CHAPTER III

PROPOSED METHOD

An adaptive application is an application that can adapt itself to a high variable environments. According to the Quality of Service (QoS) concept, QoS management is executed to deliver service license agreement (SLA) between a client and a server. By merging of these two concepts, this chapter proposes a generic QoS management model as a conceptual framework of an adaptive applications. End-to-end QoS management functions on the server site and the client site are provided to maximize all users' satisfaction and each user satisfaction, respectively. The end-to-end transmission delay is considered as a measurement unit in the case study of the QoS management functions.

3.1 Reference System

The reference system is based on a wireless Internet infrastructure. This is an interesting environment because of its characteristics, such as limited bandwidth, high resource variation, and intermittent connection, while the usage growth rate is rapidly high [70, 71]. In this infrastructure, inter-domain agreement of the traditional QoS system is unsuitable because of complexity and uncontrollable of the Internet domains. The system can be separated into two parts: a Web server on wireline network and clients, which are wireless terminals, on wireless networks which connected to the server via the Internet as shown in Figure 3.1. Currently, there are many wireless communication standards with different data rates. For examples, data rates in the IEEE 802.11b [72] is up to 11

Mbps, the GPRS [73] is up to 171 Kbps, the EDGE [74] is up to 384 Kbps, the UMTS [27] is up to 2 Mbps, and 802.11g [75] is up to 54 Mbps. Whereas it is up to Gbps in the wireline system. The system concerns the end-to-end communication manner which leaves the other communication parts, the Internet, as a black box.

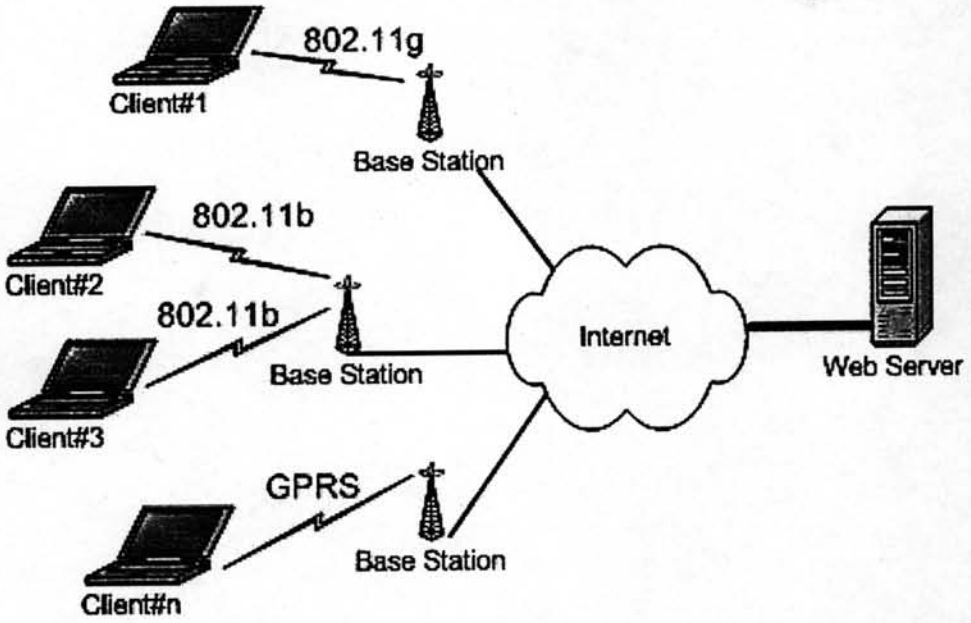

Figure 3.1: An instance of reference system.

Similar to other general web servers, the server supports three QoS classes which are interactive, background, and streaming. Each service class has its own requirements which normally lie on timeliness, accuracy, capacity, and security. For example, the streaming class such as real-time audio/video, requires low jitter, the interactive class such as web browsing, and the background class such as file transfer, both require low bit error rate but the interactive class concerns round-trip delay time.

It's the truth that every service class and users need high performance communication but the network resources are limited. Resource management is responsible to handle these requirements; QoS management is offered to this system to achieve the customers, which are users on the client site, satisfaction.

3.2 A Generic QoS Management Model

In the general adaptive system, adaptation process will be invoked when the system status exceeds a parameter's threshold. In this study, QoS management functions are integrated to support the system's adaptation to satisfy the users. The QoS management functions are installed on both end-to-end communications sites: the client site and the server site. At the client site, the QoS management function is applied to maximize individual user's satisfaction, which named maximum ratio of a unit satisfaction, while the management function at the server site attempts to maximize the overall users' satisfaction.

QoS specification is concerned with capturing application-level QoS requirements and management policies [17]. There are many papers [76, 77] that study about this topic. Many features are interested depend on the communication level. At the user level, a user concerns about, for examples, availability, performance, accuracy and reliability. At the network level, evaluation factors that always concerned are throughput, delay, jitter, response time, and loss rate. In this study, end-to-end transmission delay is concerned as a case study of user requirements because it is one of an important communication factor of users' requirements. The system provides the following QoS policies to control and react to the system's events:

- Every user has single role, they have equivalent priority to get the services.
- Admission for a new user and renegotiation must regard to other users who occupied the system resources.

3.2.1 Agent-based QoS Management Architecture

In this study, a communication session viewpoint is considered, starting from receiving a new request until the request is terminated. According the reference system in Section

3.1, the agent-based QoS management architecture is presented in Figure 3.2. The details of system flow and related algorithms are expressed with QoS management cycle and QoS management functions, respectively, in the following sections.

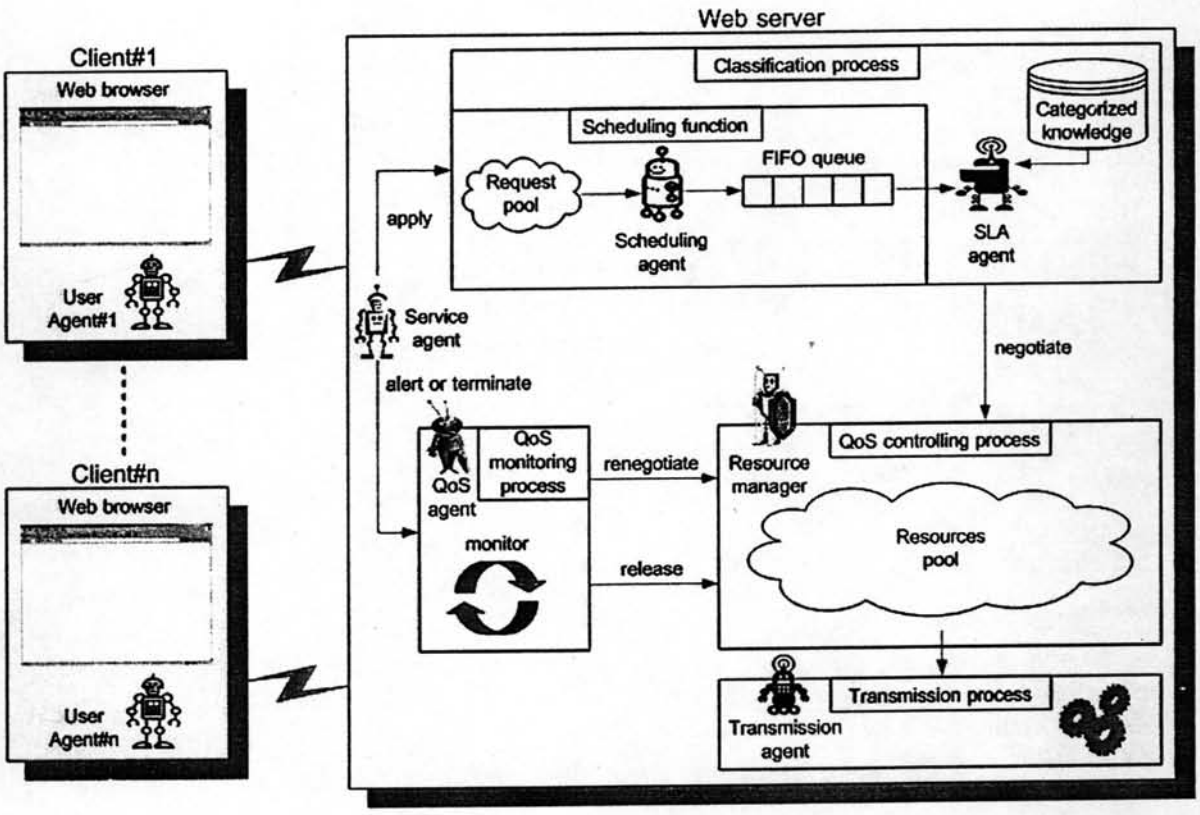

Figure 3.2: Agent-based QoS management architecture.

The architecture is based on the Internet platform. Suppose a web application is considered, a web browser is the application on a client and a web server is the application on a server. There are six agent types that work towards the system objective. Each agent type has its own role as follows:

- User agent works on the client to sent apply/alert/terminate signals to the server.
- Service agents work on the server to handle the client requests. One service agent is generated for one client's request. The service agent holds the client requirements

and contacts with other agents to handle the client request. The service agent is destroyed when the client request is terminated.

- Scheduling agent works on the server to schedule the incoming requests in the classification process. It can work as an adaptive agent to adjust the scheduling policy when quality of the system's service utilization, such as number of success request, is degraded.

- SLA agent works on the server to find out SLA parameters' threshold from the categorized knowledge database depending on the requirements in the service agent that is processed. The selected SLA parameters threshold are assigned to the service agent.

- Resource manager works on the server to manage the system resources. It has duties to negotiate, renegotiate, and release the resources depending on the request. Moreover, if the resource capacity is over the resource's reserved threshold, which is reserved for any new requests, it has duty to distribute the exceeding resource capacity to the requests that work in the system.

- Transmission agent works on the server to handle the transmission process of service agents.

- QoS agent works on the server to control the system's quality of service. It waits for a signal from the service agent, such as an alert signal to renegotiate more resources and a terminate signal to release the resources. Furthermore, it monitors other agents. For instance, it monitors illegible service agent that did not contact its user agent for a long time, then terminates that agent to protect the impatient's user problem.

3.2.2 QoS Management Cycle

In this study, a communication session viewpoint is considered, starting from receiving a new request until the process is terminated. The QoS management model is presented by data flow diagram as shown in Figure 3.3 and the main components of the model is shown in Figure 3.4.

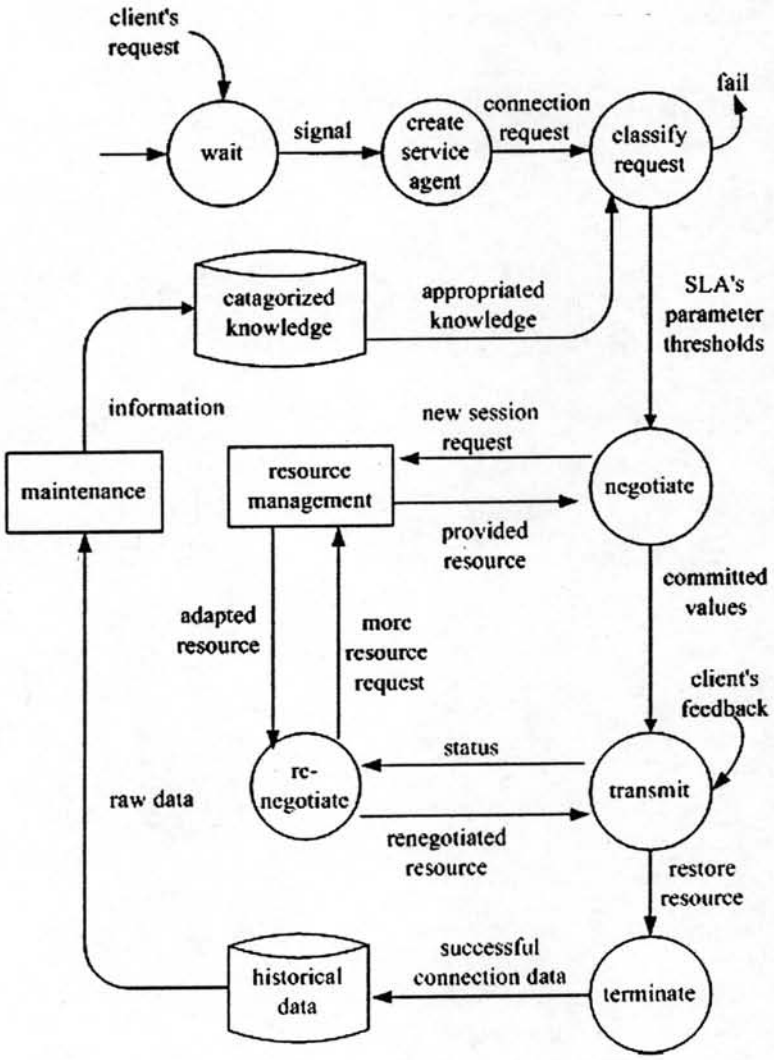

Figure 3.3: QoS management cycle in the system.

In the Figures, when a client requests a service, the server creates a service agent to serve that request. The service agent classifies request based on the request's parameters

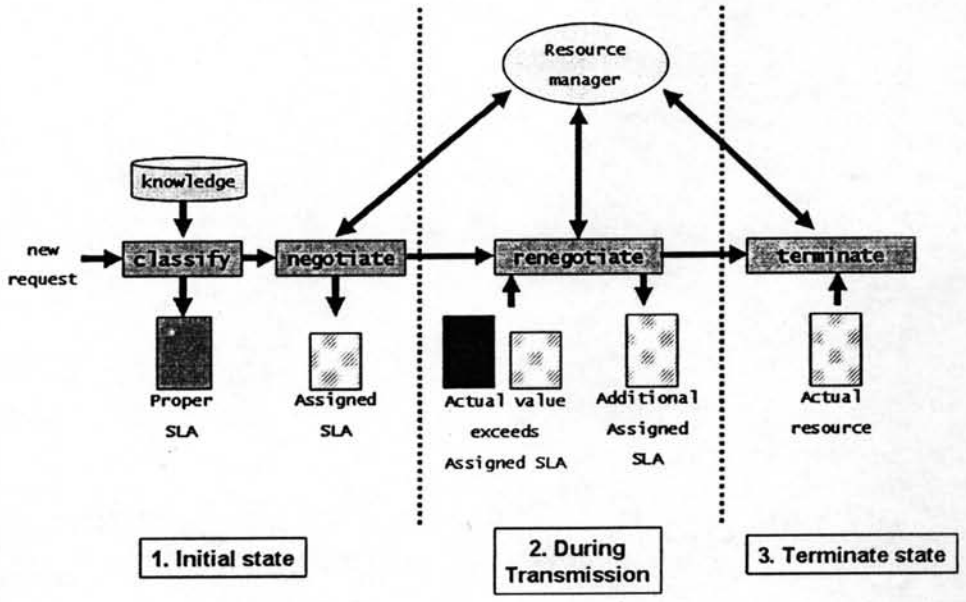

Figure 3.4: Main components of the QoS management cycle.

and the categorized knowledge. This classfication process generates network parameters' threshold, which is an SLA, for the request. Next stage, the service agent negotiates network resources with resource manager. The resource manager is an agent that manages the network resources in the system. The output of the negotiation process is an assigned SLA. During transmission, the client can send a request to activate the server that it has some communication problem, such as the transmission delay is too high. Once the server is activated, renegotiation process is then invoked to adapt the transmission to the changing situation. Last, when the transmission is terminated, either success or failure of the user' request, the system provides the termination process to release the allocated network resources and end the service agent. Additionally, successful situation is kept as historical data. The maintenance process works in a long-term period (e.g. a week or a month) to analyze the historical data and classify them to the categorized knowledge.

3.2.3 Analysis of End-to-End Application-level Communication

The QoS architecture is proposed on the Internet platform at the application level. On this platform, there are many sources that have effect to the delay performance of a communication. Spohn [77] introduced the primary contributors to delay include:

- Propagation path length
- Line speed
- Number of access hops
- Hardware and software interface buffers
- Load on every components across the path
- Hardware/processor elements traversed (each adds delay)
- Window sizes
- Bit-setting selection (e.g., D-bit)
- Memory and buffers
- Pad functions
- Address database look-up
- Address verification
- Changes in traffic load
- Filtering, forwarding, and processing packets, frames, and cells.

Heidemann et al. [78] researched on modeling the performance of HTTP over serverl transport protocols. For a transport protocol, such as TCP and UDP, they suggested that the minimum possible transaction time is the one roundtrip time delay inherent in communication, plus the time it takes to send the request and receive the reply, and any time spent at the server:

$$T_{min} = rtt + req_{min} + processing + reply_{min}$$

$$req_{min} = req_{size}/bw$$

$$reply_{min} = reply_{size}/bw$$

where T_{min} is the minimum transaction time, rtt is one roundtrip time delay inherent in communication, $processing$ is the time that spent at the server, req_{min} is the time that take to send the request, req_{size} is the request's data size, bw is bandwidth of the considerate module, $reply_{min}$ is the time that take to receive the reply, and $reply_{size}$ is the replay's data size.

In this study, the consider period of time is the processing time in a server. According to our assumption that every components under the application layer is leaved as a black box, the delay in this black box, which is delay in the physical layer and other communication layers, is determined as the communication delay time (D_{comm}). Thus, the total processing delay time for an end-to-end application level performance can be denoted as:

$$D_{processing} = D_{comm} + D_q + D_{serv} + D_{reply}$$

where $D_{processing}$ is the total processing delay time for an end-to-end application, D_{comm} is the time that take in communication under the application layer (both on request and reply process), D_q is the time that the session take in the waiting queue before executed, D_{serv} is the time that takes to determine SLA parameters' threshold, to negotiate required resources, and to renegotiate more resources, and D_{reply} is the time that the

application layer replies the session to the lower communication layer.

Normally, request arrival time of a client/server application is assumed to be a *markovian system*, which the interarrival times is an exponential distribution (Poisson) that exhibit markov (memoryless) property. Because of this study intends on worst study analysis of the resources' usage, the exponential service time is considered. The communication in this end-to-end application is modeled as the M/M/1 queue according to the analysis at system utilization and capacity [77, 79]. Liu et al. [79] analysted the performance at application level. They expressed the steady state probabilities, mean value of customers in the system, and mean response time as follows:

- the steady state probabilities can be denoted as:

$$p_0 = 1 - \frac{\lambda}{\mu} =: 1 - \rho$$
$$p_k = \left(\frac{\lambda}{\mu}\right)^k p_0 =: p^k p_0$$

  where p_k denotes the probability that there are k customers in the system, λ is the arrival rate of the client request, μ is the system service rate and $\rho = \lambda/\mu$.

- the mean value of customers in the system can be denoted as:

$$\bar{N} = E[N] = \sum_k k p_k = \frac{\rho}{1-\rho}$$

- By applying Little's law, the mean response time can be denoted as:

$$\bar{R} = \frac{\bar{N}}{\lambda} = \frac{1}{\mu - \lambda}$$

  where $\lambda \neq \mu$.

3.2.4 Advantages of Adaptive Application Beyond Best Effort System

For the best effort system, after a communication session is established, if there are communication problems, the probability of the enduring session is 0.5.

Adaptive application is a beyond best effort system that intends to adapt itself to the changing environment, such as network problems. The assumption of this study is an adaptive process to handle quality of service situation is a tiny time that promptly fixes the problem.

There are three situations for the adaptive process as follows:

- Adapt on a feature that is not related any resources, such as packet length adaptation.
- Adapt in rich resources situation. In this situation, the system may be lightly loaded or has enough resources capacities to provide for any request. The requests that occupied resources are not disturbed in the regotiation process.
- Adapt in scarce resources situation. In this situation, the system must negotiate with the requests that occupied the excess resources for more resources.

It can be expected that if the probability of the adative process is x then the probability of the enduring session is $0.5+x$.

3.2.5 QoS Management Functions

The QoS management functions are the processes to maximize overall and individual users' satisfaction, according to SLA. The SLA is an agreement between a server and a client. Hence, the service that the client received must be maintained in the range of

the agreed threshold in the SLA. The QoS management functions must be concerned in both server and client sites.

The server site QoS management function

A related definition of the server site QoS management function is defined as follows:

Definition 1. *Feature (feature)*

Feature is capacity of the resource that system provides for a session, such as transmission rate, packet size.

The server site QoS management function can be defined as an optimization function, where the objective is to maximize the overall satisfaction of users that currently connect to the particular server. The optimization function of the server site QoS management function can be modeled as follows:

$$\max\ satisfaction = \sum_{i=1}^{n} S(i) \tag{3.1}$$

subject to:

- $\forall S(i) \geq SLA_{min}(i)\ for\ i = 1\ to\ n$,
- $\forall(j)(\sum_{i=1}^{n} feature(i,j) \leq\ overallFeature(j))\ for\ j = 1\ to\ p$

where

- n be the number of the sessions that currently connected to the specified server,
- $S(i)$ be a satisfaction value of the i^{th} session,
- $SLA_{min}(i)$ be a minimum satisfaction value of the i^{th} session,
- p be the number of the features in the system,

- $feature(i, j)$ be a resource j^{th} feature's capacity which is occupied by the i^{th} session,
- $overallFeature(j)$ be an overall resource j^{th} feature's capacity which provided by the system.

The client site QoS management function

Related definitions of the client site QoS management function are defined as follows:

Definition 2. *SLA Impact Factor*

An SLA impact factor, abbreviates $factor$, is a value impacts to satisfaction level, such as delay, jitter, loss rate, and throughput. Factors are also referred to as QoS metrics, which is reviewed in Section 2.1.2. The factor can be categorized into two types:

- Positive factor. If this factor value increases, the system performance will be increased, such as throughput.
- Negative factor. If this factor value increases, the system performance will be decreased, such as delay and loss rate.

Each factor has its own threshold values depending on its requirements. These values are defined as a maximum threshold and/or a minimum threshold that indicates a satisfying bound of the transmission and deviation values that indicates a critical situation of the transmission.

- For positive factor, the minimum threshold value is concerned because the server must provide more resources to handle this transmission situation.
- For negative factor, the maximum threshold value is concerned because the server must provide more resources to handle this transmission situation.

Definition 3. *Acceptance Level*

Acceptance level of a factor or maximum ratio for a unit of satisfaction, *acceptanceLevel*(), is a calculating function that calculates the accept value of a session's factor. A factor value can be measured at a client site during a data transmission between a client and a server. It returns a value in degree 0, 0.5, and 1, where 0 is an unaccepted value, 0.5 is a poorly accepted value, and 1 is a good accepted value. Acceptance level can be denoted, according to the factor types, as:

- For positive factor, the acceptance level is shown in Figure 3.5.

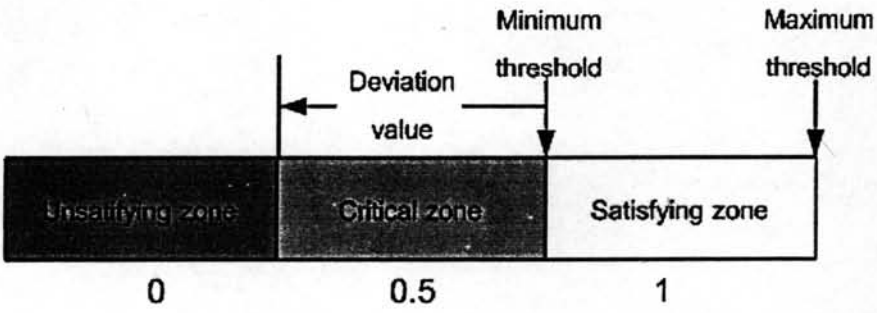

Figure 3.5: Positive factor's acceptance level.

If an actual detected value ($factor_{act}$) is more than or equal to the minimum threshold value ($factor_{thres_{min}}$), this is a satisfying situation and the acceptance level is 1. If an actual detected value is less than the minimum threshold minus the deviation value ($factor_{dev}$), the transmission can not be endured that is an unsatisfying situation and the acceptance level is 0. If an actual detected value is in the deviation range, this is a critical transmission situation that the server must provide more features to handle this transmission session via the renegotiation process. If the transmission is in this poor situation, the acceptance level is 0.5.

The acceptance level function for the positive factor can be denoted as:

$$acceptancelevel() = \begin{cases} 1 & \text{if } factor_{act} \geq factor_{thres_{min}} \\ 0.5 & \text{if } factor_{thres_{min}} > factor_{act} \geq (factor_{thres_{min}} - factor_{dev}) \\ 0 & \text{if } factor_{act} < (factor_{thres_{min}} - factor_{dev}) \end{cases} \tag{3.2}$$

where $factor_{act}$ is an actual detected value of the factor and $factor_{dev}$ is a deviation value of the factor.

- For negative factor, the acceptance level is shown in Figure 3.6.

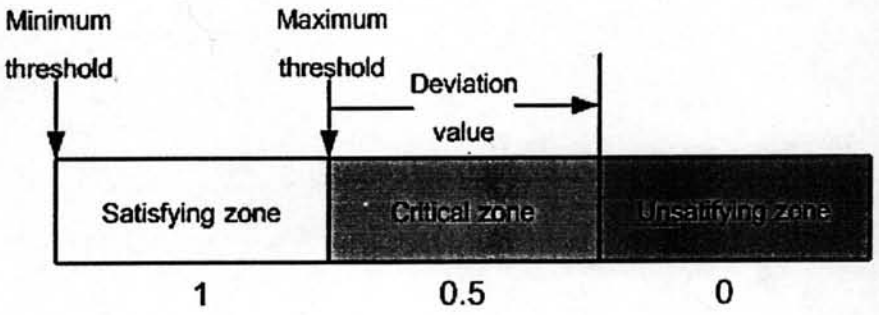

Figure 3.6: Negative factor acceptance level.

If an actual detected value is less than or equal to the maximum threshold value ($factor_{thres_{max}}$), this is a satisfying situation and the acceptance level is 1. If an actual detected value is more than the maximum threshold plus the deviation value, the transmission cannot be endured that is an unsatisfying situation and the acceptance level is 0. If an actual detected value is in the deviation range, this is a critical transmission situation that the server must provide more features to handle this transmission session via the renegotiation process. If the transmission is in this poor situation, the acceptance level is 0.5. The acceptance level function

can be denoted as:

$$acceptancelevel() = \begin{cases} 1 & \text{if } factor_{act} \leq factor_{thres_{max}} \\ 0.5 & \text{if } factor_{thres_{max}} < factor_{act} \leq (factor_{thres_{max}} + factor_{dev}) \\ 0 & \text{if } factor_{act} > (factor_{thres} + factor_{dev}) \end{cases} \tag{3.3}$$

where $factor_{act}$ is an actual detected value of a factor, $factor_{thres}$ is a threshold value of a factor and $factor_{dev}$ is a deviation value of a factor.

Furthermore, for any communication session k, the QoS management function, which has an aim to find the maximum ratio for a unit of satisfaction ($\max S(k)$), can also be defined as an optimization model as follows:

$$\max S(k) = \frac{\sum_{j=1}^{m} acceptanceLevel(factor(k,j))}{m} \tag{3.4}$$

subject to :

- $\forall factor(k,j),\ acceptanceLevel(factor(k,j)) > 0$

where

- $S(k)$ be the user's satisfaction value of the client who owns the k^{th} session,
- $factor(k,j)$ be an impact j^{th} factor of the provided services for the k^{th} session,
- m be the number of concerning factors,

The objective of this function is to maximize individual user's satisfaction, the k^{th} session. The constraints of this function are the maximum or minimum thresholds of all provided service factors, all sessions must held these values, and these values must not exceed the capacity that the server can provide, which is a satisfy situation. If each factor exceeds its threshold, then, the agent at the client site sends signal to activate the renegotiation process at the server site; that is an unsatisfying situation.

3.3 The QoS Management Functions: a Case Study

At the network-level, there are many evaluation factors that always concerned, such as jitter, throughput, and loss rate. This study uses feature adaptation technique to adjust the service in this system under an assumption that "features adaptation have positive impact to the factors which will increase user satisfaction". One key indicator of the user's satisfaction depends on the transmission delay, which related to the transmission rate and the packet length. This key indicator is used as a case study of the QoS management functions. Applications of the server site and the client site QoS management functions, in Section 3.2.5, are explained by replacing of features and factors that affected by the transmission rate adaptation. The related definitions are defined as follows:

Definition 4. *Maximum Transmission Delay*

A maximum transmission delay value is the maximum transmission delay that a user can accept. This value is depended on the service class that the user requests, denoted by D_{max}.

Definition 5. *Transmission Delay*

Transmission delay, defined in [11], is a amount of time required to put all of the packet's bits into the link.

$$D(i) = L(i)/R(i)$$

where

- $D(i)$ be a transmission delay of the i^{th} session,
- $L(i)$ be a length of a packet of the i^{th} session,
- $R(i)$ be a transmission rate of the i^{th} session.

3.3.1 Transmission Rate Adaptation

In this study, end-to-end delay, which considers delay from a source to a destination, is presented. Assume that networks between the source and the destination hosts are as a black box. Furthermore, every node in the network has high performance computing and the network between the two hosts is uncongestion. Therefore, the processing delay, and the queuing delay are negligible, while the propagation delay is constant. Only the transmission rate out of the source host, R bits/sec is considered.

From Definition 5, the transmission delay is the amount of time required to push (that is, transmit) all of the packet's bits into the link. The transmission delay is

$$D = L/R \tag{3.5}$$

where

- D be the transmission delay of a session,
- L be the length of the packet of a session,
- R be the transmission rate of a session.

In this case study, L and R are the considerate features and D is the considerate factor. With feature adaptation, if L or R changes, it has an effect on the ralated factor, D. Suppose R is considered and L is a constant, R is an inverse of D. If R is increased, D will be decreased and if R is decreased, D will be increased. This result is used as a strategy in this research. The relationships of D, L, and R in the wireless communication are observed by simulation and reported in Chapter 4.

3.3.2 Server Site QoS Management Functions

When a client sends a request to a server, the request classification process is invoked. In this stage, the maximum transmission delay which is suitable for the request service class

is calculated. Then, system starts the connection. For this case study, the server site QoS management function will concern only on the transmission rate (R) as a feature that impacts to the transmission delay (D), where the other features are left in term of $otherFeature$. From (3.1), the function is defined as follows:

$$\max\ satisfaction = \sum_{i=1}^{n} S(i)$$

subject to:

- $\forall S(i) \geq SLA_{min}(i)\ for\ i = 1\ to\ n$,
- $\sum_{i=1}^{n} R(i) \leq overallR$,
- $\sum_{i=1}^{n} otherFeature(i,j) \leq overallOtherFeature(j)$,

where

- n be the number of the sessions,
- $R(i)$ be a transmission rate that system provides for the i^{th} session,
- $overallR$ be the overall transmission rate of the system,
- $otherFeature = \bigcup_{j=1}^{m} feature(j) - R$,
- m be the number of the system's features,
- $overallOtherFeature$ be the overall of $feature(j)$ which is not the transmission rate.

Algorithm 2 describes the method to finding maximum transmission delay. $sim()$ is a function that finds a similar value of requested class and other class. The result of this algorithm is $class_{sel}$ which is an SLA between the server and a session. The SLA

Algorithm 2 Find maximum transmission delay

repeat

if $sim(class_{req}, class_{db}) > sim(class_{req}, class_{sel})$ **then**

$class_{sel} = class_{db}$

end if

until (searching all categorized knowledge)

return $class_{sel}$

consists of the related network parameters' threshold, $request_{min}$ and $request_{max}$ in the following algorithms.

Adaptation algorithms are decision making algorithms at the server site. These algorithms are invoked by three main situations:

1. ***New session*** When a new user requests for sharing the resources, the server processes the request by maintaining each user's satisfaction rate. *avail* is the available system resource that can be provided. Algorithm 3 describes a method for serving a new session request.

2. ***Renegotiation*** When the transmission seem to have a problem, the agent at the client site of the k^{th} session will send a request to ask for more resources. The server must process the request by maintaining the satisfaction rate of each user at less than or equal to the SLA. Algorithm 4 is the method for serving renegotiation process from the client.

3. ***Terminate a session*** When a session terminates its connection, the capacity which reserved for that session will be available for other sessions.

Algorithm 3 Serve a new session request

n = number of request that occupied the capacity

$i = 1$

while $((avail < request_{min})$ *and* $(i \leq n)$ *and* $(NOTTIMEOUT))$ **do**

if $(S(i) > SLA_{min}(i))$ **then**

reduce $S(i)$ by reducing $R(i)$

$R(i) = R(i) - reducedR(i)$

$avail = avail + reducedR(i)$

end if

$i = i + 1$

end while

if $(avail \geq request_{min})$ **then**

$assign = \min\{request_{max}, avail\}$

$avail = avail - assign$

else

reject request

end if

Algorithm 4 Serve renegotiation process from the client

n = number of requests that occupied the capacity

$i = 1$

$require_{min}(k) = request_{min}(k) - current(k)$

$require_{max}(k) = request_{max}(k) - current(k)$

while $((avail < require_{min}(k))$ *and* $(i \leq n)$ *and* $(NOTTIMEOUT))$ **do**

  if $((i \neq k)$ *and* $(S(i) > SLA_{min}(i))$ **then**

    reduce $S(i)$ by reducing $R(i)$

    $R(i) = R(i) - reducedR(i)$

    $avail = avail + reducedR(i)$

  end if

  $i = i + 1$

end while

if $(avail \geq require_{min}(k))$ **then**

  $assign = \min\{require_{max}, avail\}$

  $avail = avail - assign$

else

  reject request

end if

3.3.3 Client Site QoS Management Function

From (3.4), for a session k, the objective function of each user can be modeled in term of transmission delay as follows:

$$\max S(k) = \frac{acceptanceLevel(D(k)) + \sum_{j=1}^{m-1} acceptanceLevel(otherFactor(k,j))}{m}$$

subject to :

- $acceptanceLevel(D(k)) > 0$
- $\forall otherFactor(k,j),\ acceptanceLevel(otherFactor(k,j)) > 0$

where

- m be the number of factors,
- $D(k)$ be the transmission delay of the k^{th} session, and
- $otherFactor = \bigcup_{j=1}^{m} factor(j) - D$.

At the client site, the QoS management function is applied to maximize individual user's satisfaction. During transmission, the user agent monitors whether an average transmission delay, $D(k)$, is larger than the maximum transmission delay, $D_{max}(k)$, or not. If the average delay is larger than the maximum delay, the user agent sends the request to invoke the renegotiation process at the server site. The simple moving average technique in [80] is used to find the average delay. For an example, the last three delay values are used to calculate the average delay. Algorithm 5 is the method of how an agent at the client site detects the deviation. For this case study, the other remaining factors are left in term of $otherFactor$.

Algorithm 5 Monitor the moving average of transmission delay

repeat

if $[D_{t_{n-3}} + D_{t_{n-2}} + D_{t_{n-1}}/3] > D_{max}(k)$ **then**

renegotiation for lower transmission delay

end if

until (Terminate)

3.4 Incoming Request Scheduling

The Internet has important roles for everyday life, such as business, communication, education, and entertainment. The population of the Internet users grow rapidly which lead to high competition in every layer of service providers. For service provider's point of view, Quality of Service (QoS) can be used as a tool to improve and guarantee services to customers. At the same time, for customer's point of view, the QoS can be used as a key factor to choose the most satisfied service provider. The main function of the QoS is to manage services according to the SLA in order to achieve the customers' satisfaction.

This research concerns web server, which is a layer of the Internet's service provider. When users contact a web server, the first step that every user must face is an incoming request handling process. An important problem at this first step is the impatient user problem. The problem occurs when the users got none of the server response for long time and then abandon their request. This users' unsatisfaction lead the server deadlock situation because the server wastes its resources to the abandoned requests. Extending web server approaches are studied in many research areas to solve the problem both on server-centric and client-centric constraints.

Enhancing incoming request scheduling, instead of the best effort process, is a considerate approach. Incoming request scheduling is the approach to handle the processing order beneath the scarce resources system. Generally, web server uses the best-effort

policy to handle the requests. There are three widely used policies: First-In-First-Out (FIFO), Earliest-Deadline-First (EDF), and Shortest-Processing-Time-First (SPT). Typically, a scheduling policy dependents on only one criterion. For examples, FIFO method depends on arrival time, EDF method depends on deadline, and SPT method depends on processing time. Each policy has its own advantages and disadvantages. Although SPT has low average waiting time and number of rejected request but it has high maximum and standard deviation (Sd.) waiting times. FIFO and EDF, on the other hand, have low maximum and Sd. waiting times but they have high average waiting time and number of rejected request. It can be seen that the average waiting time and the number of rejected request conflict with the maximum and Sd. waiting times.

In this study, we propose a new scheduling policy for incoming web requests, called Multicriteria-Based (MCB), to solve the conflicting problem by considering more than one criterion and providing optimal results. This proposed policy is based on Multi-Criteria Decision Making (MCDM) concept. Because FIFO, EDF, and SPT are studied compare with MCB, therefore, arrival time, deadline, and processing time are determined as MCB's criteria. The problem is modeled and simulated in an M/G/1 queuing system. Waiting time information: average, maximum, and Sd. waiting times; are manipulated as factors of the performance measurement. Weighted aggregation approach is applied to the measurement function to quantify the performance of each comparative policy.

3.4.1 Reference Model

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 3.7.

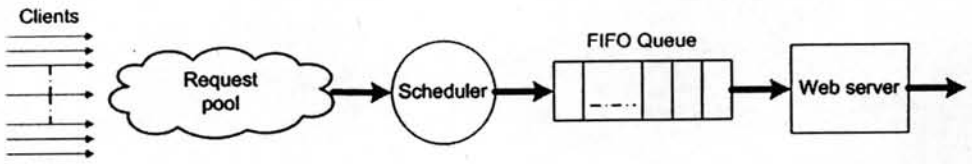

Figure 3.7: Incoming request scheduling reference model

The time-line of each request is shown in Figure 3.8. Arrival time is the point of time that a request arrives at the system's request pool. Waiting time is the duration of time that the request waits for execution. Launch time is the point of time that the request is executed. Service time is the duration of time using to execute the request. Completion time is the point of time that the request's execution completed. Response time is the total time that the request takes when it arrives at the system until its execution completed. Finally, due time or deadline is the point of time that the request is expected to complete its job.

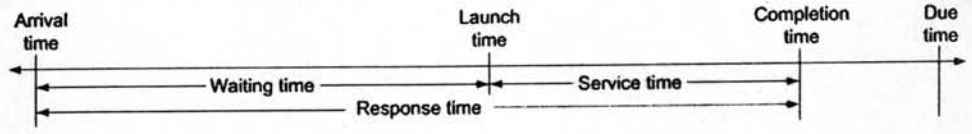

Figure 3.8: Timeline of each request in the reference model

3.4.2 Mathematical Model

The considerate problem consists of two parts: scheduling level and performance measurement level. At scheduling level, weighted aggregation method is applied to decide a scalar constraint value for the policy. This value is used to organize the requests in the FIFO queue where the request with minimum value gets the highest priority to execute. Additionally, at performance measurement level, the weighted aggregation method is

also used to conduct the different measurements into a single figure of merit. This value is used to discover the performance of each scheduling policy. The policy with the lowest value is a preference among the comparative policies. Both of the levels have the same objective to minimize a linear combination of K criteria [63]. To reduce the bias, each individual criterion is normalized to the same magnitude. A general function of the problems can be denoted as:

$$f_i = \sum_{j=1}^{K} w_j \tau_{i_j}^{'} \tag{3.6}$$

where i is a feature index, f_i is a scalar constraint value of i^{th} feature, j is a criterion index, K is number of considerate criteria, w_j is weight of j^{th} criterion, $w_j \geq 0 \ \forall j$ and $\sum_{j=1}^{K} w_j = 1$, and $\tau_{i_j}^{'}$ is a normalized function of j^{th} criterion of i^{th} feature. Weight summation can be assigned to any value. In this study, it is determined as the priority of the criteria. It is assigned to be 10 in the scheduling function for a reason of presentation and 1 in performance measurement function.

At Scheduling Level

In this level, the order of request to be executed is determined. Instead of using one criterion for the scheduling condition as the traditions, multiple criteria are considered. In this study, processing time, deadline, and arrival time are concerned as critical criteria of the condition. Each request is characterized by the following parameters:

$$Request_i(\pi_i,\ \delta_i,\ \alpha_i) \tag{3.7}$$

where i is a request index, π_i is processing time of i^{th} request, δ_i is deadline of i^{th} request, and α_i is arrival time of i^{th} request. From (3.6), mapping to the MCB, $\tau_{i_1}^{'}$ is normalized processing time of i^{th} request, $\tau_{i_2}^{'}$ is normalized deadline of i^{th} request, and $\tau_{i_3}^{'}$ is normalized arrival time of i^{th} request. For MCB, the request with the minimum constraint value has the highest priority to be executed. The scheduling function of

MCB policy can be represented as:

$$g_i = w_1\pi_i^{'} + w_2\delta_i^{'} + w_3\alpha_i^{'} \tag{3.8}$$

where g_i is a scalar value of i^{th} request, $\pi_i^{'}$ is normalized processing time of i^{th} request, $\delta_i^{'}$ is normalized deadline of i^{th} request, $\alpha_i^{'}$ is normalized arrival time of i^{th} request, and $w_1, w_2, w_3 \geq 0$ and $w_1 + w_2 + w_3 = 10$. This MCB's scheduling function can be mapped to the other scheduling policies as follows: in FIFO approach, arrival time is weighted with w_3 equals to ten while w_1 and w_2 equal to zero, in EDF approach, deadline is weighted with w_2 equals to ten while w_1 and w_3 equal to zero, and lastly, in SPT approach, processing time is weighted with w_1 equals to ten while w_2 and w_3 equal to zero.

For the scheduling complexity analysis, suppose a general sorting algorithm likes Quick sort is applied to EDF and SPT algorithms, worst-case running time of each scheduling cycle is $\Theta(n^2)$. In additional, MCB algorithm is added up with normalization processes that cause linear function complexity while FIFO has no overhead of normalization and sorting processes.

At Performance Level

From the scheduling level, waiting time information: average waiting time, maximum waiting time, and Sd. waiting time; are produced. The problem of the traditional scheduling policies is the average waiting time always contradicts to maximum and Sd. waiting times. These information are used as the criteria of the performance measurement function to determine the preferred scheduling policy. From (3.6), mapping to the performance measurement criteria, $\tau_{i_1}^{'}$ is normalized average waiting time of i^{th} policy, $\tau_{i_2}^{'}$ is normalized maximum waiting time of i^{th} policy, and $\tau_{i_3}^{'}$ is normalized Sd. waiting

time of i^{th} policy. The performance function can be represented as:

$$h_i = \lambda_1 A_i^{'} + \lambda_2 M_i^{'} + \lambda_3 X_i^{'} \tag{3.9}$$

where h_i is a scalar value of i^{th} policy, $A_i^{'}$ is normalized average waiting time of i^{th} policy, $M_i^{'}$ is normalized maximum waiting of i^{th} policy, $X_i^{'}$ is normalized Sd. waiting time of i^{th} policy, λ_j is weight of the criterion, and $\lambda_1, \lambda_2, \lambda_3 \geq 0$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

3.5 Mapping of Incoming Request Scheduling Problem

Incoming request scheduling as a function in the classification process in Figure 3.3 is an interesting problem because it is the first function that every users must face and contribution of improving this function has impact on a wide range of user. The increasing interest in scheduling problem has heightened the need for enhancing the satisfactory criteria in both server and client sites, instead of modeling the solution of this problem as the best effort algorithm. From [81, 82], many conventional approaches have been generated wide interests in server site as follows:

- First-In-First-Out (FIFO) emphasizes its work on fair service, which can be inferred to as the minimum variance of waiting time.
- Earliest-Deadline-First (EDF) states its work to minimize the maximum lateness of execution time, which can be inferred to as the minimum amount of tardy requests, i.e. the number of jobs that complete after their due date.
- Shortest-Processing-Time-First (SPT) attempts to minimize mean flow time, which can be inferred to as the maximum amount of serviceable requests.

This findings suggest that both EDF and SPT devote to model the scheduling problem in sever site view point, which are emphasized on the overall satisfaction of most

users, rather less attention has been paid to the individual user satisfaction at client site.

Although considerable research such as FIFO, states its attempt on serving the incoming request according to the request's arrival time and guarantee the fairness among user, focusing on maximize individual user satisfaction degree, it remains unclear whether this can be inferred to the maximizing of overall user satisfaction at client site.

This study attends to balance both client site and server site requirement in the server's provided QoS. By concerning on more dimensions, we propose a new multicriteria scheduling policy, called Multicriteria-Base (MCB). Mapping of incoming request scheduling problem to the QoS management function and its definitions are presented as follows.

3.5.1 Definitions for a priority changing of an incoming request

In order to model the proposed algorithm, some following definitions relevant to a priority changing of an incoming request are defined in this section.

Definition 6. *A priority changing of an incoming request vector*

In this study, we define the priority changing of incoming request at time period Δt in term of vector, hereafter referred to as vector $\vec{r}$. This definition is used as the general form of the consideration vector of either server site or client site.

$$\vec{r} = \nu_1 \hat{i} + \nu_2 \hat{j} + \nu_3 \hat{k} + \nu_4 \hat{l}$$

where

- $\hat{i}$ be the unit vector and has the positive direction according to x-axis,

- ν_1 be the scalar component of vector $\vec{r}$ corresponding to the direction of $\hat{i}$, representing priority deviation of processing time,
- $\hat{j}$ be the unit vector and has the positive direction according to y-axis,
- ν_2 be the scalar component of vector $\vec{r}$ corresponding to the direction of $\hat{j}$, representing priority deviation deadline,
- $\hat{k}$ be the unit vector and has the positive direction according to z-axis,
- ν_3 be the scalar component of vector $\vec{r}$ corresponding to the direction of $\hat{k}$, representing about time, denoted in this case as priority deviation arrival time,
- $\hat{l}$ be the unit vector and also has the positive direction according to z-axis, and
- ν_4 be the scalar component of vector $\vec{r}$ corresponding to the direction of $\hat{l}$, representing priority deviation of waiting time

Definition 7. *Function for modeling priority changing of incoming requests*

The behavior of priority changing of the overall incoming requests, hereafter referred to as function $\mathcal{R}(\pi', \delta', \alpha', \mu')$, is defined as follows:

$$\mathcal{R}(\pi', \delta', \alpha', \mu') = \nu_1\pi' + \nu_2\delta' + \nu_3\alpha' + \nu_4\mu'$$

where

- π' be the variable representing a normalized value of amount of processing time,
- ν_1 be priority deviation of processing time,
- δ' be the variable representing a normalized value of deadline,
- ν_2 be priority deviation of deadline,

- α' be the variable representing a normalized value of arrival time,
- ν_3 be priority deviation of arrival time,
- μ' be the variable representing a normalized value of waiting time, and
- ν_4 be priority deviation of waiting time

Definition 8. *A direction of priority changing for incoming requests*

The general form of a direction of of priority changing for incoming requests, hereafter referred to as $\nabla\mathcal{R}$, can be modeled by means of a gradient of function $\mathcal{R}(\pi', \delta', \alpha', \mu')$ as follows:

$$\nabla\mathcal{R} = \frac{\partial\mathcal{R}}{\partial\pi'}\hat{i} + \frac{\partial\mathcal{R}}{\partial\delta'}\hat{j} + \frac{\partial\mathcal{R}}{\partial\alpha'}\hat{k} + \frac{\partial\mathcal{R}}{\partial\mu'}\hat{l}$$

3.5.2 Definitions for a Scheduling Algorithm

In order to model the proposed algorithm, some following definitions relevant to a scheduling algorithm are defined in this section.

Definition 9. *An attempt of scheduling algorithm*

An attempt of scheduling algorithm, hereafter referred as function $\mathcal{A}(\pi', \delta', \alpha', \mu')$, is defined as follows:

$$\mathcal{A}(\pi', \delta', \alpha', \mu') = w_1\pi' + w_2\delta' + w_3\alpha' + w_4\mu'$$

where

- π' be the variable representing a normalized value of amount of processing time,
- w_1 be a concerning weight of processing time,
- δ' be the variable representing a normalized value of deadline,

- w_2 be a concerning weight of deadline,
- α' be the variable representing a normalized value of arrival time,
- w_3 be a concerning weight of arrival time,
- μ' be the variable representing a normalized value of waiting time, and
- w_4 be a concerning weight of waiting time

Definition 10. *A direction of scheduling attempt*

The general form of a direction of scheduling attempt, hereafter referred to as $\nabla\mathcal{A}$, can be modeled by means of a gradient of function $\mathcal{A}(\pi', \delta', \alpha', \mu')$ as follows:

$$\nabla\mathcal{A} = -\left(\frac{\partial\mathcal{A}}{\partial\pi'}\hat{i} + \frac{\partial\mathcal{A}}{\partial\delta'}\hat{j} + \frac{\partial\mathcal{A}}{\partial\alpha'}\hat{k} + \frac{\partial\mathcal{A}}{\partial\mu'}\hat{l}\right)$$

The direction of $\nabla\mathcal{A}$ is defined in the *reverse* direction, since the purpose of algorithm dealing with *decreasing* the value of concerning variable.

Definition 11. *Result vector from scheduling algorithm*

The results from scheduling algorithm are represented by vector, whose magnitudes and direction are quantified for *a concerning weight of* and a *direction* of the residual requests.

$$\vec{\mathcal{D}} = \nabla\mathcal{R} + \nabla\mathcal{A} \tag{3.10}$$

Definition 12. *A complete serving algorithm*

Let θ be an angle of $\nabla\mathcal{R}$ according to $\nabla\mathcal{A}$. The scheduling algorithm $\mathcal{A}(\pi', \delta', \alpha', \mu')$ is said to be a **complete serving algorithm** over the incoming requests representing by $\mathcal{R}(\pi', \delta', \alpha', \mu')$, if for any $\vec{r}$ the result after applying $\mathcal{A}(\pi', \delta', \alpha', \mu')$ must fall into one of these two sets of conditions:

1. **Conditions set 1: A *complete* serving algorithm with the *total serving results***

   (a) $\vec{\mathcal{D}} \in$ set of nonzero vectors, $\left\|\vec{\mathcal{D}}\right\| = 0$

   (b) $\theta = 180°$

   A *complete* serving algorithm with the *total serving results* will apply when the amount of incoming requests is less than or equal to the amount of serving capacity. This can refer that $\|\nabla\mathcal{R}\| \leq \|\nabla\mathcal{A}\|$. Thus, all incoming requests,$\vec{r}_{(1)}, \ldots, \vec{r}_{(n)}$, can be *serveable requests* by $\mathcal{A}(\pi', \delta', \alpha', \mu')$ at time period Δt.

2. **Conditions set 2: A *complete* serving algorithm with the *partial serving results***

   (a) $\vec{\mathcal{D}} \in$ set of nonzero vectors, $\left\|\vec{\mathcal{D}}\right\| > 0$

   (b) $\theta = 180°$

   A *complete* serving algorithm with the *partial serving results* will apply when the amount of incoming requests is greater than the amount of serving capacity, referring as $\|\nabla\mathcal{R}\| > \|\nabla\mathcal{A}\|$. Thus, the direction of $\vec{r}_{(n+1)}, \ldots, \vec{r}_{(m)}$, representing the *remaining requests* at time period Δt after applying $\mathcal{A}(\pi', \delta', \alpha', \mu')$, still *retain* their original directions.

Definition 13. *An incomplete serving algorithm*

Let θ be an angle of $\nabla\mathcal{R}$ according to $\nabla\mathcal{A}$. The scheduling algorithm $\mathcal{A}(\pi', \delta', \alpha', \mu')$ is said to be an **incomplete serving algorithm** over the incoming requests representing by $\mathcal{R}(\pi', \delta', \alpha', \mu')$, if for any $\vec{r}$ the result after applying $\mathcal{A}(\pi', \delta', \alpha', \mu')$ must fall into one of these set of conditions:

1. **Conditions set 1: An *incomplete* serving algorithm with the *total serving results***

   (a) $\vec{\mathcal{D}} \in$ set of nonzero vectors, $\left\|\vec{\mathcal{D}}\right\| = 0$

   (b) $\theta \neq 180°$

   An *incomplete* serving algorithm with the *total serving results* will apply when the amount of incoming requests is less than or equal to the amount of serving capacity, again, referring as $\|\nabla\mathcal{R}\| \leq \|\nabla\mathcal{A}\|$. Hence, all incoming requests,$\vec{r}_{(1)}, \ldots, \vec{r}_{(n)}$, can be *serveable requests* by $\mathcal{A}(\pi', \delta', \alpha', \mu')$ at time period Δt.

2. **Conditions set 2: An *incomplete* serving algorithm with the *partial serving results***

   (a) $\vec{\mathcal{D}} \in$ set of nonzero vectors, $\left\|\vec{\mathcal{D}}\right\| > 0$

   (b) $\theta \neq 180°$

   An *incomplete* serving algorithm with the *partial serving results* will apply when the amount of incoming requests is greater than the amount of serving capacity. From the above definition, it can infer that the direction of $\vec{r}_{(n+1)}, \ldots, \vec{r}_{(m)}$, representing the *remaining requests* at time period Δt after applying $\mathcal{A}(\pi', \delta', \alpha', \mu')$, *diverse* from their original directions.

3.5.3 Evaluation of Server Site and Client Site Scheduling Algorithm

The specific type of scheduling algorithm for both server site and client site can be modeled as follows:

1. **An attempt of server site algorithm**, hereafter referred as function $\mathcal{S}(\pi', \delta', \alpha', \mu')$, is defined as follows:

$$\mathcal{S}(\pi', \delta', \alpha', \mu') = w_1\pi' + w_2\delta' + w_3\alpha' + w_4\mu'$$

The main objective of the server site algorithm is to maximize the **overall** user satisfaction for all of the user encounter in the system. In this research, the overall user satisfaction can be inferred as number of successful user, lateness and fairness. On the other hand, the priority deviation of waiting time is not taken into account, this leads the value of w_4 is equal to zero. Hence, the definition for an attempt of server site algorithm becomes as follows:

$$\mathcal{S}(\pi', \delta', \alpha', \mu') = w_1\pi' + w_2\delta' + w_3\alpha'$$

2. **An attempt of client site algorithm**, hereafter referred as function $\mathcal{C}(\pi', \delta', \alpha', \mu')$, is defined as follows:

$$\mathcal{C}(\pi', \delta', \alpha', \mu') = w_1\pi' + w_2\delta' + w_3\alpha' + w_4\mu'$$

The main objective of the client site algorithm is to maximize the **individual** user satisfaction. In addition, the individual user satisfaction can be inferred as waiting time. Nevertheless, the successful user, lateness and fairness do not be concerned. This leads the value of w_1, w_2 and w_3 are equal to zero. Hence, the definition of an attempt of client site algorithm becomes as follows:

$$\mathcal{C}(\pi', \delta', \alpha', \mu') = w_4\mu'$$

3.5.4 Evaluation of Conventional Scheduling Algorithm

In additional, the evaluation of the conventional scheduling algorithms can be summarized as the following description.

1. **Attempt direction of SPT algorithm:** The attempt direction of SPT scheduling algorithm, hereafter referred as vector $\nabla \mathcal{A}_{SPT}$, can be modeled as the following equation

$$\begin{aligned}\nabla \mathcal{A}_{SPT} &= -\left(\frac{\partial \mathcal{A}}{\partial \pi'}\hat{i} + \frac{\partial \mathcal{A}}{\partial \delta'}\hat{j} + \frac{\partial \mathcal{A}}{\partial \alpha'}\hat{k} + \frac{\partial \mathcal{A}}{\partial \mu'}\hat{l}\right) \\ &= -\left(\frac{\partial \mathcal{A}}{\partial \pi'}\hat{i} + 0 + 0 + 0\right) \\ &= -\frac{\partial \mathcal{A}}{\partial \pi'}\hat{i} \\ &= -w_1\hat{i}\end{aligned}$$

It can be noted that for SPT algorithm the concerning attribute π', representing the amount of processing time, is taken into account, while the others are less consideration. The value of δ', α' and μ' respecting to this algorithm are considered as constant number. In this theoretical point of view, the attempt direction of this algorithm relies on the x-axis on minus direction.

Moreover, the discrepancy between the dynamic direction of incoming requests attributes and the attempt direction of SPT scheduling problem can be derived according to the definition of the *result vector of scheduling algorithm* as the following equation.

$$\begin{aligned}\vec{\mathcal{D}}_{SPT} &= \nabla \mathcal{R} + \nabla \mathcal{A}_{SPT} \\ &= \left(\nu_1\hat{i} + \nu_2\hat{j} + \nu_3\hat{k} + \nu_4\hat{l}\right) - (w_1\hat{i}) \\ &= (\nu_1 - w_1)\hat{i} + (\nu_2)\hat{j} + (\nu_3)\hat{k} + (\nu_4)\hat{l}\end{aligned}$$

The notable assumption is relies on the best case algorithm, in which $\mathcal{A}_{SPT}$ is an incomplete serving algorithm with the *total serving results*. Hence, the weight of a concerning attribute π' of attempt algorithm is equivalent to the weight of

processing time attribute for the incoming requests. The result of $\vec{\mathcal{D}}_{SPT}$ can be derived as:

$$\vec{\mathcal{D}}_{SPT} \approx \nu_2\hat{j} + \nu_3\hat{k} + \nu_4\hat{l}$$

The deriving result shows that $\vec{\mathcal{D}}_{SPT}$ have these following properties.

(a) $\vec{\mathcal{D}}_{SPT} \in$ set of nonzero vectors. Since $\left\|\vec{\mathcal{D}}_{SPT}\right\| = \sqrt{\nu_2^2 + \nu_3^2 + \nu_4^2}$, which does not equal to zero.

(b) The direction of $\vec{\mathcal{R}}$ is relies on both y-axis and z-axis, whereas the direction of $\nabla\mathcal{A}_{SPT}$ relies on x-axis. Thus, this leads to state that $\theta \neq 180°$.

This finding confirmed that the attempt of SPT algorithm covers some part of server site concerning attributes, while both of others server site concerning attributes and client site concerning attributes seem aimless. From the above definition, the SPT algorithm is said to be an incomplete serving algorithm.

2. **Attempt direction of EDF algorithm:** The attempt direction of EDF scheduling algorithm, hereafter referred as vector $\nabla\mathcal{A}_{EDF}$, can be modeled as the following equation

$$\begin{aligned}
\nabla\mathcal{A}_{EDF} &= -\left(\frac{\partial\mathcal{A}}{\partial\pi'}\hat{i} + \frac{\partial\mathcal{A}}{\partial\delta'}\hat{j} + \frac{\partial\mathcal{A}}{\partial\alpha'}\hat{k} + \frac{\partial\mathcal{A}}{\partial\mu'}\hat{l}\right) \\
&= -\left(0 + \frac{\partial\mathcal{A}}{\partial\delta'}\hat{j} + 0 + 0\right) \\
&= -\frac{\partial\mathcal{A}}{\partial\delta'}\hat{j} \\
&= -w_2\hat{j}
\end{aligned}$$

For EDF algorithm, the concerning attribute δ', representing deadline of incoming request, is taken into account, while the others are less consideration. Hence, the value of π', α' and μ' respecting to this algorithm are also considered as constant

number. In this theoretical view point, the attempt direction of this algorithm relies on the y-axis on minus direction, showing as the result from the above equation.

Additionally, the discrepancy between the dynamic direction of incoming requests attributes and the attempt direction of EDF scheduling problem namely as $\vec{\mathcal{D}}_{EDF}$, can be derived as follows:

$$\begin{aligned}\vec{\mathcal{D}}_{EDF} &= \nabla\mathcal{R} + \nabla\mathcal{A}_{EDF} \\ &= \left(\nu_1\hat{i} + \nu_2\hat{j} + \nu_3\hat{k} + \nu_4\hat{l}\right) - (w_2\hat{j}) \\ &= (\nu_1)\hat{i} + (\nu_2 - w_2)\hat{j} + (\nu_3)\hat{k} + (\nu_4)\hat{l}\end{aligned}$$

The notable assumption is relies on the best case algorithm, in which $\mathcal{A}_{EDF}$ is an incomplete serving algorithm with the *total serving results*. Thus, the weight of a concerning attribute δ' of attempt algorithm is equivalent to the weight of deadline attribute for the incoming requests. The result of $\mathcal{D}_{EDF}$ can be derived as:

$$\vec{\mathcal{D}}_{EDF} \approx \nu_1\hat{i} + \nu_3\hat{k} + \nu_4\hat{l}$$

The deriving result refines that $\vec{\mathcal{D}}_{EDF}$ have these following properties.

(a) $\vec{\mathcal{D}}_{EDF} \in$ set of nonzero vectors. Since $\left\|\vec{\mathcal{D}}_{EDF}\right\| = \sqrt{\nu_1^2 + \nu_3^2 + \nu_4^2}$, which does not equal to zero.

(b) The direction of $\vec{\mathcal{R}}$ relies on x-axis, y-axis and z-axis, whereas the direction of $\nabla\mathcal{A}_{EDF}$ relies on y-axis. Thus, this can infer that $\theta \neq 180°$.

This finding is the evidence showing that the attempt of EDF algorithm also covers some part of server site concerning attributes, while less attention has been paid to

either others server site concerning attributes or client site concerning attributes. From the above definition, the EDF algorithm is said to be an incomplete serving algorithm.

3. **Attempt direction of FIFO algorithm:** The attempt direction of FIFO scheduling algorithm, hereafter referred as vector $\nabla \mathcal{A}_{FIFO}$, can be modeled as the following equation

$$\begin{aligned}
\nabla \mathcal{A}_{FIFO} &= -\left(\frac{\partial \mathcal{A}}{\partial \pi'}\hat{i} + \frac{\partial \mathcal{A}}{\partial \delta'}\hat{j} + \frac{\partial \mathcal{A}}{\partial \alpha'}\hat{k} + \frac{\partial \mathcal{A}}{\partial \mu'}\hat{l}\right) \\
&= -\left(0 + 0 + \frac{\partial \mathcal{A}}{\partial \alpha'}\hat{k} + \frac{\partial \mathcal{A}}{\partial \mu'}\hat{l}\right) \\
&= -\left(\frac{\partial \mathcal{A}}{\partial \alpha'}\hat{k} + \frac{\partial \mathcal{A}}{\partial \mu'}\hat{l}\right) \\
&= -\left(w_3\hat{k} + w_4\hat{l}\right)
\end{aligned}$$

Finally, FIFO algorithm emphasizing it effort on consideration about the timing criteria, stated above as arrival time. This type of criteria is significantly corresponding to the waiting time criteria, which is a strong criteria indicating individual user satisfaction. Because the earlier time incoming request, is also inferred as the higher waiting time of that particular request. Thus, the concerning attributes α' and μ' are devoted to the attempt direction, while the others are less consideration. The value of π' and δ' respecting to this algorithm are also considered as constant number. In the theoretical view point, the attempt direction of this algorithm relies on the z-axis on minus direction, stating as the result from the above equation.

Furthermore, the discrepancy between the dynamic direction of incoming requests attributes and the attempt direction of FIFO scheduling problem namely as $\vec{\mathcal{D}}_{FIFO}$,

can be derived as follows:

$$\begin{aligned}\vec{\mathcal{D}}_{FIFO} &= \nabla\mathcal{R} + \nabla\mathcal{A}_{FIFO} \\ &= \left(\nu_1\hat{i} + \nu_2\hat{j} + \nu_3\hat{k} + \nu_4\hat{l}\right) - \left(w_3\hat{k} + w_4\hat{l}\right) \\ &= (\nu_1)\hat{i} + (\nu_2)\hat{j} + (\nu_3 - w_3)\hat{k} + (\nu_4 - w_4)\hat{l}\end{aligned}$$

The notable assumption is relies on the best case algorithm, in which $\mathcal{A}_{FIFO}$ is an incomplete serving algorithm with the *total serving results*, thus the weight of a concerning attribute α' and μ' of attempt algorithm is equivalent to the weight of arrival time attribute for the incoming requests. Thus, the result of $\vec{\mathcal{D}}_{FIFO}$ can be derived as:

$$\vec{\mathcal{D}}_{FIFO} \approx \nu_1\hat{i} + \nu_2\hat{j}$$

Moreover, the deriving result presents that $\vec{\mathcal{D}}_{FIFO}$ have these following properties.

(a) $\vec{\mathcal{D}}_{FIFO} \in$ set of nonzero vectors. Since $\left\|\vec{\mathcal{D}}_{EDF}\right\| = \sqrt{\nu_1^2 + \nu_2^2}$, which does not equal to zero.

(b) The direction of $\vec{\mathcal{R}}$ relies on x-axis, y-axis and z-axis, whereas the direction of $\nabla\mathcal{A}_{FIFO}$ relies on z-axis. Hence, this leads to state that $\theta \neq 180°$.

This leads to confirm that the attempt of FIFO algorithm covers client site concerning attribute and also encounters some part of server site concerning attributes. However, some others server site concerning attributes, such as deadline and processing time, remains aimless consideration. From the above definition, the FIFO algorithm is also said to be an incomplete serving algorithm, as SPT and EDF.

3.5.5 The Proposed Approach - MCB

The attempt direction of MCB scheduling algorithm, hereafter referred as vector $\nabla \mathcal{A}_{MCB}$, can be modeled as the following equation

$$\begin{aligned}\nabla \mathcal{A}_{MCB} &= -\left(\frac{\partial \mathcal{A}}{\partial \pi'}\hat{i} + \frac{\partial \mathcal{A}}{\partial \delta'}\hat{j} + \frac{\partial \mathcal{A}}{\partial \alpha'}\hat{k} + \frac{\partial \mathcal{A}}{\partial \mu'}\hat{l}\right) \\ &= -(w_1\hat{i} + w_2\hat{j} + w_3\hat{k} + w_4\hat{l})\end{aligned}$$

where

- π' be the variable representing a normalized value of amount of processing time,
- δ' be the variable representing a normalized value of deadline,
- α' be the variable representing a normalized value of arrival time, and
- μ' be the variable representing a normalized value of waiting time

In order to model the algorithm that can serve all direction of the changing in attributes value for incoming requests, the proposed algorithm should overcome the discrepancy in all aspects. Hence, the result vector from scheduling algorithm representing by $\vec{\mathcal{D}}_{MCB}$ can be defined as follows:

$$\vec{\mathcal{D}}_{MCB} = \left(\nu_1\hat{i} + \nu_2\hat{j} + \nu_3\hat{k} + \nu_4\hat{l}\right) - (w_1\hat{i} + w_2\hat{j} + w_3\hat{k} + w_4\hat{l})$$

The result vector from scheduling algorithm representing by $\vec{\mathcal{D}}_{MCB}$ have these following properties.

1. $\vec{\mathcal{D}} \in$ set of zero vectors, for the MCB algorithm with the total serving results. $\vec{\mathcal{D}}_{MCB}$ can derived as follows:

$$\begin{aligned}\vec{\mathcal{D}}_{MCB} &= \left(\nu_1\hat{i} + \nu_2\hat{j} + \nu_3\hat{k} + \nu_4\hat{l}\right) - (w_1\hat{i} + w_2\hat{j} + w_3\hat{k} + w_4\hat{l}) \\ &\approx 0\end{aligned}$$

The notable assumption for the total serving results algorithm relies on the equivalent on weight of a concerning attribute π', δ', α' and μ' and weight of processing time, deadline, arrival time and waiting time of the incoming requests, respectively. This can also be stated as $\|\nabla \mathcal{A}_{MCB}\|$ is equal to or greater than $\|\nabla \mathcal{R}\|$. In other words, this means the incoming requests is not exceed the serving capacity. Thus, all requests will be served.

2. $\vec{\mathcal{D}} \in$ set of nonzero vectors, for the MCB algorithm with the partial serving results.

$$\mathcal{D}_{MCB} = (\nu_1 - w_1)\hat{i} + (\nu_2 - w_2)\hat{j} + (\nu_3 - w_3)\hat{k} + (\nu_4 - w_4)\hat{l}$$

The notable assumption for the partial serving results algorithm relies on the inequivalent on weight of a concerning attribute π', δ', α' and μ' and the weight of processing time, deadline, arrival time and waiting time of the incoming requests, respectively. This can be stated as $\|\nabla \mathcal{A}_{MCB}\|$ is less than $\|\nabla \mathcal{R}\|$. In other words, this means the incoming requests is exceed the serving capacity. Thus, there are some incoming requests remaining in the queue.

However, the direction of $\nabla \mathcal{A}_{MCB}$ also relies on x-axis, y-axis and z-axis as the direction of $\vec{\mathcal{R}}$. Hence, this can infer as $\theta = 180°$.

This leads to confirm that the attempt of MCB algorithm covers the client site concerning attributes and also encounters all concerning attributes of a service site. Moreover, the MCB algorithm can be stated as a complete serving algorithm. Because the properties of the result vector from MCB scheduling algorithm, indicated in the above paragraph, satisfy the conditions of the complete serving algorithm.

Moreover, MCB scheduling algorithm proposed the performance level function, whose objective is stated as finding the optimal values according to the different situations. Additionally, this upcoming values must balance the individual user satisfaction concerning

at client site and the overall user satisfaction concerning at server site. In order to devise the performance level of MCB scheduling algorithm, some variables will be adjusted as follows:

1. Firstly, this can be done by increasing the serving capacity. This leads to increase the magnitude of $\left\|\vec{\mathcal{D}_{MCB}}\right\|$. Thus, the number of successful requests will increase. However, this is not practical in the real world environment

2. Secondly, devising the performance level function by *conversing* the direction of $\nabla\mathcal{A}$ into the direction of $\nabla\mathcal{R}$. The angle conversing intention can be conducted by adjusting an angle of $\nabla\mathcal{R}$ according to $\nabla\mathcal{A}$ to 180°. In other words, this declared intention can be done by *revising the weight*,w_1, w_2, w_3 and w_4, of each concerning attribute respecting to the direction of $\nabla\mathcal{R}$. This would seem more practical than the first solution.

Finally, we can conclude that the MCB algorithm is a complete serving algorithm. This algorithm focuses on serving the incoming request according to the multicriteria derived priority. Both of scheduling level function and performance function proposed by this MCB algorithm will result in the feasible dynamic adaptation algorithm.