Yes

IDA: Inter-Domain's Agreement

2.2 Incoming Request Scheduling

Improvement of incoming request service of a web server is an interesting research area because many principles, such as queuing management, resource management, scheduling management, QoS, and so on, can be applied to the study and contribution of the improvement has direct impact on a wide range of users. Generally, application developers develop their application based on functionality of the application by leaving server's resource control duties to the operating systems. When the user requests become more diversely and vastly; either working on the Internet or expanding into mobile and/or wireless computing, it leads a critical service problem to the applications, such as web sever.

Dalal and Jordan [42] considered an *impatient user problem* that occurs when a user abandons a pending web request if no response for that request in a matter of second. The problem causes the server to waste resources on a request that no need to complete. These situations may ultimately prove disastrous and lead to a server deadlock crisis on a heavily loaded server.

Due to this critical situation and limited resources of the server, such as memory, disk bandwidth, communication bandwidth, and CPU cycle, extending web server to

do resource management is very important. There are many techniques to extend web servers. For server responsibility viewpoint, server-centric and client-centric constraints are two main classes of the classification.

- For server-centric constraint, requests are differentiated depends on some attributes of the server. For example, Pandey et al. [43] informed a notion of quality of service that enables an HTTP server to respond the external requests by setting priorities among page requests and allocating server resources.

- For client-centric constraint, the different attributes of client are used to determine the server's response [44, 45, 46, 47]. For example, Chandra et al. [44] used transcoding technique to allow web servers to customize the size of objects constituting a web page and hence the bandwidth consumed by that page, by dynamically varying the size of multimedia objects on a per-client basis.

Further, a considerate alternative to extend web server in this study is *incoming web request scheduling* [41, 42, 46, 48, 49]. This is an approach to handle the processing order of the incoming request on the scarce resources system. Many interesting scheduling techniques are proposed as a part of many systems. For examples, Almeida et al. [46] implemented a priority-based scheduling by assigning priorities to the user requests according to the requested documents. Next, Wang et al. [41] implemented a system that classifies incoming requests into different priority queues according to their pre-negotiated priorities, and uses a scheduler in queuing part to reschedule the classified requests before sending them to the destination application.

There are many policies to handle incoming request, such as First-In-First-Out (FIFO), Earliest-Deadline-First (EDF), Shortest-Processing-Time-First (SPT), Last-Come-First-Served (LCFS), Round-Robin (RR) with a fixed quantum, Lottery, Fair-Share, Biggest-In-First-Served (BIFS), and so on.

The scheduling policy can be separated into two running characteristics: *preemptive scheme* and *non-preemptive scheme*. In *preemptive scheme*, another high priority process can interrupt the running process, whereas in *non-preemptive scheme*, the running process must be completed before the another is executed.

In addition, Stankovic et al. [50] presented that the knowledge of the request and its factors, such as deadline, processing time, precedence, future release times, and so on, play importance role to the scheduling policy for both uniprocessor and multiprocessor real-time systems.

Balachandran et al. [51] analyzed user behavior and network performance in a public wireless LAN - a well-attended ACM conference over 3 days and concluded that web browsing (HTTP) contributes 46% of the total bytes transferred, mean web object sizes are 8KB, and 60% of the user sessions are less than 10 min. According to analyzing the browse patterns of mobile clients (cellular phones and Personal Digital Assistances-PDAs) by Adya et al. [52], they found that the majority of client requests for wireless clients are concentrated on a small number of documents. Most of the replies to wireless clients are less than 3 Kbytes and to offline clients are 6Kbytes and users tend to have short sessions when interacting with the website: the largest session time for 95% of user was less than three minutes. They empirically determined the session-activity threshold to be someware between 30-45 seconds. However, Kotz and Essien [53] analyzed a campus-wide wireless network, Dartmouth College, where each access point provides 11Mbps coverage to nearly the entire campus and users use laptop to access the network. They summarized that 53% traffics are HTTP protocol, most session are short (the median seesion length was 16.6 minutes), 71% of sessions finish in less than one hour, and 82% sessions are non-roamimg.

Generally, incoming request service works on *best effort*, FIFO, manner. When a client sends a request to a web server, the request is held in the web server's request

pool waiting for a processing cycle. After it got the signal, it is then pushed to the FIFO queue and waited for the server's processing. To improve this incoming service, a scheduler is added to organize the requests into the FIFO queue as shown in Figure 2.2.

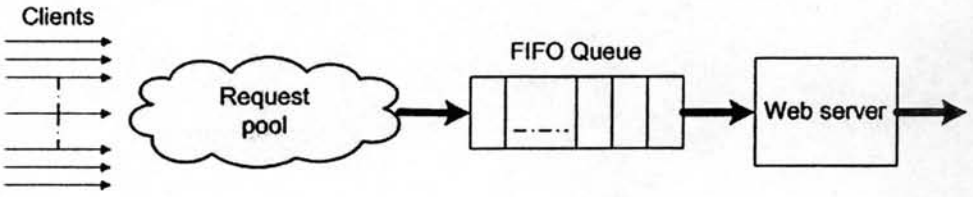

Figure 2.2: Best effort web server's reference model

2.2.1 Queuing system

Queuing system is a fundamental component of the incoming request scheduling system. Customer arrival, service facility, service time are characteristics that use to determine the system.

Notations

Willig [54] expressed the *Kendall Notation*, which explains a short characterization of queuing system, as follows:

$$A/B/m/N - S$$

where A denotes the interarrival time distribution, B denotes the distribution of the service times, m denotes the number of servers, N denotes the maximum size of waiting time line in the finite case (if $N = \infty$ then this letter is omitted) and the optional S denotes the service discipline used (FIFO, EDF, and so on). If S is omitted, the service discipline is always FIFO. For A and B the following abbreviations are very common:

- M (Markov): this denotes the exponential distribution with $A(t) = 1 - e^{-\lambda t}$ and $a(t) = \lambda e^{-\lambda t}$, where $\lambda > 0$ is a parameter. The name M stems from the fact that

the exponential distribution is the only continuous distribution with the markov property, i.e. it is *memoryless.*

- D (Deterministic): all values from a deterministic "distribution" are constant, ie. have the same value.

- G (General): general distribution, not further specified. In most cases at least the mean and the variance are known.

The *memoryless* attributed is explained by Gelenbe and Pujolle [55] as follows. Suppose that we are dealing with a time bomb which explodes automatically after a time X distributed according to an exponential distribution:

$$P\{X < x\} = 1 - e^{-\lambda x}, \infty > \lambda > 0.$$

We trigger the mechanism at time $t = 0$ to cause an explosion at time $t = X$. At an intermediate time $t = y$, before the explosion occurs, we would like to know the time remaining before the explosion.

This simply means that we with to know the distribution of $X - y$ knowing that $X\,y$ since the explosion has not occurred at time $t = y$. We calculate:

$$\begin{aligned} P\{X - y < x | X > y\} &= P\{y < X < y + x\}/P\{X > y\}, \\ &= \frac{1 - e^{-\lambda(y+x)} - (1 - e^{-\lambda y})}{e^{-\lambda y}}, \\ &= 1 - e^{-\lambda x} = P\{X < x\} \end{aligned}$$

and we discover that the fact that the explosion has not occurred up to time t allows us to establish simply that $X - y$ has the same distribution as X. This is called the *Markovian* or *memoryless* property of the exponential distribution.

The interested queuing system in this study, the *M/G/1*, can then be described as follows: the system has a single server, an infinite waiting line, the customer interarrival

times are *independent and identically distributed (iid)* and exponentially distributed with some parameter μ, and the service time are general distributed with a known mean and variance.

2.2.2 Poisson Process

Jarrett and Kraft [56] described the *Poisson probability distribution* as the probability of a random variable X taking on a finite number of values over a continuous interval. Random variables whose occurrence can be calculated by the Poisson probability distribution behave according to a *Poisson process.* In a Poisson process, the events occur at a constant rate m per *unit* interval. For a Poisson process with event occurring at a constant rate per unit interval, the probability of occurrence of an event in an interval is proportional to the length of the interval. Thus, in a *Poisson probability interval* with t such unit intervals, the events occur at a constant rate mt, which is the average number of occurrences in the interval. Since the parameter μ is the average, it is equal to mt. The Poisson probability distribution can be denoted as:

$$P(X = x) = \frac{e^{-\mu}\mu^{x}}{X!}$$

where X may take on the values 0, 1, 2, ... The symbol e is the base of natural logarithms and is approximately equal to 2.7183.

Andersson et al. [57] investigated the *web traffic* logs during the crisis conditions, the London underground system bombing - July, 2005. The results show that the Poisson process is a good candidate for the arrival process of document requests. The crisis conditions lead medium and heavily loaded services to the web server, which is the concerning situation in this dissertation.

Vincent [58] introduced the Poisson process generation algorithm as follows:

Algorithm 1 Poisson process generation

n = 0

T = 0

repeat

A = $\frac{1}{\lambda}(-log(1 - random(0, 1)))$

T = T + A ;Next arrival time

n = n + 1

until (end of simulation)

If U is uniformly distribution on [0,1(random function)] then $\frac{1}{\lambda}(-log(1 - U))$ is exponentially distributed with rate λ.

2.2.3 The M/G/1 System

Sidi and Khamisy [59] summarized the characteristics of the *M/G/1 queuing system* as the following detail.

The arrival process of the M/G/1 system is Poisson with rate λ. The service time has arbitrary distribution with Laplace Stieltjes Transform (LST) denoted by $\mathcal{B}(s)$. Let λ and x^2 be the average and the second moments of the service time. The analysis is generally based on the method of the embedded Markov chain at the departure instants from service. The results for the M/G/1 system can be summarized as:

- Utilization factor (proportion of time the server is busy) $\rho = \frac{\lambda}{\mu}$.
- The generating function $\mathcal{Q}(z) \triangleq \sum_{i=0}^{\infty} p_i z^i$ of the probability distribution $p_i, i \geq 0$ of the number of customers in the system in steady-state (also at departure and arrival instants) $\mathcal{Q}(z) = \mathcal{B}(\lambda - \lambda z)\frac{(1-\rho)(1-z)}{\mathcal{B}(\lambda-\lambda z)-z}$.

- Average number of customers in the system $N = \rho + \frac{\lambda^2 x^2}{2(1-\rho)}$.
- Average customer time in the system $T = \frac{1}{\mu} + \frac{\lambda x^2}{2(1-\rho)}$.
- Average number of customers in queue $N_Q = \frac{\lambda^2 x^2}{2(1-\rho)}$.
- LST for the system time $\mathcal{T}(s) = \mathcal{B}(s)\frac{s(1-\rho)}{s-\lambda+\lambda B(s)}$.
- LST for the waiting time $\mathcal{W}(s) = \frac{s(1-\rho}{s-\lambda+\lambda B(s)}$.

2.2.4 Performance Measurement of Queuing System

Adan and Resing [60] suggested that there are some relevant measurements for measuring the performance of queuing system as follows:

- The distribution of the waiting time and the sojourn time of a customer, The sojourn time is the waiting time plus the service time.
- The distribution of the number of customers in the system.

The relations between these measurements can be expressed by *Little law* [61, 62]. The relation between $E(L)$, the mean number of customers in the system, $E(S)$, the mean sojourn time, and λ, the average number of customers entering the system per unit time as follows:

$$E(L) = \lambda E(S).$$

Applying *Little law* to the queue (excluding the server), a relation between the queue length L_q and the waiting time W, namely

$$E(L_q) = \lambda E(W).$$

If *Little law* is applied to the server only, it can be obtained that

$$\rho = \lambda E(B)$$

where ρ is the mean number of customers at the server and $E(B)$ is the mean service time.

2.2.5 Literature Reviews on Incoming Request Scheduling

Normally, as mentioned earlier, most scheduling policies intend to meet merely single criterion. It is always expected that the single criterion method can satisfy the user requirements. This unidimensional viewpoint responses the policy objective in particular criterion, while usually ignores other criteria, such as in SPT policy, shortest processing time of the request is the determining criterion that increases the number of successful requests, but at the same time, deadline and arrival order of the request are ignored.

T'kindt and Billaut [63] introduced a definition of a multicriteria scheduling problem. They recommended that the minimization of several conflicting criteria change the way to handle scheduling problems. The definitions of weak and strict Pareto optimality are encountered to minimize optimally all the criteria, especially the strict Pareto optimum that is necessary for the multicriteria scheduling in computing system.

There are many techniques to determine a single solution for multicriteria problem, such as the weighted aggregation, fuzzy set theory, goal programming, and utility theory.

Multicriteria scheduling concept has been applied to many computing systems and communication layers. For example, Fan and Huang [64] proposed a multicriteria scheduler algorithm that explores the trade off between the system throughput, user transmission rate, and user transmitted slots, in an access point based on 1xEV-DO system (an evolution of cdma2000 for a data-only system). Cherkasova [65] proposed a tunable scheduling strategy lies between FIFO and shortest-first, based on the value of a coefficient Alpha. This policy is called Alpha scheduling with non-preemption, allows to improve overall response time per HTTP request more than three times under heav-

ily loaded. However, T'kindt and Billaut [63] suggested that only a small number of problem consider the minimization more than two criteria.

According to the goal of a web server that needs to serve as many users as possible under limited resources and various user requirements that have many objectives and usually conflict, the multicriteria concept is an appropriate method to provide an optimal solution.

In this paper, we propose a multicriteria scheduling, MCB for short, by applying the MCDM concept [63, 66]. The objective of this study is not to find the Pareto optimum, but to present an alternative optimal scheduling policy. We investigate the incoming request service on application-level of a web server in order to solve the impatient user problem. To achieve the user satisfaction, the policy considers more than one characteristic of the incoming request and sets them as criteria for scheduling. These characteristics dominate the system's quality of service; waiting time. The well-known factors of the request like arrival time, deadline, and processing time are concerned as instances of MCB policy's criteria compare to the other three basic policies: FIFO, EDF, and SPT. Each considerate factor is also the monopolizing factor of each comparative policy, such as FIFO depends on arrival time, EDF depends on deadline, and SPT depends on processing time.

These three scheduling policies have single criterion objective. First, FIFO policy intends to prioritize account by using the arrival time of each request. Second, EDF policy aims to keep each request on their deadline. Finally, SPT policy considers to do the request with the shortest processing time first. Comparative performance studies of EDF compares to Weighted SPT and FIFO, and SPT compares to FIFO are reported. Ye et al.[48] applied Weighted SPT and EDF queuing disciplines to differentiate the web services and concluded that Weighted SPT and EDF disciplines have overall waiting time shorter than FIFO discipline. Elnikety et al. [49] introduced SPT scheduling discipline

for admission control at proxy server and concluded that average response time can be reduced by SPT. Correlations of each criterion are not covered in this paper, see [59, 67, 68] for more detail.

Normally, general systems are concerned on overall user satisfaction in form of average response time or average waiting time while ignoring individual user satisfaction such as maximum and standard deviation (or variance) waiting times. The proposed scheduling policy considers both overall and individual user satisfactions by defining average, maximum, and standard deviation waiting times as its criteria to measure the performance of the policy.

2.2.6 The Proposed Scheduling Policy

Comparing our study on incoming request scheduling to the other related works are shown in Table 2.5. Our study has intention to reduce the QoS gaps, therefore overall user satisfaction and individual user satisfaction are balanced with weight aggregation technique.

Table 2.5: Comparison between the considerate scheduling system and other related works.

Title	Application	Strategy	Technique	OUS	IUS
Elnikety et al. [49]	proxy server	SPT	-	Yes	No
Dalal & Jordan [42]	non-processor sharing & processing sharing web server	single criterion policy	-	Yes	No

continue on the next page

Table 2.5: (*continued*)

Title	Application	Strategy	Technique	OUS	IUS
Duenas & Petrovic [69]	single machine production scheduling (kiln)	single criteria + multiobjective	Genetic algorithm (GA)	Yes	No
Fan & Huang [64]	1xEV-DO	multicriteria	weighted aggregation	Yes	No
Cherkasova [65] (Alpha scheduling)	sequential web server	bi-criterion policies: FIFO, SPT	weighted aggregation (tunable)	Yes	No
Ye et al. [48]	sequential web server	single criterion policies: WSPT, ATC, EDF	-	Yes	No
Our study	sequential web server	multicriteria	weighted aggregation	Yes	Yes

OUS: consider Overall User Satisfaction , IUS: consider Individual User Satisfaction

Comparison MCB to FIFO, EDF, SPT, and Alpha are shown in Table 2.6. The distinct attributes of our propose method are that we concern more criteria, which have impact on overall and individual user satisfactions, and use compromising strategy among these conflicting criteria.

Table 2.6: Comparison between FIFO, EDF, SPT, Alpha, and MCB.

Policy	Objective	Criteria	OUS	IUS	Advantages
FIFO	minimize variance of waiting time	arrival time	No	Yes	fair service
EDF	minimize number of tardy jobs	deadline	Yes	No	Optimal for n independent, single-operation jobs which is available for processing at time zero. [48]
SPT	minimize mean flow time	processing time	Yes	No	serve users with minimum mean flow time
Alpha [65]	optimize response time	arrival time, processing time	Yes	No	improve overall response time
MCB	compromise all criteria	arrival time, deadline, processing time	Yes	Yes	concern both OUS and IUS

OUS: consider Overall User Satisfaction, IUS: consider Individual User Satisfaction