CHAPTER 6
ADAPTIVE WEB TRAFFIC/LATENCY REDUCTION

6.1 Introduction

Users interact with web browsers to fetch the web pages. The browser can store the history of each user’s behaviour and it can take intelligent decision depending upon the user behaviour and traffic condition with the help of intelligent agents. In the adaptive traffic/latency reduction technique, intelligent agents are used at the client and web server side to assist them to take intelligent decision on reducing the size of the document for each web request. In this technique, most of all existing web traffic/latency reduction algorithms are incorporated in order to improve the web performance. In this chapter, section 2 describes the adaptive web traffic/latency reduction algorithms at client and web server side. Section 3 describes the simulation environment and the results obtained. The work reported in this chapter appears in two publications [230] and [231].

6.2 Adaptive Web Traffic/Latency Reduction Algorithms

A hybrid technique for web traffic/latency reduction called adaptive traffic/latency reduction technique is reported in this section. In this technique, the existing traffic/latency reduction techniques such as web caching, web cache sharing, dynamic pre-fetching, optimization of HTML codes, compression, bandwidth controlling and traffic dispersion are combined. Web traffic/latency
reduction is achieved in adaptive manner by monitoring the user's preferences and bandwidth usage. In this new adaptive technique, intelligent agents are maintained at the client and server side for monitoring the web traffic. In this technique, intelligent agents monitor the bandwidth usage and select appropriate settings at the client side and server side to achieve maximum traffic/latency reduction. Adaptive web traffic/latency reduction algorithm at client side is incorporated into the dynamic pre-fetching browser (DPB). It can be used for single and multi user systems. When DPB is used for a LAN, the user can access DPB at any of the current systems in that LAN. Wherever a user logged in, that user's profile is taken from the user profile database of proxy server. Schematic block diagram of the functions of intelligent agents at web server and browser side is shown in Figure 6.1.

6.2.1 Web Traffic/latency Reduction at Client Side

The intelligent agents help the browser to make it intelligent. Agents at the client side monitor the user's preferences and bandwidth usage, then takes decision dynamically for reducing latency and web traffic. Web caching, pre-fetching, bandwidth controlling, compression, optimization of HTML codes and maintaining user preferences are incorporated into the client side adaptive traffic/latency reduction algorithm. In adaptive traffic/latency reduction technique, cacheable requests only stored in cache. Dynamic and secured documents are not searched in cache, such URL requests are directly forward to the web server. Caching is done in different levels such as in browsers and proxy servers. Browser is implemented with browser cache for reducing latency.
Figure 6.1 Schematic block diagram of the functions of Intelligent Agents at Web server and Browser side
Proxy's hard disk or main memory is used for storing the cacheable documents at client side. If several proxy servers are collaborated to share their cache contents, caching protocols are used for exchanging the cache directory information. In adaptive traffic/latency reduction technique, pre-fetching technique is integrated into the caching technique. Whenever a user is fetching a page and bandwidth usage of the internet connection link is less, subsequent links can be pre-fetched. The number of links to be pre-fetched is decided using dynamic pre-fetching technique. Traffic threshold levels say x1, x2 can be set by the administrator or default values can be used. If the traffic monitoring agents reported that bandwidth utilization is greater than x2, the web traffic is high. Then pre-fetching is not allowed. Dynamic refreshing of cache contents is also not permitted. Images are not downloaded. Only link of images are displayed in the web page. If the user clicks on that links, corresponding image are downloaded. The user can set an option, whether images are to be downloaded when bandwidth utilization is greater than x2. If threshold is between x1 and x2, then pre-fetching is not allowed. But cache refreshing is allowed and images are also downloaded. If traffic is below x1, bandwidth utilization is low and all traffic is allowed. At client side, optimization of HTML codes and compression techniques are used for reducing the size of the files. Intelligent agents are used for identifying user preferences from log information, when searching the user preferred sites are displayed first.
Dynamic pre-fetching browser is designed to work both in PC and LAN based on client side adaptive traffic reduction algorithm. When DPB is used in PC, proxy server works on the same system in which the browser runs. When DPB is used in LAN, the proxy server and browsers may be run in different systems. Even if DPB can be used both in PC and LAN, pre-fetching can be done only if the bandwidth usage is less than a particular threshold. Since dynamic pre-fetching technique is used, user access patterns are stored as weights in hash table by considering the entire LAN as a single network. A single hash table is only used for maintaining the weights of URLs accessed by all users in the LAN. Pseudo code of client side algorithm for web traffic/latency reduction is given below.

For each URL

Begin

Flag = 0

Begin

If (request = "search")

Begin

Replycode = forward (URL, webserver)

If (Reply_code = "OK")

Begin

Doc = Accept (Document)

Display (Sort (Doc, Preference_keyword))

End

End

If (request = "dynamic" or request = "secure") forward (URL, webserver)

Else

Begin

found = Search (URL, browser_cache)

If (found)

Begin
Document = fetch (URL, browser_cache)
Display (Document)
Flag = 1
End

found = Search(URL, local_cache)
If (found)
Begin
  Document = fetch (URL, local_cache)
  Display (Document);
  Flag = 1
End

Found = search (URL, remote_summary)
If (found) replycode = send (proxy, queryURL)
If (replycode = "OK")
Begin
  Accept (Proxy, Document)
  Store (URL, Document, local_cache)
  Display (Document)
  Flag = 1
End
End
If (flag = 0)
Begin
  Replycode = forward (URL, webserver)
  If (Reply_code = "OK")
  Begin
    Accept (Document)
    Store (URL, Document, local_cache)
    Display (Document)
  End
End
End

Bandwidth_usage = Monitor (web traffic)
// x1, x2, x3 are threshold values set by administrator
If (Bandwidth_usage < x1)
Begin
  Allow (alltraffic)
  Dynamic_Pre-fetch (SubsequentURL)
Else
If (Bandwidth_usage < x2) DonotAllow (pre-fetching)
Else
Begin
  DonotAllow (pre-fetching, cache_refreshing)
  Display (link_images) // display only links of images
  If clickon (link_images) Fetch (images)
End
If (upload(file))
Begin
  file = Optimize(file)
  // limit - Maximum file size allowed to send without compression
  If (filesize (file) > limit) file = Compress(file)
  Send (file)
End
End

Pseudo code of Intelligent Agent 1:
If (request = "search")
Begin
  Replycode = forward (URL, webserver)
  If (Reply_code = "OK")
  Begin
    Doc = Accept (Document)
    Display (Sort (Doc, Preference_keyword))
  End
End

Pseudo code of Intelligent Agent 2:
If (request = "dynamic" or request = "secure") forward (URL, webserver)
Else
Begin
found = Search (URL, browser_cache)
If (found)
Begin
    Document = fetch (URL, browser_cache)
    Display (Document)
    Flag = 1
End
found = Search(URL, local_cache)
If (found)
Begin
    Document = fetch (URL, local_cache)
    Display (Document);
    Flag = 1
End
found = search (URL, remote_summary)
If (found) replycode = send (proxy, queryURL)
If (replycode = "OK")
Begin
    Accept (Proxy, Document)
    Store (URL, Document, local_cache)
    Display (Document)
    Flag = 1
End
End
If (flag = 0)
Begin
    Replycode = forward (URL, webserver)
    If (Reply_code = "OK")
    Begin
        Accept (Document)
        Store (URL, Document, local_cache)
        Display (Document)
    End
End
Pseudo code of Intelligent Agent 3:

Bandwidth_usage = Monitor (web traffic)
// x1, x2, x3 are threshold values set by administrator
If (Bandwidth_usage < x1)
Begin
   Allow (alltraffic)
   Dynamic_Pre-fetch (SubsequentURL)
Else
   If (Bandwidth_usage < x2) DonotAllow (pre-fetching)
   Else
   Begin
      DonotAllow(pre-fetching, cache_refreshing)
      Display (link_images) // display only links of images
      If clickon(link_images) Fetch (images)
   End
End

Pseudo code of Intelligent Agent 4:

If (upload(file))
Begin
   file = Optimize(file)
   // limit – Maximum file size allowed to send without compression
   If (filesize (file) > limit) file = Compress(file)
   Else Send (file, client)
End

Pseudo code of Intelligent Agent 5:

Monitor (user, browser_settings, count)
If (count = 3 & same(browser_settings) store UserProfile(user,
   browser_settings)
When user login searchUserProfile(user, browser_settings)
If exixts load(user, browser_settings)
Else load default settings
6.2.2 Web Traffic/Latency Reduction at Web Server Side

The intelligent agents at the web server side identify the web browser capabilities for sending the documents in compressed form. Controlling the quality of image, audio and video files, compression, optimization of HTML codes and traffic dispersion technique are incorporated into the server side traffic/latency reduction algorithm. In adaptive traffic/latency reduction technique, intelligent agents at server side monitor the bandwidth usage. Depending upon the bandwidth usage, the quality of image, audio and video files are varied in order to avoid congestion. Intelligent agents at web servers identify the bloated HTML codes and optimize these codes to reduce the size of the file to be sent. Bursty traffic is divided into sub bursts and each sub bursts are transmitted through different routes in order to avoid congestion. The pseudo code for web traffic/latency reduction algorithm at web server side is given below.

For each URL
Begin
    // x1, x2, x3 are threshold values set by administrator
    // Web server contains image, audio and video files in varying qualities
    If (request = “image or audio or video files”)
    Begin
        Bandwidth_usage = Monitor (web traffic)
        If (Bandwidth_usage < x1) Send (good_quality(file))
        Else If (Bandwidth_usage < x2) Send (medium_quality(file))
        Else Send (low_quality(file), client)
    End
    Else If (request = “dynamic”)
    Begin
        (staticfile, dynamicfile) = Split (file)
Send (staticfile, client)
Send (dynamicfile, client)

// burst_limit – Maximum file size allowed to send as a single file
Else If (filesize (file) > burst_limit)
Begin
    Split(file, sub_burst)
    Send (sub_burst, Select(route), client) //Send each sub burst in
different route
End
Else If (upload(file))
Begin
    file = Optimize(file)
    // limit – Maximum file size allowed to send without compression
    If (filesize (file) > limit) file = Compress(file)
    Send (file, client)
Else
    Send (file, client)
End

Pseudo code of Intelligent Agent 1:
If (request = "image or audio or video files")
Begin
    Bandwidth_usage = Monitor (web traffic)
    If (Bandwidth_usage < x1) Send (good_quality(file))
    Else
        If (Bandwidth_usage < x2) Send (medium_quality(file))
        Else Send (low_quality(file), client)
    End

Pseudo code of Intelligent Agent 2:
If (request = "dynamic")
Begin
    (staticfile, dynamicfile) = Split (file)
Send (staticfile, client)
Send (dynamicfile, client)
End

**Pseudo code of Intelligent Agent 3:**

If (filesize (file) > burst_limit)
Begin
Split(file, sub_burst)
Send (sub_burst, Select(route), client) //Send each sub burst in different route
End

**Pseudo code of Intelligent Agent 4:**

If (upload(file))
Begin
file = Optimize(file)
// limit - Maximum file size allowed to send without compression
If (filesize (file) > limit) file = Compress(file)
Send (file, client)
End

### 6.3 Simulation and Discussion of Results

The adaptive traffic/latency reduction technique is simulated by incorporating client side adaptive traffic reduction algorithm into the dynamic pre-fetching browser (DPB) and server side traffic reduction algorithm in web servers. This new technique is simulated in an environment of medium sized network containing 90 nodes and trace driven data are collected. Java and C# languages are used for developing the adaptive traffic/latency reduction algorithms. In DPB, an intelligence menu is included for setting traffic threshold values, user's
willingness for pre-fetching, level of pre-fetching, quality of the files, cache size and size of pre-fetch area as shown in Figure 6.2. This new adaptive traffic/latency reduction technique makes DPB to monitor the current network traffic and user’s preferences and helps the browser to utilize the maximum bandwidth available effectively. The pseudo codes for the simulation setup of adaptive web traffic/latency reduction technique are given below.

**At Client Side**

Open (browser)
Start Brower Thread
Start Intelligent Agents
verified = Authenticate(username, password)
If (verified)
Begin
PreferenceList = search(UserProfile)
Found = search (PreferenceList, local_cache)
If !(found) found = search (PreferenceList, prefetch_area)
If !(found) Call Dynamic_Prefetch(PreferenceList)
End
While (request != “Exit”)
Begin
If (request = “URL”) search (URL, local_cache)
If !(found) found = search (URL, prefetch_area)
If !(found)
Begin
Document = Fetch (URL, webserver)
Display (Document)
Bandwidth_usage = Monitor (web traffic)
If (Bandwidth_usage < ThresholdLevel x1)
Begin
Call Dynamic_Prefetch (URL)
Call AutoCacheRefresh (URL)

End

Else

If (Bandwidth_usage < ThresholdLevel x2)

Begin

Call AutoCacheRefresh (URL)

DonotAllow Dynamic_Prefetch (URL)

Else

Begin

DonotAllow Dynamic_Prefetch

DonotAllow AutoCacheRefresh(URL)

End

Display (link_images)

End

End

If (request = “FreeCacheSpace”) UseLRU(SpaceNeed, clean_local_cache)

If (request = “Prefetch_URL”) Call Dynamic_Prefetch(PrefecthURL)

If (request = “search”)

Begin

Replycode = forward (URL, webserver)

If (Reply_code = “OK”) Display(sort(document, preference_keyword))

End

If (request = “dynamic” or request = “secure”) forward (URL, webserver)

Else

Begin

found = Search (URL, browser_cache)

If (found) Display (Document)

Else found = Search(URL, local_cache)

If (found) Display (Document);

Else Found = search (URL, remote_summary)

If (found) Display (Document)

Else Replycode = forward (URL, webserver)

If (Reply_code = “OK”) Display (Document)

End
End
If (upload(file))
Begin
    file = Optimize(file)
    If (filesize (file) > limit) file = Compress(file)
    Send (file, client)
End
End

At Server Side

Open (server)
Start WebServerThread
Start IntelligentAgents
For each URL
Begin
    If (request = "image or audio or video files")
    Begin
        Bandwidth_usage = Monitor (web traffic)
        If (Bandwidth_usage < x1) Send (good_quality(file))
        Else
            If (Bandwidth_usage < x2) Send (medium_quality(file))
            Else Send (low_quality(file))
    End
    Else If (request = "dynamic") Split (file)
    Else If (filesize (file) > burst_limit) Split(file, sub_burst)
End
Else If (upload(file))
Begin
    file = Optimize(file)
    If (filesize (file) > limit) file = Compress(file)
    Send (file, client)
End
Else Send (file, client)
Parameters used for simulation of adaptive web traffic/latency Reduction are given below.

Threshold Level $x_1$: 75% of maximum bandwidth capacity

Threshold Level $x_2$: 90% of maximum bandwidth capacity

Proxy Cache Size: 10 GB

Maximum allowed space for Hash table: 32 MB

Maximum Bandwidth capacity: 1 Mbps Internet connection

Computer Configuration Used: Pentium IV, 2.6 GHz machine running windows ME, 256 MB RAM and 40 GB hard disk.

Number of users required for testing, 50 to 90 users

Default PageSize: 1 MB

Multilevel pre-fetching: Yes, Level:3

Limit of file size: 0.5 MB

Burst_limit: 2 MB

Graphical user interface used for monitoring the bandwidth and changing the settings of pre-fetching are given in Figure 6.3. DPB is run in various threshold values and found that best threshold values for $x_1$ and $x_2$ are 75 and 90 respectively. An option has been provided to users for choosing whether good quality files are needed even if bandwidth usage is greater. It is observed that the average bandwidth usage using adaptive traffic/latency reduction technique is about 70 to 90% of the maximum bandwidth available of the current network as shown in Figure 6.4.
Figure 6.2 Dynamic Pre-fetching Browser

Figure 6.3 Bandwidth monitoring and settings of Pre-fetching
In figure 6.4, Bandwidth usage of DPB is compared with Internet Explorer (IE) version 6.0 and Netscape Navigator version 7.0. The bandwidth usage in DPB is almost constant. Other browsers attempts to fetch more objects than the network can afford without considering the current network traffic. But in DPB, intelligent agents make the browser to monitor the current network traffic and user's preferences and help the browser to utilize the maximum bandwidth available effectively. There is no bandwidth monitoring features available in Internet Explorer and Netscape Navigator. If the network bandwidth is almost idle, other IE and netscape fetches only requested URL's, but DPB pre-fetches subsequent links until bandwidth usage exceeds a pre-defined threshold. So DPB utilizes the existing maximum bandwidth capacity efficiently. Thus DPB increases pre-fetch hit ratio at light traffic and reduces pre-fetch hit ratio at heavy traffic. There is no controlling of pre-fetch hit ratio in IE and netscape. Comparison of DPB with Internet explorer and netscape navigator are given in Table 6.1.
Table 6.1 Comparison of Dynamic Pre-fetching Browser, Internet Explorer and Netscape Navigator

<table>
<thead>
<tr>
<th>Category</th>
<th>Dynamic Pre-fetching Browser</th>
<th>Internet Explorer</th>
<th>Netscape Navigator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity and ease of use</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Disk Space</td>
<td>Only less space is needed to store DPB and its required modules, but some of the features available in other browsers are not implemented</td>
<td>Compared to DPB, it takes more space, but less than Netscape Navigator</td>
<td>Takes more disk space to store browser</td>
</tr>
<tr>
<td>Main Memory Space</td>
<td>Only less space is needed to store DPB in memory, but some of the features available in other browsers are not implemented</td>
<td>Compared to DPB, it takes more space, but less than Netscape Navigator</td>
<td>Netscape sucks up more system resources</td>
</tr>
<tr>
<td>Speed</td>
<td>Faster</td>
<td>Comparatively slower</td>
<td>Faster</td>
</tr>
<tr>
<td>Traffic Monitoring and congestion avoidance Facility</td>
<td>Available</td>
<td>Not available</td>
<td>Not available</td>
</tr>
</tbody>
</table>

The simulation results show that the incorporation of the traffic/latency reduction algorithms in the browser and web server side, increases cache hit ratio to 40 – 82% and reduces latency to 20 – 65% as shown in Figures 6.5 and 6.6. Congestion will not occur using dynamic pre-fetching technique or using adaptive web traffic/latency reduction technique. If the bandwidth usage is greater than 90%, pre-fetching and automatic cache refreshing are not allowed, only links of the images will be displayed. Even if image, audio and video files are sent through the network, only they send in low quality. If bandwidth usage is greater than 97%, prompt the user that heavy bandwidth usage and make them to wait.
Adaptive web traffic/latency reduction technique guarantees that bandwidth usage of the existing network will be below the maximum capacity of the bandwidth available. This technique, checks only the incoming and outgoing links bandwidth capacity and its usage. It could not check the intermediate link's bandwidth. Congestion can occur in intermediate links.

![Graph showing Cache Hit Ratio](image)

**Figure 6.5 Analysis of Cache Hit Ratio**

![Graph showing Latency Reduction](image)

**Figure 6.6 Analysis of Latency Reduction**
6.4 Concluding Remarks

Since adaptive web traffic/latency reduction is a hybrid technique, it enhances the web performance. Web caching, dynamic pre-fetching, bandwidth controlling, optimization of HTML codes, compression and traffic dispersion techniques are incorporated into the adaptive traffic/latency reduction algorithms. Since dynamic pre-fetching technique is incorporated into the client side adaptive traffic/latency reduction algorithm, it is capable of reducing web traffic and maintaining the bandwidth usage almost constant. The intelligent agents monitor the bandwidth usage and adjust pre-fetching to make the maximum utilization of the existing capacity of the network. Using this adaptive traffic/latency reduction technique, cache hit ratio is increased up to 82% and latency is reduced up to 65%.