CHAPTER 5

PROPOSED MODIFIED-MPR PROXY CACHING SYSTEM

The revolution of mobile communication technology has enabled the users to access the Internet anywhere, anytime, with heterogeneous mobile devices, such as hand-held PCs, Personal Digital Assistants (PDAs) and WAP(Wireless Application Protocol)-enabled cellular phones. These devices are different from one another in their computing capability, network connectivity and screen size. Due to the limited bandwidth of mobile communication, the traditional content of a web object for desktop computers might not be suitable for a mobile device. Hence, there is a desire to transcode the content to a reduced size format that is more appropriate to be presented on the mobile devices. The proposed Modified-MPR(Maximum Profit Replacement) caching system incorporated a transcoding proxy architecture with an improved caching mechanism for serving the user request.

5.1 INTRODUCTION TO TRANSCODING

The transcoding technology converts a multimedia object from one form to another, a lossy process, which trades object fidelity for size and preserves the important information. A proper transcoding process will effectively reduce the transmission time of a web object. However, a transcoding system should strike a compromise between the data information and the transmission efficiency. The time spent in transmission of the object will be long with small amount of distillation. In contrast, with large amount of distillation, useful information residing in the message might be lost. The
transcoding is useful when the device does not have the capability of processing the content, e.g., for a text-only mobile device, the image has to be transcoded into a text file before sending the content to the device.

5.1.1 Transcoding Systems

Transcoding systems can be divided into three classes according to the location where the transcoding process takes places, i.e., the client-based, the server-based, and the proxy-based transcoding systems. It is noted that transcoding at client side is costly due to the limitation in both bandwidth and computing power. On the other hand, transcoding at server side is not flexible to satisfy the client’s requirement and will require too much of unnecessary storage. Thus, the transcoding system is often implemented at an intermediate proxy.

5.1.2 Transcoding Proxies

Conventional transcoding proxies can be further divided into two different classes according to the way how the transformation logic is applied. The first class is referred to as the fixed transcoding proxy, where the transcoding proxy merely transcodes the input into the output without any context-aware processing. The second class is referred to as the heuristic based transcoding proxy, which is able to read the capability profile from the client device and tries to transform the content according to the device capability. Since the proxy does not have any knowledge about which information is important, it is difficult to determine the transformation strategy of the content.

Although recent researchers (Chandra et al 2001) have proposed various heuristics, these heuristics still suffer from the loss of important information or the loss of opportunities for better transcoding. It is noted that both the fixed transcoding proxy and the heuristic transcoding proxy belong to monolithic transcoders, meaning that they can only provide transcoding
services to the content type or the protocol which can be recognized in advance. When there is a desire to deal with a new content type or a communication protocol, the upgradation of the whole architecture is inevitable. This consideration makes it rather difficult to maintain the transcoding proxy system.

5.2 VERSATILE TRANSCODING PROXY SYSTEM

The schematic block diagram of the Versatile Transcoding Proxy (VTP) for Internet content adaptation is shown in Figure 5.1. In the VTP framework, the proxy can accept and execute the transcoding preference script provided by the client or the server to transform the corresponding data or protocol according to the user’s specification, so that the proxy server can avoid the uncertainty of the heuristic transcoding proxy. In addition, the system had been designed to provide the capability of supporting arbitrary type of data by utilizing the well-defined software modules in which the new plug-in components can be automatically downloaded from the Internet. The VTP architecture is used to transcode many types of client-server systems, without being limited to Web contents with HTTP protocol.

![Figure 5.1 Schematic Of The Versatile Transcoding Proxy](image-url)
5.2.1 Components Of VTP

The versatile transcoding proxy is a generic software agent container which supports heterogeneous data types and protocols. An agent is a clip of small Java bytecodes which can be downloaded from the Internet to provide extension and enhance the functionality of the original transcoding proxy. When the transcoding proxy needs to process unknown data type or protocol, the proxy automatically downloads the suitable software agent from the Internet according to the instructions specified in the transcoding preference script. Figure 5.2 shows the detailed architecture of the VTP system.

![Overall Architecture Of The VTP System](image)

**Figure 5.2 Overall Architecture Of The VTP System**

The major components of the architecture are:

- Transcoding Agent
- Service Agent
- Transcoding Preference Script
- Cache manager

Service agents are used to abstract the protocol details from the proxy system, while transcoding agents are used to abstract the transcoding actions. The service agent and the transcoding agent enable the VTP system to be adaptable to heterogeneous protocols and data types. The transcoding preference script is programmable, so that the user or the content provider can configure the related methods when a web object is transcoded. The cache manager performs the cache allocation and replacement of objects from the cache memory.

5.2.1.1 Transcoding Agent

The transcoding agent is responsible for the data transformation. It is capable of transcoding the content from one MIME (Multipurpose Internet Mail Extensions) content type to another MIME content type or to the same MIME type with degraded quality. The transcoding agent basically functions as a filter of the input data and multiple transcoding agents are able to cooperate with each other to accomplish a transcoding task. The transcoding agent in VTP can be further divided into two different types according to the awareness of the context.

The first type is context-aware transcoding agent. This type of transcoding agent will perform transcoding based on the client device’s CC/PP profile. In this type of transcoding agent, there is less but simpler control from transcoding preference script. The flexibility is high for the agent author but low for the user. The other type is dumb transcoding agent. This type of agent does not read the client device’s CC/PP profile. It just transcodes the content by the transcoding hint provided by the transcoding preference script. This type of transcoding agent leaves most of the flexibility to the transcoding preference script provider.
5.2.1.2 Service Agent

The service agent in the framework had been used to abstract the protocol details. The service agent applies the transcoding technology to many existing or future Internet application protocols. For example, to transcode the HTTP protocol, the proxy system can be tuned to employ a HTTP service agent. Thus the user can dynamically instruct the proxy to download the required service agent, to extend the functionality of the proxy system, and to transcode the protocol that is understood by the service agent.

5.2.1.3 Transcoding Preference Script

The Transcoding Preference Script (TPS) in VTP is a scripting language based on the JavaScript to provide the mechanism which allows the user or the server to determine the agent of processing the desired content, the time to perform transcoding and also the location where the agent should be downloaded. A transcoding preference script can pass the transcoding hints, i.e., TPS parameters, to the software agent. Users can load/unload agents by TPS and assign different agents to process the transcoding task in different contexts. TPS is able to read the attribute of CC/PP. Thus TPS can be used to determine the property of a device and ask the associated software agents to provide the suitable parameters for transcoding. CC/PP can be viewed as a vocabulary collection to describe the device capability without providing any transcoding function.

5.2.1.4 Cache Manager

The proposed system employs a transcoding enabled caching mechanism to store the multimedia objects requested by the clients. The caching system consists of the cache allocation and cache replacement schemes. The cache allocation is based on the profit associated with the
versions, where the profit depends on the popularity and the transcoding relationship among the versions of the objects. The proposed system performs the transcoding operation dynamically to serve the request.

### 5.2.2 Execution Scenario Of VTP

The execution scenario of the VTP system is depicted in Figure 5.3. Initially, a mobile device needs to register in the server. During the registration process, the device capability profile and the transcoding preference script are sent to the proxy server. The server will execute the transcoding preference script and download the designated service agent so as to set up the proxy server for the user. The connection ports in VTP can be shared among different users.

![Figure 5.3 Execution Scenario Of VTP](image-url)
The proxy system is able to identify the users by their source IP addresses of inbound connection and to dispatch the data stream to the corresponding service agent. After the registration process, the proxy is ready for transcoding. The transcoding preference script instructs the proxy server in performing the transcoding process. The service agent is responsible for decoding the TPS from the data stream. In addition to the default software agents and the agents configured by TPS, VTP can also download the software agents from the Internet. An additional server can be established to provide the software agents for special requirements in the Internet. The situation of downloading agents from the Internet occurs when the VTP encounters unknown protocols or data types which the client or the server are not able to develop the transcoding agent by TPS.

5.3 PROPOSED MODIFIED-MPR CACHING SCHEME

In order to enhance the effectiveness of the VTP architecture, a caching scheme namely Modified-MPR algorithm had been proposed to cache and replace the objects in the transcoding proxy.

The proposed system consists of the version database $D$ which contains all possible objects and their relevant versions. For each object $i$, the number of transcodable versions, i.e., the categories, is denoted by $n_i$. Let $O_{i,j}$ denote the $j^{th}$ version of a given object $i$. The original version of object $i$ is denoted as $O_{i,1}$, whereas the least detailed version which cannot be transcoded any more is denoted as $O_{i,n_i}$. 
Figure 5.4 Architecture Of The Proposed Modified-MPR Caching Scheme

The proposed scheme consists of two major components namely:

i. Dynamic Cache Categories (DCC) mechanism
ii. Maximum Profit Replacement (MPR) algorithm

The functional block diagram of the proposed caching scheme is shown in the Figure 5.4.

5.3.1 Dynamic Cache Categories (DCC) Mechanism

Scheme DCC offers fine granularity control in the number of cache categories by building a Weighted Transcoding Graph (WTG) which depicts the transcoding relationship among transcodable versions dynamically. The proposed mechanism DCC contains the procedure to maintain the cache categories of the transcoding proxy. The weighted transcoding graph represents the relationship among different versions of an object. Table 5.1 explains the terms used in the DCC mechanism.
Table 5.1 Description Of Terms Used In The DCC Mechanism

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>Object number</td>
</tr>
<tr>
<td>$G_i$</td>
<td>Weighted transcoding graph of object $i$</td>
</tr>
<tr>
<td>$o_{i,j}$</td>
<td>$j$th version of the object $i$</td>
</tr>
<tr>
<td>$v$</td>
<td>Individual vertex in the transcoding graph</td>
</tr>
<tr>
<td>$V$</td>
<td>Set of vertices in the graph</td>
</tr>
<tr>
<td>$(u,v)$</td>
<td>Individual edge in the transcoding graph</td>
</tr>
<tr>
<td>$E[G]$</td>
<td>Set of edges in the graph</td>
</tr>
<tr>
<td>$w(u,v)$</td>
<td>Transcoding cost from version $u$ to $v$</td>
</tr>
</tbody>
</table>

The WTG, $G_i$, is a directed graph with weight function $w_i$. $G_i$ depicts the transcoding relationship among transcodable versions of object $i$. Each vertex $v \in V[G_i]$ represents a transcodable version of object $i$. Version $u$ of object $i$, i.e., $o_{i,u}$ is transcodable to version $v$, i.e., $o_{i,v}$ if there is a directed edge $(u,v) \in E[G_i]$. The transcoding cost from version $u$ to $v$ is given by $w_i(u,v)$, i.e., the weight of the edge from $u$ to $v$. Figure 5.5 (a) illustrates the example of a weighted transcoding graph.

In the VTP architecture, since the proxy is able to perform the transcoding according to the user preferences, the categories of the cached object will change dynamically. In order to maintain dynamic cache categories, three procedures, i.e., AddCate($G_i$), RemoveCate($G_i$), and GetSubgraph($G_i$, $V'$) are defined. Procedure AddCate($G_i$) adds a new vertex to $G_i$ whenever a new version is created. The transcoding relationship will be updated simultaneously once a vertex is added. Figure 5.5(b) shows the result of procedure AddCate($G_i$) after the vertex $v_5$ is added to the original graph in Figure 5.5(a).

On the other hand, procedure RemoveCate($G_i$) is used to update the transcoding graph after a category is removed. Figure 5.5(c) depicts the result
of procedure RemoveCate($G_i$) after the vertex $v_4$ is removed from the original graph in Figure 5.5(a). Procedure GetSubgraph($G_i$, $V'$) derives the subgraph which minimizes the transcoding delay when caching a certain set of versions of the object $i$. Here $V'$ denotes the set of the versions that the transcoding proxy tries to cache. Figure 5.5(d) illustrates the result of procedure GetSubgraph($G_i$, $V'$). Given $V' = \{v_1, v_2\}$ and $G_i$ in Figure 5.5(a), procedure GetSubgraph($G_i$, $V'$) finds the edge with minimum weight to reach $v_3$ and $v_4$.

(a) Weighted Transcoding Graph   (b) Procedure AddCate($G$)

(c) Procedure RemoveCate($G$)   (d) Procedure GetSubgraph($G$)

Figure 5.5 Illustrative Example Of The DCC Mechanism

5.3.2 Modified-Maximum Profit Replacement (MMPR) Algorithm

With the transcoding relationship among categories expressed as Weighted Transcoding Graph (WTG), the Modified-MPR algorithm first
identifies the Caching Candidate Set (CCS), which had been generated by the concept of Dynamic Programming (DP). Then the algorithm MPR performs the cache allocation and replacement based on the Caching Candidate Set that was obtained using DP. The Modified-MPR was divided into two phases. The first phase was performed when the proxy has sufficient space. In this phase, once the object is requested, it will be cached to increase the profit for future access. The second phase had been performed when the proxy has insufficient space. In this phase, the cache replacement is performed according to the priority of the requested object.

The Modified-MPR algorithm had been designed according to the weighted transcoding graph defined in mechanism DCC. A transcoding proxy should perform cache replacement according to the following concerns:

i) A version with high popularity is profitable to be cached.

ii) A version with small item size is profitable to be cached.

iii) A version which is helpful to reduce the transcoding delay of other versions is profitable to be cached.

The overview of the Modified-MPR algorithm is given below:

1. For each incoming query, check the cached items. If cache hit occurs, return the cached item.

2. If cache miss occurs, check the requested item size and the available space in the cache. If the available space of the cache is larger than or equal to the requested item size, then cache the item after retrieving it either from the server or from other cached versions. Finally, update the profit of the associated objects and the objects in the high priority set $D_H$. 


3. If the available space of the cache is smaller than the requested item size, check the caching candidate set $D_H$. If the requested item belongs to $D_H$, remove the item with the smallest generalized profit until the available space of the cache is larger than or equal to the size of the requested item. Place the requested item into the cache in the same way as in (2), and update the profit of the associated objects and the objects in the high priority set $D_H$.

Thus the algorithm MPR can be divided into two phases. The first phase was performed when the proxy has sufficient space to cache the requested item. The second phase was performed when the proxy has insufficient space to cache the requested item.

5.3.2.1 Symbol Definition

**Singular Profit** :- $PF(o_{i,j})$ : It is the profit of caching $o_{i,j}$ while no other version of object $i$ is cached.

$$PF(o_{i,j}) = \sum_{(j,x) \in E[G_i]} r_{i,x} \cdot (d_i + w_i(1,x) - w_i(j,x))$$  \hspace{1cm} (5.1)

where $E[G_i]$ represents the collection of all edges in graph $G_i$. It is noted that the reference rate $r_{i,x}$ reflects the popularity, whereas the term $(d_i + w_i(1,x) - w_i(j,x))$ is regarded as the delay saving.

**Aggregate Profit** : $PF(o_{i,1}, o_{i,2}, \ldots o_{i,k})$ : It is defined as the profit of caching $(o_{i,1}, o_{i,2}, \ldots o_{i,k})$ simultaneously.

$$PF(o_{i,1}, o_{i,2}, \ldots o_{i,k}) = \sum_{v \in V[G'_i]} \sum_{(v,x) \in E[G'_i]} r_{i,x} \cdot (d_i + w_i(1,x) - w_i(v,x))$$  \hspace{1cm} (5.2)
where $G'_{1}$ is the subgraph derived from the procedure \text{GetSubgraph}(G, \{ o_{i,1}, o_{i,2}, \ldots o_{i,k} \})

\textbf{Marginal Profit}: $PF(o_{i,j} \mid o_{i,1}, o_{i,2}, \ldots o_{i,k})$ : It is the profit of caching $o_{i,j}$, given that $o_{i,1}, o_{i,2}, \ldots o_{i,k}$ are already cached where $j \neq 1,2, \ldots, k$.

\[
PF(o_{i,j} \mid o_{i,1}, o_{i,2}, \ldots o_{i,k}) = PF(o_{i,j}, o_{i,1}, o_{i,2}, \ldots o_{i,k}) - PF(o_{i,1}, o_{i,2}, \ldots o_{i,k}) \tag{5.3}
\]

Here each item $o_{i,j}$ contains two attributes namely the profit $p_{i,j}$ and object size $s_{i,j}$.

Now $p_{i,j} = PF(o_{i,j})$ if no other version of object $i$ is cached and $p_{i,j} = PF(o_{i,j} \mid o_{i,1}, o_{i,2}, \ldots o_{i,k})$ if there are some other versions namely $o_{i,1}, o_{i,2}, \ldots o_{i,k}$ cached.

\textbf{Generalized Profit} :- $g_{i,j}$ : It is the ratio of the profit to the size of the object $o_{i,j}$,

\[
g_{i,j} = \frac{p_{i,j}}{s_{i,j}} \tag{5.5}
\]

\textbf{5.3.2.2 Caching Candidate Set (CCS) Selection}

The Modified-MPR algorithm tries to cache the objects that increases the profit value which results in the improvement of the system performance. The object that has higher profit value are given high priority for caching and is called as a caching candidate. The set of objects that has high priority for caching forms the Caching Candidate Set (CCS), termed as $D_{H}$. Thus the caching candidate set $D_{H}$ is a subset of $D$ which contains the objects with high priority that can be given preference for caching. The low priority objects which are not in $D_{H}$ are termed as Non Candidate Set (NCS).
Total profit of $D_H (P_H)$: It is the summation of the profit of all data items, including the original and transcoded ones, in $D_H$.

Total size of $D_H (S_H)$: It is the summation of the object size of all data items in $D_H$.

The problem of determining $D_H$ can be formulated as a 0–1 knapsack problem, in which $P_H$ and $S_H$ can be viewed as the entire value and entire weight of the items carried in the knapsack. This 0–1 knapsack problem can be optimally solved by the concept of dynamic programming. The set of versions which improves the profit of caching with respect to different cache sizes had been identified. The determination of the caching candidate set using the dynamic programming procedure is shown in Figure 5.6.

```
Procedure Caching Candidate Set Selection
Input: database D, cache size constraint $Z_c$
/* $|D| = m*n$, where there are $m$ kinds of objects, and $n = \max \{n_i\}$ */
Output: caching candidate set $D_H$
Create two-dimensional $(m*n + 1) \times (Z_c + 1)$ array $c[]$
for $i=1$ to $m$
  for $j=1$ to $n$
    for $z=1$ to $Z_c$
      if $s_i \leq z$
        if
          $P_{i,j} + c[(i-1)*n+j-1][z-s_i] > c[(i-1)*n+j-1][z]
          c[(i-1)*n+j][z] = P_{i,j} + c[(i-1)*n+j-1][z-s_i]
        else
          $c[(i-1)*n+j][z] = c[(i-1)*n+j-1][z]
        else
          $c[(i-1)*n+i][z] = c[(i-1)*n+i-1][z]
```

Figure 5.6 Algorithm For Caching Candidate Set Selection
5.3.2.3 Modified-MPR Algorithm Phase-I

The first phase of the Modified-MPR algorithm performs the caching function when the proxy has sufficient space to accommodate the requested item. In this phase, once an item is queried, it will be cached to increase the profit for future access. That is, the proxy will cache as many items as it can.

The details of the algorithm is summarized as follows.

i. For each user request, check for the requested version of the object i.e the item in the cache. If cache hit occurs, then return the requested version.

ii. If cache miss occurs, check if sufficient cache space is available for the requested version. If the available space of the cache is larger than or equal to the requested version size, then fetch the item. There are two cases in retrieving the required version:

a) If the requested version can be transcoded from some current version in the cache, then the decision has to be taken whether to perform the transcode operation to get the required version or to download it from the server itself. The decision is based on the tradeoff between the transcoding time and the transmission time of the object.

b) If the requested object cannot be transcoded from any item in the cache, then download the requested version from the remote server.

iii. Place the retrieved item in the cache.

iv. Finally, update the profit of the associated objects and $D_H$. 
The operation of the Modified-MPR algorithm Phase-I is shown in Figure 5.7.

Figure 5.7 Flowchart Of Modified-MPR Algorithm Phase-I
Considering The NCS Versions

The low priority versions that are not part of the Caching Candidate Set are referred to as the Non Candidate Set (NCS). In the existing MPR algorithm the NCS had not been considered for caching even when there is enough cache space. Thus the proposed modified-MPR algorithm caches the NCS versions if there is sufficient cache space. These NCS objects had been cached so as to increase the number of objects that are retrieved from the cache, which improves the response time of the system. The efficiency of this method can be determined by analyzing the Byte Hit Ratio (BHR). BHR is defined as the ratio of the amount of data served from cache to the amount of data requested by the user. The BHR of the system had been analyzed with respect to different cache sizes. The proposed system shows an improvement of 2% in terms of BHR when compared to the existing system.

Retrieving The Required Version

When the required version is not present in the cache then the version has to be retrieved either from the server directly or by transcoding from the existing transcodable versions. The decision of whether to download from the server or to transcode depends on the transmission time and the transcoding time of the object. If the transcoding takes more time than actually fetching the version from server then the object is retrieved from the server. Otherwise the transcoding is performed for getting the object.

In the existing MPR scheme, the object is always transcoded if any transcodable version is available. In the proposed Modified-MPR scheme the object is retrieved efficiently by making the decision dynamically. Particularly for a least detailed version, which is the smallest in size can be
downloaded instead of transcoding it in the proxy. This reduces the response
time of the system. The efficiency of the system is determined by considering
the Communication Delay Ratio (CDR). CDR is defined as the ratio between
the delay due to transmission and the delay without transmission.

5.3.2.4 Modified-MPR Algorithm Phase-II

The second phase is meant for caching objects when the proxy has
insufficient space to cache the requested item. Thus the cache replacement is
carried out to store the requested item. The replacement policy of Modified-
MPR is to keep the object with high priority and to remove the object with
low priority. If the requested object belongs to the Caching Candidate Set
(CCS), then replacement has to be done to keep the requested item, since CCS
contains the items with high priority,

The operation of the Modified-MPR algorithm Phase-II is shown in
Figure 5.8. First the objects from NCS if any available in the cache had been
considered for replacement to accommodate the requested object. Still if the
freed cache space had not been sufficient, then replacement had been carried
out with the high priority object itself. Now the generalized profit value had
been used to decide the items to be removed from the cache. The item with
the smallest generalized profit in the cache will be removed first. The removal
procedure continues until the proxy has enough space to cache the requested
object.
Figure 5.8 Flowchart of Modified-MPR Algorithm Phase - II
5.4 EXPERIMENTS AND RESULTS

The proposed system had been analyzed in terms of the improvement in cache benefit and the user latency which had been evaluated using the factors namely the Byte Hit Ratio (BHR) and the Communication Delay Ratio (CDR).

5.4.1 Experimental Setup

The sizes of the five versions were set to 100%, 80%, 50%, 30%, and 10% of the original object size. A more detailed version can be transcoded to a less detailed one. In the transcoding proxy model, the number of different objects taken was 100 with 5 versions in each, leading to 500 different versions. The different file formats considered for the video objects had been AVI (Audio Video Interleaved), MPEG (Moving Pictures Expert Group) and QuickTime. The cache capacity of the system had been set to 10% of the original object database. Table 5.2 shows the experimental parameters for the proposed caching scheme.

Table 5.2 Experimental Parameters For The Modified-MPR algorithm

<table>
<thead>
<tr>
<th>Experimental Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of videos</td>
<td>100</td>
</tr>
<tr>
<td>Number of versions in each video</td>
<td>5</td>
</tr>
<tr>
<td>Skewness parameter</td>
<td>0.5 to 1.3</td>
</tr>
<tr>
<td>Diversity parameter</td>
<td>100 to 200</td>
</tr>
<tr>
<td>Number of requests</td>
<td>1500</td>
</tr>
<tr>
<td>Cache size with respect to database</td>
<td>10%</td>
</tr>
<tr>
<td>Channel bandwidth</td>
<td>1 unit / second</td>
</tr>
<tr>
<td>Transcoding rate</td>
<td>5 to 10 units / second</td>
</tr>
<tr>
<td>Ratio of version sizes of video</td>
<td>100%, 80%, 50%, 30%, 10%</td>
</tr>
</tbody>
</table>
5.4.2 Results And Discussion Of Modified-MPR Caching Scheme

The experiment had been designed to examine the performance of the proposed MPR scheme. The various metrics used to evaluate the performance of the system are the Exact Hit Ratio (EHR), Byte Hit Ratio (BHR) and the Communication Delay Ratio (CDR). The results had been obtained and compared between the existing MPR scheme and the proposed Modified-MPR (MMPR) scheme.

5.4.2.1 Analyzing The Exact Hit Ratio

The exact hit ratio is defined as the fraction of requests which are satisfied by the exact versions of the objects cached. This metric is motivated by the fact that the users are provided only with the exact version which they required (rather than an overqualified one) for effective bandwidth use.

The influence of the reference rate of the object on the exact hit ratio had been analyzed. The exact hit ratio increases as the value of skewness factor ($\ell$) increases which is shown in Figure 5.9.

![Figure 5.9 Influence Of Skewness Factor](image-url)
The reference rate of each object is generated by the Zipf distribution function \( r_i = \frac{1}{i^\theta} \sum_{j=1}^{N} (1/j) \), where \( \theta \) is viewed as a skewness parameter. The profit of the system depends on the reference rate of the objects requested. Thus the increase of the skewness improves the profit. Also, the increasing skewness enhances the locality of the object requests. Therefore, the exact hit ratio increases as the value of skewness factor increases. The proposed method showed an improvement of around 2% in terms of EHR when compared with the existing scheme for different values of skewness.

The size of each object is represented by the uniform distribution between \([0, \phi]\) units, where \( \phi \) is defined as the diversity parameter. Figure 5.10 depicts that the EHR decreases as the value of diversity parameter increases. However the proposed Modified-MPR method outperformed the existing method by 2% in terms of EHR for different diversity values. Since the increase of \( \phi \) granulates the object size, it will be more likely that the cache space is not utilized efficiently, which affects the performance of the system.

![Figure 5.10 Influence Of Diversity Parameter](image-url)
5.4.2.2 Analyzing The Byte Hit Ratio

Byte Hit Ratio is calculated by considering the size of the objects that are requested by different clients. An increase in the BHR value denotes the increase in the efficiency of the system. Figure 5.11 depicts the influence of cache size on BHR with the inclusion of the Non-Candidate Set objects. The BHR increases with the increase in cache size and the proposed method outperforms the existing method by an amount of 2% by considering the NCS objects.

![Byte Hit Ratio](image)

**Figure 5.11 Influence Of Cache Size On BHR**

5.4.2.3 Analyzing The Communication Delay Ratio

Communication Delay Ratio (CDR) takes into account the transmission time of the object from the server and the transcoding time of the object at the proxy. The decision of downloading the version instead of
transcoding always, leads to the reduction of the CDR value which results in
the improvement of the response time of the system.

Figure 5.12 depicts the influence of the cache size on CDR of the
system. A decrease in CDR value denotes the increase in the efficiency of the
system in terms of the response time. It had been observed that the CDR
decreases with an increase in cache size. The proposed Modified-
MPR(MMPR) method outperformed the existing MPR method by 4% in
terms of CDR.

![Communication Delay Ratio](image)

**Figure 5.12 Influence Of Cache Size On CDR**

5.5 **SUMMARY**

The proposed Modified-MPR caching scheme has been used to
cache the individual versions of the objects. This proposed algorithm
efficiently handles the requests of heterogeneous clients with different
resource capabilities. The versions from low priority set is also considered for
caching when there is sufficient cache space in the proxy which leads to
improved cacheability. Dynamic transcoding decision had been taken to
reduce the user latency. This enables the proxy to dynamically transcode the object on demand and thereby reducing the user latency.

The Modified-MPR algorithm consists of two phases. The first phase deals with cache allocation when the proxy has sufficient cache space for storing the requested object. The NCS versions had been cached when the cache space is available. The second phase deals with cache replacement when the cache space is insufficient for storing the requested object. The Modified-MPR algorithm performs the cache replacement by giving preference to the content in the CCS.

The proposed Modified-MPR caching system had been analyzed in terms of the evaluation factors namely EHR, BHR and CDR. The proposed method showed an improvement of around 2% in terms of EHR when compared to the existing scheme for different skewness and diversity values. The proposed system has been observed with different cache sizes for the BHR. There had been an improvement of 2% in terms of BHR when the NCS versions had been cached. While analyzing the system in terms of reduction in CDR, it had been found that the proposed method outperforms the existing method by 4%.