Dissertation for the Degree of Doctor

A Study on Streaming Media Service for Mobile Clients

by

Dongmahn Seo

Department of Computer and Communications Engineering

Graduate School

Kangwon National University

February, 2010

Under the Guidance of
Professor Inbum Jung

A Study on Streaming Media Service for Mobile Clients

A DISSERTATION
Submitted to the Graduate School of
Kangwon National University in Partial
Fulfillment of the Requirements
for the Degree of

Doctor of Engineering

by
Dongmahn Seo

Department of Computer Engineering

February, 2010

Approved by Committee of the Graduated School of Kangwon National University in Partial Fulfillment of the Requirements for the Degree of Doctor of Engineering

Dongmahn Seo

December, 2009

Dissertation Committee ;

Inbum Jung (signature)
(Chairman of Committee)

ChangGeun Song (signature)
(Committeeman)

YongSeok Kim (signature)
(Committeeman)

HeonGuil Lee (signature)
(Committeeman)

HarkSoo Kim (signature)
(Committeeman)

A Study on Streaming Media Service for Mobile Clients

Dongmahn Seo

Department of Computer Engineering

Graduate School, Kangwon National University

Abstract

Based on recently the amazing growth of telecommunication, computer and image compression technologies, the streaming media service has been spotlighted in many multimedia applications. In particular, the advancement in wireless network technologies has enabled the streaming media service on the mobile devices such as PDAs, laptops, navigation, wireless IP-TV, and mobile phones. The streaming media have larger and more complex data than the traditional text, picture image data. Furthermore, since the wireless network has low bandwidth channels and many mobile devices compose of limited hardware specifications, the streaming media service for mobile clients is needed to study.

In this dissertation, a integrated transcoding media streaming service system is proposed in order to provide streaming media service for mobile clients. The proposed system needs to provide a unstable wireless network-adaptive transcoding method and a load distribution method and a admission control method for various clients on server side in order to guarantee QoS to mobile clients using streaming media service. Furthermore, it is necessary to provide a QoS-guarantee method with low bandwidth in cell boundary area and frequent hand-over of fast moving clients.

In this dissertation, three methods for streaming media service for mobile clients are proposed to solve these problems. At first, the Network Adaptive Autonomic Transcoding Algorithm (NAATA) is proposed to support streaming media service for mobile clients. The proposed algorithm decides on target transcoding bit-rates in real-time according to the wireless network state. Since it protects continuous transmission failures for streaming media data, seamless and stable streaming media services are provided for mobile clients.

Secondly, a new load distribution method is proposed for fair transcoding load distribution in the distributed transcoding servers. The proposed method controls and distributes transcoding requests based on transcoding time estimation for distributed transcoding environments. In our experiments, the proposed method shows better scalable performance than other load distribution methods, because the characteristics of transcoding servers, transcoding requirements and streaming media data are considered.

Finally, a client mobility-based media stream prefetching method to guarantee stable QoS in high speed internet environment like a Mobile WiMAX, is proposed. In proposed method, high speed moving clients is joint a group depending on their characteristics and provided streaming media service according to a joint group. Mobile clients can change their group depending on their situation. In each group,

clients' direction is predicted, and media streams are prefetched against disconnect and latency caused by handover. The proposed method is experimented and evaluated that buffer state is stable with handover for streaming media service.

Acknowledgement

※ This study is partially supported by “2005 Korea Sanhak Foundation Scholarship” provided by the Korea Sanhak Foundation.

※ This study is partially supported by “2007 Engineering Graduate School Research Scholarship” provided by the Korea Science and Engineering Foundation.

CONTENTS

List of Tables

List of Figures

Chapter I

Introduction

Based on recently the amazing growth of telecommunication, computer and image compression technologies, the streaming media service has been spotlighted in many multimedia applications. In particular, the advancement in wireless network technologies has enabled the streaming media service on the mobile devices such as PDAs, laptops, navigation, wireless IP-TV, and mobile phones. Figure 1-1 shows various streaming media services and transcoding systems for mobile clients. As shown as the figure, various kinds of clients are used for streaming media service.

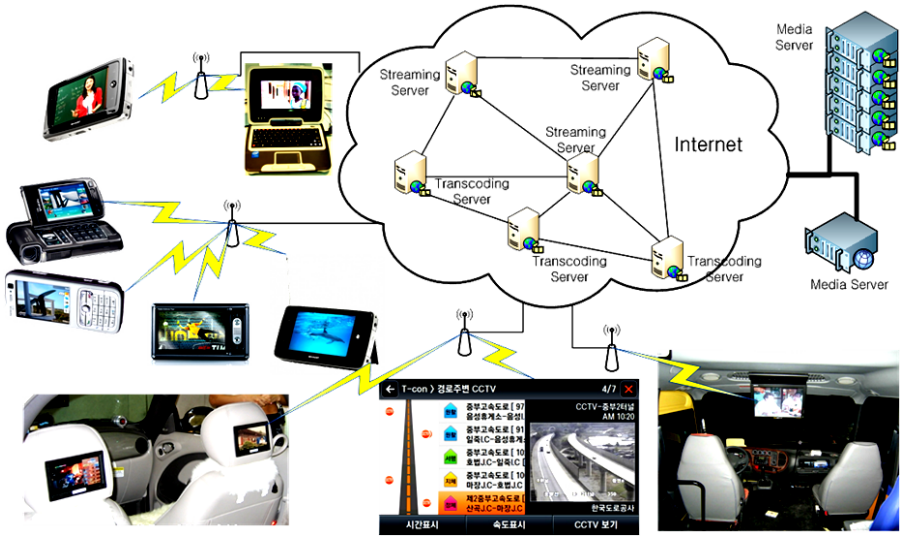

Figure 1-1. Streaming media services for mobile clients.

The streaming media have larger and more complex data than the traditional text, picture image data. Thus, the large amount of network traffics and the high performance computing ability are inevitable to support the QoS streams [1-3]. However, since the wireless network has low bandwidth channels and many mobile devices compose of limited hardware specifications, the transcoding technology is needed to adapt the originally encoded MPEG media to the given mobile devices. The ranges of adaptation include changing the frame rates, bit rates, video sizes and the re-encoding MPEG I, II media into the MPEG IV.

The transcoding system is usually composed of both the multimedia server with the originally encoded MPEG media and the transcoding servers to perform the adapting to the given environment. The multimedia server retrieves the MPEG media and sends them to the selected transcoding server. The transcoding server performs the transcoding to original MPEG video and also sustains the streaming service to the corresponding client. In particular, to provide QoS for clients, it is inevitable to guarantee streaming media without ceasing and jittering phenomena [3-5].

As the transcoding systems, there have been several approaches such as the source based static encoding system, the static transcoding server system, and load distribution transcoding system [6-8]. In the source based static encoding system, the server stores the MPEG videos encoded by all client grades. Due to the absence of on-line overheads for transcoding, this approach takes an advantage on the side of streaming service. However, it is difficult to prepare encoded videos that are adapted for all kinds of mobile clients. And also, it has the burden of storing all encoded grades to the same title MPEG movies.

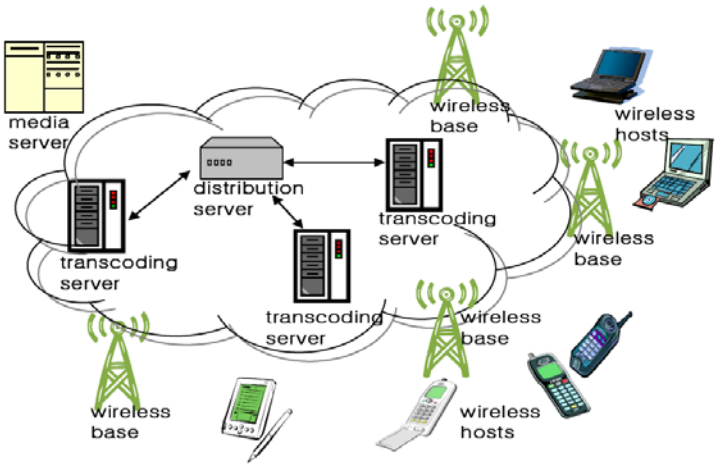

Figure 1-2. Load distribution transcoding system.

The static transcoding server system chooses the transcoding server close to the wireless base of clients. In this approach, the specific servers suffer from the heavy congested transcoding jobs. To address this problem, the load distribution transcoding system uses the load distribution server. The load distribution server monitors the loads of transcoding servers. The arrived transcoding jobs are distributed to transcoding servers based on the load distribution strategies. As shown as Figure 1-2, the transcoding requests from mobile clients are sent to the selected transcoding server by the load distribution server.

MPEG media are usually used for streaming media service due to their highest compression and decompression ratio. On the basis of the characteristics of mobile clients, the MPEG media streams are classified as several grades. To provide the streaming service for the various mobile devices, the transcoding server should adapt original MPEG media to the corresponding client grades. The transcoding jobs cause the servers to exhaust the high amount of their resources. In particular, depending on the transcoding grades, the different amounts of CPU, memory and

network bandwidth are consumed in transcoding servers. If the transcoding grades incurring the high resource consumption are concentrated on specific transcoding servers, the load imbalance among transcoding servers may occur. In this case, it is hard to expect the efficient resource utilization of transcoding servers. To address this problem, the load weight by transcoding grades should be reflected on the load distribution strategy.

The streaming media have its intrinsic characteristic such as the real time specification. Within the limited time, if the streaming media cannot be read, transcoded, sent and decoded in the client terminal, the ceasing and jittering streaming is displayed on the screen. Thus, the real time requirement for the QoS means that the transcoding operation should be completed within the limited time and also the transcoded media can be streamed to clients without ceasing and jittering events. In addition, it is essential that new transcoding requests have not negative impact on the QoS of all clients being serviced.

In this dissertation, three issues in streaming media service for mobile clients is studied for a integrated transcoding media streaming service system as shown as Figure 1-3. First issue is how to guarantee quality of service (QoS) over wireless network. Second issue is hot to guarantee QoS for various clients. Last issue is how to guarantee QoS for fast moving clients.

For first issue, the Network Adaptive Autonomic Transcoding Algorithm (NAATA) is proposed to support streaming media service for mobile clients. The proposed algorithm decides on target transcoding bit-rates in real-time according to the wireless network state. Since it protects continuous transmission failures for streaming media data, seamless and stable streaming media services are provided for mobile clients.

Figure 1-3. Three issues in streaming media service for mobile clients.

For second issue, a new load distribution method is proposed for fair transcoding load distribution in the distributed transcoding servers. The proposed method controls and distributes transcoding requests based on transcoding time estimation for distributed transcoding environments. In our experiments, the proposed method shows better scalable performance than other load distribution methods, because the characteristics of transcoding servers, transcoding requirements and streaming media data are considered.

For third issue, a client mobility-based media stream prefetching method to guarantee stable QoS in high speed internet environment like a Mobile WiMAX, is proposed. In proposed method, high speed moving clients is joint a group depending on their characteristics and provided streaming media service according to a joint group. Mobile clients can change their group depending on their situation. In each group, clients' direction is predicted, and media streams are

prefetched against disconnect and latency caused by handover.

The rest of this dissertation is organized as follows. In Chapter 2, the Network Adaptive Autonomic Transcoding Algorithm (NAATA) is proposed to support streaming media service for mobile clients. We evaluate that the NAATA provides a more seamless streaming media service for mobile clients, without jittering or ceasing phenomena. Chapter 3 propose a new load distribution method is proposed for fair transcoding load distribution in the distributed transcoding servers. We show that the proposed method provides more linear performance scalability than other load distribution. In Chapter 4, we propose a client mobility-based media stream prefetching method to guarantee stable QoS in high speed internet environment. Finally, conclusions and future works are presented in Chapter 5.

Chapter II

Network-adaptive Autonomic Transcoding Algorithm for Seamless Streaming Media Service of Mobile Clients

1. Motivation

Based on the recent significant growth of telecommunication, computer, and image compression technologies, the streaming media service has been spotlighted in many multimedia applications. In particular, the advances in wireless network technologies have enabled streaming media service on mobile devices such as PDAs and cellular phones. Streaming media need larger and more complex data than the traditional text and image data. Thus, a large network traffic bandwidth and high performance computing ability are inevitably required to support the Quality of Service (QoS) streams [1-3,9,10]. However, since wireless networks have low and unstable bandwidth channels compared to wired networks, and many mobile devices have limited CPU performance, transcoding technology is needed to adapt the originally encoded media to the given mobile devices. The range of adaptations includes changing the frame rates, bit rates, video sizes and compress format such as re-encoding MPEG Ⅰ, Ⅱ media into MPEG Ⅳ [9,11,12]. Moreover, transcoding of MPEG Ⅳ encoded media data is necessary to provide QoS guarantee streaming server for mobile clients.

The transcoding system is usually composed of both the multimedia server with

the originally encoded media and the transcoding servers to perform the adaptation to the given environment. The multimedia server retrieves the media and sends them to the selected transcoding server. The transcoding server performs the transcoding original media and also sustains the streaming service to the corresponding client. In particular, a critical requirement for providing QoS for clients is to guarantee streaming media quality consistently and without jittering phenomena.

However, mobile clients work in the wireless network environment, which is unstable and has low bandwidth compared to wired network. Since the distance between mobile clients and an Access Point (AP) fluctuates according to the movement of mobile clients, the available network bandwidth is not kept stable. Therefore, it is hard to guarantee a stable QoS level and the continuity of media data in the streaming service for mobile clients.

In this chapter, the Network Adaptive Autonomic Transcoding Algorithm (NAATA) is proposed to support streaming media service for mobile clients. The proposed algorithm decides on target transcoding bit-rates in real-time according to the wireless network state. Since it protects continuous transmission failures for streaming media data, seamless and stable streaming media services are provided for mobile clients.

2. Related Work

2.1. Transcoding System

There have been several approaches for transcoding systems, including source based static encoding system and static transcoding server systems [6,9]. In the

source based static encoding system, the server stores the MPEG videos encoded by all client grades. Due to the absence of on-line overhead for transcoding, this approach has an advantage on the streaming service side. However, it is difficult to prepare encoded videos that are adapted for all kinds of mobile clients. Also, the method has the disadvantage of storing all encoded client grades to the same MPEG movies title.

The static transcoding server system chooses the transcoding server closest to the wireless base of the mobile client. This approach uses the initial state of the wireless network at the point of client arrival. However, since wireless networks have variable bandwidths, it is difficult to guarantee the QoS streams in real-time with this method.

2.2. Network-adaptive QoS Guarantee Methods

The internet does its best to transmit packets among hosts, but it does not provide any guarantee. In attempting to solve this problem, several methods for streaming media services that guarantee QoS have been studied. An adaptation method for server transmission bit-rate based on packet loss information in Real-time Transport Protocol Control Protocol (RTCP) and using classification of clients is proposed. This protocol is applied in the streaming media service between a server and the clients. Especially in the Video-On-Demand (VOD) system with a Variable Bit Rate (VBR) environment, it is difficult to guarantee the adaptive bit-rate for QoS, because a special frame may take much larger bits than other frames. To address this issue, the smooth bit-rate method was proposed in order to keep stable transmission rates [13,14]. Also previously proposed were the edge

server strategy, packet transmission interval and datagram size control strategy [14,15]. These methods, however, guarantee the QoS of streaming service with only the special environment or only with some of the streaming data. However, since they are not able to reflect the network state in real-time, not only can real-time multimedia data processing not be supported, but also, it is not possible to provide a streaming service in fixed low bandwidths.

In order to enable streaming solutions that can adapt to the network state and/or to the receiver capabilities, systems often rely on network-adaptive media coding algorithms, or adaptive decoding strategies [16]. These algorithms encode and packetize the media information under a form that facilitates adaptation to the network characteristics, expressed in terms of bandwidth variation or packet loss. Such techniques include for example scalable encoding, efficient bitstream packetization, error-resilient encoding, and dynamic changes of compressed data units' dependencies. However scalable encoding approach is quit greedy in terms of computational complexity, which makes its application quite limited in practice. And adaptive error protection is difficult to design in scenarios where the loss behavior is hard to predict, or where the access bandwidth is quite heterogeneous among clients [16].

Application-layer QoS control techniques are used to deal with dynamically varying network conditions that can lead to significant data rate variations or unexpected packet losses [16]. Automatic Repeat reQuest (ARQ) systems use combinations of timeouts and positive and negative acknowledgments to determine which packets should be retransmitted. Forward error correction (FEC) means that redundancy is added to the data so that the receiver can recover from losses or errors without any further intervention from the sender. However ARQ may not be

appropriate for applications with very tight delay constraints, or in broadcast scenarios due to the bandwidth explosion phenomenon that arises when the states of the receivers are not synchronized [16].

Optimized packet scheduling at the application layer takes into account data units' dependencies and importance for the reconstruction of the media stream at the receiver when performing transmission decisions for media packets [16,17]. Congestion control further helps in preventing packet loss and reducing delays by carefully limiting the bandwidth available to the sender. TCP-friendly rate control (TFRC) provides a lower variation of throughput over time relative to TCP, while simultaneously allowing for fair sharing of the available bandwidth with competing TCP flows [16,18,19]. The Datagram Congestion Control Protocol (DCCP) is UDP to support congestion control [20]. However these methods consider only how to control network packets. They do not reduce the amount of media data for steaming service to clients. Since streaming media service has a constraint like a soft real time service, media data should be arrived at client on time. Therefore media bit-rate changing method is needed when networks are congested, because the amount of media data for 1 second is media bit-rate.

2.3. Available Network Bandwidth-adaptive Transcoding

On the wireless network, the network bandwidth between a mobile client and an AP fluctuates according to the movement of a mobile client. In particular, the bandwidth of wireless network decreases sharply when jamming or other network problem occur, or when a mobile client is far from an AP. Legacy transcoding methods have drawbacks insofar as they do not consider wireless network

bandwidth fluctuations. In order to solve this problem, available network bandwidth-based transcoding systems were proposed [6,15,21].

2.3.1. Estimation of available network bandwidth

Initial Gap Increasing (IGI) and Packet Transmission Rate (PTR) [15,21] are algorithms for estimating the end-to-end available network bandwidth. Table 2-1 shows the pseudo-code for the IGI algorithm. The IGI algorithm sends packets to a receiver to estimate the available network bandwidth. A sender increases the number of packets continuously until the amount of received packets on the receiver side is NOT same as that of the sent packets on sender side. When both sides have the different value, then the turning point has been achieved: the amount of sent packets just before the turning point is the current available network bandwidth.

Table 2-1. Pseudo-code for the IGI Algorithm

```
Algorithm IGI
{
   /* initialization */
   probe_num = PROBENUM;
   packet_size = PACKETSIZE;
   gB = GET_GB();
   init_gap = gB / 2;
   gap_step = gB / 8;
   src_gap_sum = probe_num * init_gap;
   dst_gap_sum = 0;
   /* look for probing gap value at the turning point */
   While(!GAP_EQUAL(dst_gap_sum, src_gap_sum)) {
      init_gap += gap_step;
      src_gap_sum = probe_num * init_gap;
      SEND_PROBING_PACKETS(probe_num, packet_size, init_gap);
      dst_gap_sum = GET_DST_GAPS();
   }
   /* compute the available bandwidth using IGI fomula */
   inc_gap_sum = GET_INCREASED_GAPS();
   c_bw = b_bw * inc_gap_sum / dst_gap_sum;
   a_bw = b_bw - c_bw;
}
```

2.3.2. Available network bandwidth-adaptive transcoding algorithm

The network bandwidth-adaptive transcoding technology is based on the IGI algorithm. As mentioned in the above subsection, this algorithm estimates the available network bandwidth. Table 2-2 shows the Available Network Adaptive Transcoding (ANAT) algorithm, which controls the transcoding bit-rate according to the available network bandwidth that is estimated by the IGI algorithm. Since the ANAT algorithm decides the target transcoding bit-rate by comparing the transcoding bit-rate with the available network bandwidth, it controls the bit-rate of streaming media in real-time.

Table 2-2. Pseudo-code for the ANAT Algorithm

```
Algorithm ANAT
{
  if (diff_sbyte - diff_rbyte > BIT_DIFF) {
    bit_diff_cnt++;
  } else {
    bit_same_cnt++;
  }
  if(bit_diff_cnt >= BIT_DIFF_COUNT) {
    if(initial state) {
      bit-rate reduction;
    } else if (max_bit_rate == min_bit_rate) {
      bit-rate reduction;
    } else {
      bit-rate reduction;
    }
  } else if (bit_same_cnt >= BIT_SAME_COUNT) {
    tmp = bit-rate to change;
    if(tmp < max_bit_rate && tmp < init_bit_rate) {
      bit-rate modification;
    } else if(tmp >= inin_bit_rate) {
      set bit-rate as init_bit_rate;
    } else if(tmp >= max_bit_rate) {
      set bit-rate as max_bit_rate;
      tmpcnt++;
      if(tmpcnt > threshold) {
        max_bit_rate = init_bit_rate;
        tmpcnt = 0;
      }
    }
  }
}
```

3. The Network Adaptive Autonomic Transcoding Algorithm (NAATA)

Based on the estimation of available network bandwidth in real-time, the ANAT algorithm provides a seamless streaming media service for mobile clients with changing transcoding bit-rates. However the wireless network works on a variable and low network bandwidth. Thus, the ANAT algorithm causes much overhead in order to estimate the available network bandwidth. To address this problem, we propose the NAATA to provide seamless, low overhead streaming media service to mobile clients.

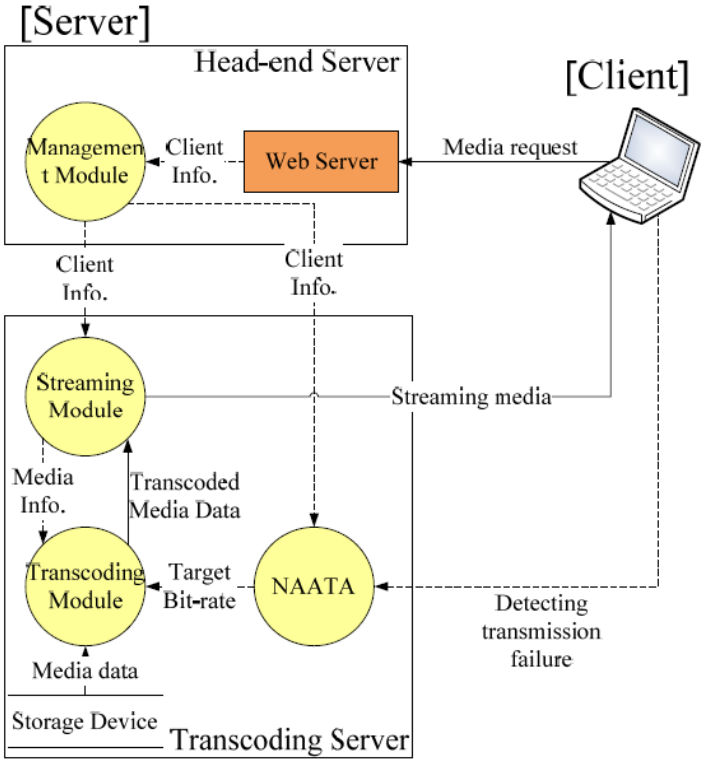

Figure 2-1. The NAATA concept using the AIMD.

As shown in Figure 2-1, the Experimental System for the NAATA is composed of the head-end server and the transcoding server. The head-end server receives user requests and controls the transcoding server. The transcoding server transcodes media data and provides streaming media services.

The transcoding server is composed a streaming module, a transcoding module and a NAATA module. The NAATA module checks transmission failures and decides on a transcoding target bit-rate for a corresponding mobile client. The transcoding target bit-rate is sent to the transcoding module. The transcoding module reads media data from storage devices and transcodes media data according to the transcoding target bit-rate. After that, it sends transcoded media data to the streaming module. The streaming module transmits the received transcoded media data to a client.

The NAATA uses the Additive Increase, Multiplicative Decrease (AIMD), which avoids continuous transmission failures in Transmission Control Protocol (TCP). This algorithm controls the transcoding bit-rate autonomically when a transmission failure occurs between a server and a client. The NAATA compares the number of transmitted packets on a server with the number of received packets on a client. If the number of received packets is below the threshold value, the NAATA changes the transcoding bit-rate. Therefore, although the wireless network bandwidth is varied, the NAATA provides a seamless streaming media service.

3.1. The AIMD congestion control algorithm in TCP

The AIMD is a part of the congestion control algorithm in TCP [22,23]. When packet losses occur, a transmitter reduces the send rate exponentially. After that,

the transmitter increases the send rate linearly. Since other TCP connections in the same congested router also suffer from the packet loss, these connections decrease their transmission rate by reducing the size of congestion window. As a result, the load of the congested router can be decreased by senders.

3.2. Characteristics of the NAATA

3.2.1. Operating behavior

To reflect media data characteristics, the NAATA is implemented with the modified AIMD. The NAATA checks the amount of transmitted/received data in each server/client and determines transmission failure. It checks the amount of transmission media data periodically: if the difference between the data received by the client and the data transmitted by the server is bigger than the threshold, the NAATA determines that a transmission problem has occurred on the network. The threshold value can be decided on operating time. When the threshold value is getting bigger, a possibility of transmission failure detection is getting lower. If the threshold value is too big, transmission failures are not detected. If 0, every transmission failures are detected. In this chapter, zero is used for the threshold value.

If a transmission failure is detected, the NAATA reduces the media data to transmit as half of the current streaming bit-rate. It uses the fast recovery method of the AIMD. If a transmission failure is not detected, the NAATA recovers the bit-rate linearly by using the slow start method. Until the streaming media bit-rate reaches the initial bit-rate, the streaming service is sustained at a level between the maximum service available bit-rate and the minimum service available bit-rate.

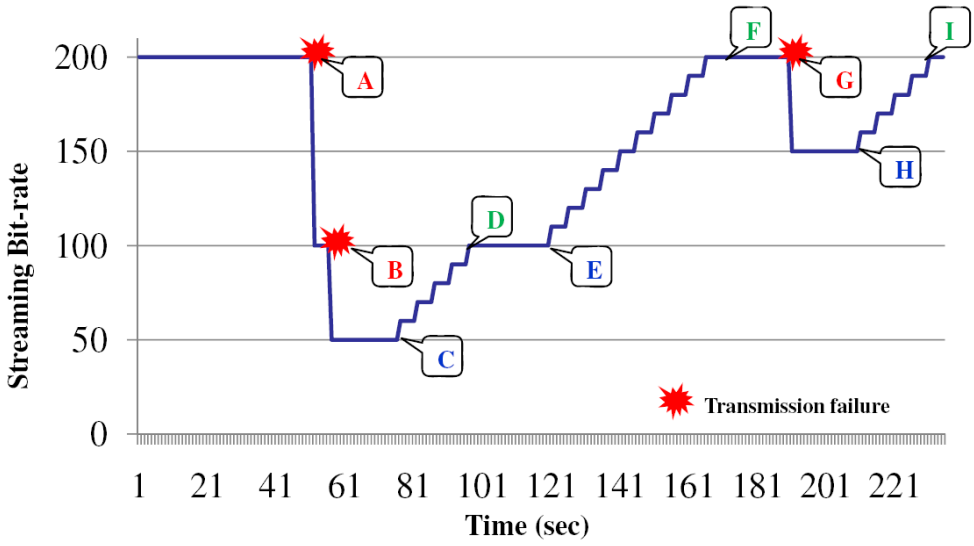

Figure 2-2. Activities in the NAATA.

Figure 2-2 shows an applied example of the NAATA. The x axis indicates the timeline of user requests; the y axis shows streaming bit-rates. In this example, a user request bit-rate is 200 Kbps. Thus, the initial bit-rate of the NAATA is also 200 Kbps. The maximum and minimum bit-rates are upper and lower bounds of the target transcoding bit-rate in the current stage as decided by the current state of the network. Point A of Figure 2-2 is the point at which the first transmission failure is found. After the failure, the streaming bit-rate is set to 100 Kbps. It is half of the maximum bit-rate of 200 Kbps. After the streaming bit-rate is changed, the recovery of the bit-rate may not be accepted immediately. From this point, the maximum bit-rate and the minimum bit-rate are both set to 100 Kbps.

Point B of Figure 2-2 shows what happens in the case of a continuous transmission failure. In this case, the streaming bit-rate is changed to 50 Kbps. It is half of the minimum bit-rate of 100 Kbps. Subsequently, the minimum bit-rate is

set with the streaming media bit-rate at 50 Kbps.

Point C in Figure 2-2 represents the point at which no transmission failure occurs after the reduction of service bit-rate, for a period of time longer than the pre-defined threshold time. At point C, the streaming bit-rate is increased. If the bit-rate reaches 100 Kbps (Point D), the previous transmission failure position, it is necessary to watch and wait to see whether another transmission failure will occur at the previous transmission failure bandwidth. For the reason, the streaming media bit-rate is held and kept, the minimum bit-rate is set with the maximum bit-rate at 100 Kbps and the maximum bit-rate is set with the initial bit-rate at 200 Kbps.

Point E one can see that, after the service bit-rate was held, no transmission failure occurred that lasted for a longer period of time than the pre-de¯ned threshold time. At Point E, the streaming bit-rate is increased. When the bit-rate reaches the initial bit-rate of 200 Kbps (Point F), it is fixed as the initial bit-rate of 200 Kbps.

The failure of Point G differs from that of Point A and B. In the case of G, since the maximum bit-rate is the current streaming bit-rate of 200 Kbps, and since the minimum bit-rate is 100 Kbps, the streaming bit-rate is changed to 150 Kbps. This is a median value between the maximum bit-rate and the minimum bit-rate. Point H is similar to the case of Point C and Point I is the same as Point F.

3.2.2. Algorithm

Table 2-3 shows the pseudo-code for the NAATA. Part A of Table 2-3 shows the variables that are used in the algorithm. The ***target_tr_bit*** is the current bit-rate of streaming media service. The ***target_tr_bit*** is set with the value of other

variables. The ***init_bit*** denotes the initial requested bit-rate by the user, and the ***max_bit*** and the ***min_bit*** represent the maximum bit-rate and the minimum bit-rate of the current streaming service, respectively. Therefore the ***target_tr_bit*** is between the ***max_bit*** and the ***min_bit*** like upper bound and lower bound. The ***tmp*** is used to check whether a change of the ***target_tr_bit*** is possible. The flag is in order to record whether the continuous transmission failures happen or not. The ***threshold_time*** indicates the threshold time to recover target bit-rate and the ***threshold_fail*** represents threshold to detect transmission failure. The count indicates the elapsed time after changing the ***target_tr_bit*** and it preserves the threshold time. The NAATA has two stages: the first is a "transmission failure stage" and the second is a "no failure stage".

Part B of Table 2-3 shows three kinds of methods for addressing the transmission failures. The first method is when the first transmission failure occurs; the second method is for when the continuous transmission failure occurs and before the recovery of bit-rate proceeds. The third method treats the other failures. Part B-1 of Table 2-3 shows the case that the first transmission failure occurs. In case of B-1, the ***target_tr_bit*** is set with a half of the ***init_bit***, and then the ***max_bit*** and the ***min_bit*** are changed by ***target_tr_bit.*** Part B-2 of Table 2-3 shows the case of that the continuous transmission failures occur. In case of B-2, since the ***min_bit*** is equal to the ***max_bit***, the ***target_tr_bit*** is changed with half of ***max_bit***. The other cases are shown in the part B-3 of Table 2-3. Since the ***min_bit*** and the ***max_bit*** are different, the ***target_tr_bit*** is changed with the medium value between the ***min_bit*** and the ***max_bit***.

Table 2-3. Pseudo-code for the NAATA

```
NAATA(int sentByte, int RcvByte) {
  static int init = 1;   // initial state flag
  static int init_bit;   // initial bit-rate (user requested bit-rate)
  static int min_bit;    // minimum bit-rate
  static int max_bit;    // maximum bit-rate
  static int tmp;        // temporal target bit-rate
  static int target_tr_bit; // target bit-rate
  static int count;      // time count
  static int flag = 0;   // flag for continous problem
  static int threshold_time = 10; // threshold time to recover target bit-rate
  int threshold_fail = 0;    // threshold to detect transmission failure
  int diff = sentByte - RcvByte;
  if(diff > threshold_fail) {      // when transmission failure occurs
    if(init) {                     // when first transmission failure occurs
      target_tr_bit = init_bit / 2;
      min_bit = max_bit = target_tr_bit;
      init = 0;
    } else if(flag == 1) {  // when continuous transmission failure occurs
      target_tr_bit = max_bit / 2;
      min_bit = 0;
      threshold_time += 10;
    } else {                // when the other transmission failure occurs
      target_tr_bit = (max_bit+min_bit)/2;
      min_bit = target_tr_bit;
      threshold_time += 10;
    }
    flag = 1;    // transmission failure occurred
    count = 0;
  } else {       // when no transmission failure occurs
    flag = 0;    // no transmission failure
    tmp = target_tr_bit + increased bit-rate;  // calculate temporal target bit-rate
    if(tmp < max_bit && tmp < init_bit){
                   // when tmp is less than maximum bit-rate and initial bit-rate
      target_tr_bit = max_bit = tmp;
    } else if(tmp >= init_bit) {  // when tmp is same as initial bit-rate
      target_tr_bit = init_bit;
      count++;
      if(count>=threshold_time) { // when count is greater than threshold
        min_bit = 0;
        threshold_time -= 10;
        count = 0;
      }
    } else if(tmp >= max_bit) {    // when tmp is greater than bandwidth
      target_tr_bit = max_bit;
      count++;
      if(count>=threshold_time) { // when count is greater than threshold
        min_bit = max_bit;
        max_bit = init_bit;
        count = 0;
        threshold_time -= 10;
      }
    }
  }

  return target_tr_bit;
}
```

A
B
1
2
3
C
1
2
3

Part C of Table 2-3, three kinds of methods are shown for recovering the dropped bit-rate after failures are detected. The ***tmp*** is the sum of the ***target_tr_bit***

and the increased bit-rate. Part C-1 shows that the ***tmp*** is smaller than the ***max_bit*** and the ***init_bit***. In this case, the ***target_tr_bit*** and the ***max_bit*** are changed with ***tmp*** for recovery service bit-rate. Part C-2 shows that the ***tmp*** is bigger than the ***init_bit***. In this case, the ***target_tr_bit*** is fixed as the ***init_bit***, because the ***init_bit*** is the user requested bit-rate and users do not need a higher service bit-rate than the user requested bit-rate. Furthermore, if the service network is continuously stable, the count exceeds the pre-defined threshold time. In that case, as the ***min_bit*** is changing to 0, the streaming service is recovered to the initial state.

Part C-3 shows that the ***tmp*** is bigger than the ***max_bit.*** In this case, the streaming service suffered from the transmission failures during the previous time-line. The ***target_tr_bit*** is fixed as the ***max_bit***, and the traffic state for streaming service will be checked again for a limited period. If the service network is continually stable, the count exceeds the pre-defined threshold time. In that case, as the ***min_bit*** is changing to the ***max_bit*** and the ***max_bit*** is changing to the ***init_bit***, the streaming service jumps to the previous bit-rate level suffering a failure.

4. Performance Evaluation

This section shows the performance of the NAATA. We compare the NAATA with the ANAT algorithm and legacy transcoding method. In order to evaluate the performance of the NAATA, three experimental metrics are identified: 1) the number of transmission failures; 2) the average time interval among transmission failures; and, 3) the overhead.

In order to evaluate the performance of the NAATA, the ANAT algorithm is

also implemented based on the IGI algorithm. In the ANAT method, an estimation module, based on the IGI algorithm, estimates the available network bandwidth [15, 21]. This module also sets the transcoding target bit-rate according to the available network bandwidth, and sends the bit rate to a transcoding module. A client also has an IGI client module to cooperate with the ANAT estimation module.

Every module in each server is implemented using C language. The ffmpeg and the ffserver are modified and applied to the transcoding module and the streaming module, respectively [24]. The mplayer is modified and applied to the client programs [25].

Table 2-4 shows the hardware specification for the server and the client. Table 2-5 shows information about the media used in the experiment. Real network environments with IEEE 802.11 b and g are used for experiments.

Table 2-4. Server and Client Hardware Specification

	Server	Mobile Client 1	Mobile Client 2
CPU	AMD Athlon MP 2200+ 1.8GHz	Intel Pentium 4 Mobile 1.8GHz	Intel Xscale PXA270 416MHz
Memory	1 GB	768MS	16MB flash, 64MB SDRAM
Network	100Mbps fast ethernet	IEEE 802.11 b/g	IEEE 802.11b
Linux Kernel	2.6.9-71	2.6.9-34	2.4.24

Table 2-5. Experiment Media Information

	Bit-rate	Frame-rate	Resolution
Movie 1	871.7 Kbps	23.97 fps	576 × 256
Movie 2	760.6 Kbps	23.97 fps	640 × 304

4.1. Performance of the NAATA

Figure 2-3 shows the streaming bit-rates of movie 1 as served by the ANATS and Figure 2-4 and 2-5 show the streaming bit-rates of movies 1 and 2 as served by the NAATA. The streaming media bit-rate in Figure 2-4 and 2-5 are changed in real-time according to the network states as these fluctuate by client movement or because of variable mobile environments. As shown in Figure 2-3, 2-4 and 2-5, both initial bit-rates are 200 Kbps. When the first transmission failure occurs, the streaming bit-rate is changed to 100 Kbps in Figure 2-4 and 2-5. However the streaming media bit-rate in Figure 2-3 is very unstable and lower than in the NAATA.

The middle parts of the two figures show dynamic bit-rate adaptations between the maximum bit-rate and the minimum bit-rate as decided by the current state of the network. When a transmission failure occurs, the streaming bit-rate is changed to a median value between minimum bit-rate and maximum bit-rate. After that, the bit-rate can recover linearly according to the network's degree of stability. We could see that the network is not good enough to support streaming service from 1400 to 2100 seconds in the Figure 2-4, because target bit-rate is dropped from 200 Kbps to 100 Kbps or less. Instead of bit-rate dropping, re-buffering could be considered for streaming media service in that period. If re-buffering is worked, user should wait a few seconds at least 4 times, because there are 4 transmission failures at least in that period. However more transmission failures could be occurred when target bit-rate is 200 Kbps. Because target bit-rate is between 130 Kbps and 60 Kbps in that period. Therefore user should wait a few seconds for streaming media service more than 4 times in 700 seconds (approx. 11 minutes). It means that user should be patient every 2 or 3 minutes. However the NAATA

supports seamless streaming media server without any re-buffering.

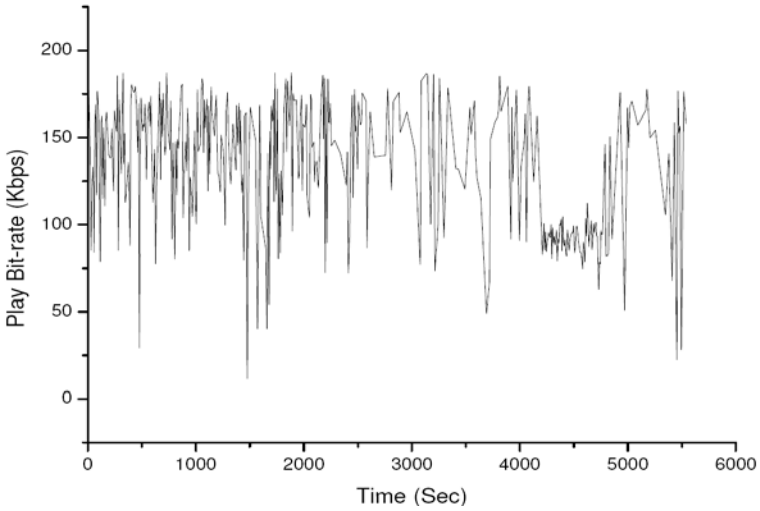

Figure 2-3. Streaming bit-rate of movie 1 with ANATS.

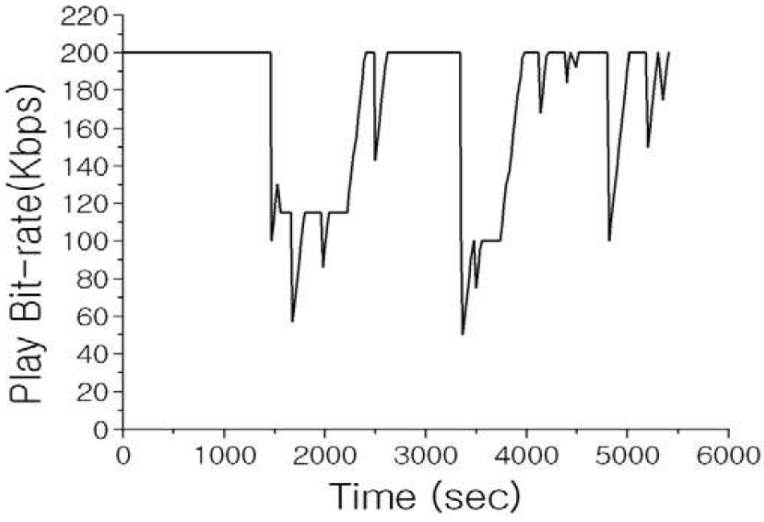

Figure 2-4. Streaming bit-rate of movie 1 with NAATA.

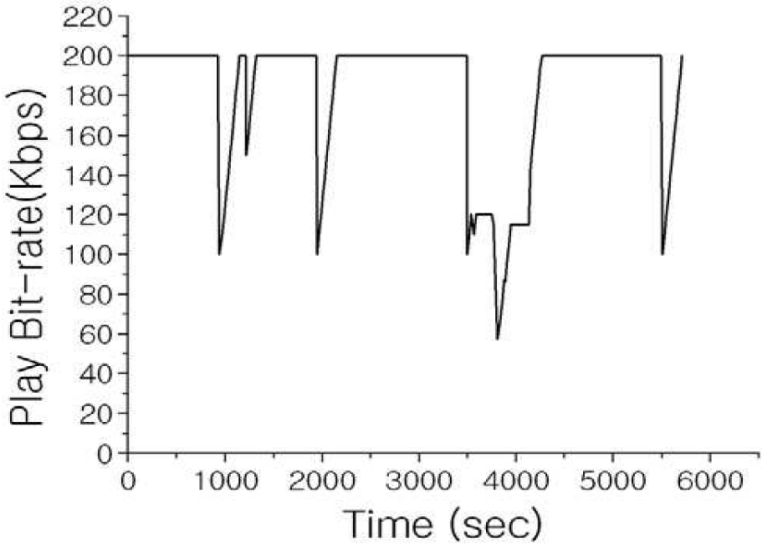

Figure 2-5. Streaming bit-rate of movie 2 with NAATA.

As shown in Figure 2-4 and 2-5, we can confirm that the NAATA serves a seamless streaming media service with dynamic bit-rate adaptation. Although the quality of play media drops due to the low bit-rate, it is better than the jittering and ceasing phenomena of media streaming. If the streaming bit-rate is kept as the initial bit-rate in the poor network bandwidth environment, the jittering and ceasing phenomena cannot be avoided.

4.2. Accumulated Number of Transmission failure

Figure 2-6 shows the accumulated number of transmission failures in the NAATA, the ANAT system and the legacy transcoding system. Various wireless network states are created for near-real service environments as mobile clients change their locations.

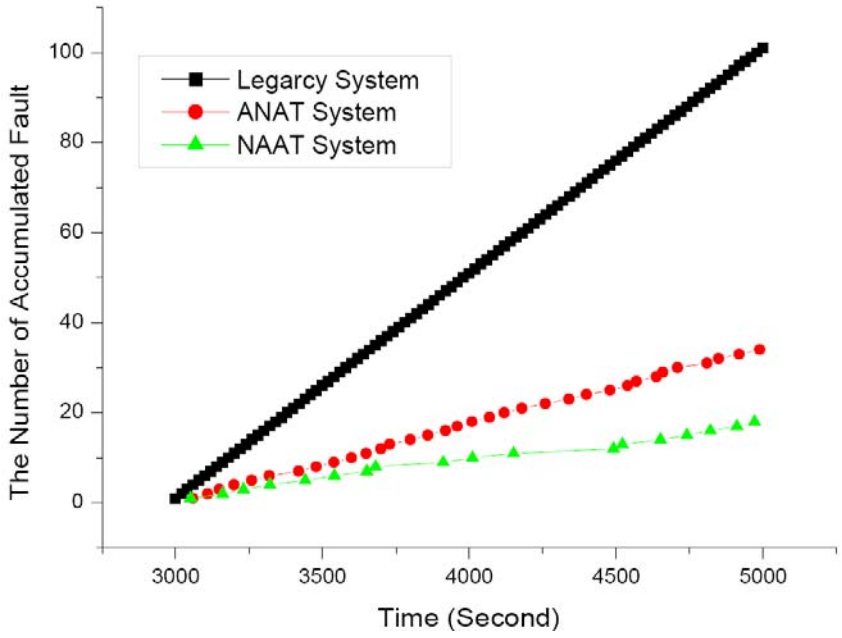

Figure 2-6. Accumulated number of transmission failures.

In the legacy transcoding system, many transmission failures are discovered and streaming service jittering and ceasing events appear. The ANAT system has fewer transmission failures than the legacy transcoding system. Since the ANAT system changes streaming bit-rates with the estimated available network bandwidth, a network-adaptive streaming is possible. However, as shown in Figure 2-6, periodic transmission failures continue to exist and jittering and ceasing phenomena also show up.

Otherwise, the NAATA results in reduced transmission failures of about 80% compared with the legacy transcoding system and of about 40% compared with the ANAT system. Furthermore, the NAATA avoids not only the continuous transmission failures but also the periodic transmission failures. Given these reliable results, the NAATA provides more seamless streaming media service than others.

4.3. Time Interval between Transmission Failures

Figure 2-7 shows the time intervals between transmission failures in the NAATA and the ANAT system. In the ANAT system, almost all of the time intervals are less than 100 seconds, with a normal distribution and with a mean of 60 seconds. Thus, transmission failures occur every 1 minute on mobile clients. This indicates that the jittering and ceasing events of streaming media occur every minute.

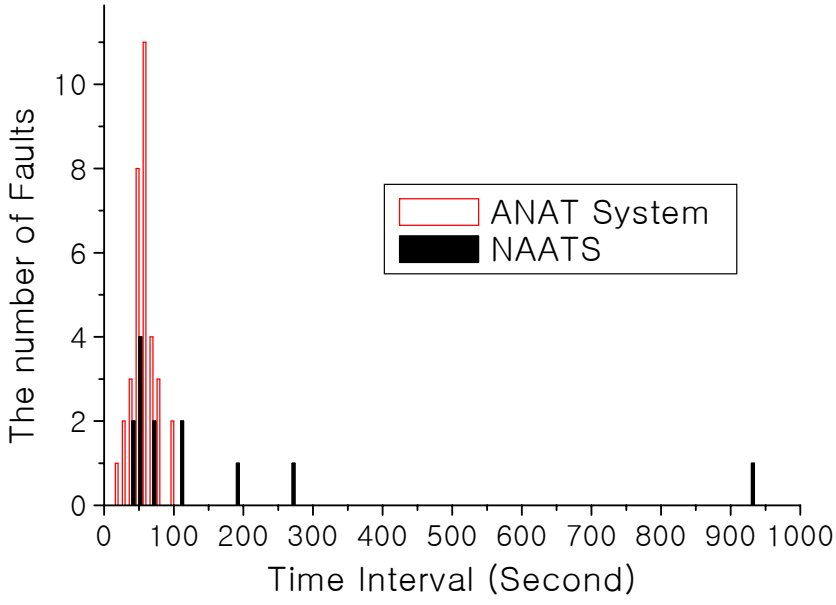

Figure 2-7. Histogram for time interval between transmission failures.

However, the proposed NAATA has long time intervals between failures when compared to the ANAT system. In our experiments, the NAATA provided a streaming media service, without any transmission failures, for 935 seconds at a maximum. In particular, the time intervals of less than 60 seconds were minimal. Compare to the ANAT system, transmission failures in the NAATA are distributed

widely across the total play time. Given these advantages, the NAATA provides seamless streaming media services to mobile clients for a longer time.

4.4. Overhead of NAATA and ANAT system

Available network bandwidth-based transcoding systems have overhead associated with estimating the available network bandwidth. The ANAT system uses the IGI algorithm to estimate available network bandwidth. As mentioned in Section 2.3, the IGI algorithm uses a lot of packets for continuous estimation. A sender makes the packets as small as possible. It also continuously increases the number of packets until the number of received packets on receiver side is same as the number of sent packets on the sender side. If a packet size is 500 bytes and at least 60 packets are required, then more than 30 Kbytes are necessary per estimation [15,21]. There is overhead in the ANAT method.

In order to provide seamless steaming service to clients, a short interval between estimations is suggested. The short interval lets the target bit-rate quickly adapt to the variable network bandwidth. However, a short interval generates more overhead. For example, as mentioned above, if the estimation process performs per 1 second, the traffic overhead of 30 Kbytes occurs per 1 second. If a streaming service requires a network bandwidth between 50 Kbps and 200 Kbps, then the overhead for 1 second is almost equal to the bandwidth of the streaming service for 1 to 4 clients. To evaluate the network state in the NAATA, a sender uses only a 24 byte payload composed of a TCP packet header and an integer variable. The integer variable is used to store the amount of data received by the client for 1 second. Although network state checking is done every second, an overhead of

only 192 bps occurs and it does not impact upon the total network traffic. From these calculations, the overhead of the ANAT is 1,250 times bigger than that of the NAATA. As a result, the NAATA not only has little overhead but it also provides network-adaptive streaming media service.

5. Summary

In a mobile client environment, a streaming media service has constraints such as low computing power, unstable wireless networks, and so on. The bandwidth of wireless network fluctuates according to mobile clients' movements and the distance from the AP; as a result, it is hard to provide a stable QoS-guaranteed streaming media service.

In this chapter, the NAATA was proposed to provide seamless streaming media services for mobile clients. The proposed method detects transmission failures, changes transcoding target bit-rates according to the network states and provides seamless steaming media services for mobile clients. In our experiments, the NAATA reduced transmission failures of 80% and 40% when compared with the legacy transcoding system and the ANAT system, respectively. We also established that the NAATA has less overhead and long time intervals between transmission failures. Based on these advantages, we confirmed that the NAATA provides a more seamless streaming media service for mobile clients, without jittering or ceasing phenomena.

Chapter III

Load Distribution Algorithm Based on Transcoding Time Estimation for Distributed Transcoding Servers

1. Motivation

Based on the recent significant growth of telecommunication, computer, and image compression technologies, the streaming media service has been spotlighted in many multimedia applications. In particular, the advances in wireless network technologies have enabled streaming media service on mobile devices such as PDAs and cellular phones. Streaming media need larger and more complex data than the traditional text and image data; thus, a large network traffic bandwidth and high-performance computing ability are required to support the quality of service (QoS) streams [1-3,9,10].

Wireless networks have lower bandwidth channels than wired networks, and mobile devices have limited hardware specifications. In this mobile environment, it is hard to provide real-time processing of a high-quality streaming media service. To solve this problem, transcoding technologies that change media data quality into mobile client capability have been studied [9,11,12]. Transcoding technology is an adaptation of the original encoded media to the given mobile devices. The range of adaptations includes changing the frame rates, bit-rates, video sizes, and re-encoding MPEG I and II media into MPEG IV [6,9,11,12,26,27].

The transcoding system is usually composed of multimedia servers and transcoding servers. The multimedia server retrieves multimedia data to the selected transcoding server, whereas the transcoding server performs transcoding jobs and sustains the streaming service to the corresponding mobile clients. Because much CPU usage is necessary for the transcoding process, a transcoding server is able to treat only a few limited requests. When many requests arrive at a transcoding server in short period, it is difficult to finish transcoding jobs within the deadline for QoS metric. For large-scale streaming services in a mobile environment, many transcoding servers are required. Arrival transcoding jobs should be distributed among transcoding servers [9-12,26,27].

In large-scale distributed transcoding systems, fair distribution of the transcoding load among transcoding servers is a critical issue to prevent disproportionate and concentrated load of a transcoding server. All media data have different transcoding process times because of size, bit-rate, frame rate, and transcoding methods. To achieve fair load distribution between distributed transcoding servers, transcoding time estimation is needed according to CPU performance of the transcoding server, transcoding requirements and media data.

In this chapter, a new load distribution method is proposed for fair transcoding load distribution in the distributed transcoding servers. The proposed method controls and distributes transcoding requests based on transcoding time estimation for distributed transcoding environments. In our experiments, the proposed method shows better scalable performance than other load distribution methods, because the characteristics of transcoding servers, transcoding requirements and streaming media data are considered.

2. Related Work

2.1. Transcoding Systems

Figure 3-1 shows the architecture of a legacy transcoding system. The client sends a transcoding request to a transcoding server. A transcoding server reads the original media data from the media server, transcodes them according to user requested resolution, bit-rate, and frame rate, and then sends the transcoded media data. For example, a transcoding server could provide quarter common intermediate format (QCIF) streaming media (176 × 144 resolution, 15 frames/second, 50 Kbps) to mobile clients, which are transcoded from common intermediate format (CIF) media (352 × 288 resolution, 25 frames/second, 100 Kbps) [9,26].

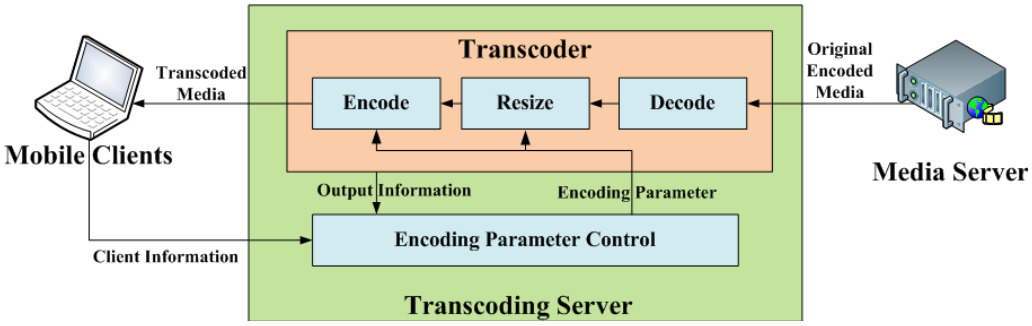

Figure 3-1. Architecture of legacy transcoding system.

There have been several approaches for transcoding systems, including source-based static encoding systems and static transcoding server systems [6,9,15]. In the source-based static encoding system, the server stores the media data encoded for all client grades. Due to the absence of on-line overhead for transcoding, this approach has an advantage on the streaming service side. However, it is difficult to prepare encoded videos that are adapted for all kinds of

mobile clients. This method has the disadvantage of storing all grade-encoded media to the same media title.

In the static transcoding server system, the closest transcoding server from the wireless base of the mobile client is selected to provide streaming media service to a given mobile client. In this approach, specific servers could be saturated by concentrated transcoding jobs. To address this problem, the transcoding load distribution method has been studied to avoid load imbalance in a static transcoding server system. In the load distribution transcoding system, a load distribution server selects transcoding server based on load distribution policy and transcoding servers' information [9,11,12].

2.2. Load Distribution Methods

Much research was undertaken on the load distribution strategies in a cluster-based server. In particular, the cluster-based server architecture has been utilized in Web server, game server, and file server areas. As representative methods in these areas, there are the round robin (RR), the least connection (LC), the weighted round robin (WRR), the dynamic weighted round robin (DWRR), the resource weighted load distribution (RWLD), among others [9,11,12,26,27].

The RR method allocates servers according to the sequence of job arrival. Because the RR does not consider the state of servers and the intrinsic features of jobs, it is difficult to attain the effective load balancing among transcoding servers.

The LC method uses the count of clients connected to each server. This method chooses the server with the least count value. However, in the case where the clients keeping a long connection are mixed with others with a short connection,

the LC method a load imbalance between transcoding servers.

The WRR method designates a different weight to each server based on the capability of transcoding servers. For example, if the basic weight is 1 and servers A, B, and C have 4, 3, and 2 weights respectively, the order of scheduling is ABCABCABA. This approach cannot reflect the state of transcoding servers that are dynamically changed. To address this problem, the DWRR method is suggested. For distributing jobs to servers, the DWRR method considers the current state of backend servers. There are two ways to monitor the state of backend server. The first is for all backend servers to send their state to the load distribution server periodically. This method has communication overheads between the load distribution server and the backend servers. The second is for a load distribution server to require the information from all backend servers whenever a client request has arrived. In this method, when some of the backend servers fail in the process of streaming service, the load distribution server is not aware of the failed servers immediately. In addition, if the number of clients increases abruptly, the heavy communication traffic appears between a load distribution server and the backend servers.

The RWLD method distributes transcoding jobs requests based on resource weight consumption rates pre-measured on three client grades. However the RWLD method is not able to support various kinds of clients, because it supports only these three grades of media data. Furthermore, this method has communication overheads to get information from transcoding servers' CPU usage.

3. Transcoding Time Estimation Based Load Distribution Method

For fair load balance between distributed transcoding servers, transcoding time estimation for user requests is necessary. In this chapter, we estimate transcoding times from target transcoding bit-rate and information of source media data, such as running time, bit-rate, and frame rate. In addition, intrinsic CPU information of the transcoding server is also used for transcoding time estimation. This approach could be used for heterogeneous distributed transcoding servers, because both the transcoding server and media are considered for transcoding time estimation. Based on transcoding time estimation, a new load distribution method among distributed heterogeneous transcoding servers is proposed.

3.1. Transcoding Time Analysis

A transcoding job consists of decoding source media data and encoding. The transcoding time is the sum of decoding time and encoding time. Decoding and encoding have various steps such as discrete cosine transform (DCT), inverse DCT (iDCT), quantization (Q), inverse Q (iQ), motion estimation/motion compensation (ME/MC), variable length coding (VLC), variable length decoding (VLD). A transcoding time is the summation of all steps' time in decoding and encoding (Equation 3-1). Table 3-1 shows symbols for equations.

In this chapter, we only consider transcoding time to change bit-rate. In a transcoding job to change bit-rate, the ***ME/MC*** and ***sampling*** items in Equation 3-1 take 0 seconds because changing the frame rate and resolution is not needed. The ***DCT*** and ***Q*** take the same time because the operations are always done. As a result, the transcoding time to change bit-rate results, as shown like Equation 3-2.

Table 3-1. Symbols for equations

Symbols	Means	Units
Tr_T	Transcoding job time	sec
Tr_R	Relative transcoding time by target bit-rate	sec
SB	Source bit-rate of media data	bps
TB	Target bit-rate for transcoding	bps
VLC, VLD	Time for variable length coding and decoding, respectively	sec
Q, iQ	Time for quantization and inverse quantization, respectively	sec
DCT, iDCT	Time for DCT and iDCT, respectively	sec
ME/MC	Time for motion estimation/motion compensation	sec
Sampling	Time for sampling	sec
WriteTime	Time for writing 1 byte to memory	sec
Clock	CPU clock of transcoding server	Hz
playtime	Running time of media data	sec
inst	The number of instructions in putAC function	ea
UnitOfWrite	Unit of data writing in putAC function	bit
STT	Source bit-rate transcoding time	sec
Resolution	Width × height	pixel
FrameRate	Frame rate of source media	Frame/sec

$$Tr_T = VLD + iQ + iDCT + MC + Sampling + ME/MC + DCT + Q + VLC \quad \text{Equation 3-1}$$

$$MC + Samplint + ME/MC = 0$$
$$\therefore Tr_T = VLC + \alpha \; (\alpha = VLD + iQ + iDCT + DCT + Q) \quad \text{Equation 3-2}$$

Table 3-2 shows a pseudo-code of VLC in MPEG-2 [28]. A macro block has a DC coefficient and 63 AC coefficients because a macro block is an 8 × 8 block. In a macro block, 63 AC coefficient values are coded by run-length coding. As the target bit-rate gets higher, the ***putAC()*** function is called on more frequently. Therefore, the transcoding request with higher target bit-rate causes a longer transcoding time.

Table 3-2. VLC pseudo-code in MPEG-2

```
Algorithm VLC {
        /* DC coefficient   */
        putDClum(dc_value);
        /* AC coefficient   */
        run = 0;
        for(i=1; i<64; i++) {
                if(ac_value != 0) {
                        putAC(run, dc_value)
                        run = 0;
                }
                else run++;
        }
}
```

Equation 3-3 shows VLC process time. ***WriteTime*** means the time to write 1 byte to file in the ***putAC()*** function. The n can be relatively calculated with the target transcoding bit-rate. To estimate ***WriteTime***, we analyzed the assembly code of the ***putAC()*** function. In this chapter, IA-32 instruction set architecture such as Intel-like and AMD-like is considered, because the number of assembly instructions depends on the instruction set architecture. When there is no error in ***putAC()*** function, we confirmed that 55 assembly instructions are executed. The ***WriteTime*** can be calculated as in Equation 3-4. From Equation 3-3 and 3-4, Equation 3-5 and 3-6 indicate the relative transcoding time. In Equation 3-6, relative transcoding time, Tr_R, is proportioned to the different value the between source bit-rate and the target bit-rate (***SB - TB***). In addition, it depends on the CPU clock rate, ***Clock***.

$$\begin{aligned} &VLC = n \times Wirte\,Time + \beta \\ &\begin{pmatrix} n = the\,\nu mber\,of\,putAC()\,function\,called, \\ \beta = constant\,time\,value \in VLC \end{pmatrix} \end{aligned} \qquad \text{Equation 3-3}$$

$$Write\,Time = \frac{inst}{\boldsymbol{Clock}} \qquad \text{Equation 3-4}$$

$$Tr_R = \frac{(\boldsymbol{SB} - \boldsymbol{TB}) \times playtime}{Unit\,Of\,Write} \times \frac{inst}{\boldsymbol{Clock}} \qquad \text{Equation 3-5}$$

$$Tr_R = \left(\frac{inst \times playtime}{Unit\,Of\,Write \times \boldsymbol{Clock}} \right) \times (\boldsymbol{SB} - \boldsymbol{TB}) \qquad \text{Equation 3-6}$$

From Equation 3-6, the relative transcoding time can be estimated, however, the relative transcoding time is not useful for load balancing in real distributed transcoding service environments. Therefore, the absolute transcoding time estimation should be derived from the relative transcoding time.

$$Tr_T = STT - x \times Tr_R \qquad \text{Equation 3-7}$$

$$x = \frac{Resolution}{\frac{SB}{FramRate}} \qquad \text{Equation 3-8}$$

$$Tr_T = STT - Tr_R \times \frac{Resolution}{\frac{SB}{FrameRate}} \qquad \text{Equation 3-9}$$

$$Tr_T = STT - \left(\frac{inst \times playtime}{Unit\,Of\,Write \times \boldsymbol{Clock}} \right) \times (\boldsymbol{SB} - \boldsymbol{TB}) \times \frac{Resolution}{\frac{SB}{FrameRate}} \qquad \text{Equation 3-10}$$

Equation 3-7, 3-8, and 3-9 show that the transcoding job time estimation, $\boldsymbol{Tr_T}$, is driven from the relative transcoding time, $\boldsymbol{Tr_R}$. In Equation 3-7, the absolute transcoding time, $\boldsymbol{Tr_T}$, is calculated by subtracting the relative transcoding time $\boldsymbol{Tr_R}$ from the source bit-rate transcoding time ***STT***. In Equation 3-8, ***Resolution*** indicates

the amount of non-encoded source media data in a picture, and the ***Sourcebitrate/FrameRate*** means the amount of source media data in a picture. Thus, Equation 3-8 shows how many times bigger non-encoded source media data are than encoded source media data. From Equation 3-7, 3-8, and 3-9, the absolute transcoding time estimation, Tr_T, is achieved in Equation 3-10. From this equation, we find that the difference between ***SB*** and ***TB*** and the CPU clock rate (***Clock***) have a great impact upon Tr_T. They are written as bold type.

3.2. Load Distribution and Admission Control

To provide streaming media service for satisfying user requested QoS, media data should be transcoded in servers and retrieved at clients within the designated deadline time. When transcoding requests enter, the transcoding job distribution among transcoding servers is needed to achieve effective load balance and to provide maximum QoS streams. In this chapter, the **T**ranscoding time **E**stimation-based **L**oad **D**istribution (TELD) method is proposed. Based on the transcoding job time estimation mentioned in Equation 3-10, the TELD method distributes transcoding jobs among distributed transcoding servers and performs admission controls to new clients.

When new transcoding request arrives, TELD estimates the transcoding times of a new request in all transcoding servers. After that, it calculates total transcoding job times in each transcoding server. A total transcoding job time means the completed time of all transcoding jobs loaded in a corresponding transcoding server. TELD assigns a new transcoding job to a selected transcoding server that has a minimal total transcoding job time among transcoding servers. In addition, based on

the total transcoding times and the running-time of new requested media, TELD performs an admission control for a new transcoding request.

Figure 3-2 shows an example of the transcoding job scheduling among N transcoding servers when a new transcoding request arrives. The horizontal striped bar indicates the transcoding estimation time currently in each transcoding servers. The vertical striped bar indicates the estimated transcoding time issues by a new transcoding job. As shown in this figure, the vertical striped bar in each transcoding server is calculated. These newly estimated transcoding times represented as vertical striped bars are based on Equation 3-10. As shown in Equation 3-10, the estimated transcoding time values are inversely depended on the CPU clock rates of corresponding transcoding servers. For a new transcoding request, the estimated transcoding time in Server 3 with a 1.8-GHz clock shows the longest value, but Server 1 with 3.0-GHz clock has the shortest transcoding estimation time.

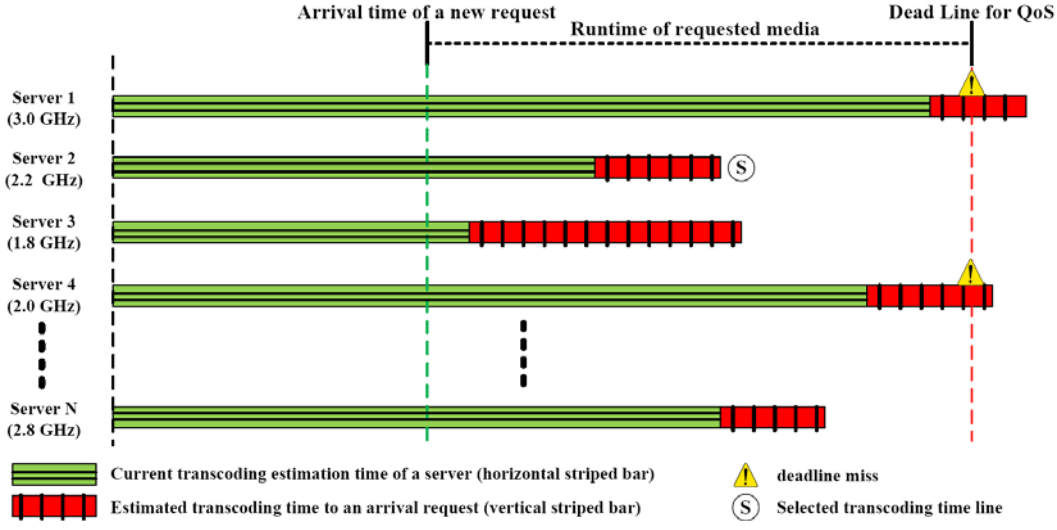

Figure 3-2. Concept of the TELD method.

For load distribution, TELD calculates total transcoding times, including the estimated transcoding time by a new request. To support QoS, the transcoding process should be finished before the runtime of requested media. Thus, the running time of new requested media is used as a deadline for completing all transcoding jobs in the transcoding server. If the total transcoding time of a transcoding server exceeds the dead line, this transcoding server is not able to provide transcoding service within the limited time for QoS. Therefore, this server cannot be selected as a transcoding server for a new transcoding job. In Figure 3-2, Server 1 and Server 4 are the transcoding servers that do not satisfy the deadline. Among the transcoding servers satisfying the deadline, the TELD method selects a transcoding server with minimal transcoding time. In our example, Server 2 is selected for a new transcoding job, because the total transcoding time of Server 2 is the shortest among transcoding servers.

3.3. Algorithm

Table 3 shows a pseudo-code of the TELD algorithm. TELD estimates transcoding times of transcoding jobs to change bit-rate, and selects an optimal transcoding server among distributed transcoding servers. The ***Time_Estimation()*** function in part C returns the estimated transcoding time value with the target transcoding bit-rate and transcoding server information. Returned time values are stored in the ***new_load[]*** and the total transcoding time values of each transcoding server are stored in the ***AccLoad[]***. The ***select_mts()*** function in part D calculates estimated total transcoding time values of each transcoding server with the sum of ***AccLoad[]*** for accumulated transcoding estimation time and ***new_load[]*** for new user requests.

Table 3-3. The TELD algorithm

```
struct mts_info mts[node_no]; // Struct variable for transcoding server information                    A
double AccLoad[node_no];     // Accumulated job time of current transcoding servers
double new_load[node_no];     // Estimated time for new request
double tmp[node_no];          // temporal variable for estimation

int TELD(struct new_job)                                                                              B
{
  for(i=0; i<node_no; i++)      // Estimate transcoding time for each transcoding server
    load[i] = Time_Estimation(new_job, mts[i].cpu_hz);
  return select_mts(load, new_job.playtime);
}

double Time_Estimation(struct newjob, double cpu)                                                     C
{
  tr_r = ((INST * newjob.p_time) / (8 * cpu * 1024)) * (newjob.SB - newjob.TB) * 1024;
                                  // Calculate relative transcoding time
  tr_t = newjob.STT- (tr_r * 8 * newjob_resol / newjob.SB / newjob.Frame) ;
                                  // Estimate transcoding time
  return tr_t;
}

int select_mts(double load[], int playtime)                                                           D
{
  time_gap = current_time - prev_time; // Calculate difference btw current time and
previous time
  for(i=0; i<node_no; i++)                    // Add new transcoding time to current      D1
transcoding time
    tmp[i] = AccLoad[i] + new_load[i] - time_gap;

  for(i=0; i<node_no; i++) {                  // Select transcoding server
    if(tmp[i] < min) {                                                                                D2
      min = tmp[i];
      selected = i;
    }
  }

  if (tmp[selected] <= playtime) {
    AccLoad[selected] = tmp[selected];
    return selected;                                                                                  D3
  }
  else return -1;      // Service denied
}
```

In part D2, a transcoding server number with the minimum estimated total transcoding time value is chosen among transcoding servers. In part D3, if the transcoding time supported by the chosen transcoding server is bigger than the running time of the requested media, no transcoding server can provide a QoS stream for new request. As a result, our admission control rejects the new user

request as returning -1 value. In other conditions, TELD assigns new user requests to selected transcoding server from the return value of select_mts() function.

Figure 3-3 shows the flow chart of TELD which distributes transcoding jobs and performs admission control. When the transcoding system is started, TELD initializes information of distributed transcoding servers and awaits user requests. When a new request arrives, transcoding times for each transcoding server are estimated and a server that has a minimal value is selected. After that, to execute the admission control, the TELD checks whether the total transcoding time supported by the selected server satisfies the running time of the requested media. After the admission control passes, a new request is assigned to a selected server.

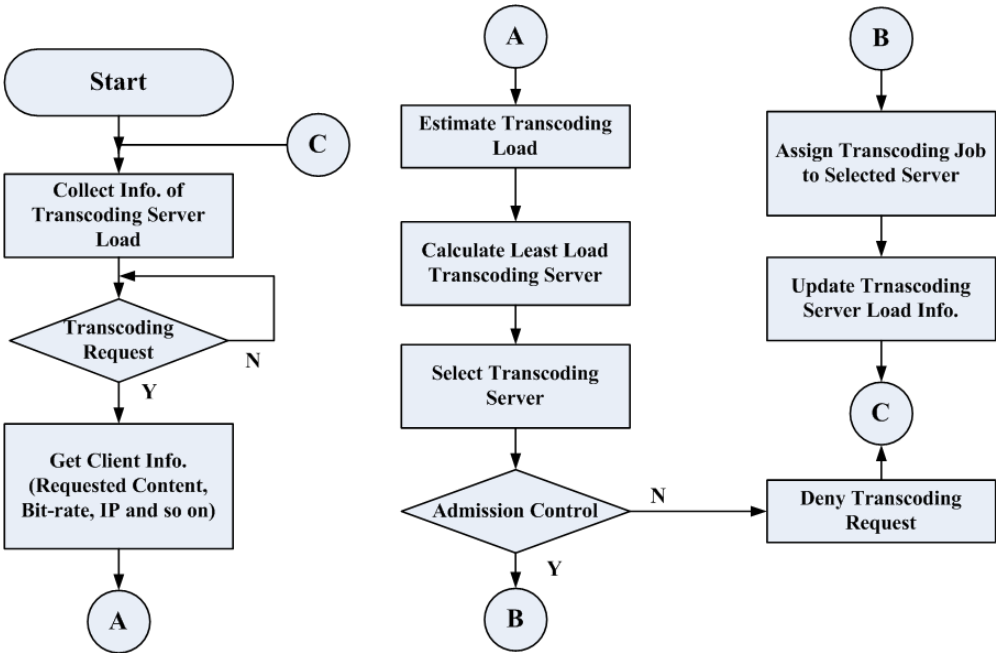

Figure 3-3. Flow chart of TELD.

4. Experimental Environment

For experiments, a distributed transcoding system is implemented that consists of a load distribution server and nine transcoding servers. A load distribution server accepts user transcoding requests and distributes transcoding jobs; transcoding servers transcode requested media data. Table 3-4 shows hardware specifications of transcoding servers: 2.2GHz and 1.8GHz AMD CPUs are used in these transcoding servers. We use the ffmpeg (version 0.4.9) [24] implemented by open-source project for transcoding. Table 3-5 shows media data for the experiment.

Figure 3-4 shows the architecture of the implemented experimental system. Transcoding servers with odd numbers use the Server 2 specification in Table 3-4, and the servers with even numbers use the server 1 specification. Nine total transcoding servers are used in the experimental system.

Table 3-4. Hardware specification of transcoding server

	CPU	Memory	Amount
Transcoding server 1	AMD Athlon MP Thoroughbred 2200+ 1.80.GHz	1 GB	4
Transcoding server 2	AMD Opteron 248 2.20GHz	1 GB	5

Table 3-5. Media data for experiment

		Bit-rate	Running time	Frame rate	Resolution
The Lord of the Rings – The Two Towers	CIF-like	1362Kbps	1448 sec	24 fps	656 × 320
	QCIF-like	769 Kbps			328 × 160
	SQCIF-like	468 Kbps			164 × 80

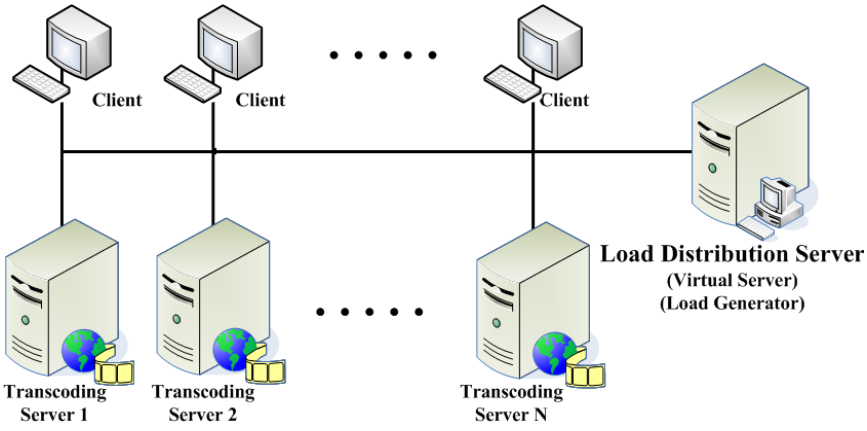

Figure 3-4. Implemented experimental system architecture.

A load generation program is implemented for the TELD experiment. This program generates user requests from a Poisson distribution with lambda 0.25 [1]. The requested target transcoding bit-rates are chosen on the present condition of wireless network bandwidth usages and mobile clients from the 2008 Survey on the Wireless Internet Use [29]. Eighty percent of user requests are 56Kbps target bit-rate; the others are randomly decided between 100Kbps and 1000Kbps.

We implemented a yardstick program [30] that consists of a virtual server and virtual clients for experiencing similar real environments. The virtual server in a load distribution server generates user requests by using a load generation program. The generated requests are sent to virtual clients, which requests streaming media service from the load distribution server. When the load distribution server receives user requests from virtual clients, TELD selects a transcoding server among distributed servers and forwards it to the selected server. The selected transcoding server performs the requested transcoding job and provides streaming media service to the virtual client.

The virtual client receives streaming media from the selected transcoding server at the requested bit-rate. Furthermore, the virtual client checks whether the amount of received data is lower than requested bit-rate; if it is, the virtual client sends an error message to the virtual server. This error message means that the serviced stream does not satisfy the QoS metric. The streams with errors are excluded in the total number of QoS streams.

5. Performance Evaluation

5.1. Transcoding Time Estimation and Measurement

CIF-like media data cannot be transcoded to 500Kbps or less, because each resolution of media data has a minimum bit-rate. Thus, we use three kinds of media data type, as shown as Table 3-5. Transcoding time values for each transcoding server can be estimated with relative transcoding time values by using Equation 3-10. Figure 3-5, 3-6, and 3-7 show the estimated transcoding time values on each transcoding server and the measured transcoding time values. As shown in these figures, the estimated transcoding time values are similar to real transcoding times on transcoding servers. Figure 3-8 shows differential rates between estimated transcoding time values and measured transcoding time values. The worst case is 2.94% in Server 1, which performed the transcoding job from SQCIF media to 300Kbps bit-rate. The average differential rate is 1.01% for all servers and all media. Based on the minimal differential rates between estimated transcoding times and measured transcoding times, we can use the estimated transcoding times derived from Equation 3-10 for the load distribution among distributed transcoding servers for arrival transcoding requests.

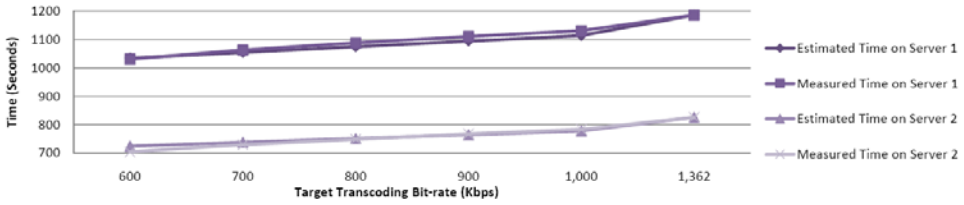

Figure 3-5. Transcoding time using CIF-like media.

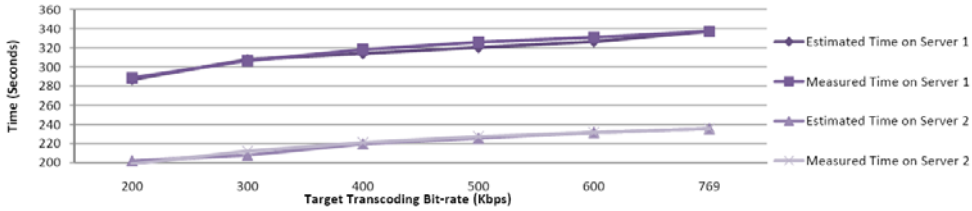

Figure 3-6. Transcoding time using SCIF-like media.

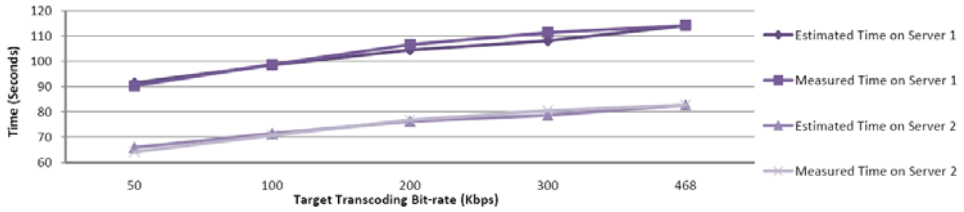

Figure 3-7. Transcoding time using SQCIF-like media.

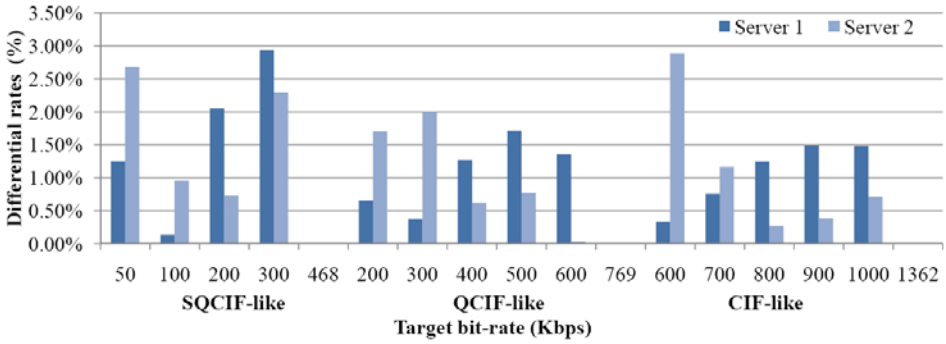

Figure 3-8. Differential rates between estimated times and measured times.

5.2. Number of QoS Stream

For performance evaluation, we measured the maximum numbers of QoS streams. We implemented the RR, WRR, RWLD, and TELD in the same working environment. Figure 3-9 shows the number of QoS streams according to the number of transcoding servers. The QoS means that clients received streaming service within the designated bit-rate. It is the most important mandatory requirement in the streaming media service. If the QoS requirement does not guarantee in the serviced streams, those streams cannot be involved in the total number of QoS streams.

As illustrated in Figure 3-9, the maximum numbers of QoS streams increases proportional to the number of transcoding servers in all methods. In this figure, the TELD method shows the best performance scalability compared to the RR, WRR, and RWLD methods.

When the RR method is used, some transcoding servers suffer from overloaded transcoding jobs, because the method does not consider media transcoding. As a result, these servers cannot satisfy the QoS requirement. In particular, new transcoding jobs allocated to these overloaded transcoding servers have negative impact on other QoS streams in progress.

For load distribution, the WRR method uses CPU performance information of transcoding servers. This method does not consider the characteristic of transcoding jobs as does the RR method. Therefore, because some specific servers suffer from a heavily transcoding load, the WRR method did not show good performance scalability.

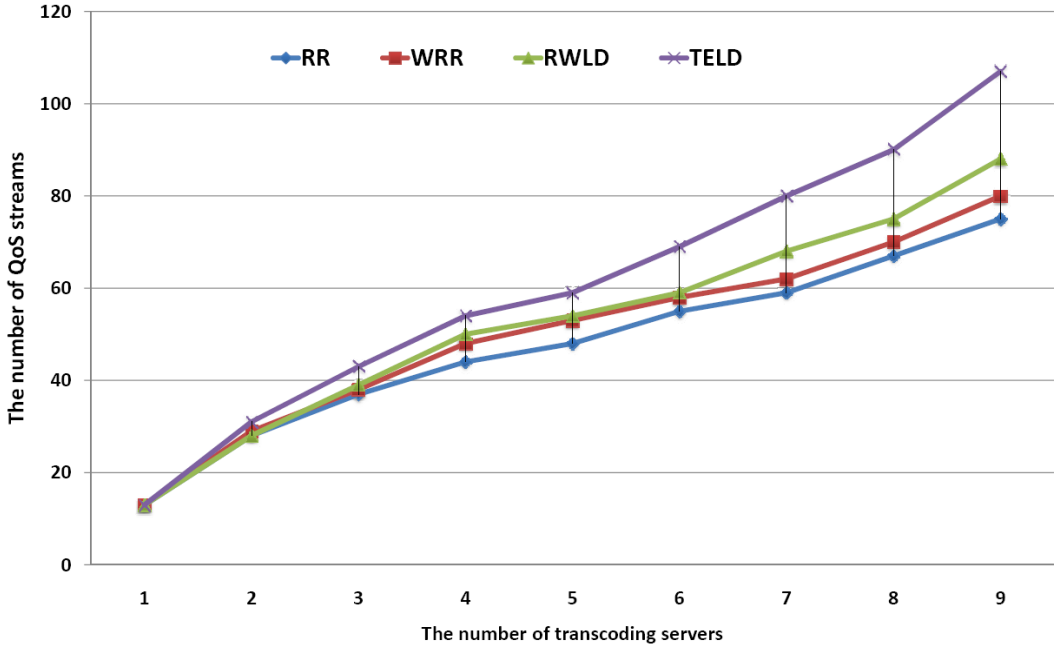

Figure 3-9. The number of QoS clients according to the number of transcoding servers.

The RWLD method uses both the resource consumption weights by three transcoding grades and the maximum number of streams in transcoding servers. However, this method considers only three kinds of transcoding media grades, which are not enough to estimate the transcoding times for various mobile devices. In addition, because the transcoding estimation times included the time for the streaming process, accurate transcoding times could not be applied in load distribution. As a result, even if the number of transcoding servers is increased, the RWLD should be able to limit the performance scalability as shown in the Figure 3-9.

On the other hand, the TELD method is based on the estimated transcoding time for various transcoding bit-rates in transcoding servers. In particular, the TELD

method considers CPU clocks of transcoding servers and any kinds of bit-rate transcoding job. In the Figure 3-9, according to the number of transcoding servers, the QoS streams supported by the TELD method increases rapidly because the TELD method distributes transcoding jobs based on the estimated transcoding time according to the characteristic of target media and transcoding servers. As the number of transcoding server increases, the available time for transcoding in servers also increases. As a result, the total transcoding job times in transcoding servers decrease and so more transcoding requests could be accepted in the TELD method.

6. Summary

In this chapter, load distribution methods were studied in distributed transcoding servers. Based on analysis of the transcoding process of MPEG-2 media, we found out that VLC is the most significant part for transcoding time estimation. We proposed the transcoding time Estimation method based on characteristics of MPEG-2 media and transcoding server. Using the estimated transcoding time, the TELD method was proposed for fair load distribution and admission control of distributed transcoding servers.

In our experimental distributed transcoding servers, we evaluated the scalable performance of the RR, WRR, RWLD and TELD methods. Because the RR and WRR method did not consider the characteristic of transcoding jobs, it cannot satisfy the QoS requirement. Therefore, because some specific servers suffer from heavy transcoding load, it did not show good performance scalability. The RWLD method considers only three kinds of transcoding media grades, which are not enough to estimate the transcoding times for various mobile devices. As a result,

even if the number of transcoding servers is increased, the RWLD should be able to limit the performance scalability.

The TELD method distributes transcoding jobs based on the estimated transcoding time according to the characteristic of target media and transcoding servers. As a result, we confirmed that the proposed TELD method provided more linear performance scalability than other load distribution in Section 5.

Chapter IV

Stream Prefetching Method on Streaming Media Service for High Speed Mobile Users

1. Motivation

As a result of improvements in mobile wireless internet technologies such as IEEE 802.16e Mobile WiMAX, high speed moving users are able to access internet service, and a streaming media service using PDAs, laptops, mobile phones and so on. However, mobile wireless networks have variable bandwidths depending on the speed and location of clients. These characteristics make it hard to support stable Quality of Service (QoS) streams for high speed mobile clients. In a streaming media service, wide and stable network bandwidths are necessary in order to retrieve large amount of media data on real-time [1,9,10].

On high speed mobile internet environments such as Mobile WiMAX, a mobile client moving under 60Km/h can be provided 3Mbps in general condition [31-33]. However, mobile clients moving 60Km/h or faster are not able to be provided 3Mbps and stable connectivity because of frequent handover [34]. To provide stable QoS guaranteed streaming media service over high speed mobile internet environments, bandwidth allocation methods [33,35-37], handover methods [37-41], mobile IP(MIP) providing mobility [42-56], client direction detecting methods [57-66], and media streaming methods [67-77] have been studied.

The Mobile WiMAX has an additional service class for real-time service such as streaming media service, VoIP and so on [33,35-37]. The service class provides a fixed bandwidth for real-time data, but no specific method is provided for streaming media service. Based on MIP-related research [42-56], a seamless connectivity is able to be provided with minimized latency caused by handover. These can be used for streaming media service on Mobile WiMAX.

Detection methods of mobile client direction have been researched to reduce latency and disconnect caused by handover [57-66]. Since not only streaming media service but also any kind of network application need seamless connectivity, these methods are used various applications. However, these methods are not reflect high speed moving clients being provided streaming media service. In addition, Streaming media service methods for mobile internet environments have been studied [67-77] such as user grouping methods for streaming media service, buffer management methods with prefetching, and data reducing methods for transmitting. However these researches consider only specific environment or do not consider dynamic clients environment and high speed client mobility. Therefore, these methods are able to provide a few specific applications.

In this chapter, a client mobility-based media stream prefetching method to guarantee stable QoS in high speed internet environment like a Mobile WiMAX, is proposed. In proposed method, high speed moving clients is joint a group depending on their characteristics and provided streaming media service according to a joint group. Mobile clients can change their group depending on their situation. In each group, clients' direction is predicted, and media streams are prefetched against disconnect and latency caused by handover. The proposed method is experimented and evaluated that buffer state is stable with handover for

streaming media service.

2. Related Work

2.1. Mobile WiMAX

The Mobile WiMAX based on IEEE 802.16e standard is wireless wide-band network to provide high mobility and high speed bandwidth with low cost [32,33]. At least 512Kbps in downstream and 128Kbps in upstream are guaranteed to a client with 60Km/h movement in cell boundary, and channel bandwidth is over 9MHz. In general condition, 3Mbps bandwidth is provided [31,33]. To guarantee QoS for various network traffics in the Mobile WiMAX, IEEE 802.16e MAX protocol provides various bandwidth request-allocate methods are defined. The Mobile WiMAX provides different services depending on three kinds class; Real-time Polling Service(RtPS), Non-real Time Polling Service(NrtPS), and Best Effort service(BE) [33,35-37]. Therefore, mobile client serviced by Mobile WiMAX should be guaranteed QoS level of own service class.

2.2. Handover in Mobile WiMAX

In the Mobile WiMAX network, a handover consists of three steps; neighbor network search step, handover preparation step, and handover execution step. A mobile station(MS) initiate handover and a network initiate handover are provided in the Mobile WiMAX [37-41].

2.2.1. Neighbor Network Searching and Information Collect

When a MS is in network, a MS receives periodic MOB_NBR_ADV (Neighbor Advertisement) messages from base station(BS) providing wireless connectivity. This messages include information and network attributes of every neighbor BS. When a MS receives MOB_NBR_ADV message, a MS collects ID, QoS parameter, channel information of neighbor BS, and prepares fast handover [37-41].

Another way to get network information is MS's scanning procedure to measure downlink signal strength from neighbor BSs. Since a MS collects IDs of neighbor BSs using MOB_NBR_ADV message and real-time link information using scanning, a MS chooses suitable BSs and manages handover candidate BS list.

To reduce handover latency time, an association procedure with neighbor BS can be performed in scanning procedure. In association procedure, a MS can do initial ranging procedure with specific BS. Ranging is a first step when new MS enters network. Using these procedures, handover processes are optimized by a MS re-using basic information related channel attributes include frequency and power control of new BS(NBS).

2.2.2. Handover Preparation

In handover preparation, a MS selects a target BS that is optimal to handover, based on collected information such as signal strength and QoS parameter, in previous step. And then, a handover is decided depending on a MS initiated handover or network initiated handover [37-41].

2.2.2.1. MS Initiated Handover Preparation

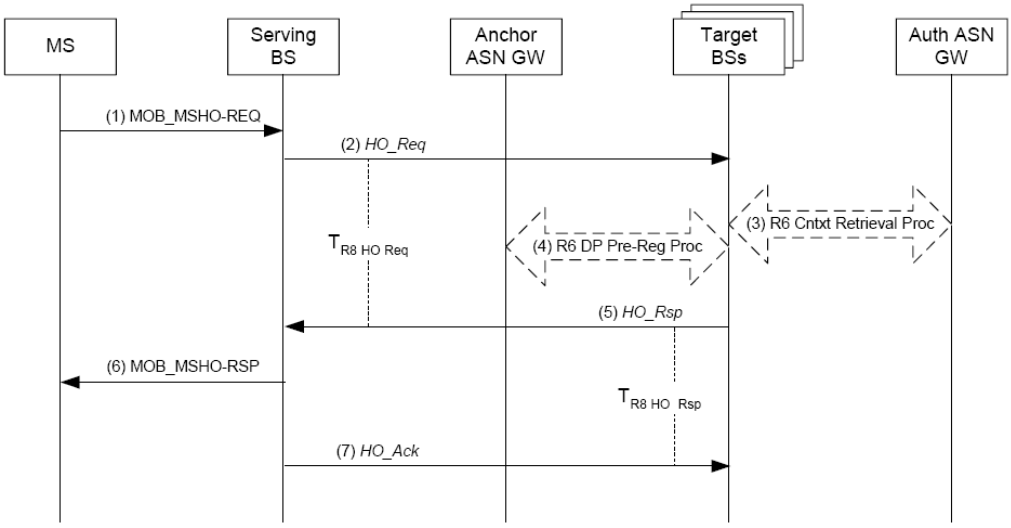

Figure 4-1. Successful MS Initiated HO Preparation[38].

STEP 1

The MS initiates a handover by sending a MOB_MSHO-REQ message to the Serving BS, which includes one or more potential target BS's.

STEP 2

The Serving BS sends an R8 HO_Req message destined to each potential Target BS''s selected for the handover and starts timer TR8-HO Req for each message. The message includes an Authenticator GW(Gateway) ID TLV(Threshold Limit Value) that points to the Authenticator/Key Distributor function at the Authenticator ASN(Access System Network) and the Anchor ASN GW ID of the Anchor Data Path function at the Anchor ASN.

STEP 3

The Target BS(s) MAY request AK(Authentication Key) context and service

authorization information for the MS by initiating a Context Retrieval procedure with the Authenticator ASN GW. Note: The Target BS(s) may optionally choose to defer this procedure to the Handover Action phase.

STEP 4

The Target BS(s) MAY initiate pre-establishment of a data path for the MS with the Anchor ASN GW. If the Anchor ASN GW does not support the Data Path Pre-Registration, the R6 Path_Prereg_Req message from the Target BS will be responded by the R6 Path_Prereg_Rsp message with an appropriate reject cause code. Note: The Target BS(s) may optionally choose to defer this procedure to the handover Action Phase.

STEP 5

The Target BS(s) sends an R8 HO_Rsp message to the Serving BS to acknowledge the handover request and starts timer TR8-HO Rsp. Upon receipt of the R8 HO_Rsp message, the Serving BS stops timer TR8-HO Req.

STEP 6

The Serving BS sends a MOB_BSHO-RSP message to the MS containing one or more potential target BS's selected by the network for the MS to handover to.

STEP 7

The Serving BS sends an R8 HO_Ack message to the Target BS(s) controlling the potential target BS(s) selected for the MS. Upon receipt of the R8 HO_Ack message, the Target BS(s) stops timer TR8-HO Rsp.

2.2.2.2. Network Initiated Handover Preparation

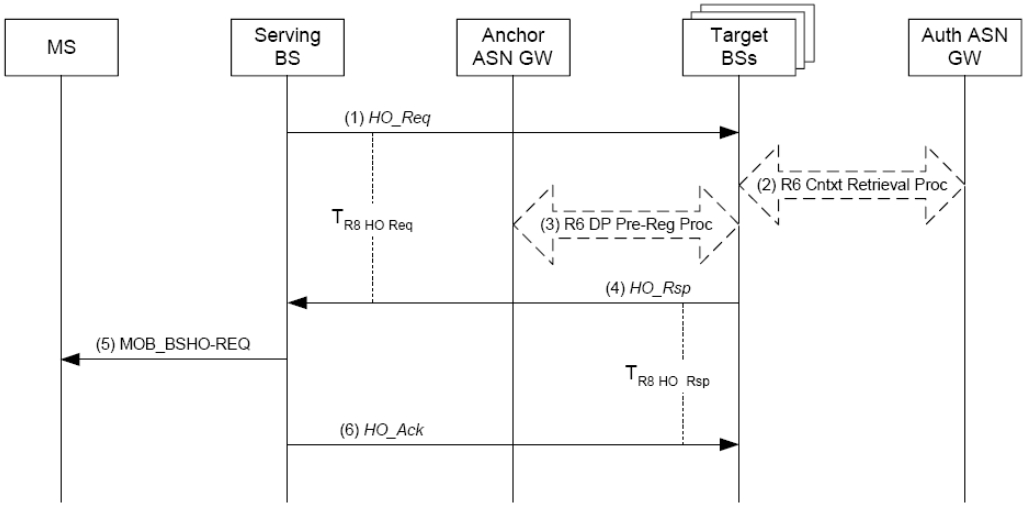

Figure 4-2. Successful Network Initiated HO Preparation Phase[38].

STEP 1

The Serving BS initiates a handover by sending an R8 HO_Req message destined to each Target BS's selected for the handover and starts timer TR8-HO Request for each message. The message includes an Authenticator GW ID TLV that points to the Authenticator/Key Distributor function at the Authenticator ASN and the Anchor ASN GW ID of the Anchor Data Path function at the Anchor ASN.

STEP 2

The Target BS(s) requests AK context and service authorization information for the MS by initiating a Context Retrieval procedure with the Authenticator ASN GW. Note: The Target BS(s) may optionally choose to defer this procedure to the Handover Action phase.

STEP 3

The Target BS(s) MAY initiate pre-establishment of a data path for the MS with the Anchor ASN GW. If the Anchor ASN does not support the Data Path Pre-Registration, the R6 Path_Prereg_Req message from the Target BS will be responded by the R6 Path_Prereg_Rsp message with an appropriate reject cause code. Note: The Target BS(s) may optionally choose to defer this procedure to the handover action phase.

STEP 4

The Target BS(s) sends an R8 HO_Rsp message to the Serving BS to acknowledge the handover request and starts timer TR8-HO Rsp. Upon receipt of the R8 HO_Rsp message, the Serving BS stops timer TR8-HO Req.

STEP 5

The Serving BS sends a MOB_BSHO-REQ message to the MS containing one or more potential target BS's selected by the network for the MS to handover to.

STEP 6

The Serving BS sends an R8 HO_Ack message to the Target BS(s) controlling the potential target BS(s) selected for the MS. Upon receipt of the R8 HO_Ack message, the Target BS(s) stops timer TR8-HO Rsp.

2.2.3. Handover Execution

When a MS select a target BS, and is ready for handover, the MS sends MOB_HO-IND message to previous BS (PBS) in order to notify, performs

handover. After sending MOB_HO-IND message, the MS can not receive a packet from PBS. After that, the MS performs a network entry procedure. At first, the MS performs ranging to synchronize link with a NBS, and negotiates capability with the NBS. After that, the MS registers into the NBS via authentication procedure. If the NBS receives the MSs' information about capability and authentication using backbone network already, the MS can reduce handover processes with skipping information sending. After the network entry procedure is done, the NBS can provide service to the MS [37,39-41].

If a MS moves to different network subnet, the MS should get a valid new care-of address (NCoA) again and perform additional IP connection re-confirmation procedure. Furthermore, the MS should perform a IP handover procedure like MIPv6 with NCoA in order to resume the session using in previous network.

2.3. Mobile IPv6 and Handover

2.3.1. Mobile IP

The Mobile IPv4 (MIPv4)[39,42] and the Mobile IPv6 (MIPv6)[39,43] established in MIP4 and MIP6 working group of Internet Engineering Task Force(IETF) is an international standard protocol to provide mobility.

MIPv6 provides clients' mobility based on binding between a home address(HoA) of client and newly created care-of address (CoA) in NBS. When a correspondent node (CN) can handle MIPv6, a binding is sent to the CN and optimal routing path are provided for data packets. However, because MIPv6 is a protocol about MS's location registration, data packet path re-routing, mobility for real-time application such as streaming media service and VoIP, is hard to guarantee in

MIPv6 [39,44-46].

MIPv6 Signaling and Handoff Optimization (MIPSHOP) working group in IETF established a fast mobile IPv6 (FMIPv6) protocol for fast IPv6 handover [39,47]. FMIPv6 provides fast service resuming based on estimating new location of a client supported by link layer and exchanging information for IPv6 handover and resuming service. Therefore FMIPv6 needs a correct mechanism for and event support method and an event exchange method from link layer. Recently, IEEE 802.21 working group [39,48-50] establishes a standard about interaction mechanism between link layer and upper layer, and an interaction method between FMIPv6 and WLAN specialized for IEEE 802.11 WLAN is proposed [39,46-52].

2.3.2. Fast Mobile IPv6

FMIPv6 is a mobility estimation based protocol. If a mobility estimation and handover preparation is done, FMIPv6 performs as a predictive mode. If not, FMIPv6 performs as a reactive mode [39,47].

2.3.2.1. Handover Process of Predictive Mode

When a MS detects a NBS, a MS sends a Router Solicitation for Proxy (RtSolPr) message to a previous access control router (PACR) in order to collect information of a new ACR (NACR). The PACR responses with a Proxy Router Advertisement (PrRtAdv) that contains IP address, MAC address and subnet prefix information of NACR based on RtSolPr. After that, the MS moves to the NACR using prefix information in a received PrRtAdv message, creates new IP address (NCoA), and then sends a fast binding update (FBU) message to the PACR for

binding between PCoA and NCoA. If the MS receives a fast binding acknowledgement (FBAck) message about the FBU message before handover, the MS performs predictive mode. If a FBU message is not sent or a FBAck is not received before handover, the MS performs reactive mode [39,47].

When the PACR receives a FBU message from a MS, the PACR exchanges a handover initiation (HI) message and a handover acknowledge (Hack) message with NACR. The PACR checks uniqueness of NCoA in NACR via Hack message, and then creates a tunnel between PCoA and NCoA. This uniqueness check result is sent to the MS via FBAck.

After the tunnel is created, all packets toward the PCoA are forwarded to the NCoA and the NACR saves forwarded packets into buffer. When a fast neighbor advertisement (FNA) message is received from the MS completed handover, the NACR sends all buffering packets.

2.3.2.2. Handover Process of Reactive Mode

If a FBU message is not sent or a FBAck is not received before handover, the MS performs reactive mode. The MS sends a FNA message including encapsulated the FBU to NACR. After the NACR receives FBU, the NACR checks uniqueness of NCoA inside FBU in the network. After that, the NACR sends FNA message include FBU to PACR. When the FBAck is arrived from PACR, the NACR creates a tunnel between PCoA and NCoA, and sends packets to the MS. IF the NCoA is already occupied, the NACR sends a router advertisement message with Neighbor Advertisement Acknowledgement (NAACK) option, and abandons FBU message [39,47].

2.3.3. HMIPv6

IETF establishes Fast Handovers for Mobile IPv6(FMIPv6), Hierarchical Mobile IPv6 Mobility Management (HMIPv6), and Fast Mobile IPv6 Handover Protocol using HMIPv6 (FHMIPv6) in order to reduce handover latency [47,53,54].

HMIPv6 guarantees small handover latency, because a registration process of MS is done only once in same MAP. However, HMIPv6 is not enough to support real-time application service. FHMIPv6 is a protocol to provide seamless service when micro mobility handover in same MAP occurs. However FHMIPv6 does not support macro mobility handover that is a handover between different MAPs. For macro mobility handover of fast moving MS, a fast handover supporting macro mobility handover in HMIPv6 (MMHHMIPv6) and various handover methods base on HMIPv6 are proposed [53,55,56].

2.4. Direction Prediction for Handover

Network layer mobility prediction schemes that are designed to facilitate pro-activity in the handover process also aid in resource allocation, flow control, call admission control, congestion control and QoS provisioning. Mobile Motion Prediction (MMP) algorithms used to predict future locations of a mobile user according to the user's movement history patterns was proposed [57,58]. This was one of the first of many techniques in the literature used to pro-actively connect services at the new location before the user's arrival. The MMP algorithms are based on the fact that human movement generally consists of regular and random movement. These algorithms use correlation analysis to match movement sequences

in a movement database. Results show that the MMP algorithm is highly accurate for regular movements but decreases linearly with increasing random movement.

A novel data mining approach for the prediction of user movements in mobile environments[57,59] is proposed based on the Apriori algorithm [60,61] and prediction algorithms[62,63]. This three stage prediction algorithm is to predict the mobility of a user travelling between the cells of a PCS (Personal Communication System) network. Simulation results reveal optimal prediction parameters for the PCS topology. Moderate prediction accuracy was achieved, decreasing only minimally with an increase in random movement. The authors focus primarily on prediction recall and precision results, and make no practical use of the movement predictions.

A prediction scheme based on the MMP algorithms[58] that uses actual movement traces taken from the campus-wide 802.11b wireless network, is proposed [57,64]. Data streams are duplicated and forwarded to predicted subnets resulting in a network layer handover latency that is close to a link-layer handover. Results show a reduced handover latency and packet loss rate compared to NeighborCasting.

A prediction mechanism that learns the mobility patterns of a mobile node according to an urban mobility model, is proposed[57,65]. The model attempts to capture realistic node movement in an urban environment, characterized by the MN's speed, direction, pause time and street coordinates. A weighted road selection process uses these parameters to predict the node's next hop, pre-emptively setting up tunnels and estimating tunnel activation times, consequently eliminating the need for a pre-trigger. This approach achieves 100% prediction accuracy only after 3000 seconds of movement over a small Manhattan-style street topology.

A prediction assisted fast handover protocol (PA-FMIP) based on reactive FMIPv6 and using data mining and the Apriori algorithm[66] is proposed[57]. PA-FMIP uses the predictions from the prediction algorithm to essentially replace the need for a pre-trigger. The prediction algorithm is run as an application at some time between handovers.

2.5. Media Streaming in Mobile Environments

Media streaming in mobile environments has been attracting much attention lately [67,68]. A real-time continuous media streaming protocol with special emphasis on dynamic transmission capacity allocation and prefetching is proposed [67,69]. NonStop middleware with partition prediction and service replication for continuous media streaming in mobile and ad hoc networks is developed [67,70]. Video streaming techniques in 3G mobile networks on top of a three-tier architecture is implemented [67,71]. These are only focused on the coordination of media servers for smooth handover and the rate adaptation technique when a base station becomes overloaded. There is no coordination at the base station level or the user level. Data management issues, prefetching in particular, are largely ignored.

Group mobility to predict the future availability of wireless links for increasing total streaming capacity is proposed [67,72]. An iterative algorithm to predict continuous link availability between mobile users is proposed [67,73]. The V3 architecture proposed for live video streaming is essentially a cooperative streaming architecture for moving vehicles [67,74]. The AO2P algorithm proposed for privacy routing uses a mechanism such that a receiver geographically closer to the destination is assigned to a class with a higher priority for contending the channel

to be the next hop [67,75]. The routing request in AO2P is sent to all neighboring nodes.

The work on moving objects databases is needed to maintain dynamic data items [67,76,77]. For representing and processing dynamic attributes such as locations and trajectories, spatial and temporal indexing methods are devised.

The headlight prefetching is proposed for media streaming in mobile environments [67]. It has headlight prefetching zone that is a virtual fan-shaped area along the direction of user movement similar to the headlight of a vehicle. All service access pointers (SAP) of the cells that overlapped with the zone are selected as the prefetching SAPs.

3. Prediction of Mobile Client's Movements

3.1. Grouping and Characteristics Analysis of Mobile Client's Movements

To provide user requested QoS guaranteed streaming media service on Mobile WiMAX, we collect and analyse characteristics of streaming media service users' mobility. Table 4-1 shows grouping of streaming media service users depending on mobility characteristics. Mobile clients on foot has limited mobility. Since they move in 4 Km/h, handovers are rarely occurred and previous methods are enough to provide stable QoS guaranteed streaming media service.

Full mobility clients are in buses, vehicles, and subway. This group can be divided into low chance of estimate movement group, Group B, and mid chance of estimate movement, Group C. Streaming media service users in buses or vehicles

in city area are in Group B. Since many intersections, junctions, and traffic signals are operated in city area, mobile clients' speed and direction are frequently changed. Therefore, clients in Group B need frequent handover, and direction and speed are hard to predict. Group C includes streaming media service users in subway or vehicles in high way. Because users in Group C move following subway lines or high way path, speed and direction can be predicted easier than Group B. However, getting off subway in stations or using out ramp in high way are hard to predict. Since vehicles in high way move very fast, handover are needed very frequently.

Table 4-1. Category for Mobile Clients.

<table>
<tr><th>Mobility</th><th colspan="2">Group</th><th>Probability of estimation</th><th>Characteristic</th><th>Policy</th><th>Example</th></tr>
<tr><td>Limited Mobility</td><td colspan="2">A</td><td>Random</td><td>Impossible to estimate (Random)
Low possibility of handover</td><td>Legacy policy</td><td>Pedestrians</td></tr>
<tr><td rowspan="3">Full Mobility</td><td colspan="2">B</td><td>Low Chance of Estimate Movement</td><td>Possible to estimate
High possibility of handover</td><td>Car movement estimation based on intersection possibility</td><td>Users in vehicle at city area</td></tr>
<tr><td rowspan="2">C</td><td>1</td><td rowspan="2">Mid Chance of Estimate Movement</td><td rowspan="2">Semi-fixed route
High possibility of handover</td><td rowspan="2">Hand-off scheduling based on routes and exception handling</td><td>Users in subway</td></tr>
<tr><td>2</td><td>Users in vehicle at highway</td></tr>
<tr><td>High Mobility</td><td colspan="2">D</td><td>High Chance of Estimate Movement</td><td>Fixed route
High possibility of handover</td><td>Hand-off scheduling based on routes</td><td>Users in train at railroad</td></tr>
</table>

High mobility clients, Group D, are streaming media service users in high speed moving trains. Users in Group D move fast and following route of trains. Since distance between stations are long, variation of speed and direction is small. In station, clients get off train less frequent than subway. However, since users in Group D have the fastest speed, handovers are occurred very frequently. Therefore, it is hard to provide stable QoS streaming media service.

3.2. Prediction of Mobile Client's Direction

In previous section, groups of streaming media service users are shown. In this section, a prediction method of clients' direction and speed is proposed.

3.2.1. Group A

The Group A is low mobility users using streaming media service. Speeds of this group users are less than 4 Km/h, directions are randomly decided by users. Therefore, handover probability of Group A is very low. In Mobile WiMAX, 3Mbps bandwidth is provided normally, and 2Mbps bandwidth is provided under 10Km/h movements [31,32,38]. Therefore, since Group A has enough bandwidth and low handover probability, legacy methods such as PA-FMIP[57], in Mobile WiMAX can provide stable QoS streaming media service.

3.2.2. Group B

Group B is full mobility users using streaming media service in buses or vehicles in city area. According to Korea Transport Database [78], average vehicles'

speed in Korean cities is approximately 35 Km/h and maximum is 60Km/h. We can get average speed and maximum speed of each intersection or junction at each time from Korea Transport Database. Therefore, speed and direction probability of client are predicted using Korea Transport Database.

However, these method predicts only probability of whole vehicles for directions. It is not able to reflect each client's characteristics. Moving speed of a client is almost same as whole vehicles flow, but moving direction is different individually. When user A uses streaming media service in office-going hour by buses, we can predict that movement of user A is always same. Therefore, individual client movement history should be considered to predict client's direction.

Equation 4-1 shows how to predict direction of clients in Group B in location ***a*** at time ***t***.

$$Ptotal_i(l,t) = \frac{No.\ of\ vehicles\ toward\ i_{th}\ direction\ at\ l}{No.\ of\ vehicles\ at\ l}$$

$$Pindv_i(l,t) = \frac{No.\ of\ moves\ toward\ i_{th}\ direction\ at\ l}{No.\ of\ visits\ at\ l}$$

$$P_i(l,t) = Ptotal_i(l,t) \times 0.2 + Pindv_i(l,t) \times 0.8 \qquad \text{Equation 4-1}$$

$$Maximum(P_1(l,t), P_2(l,t), \cdots, P_n(l,t))$$

$$(l = location,\ \ t = time,\ \ i = 1,2,\cdots,n)$$

3.2.3. Group C

Group C is full mobility users using streaming media service in subway or vehicles in high way. Users in Group C are divided into users in subway, Group C-1, and users in vehicles in high way, Group C-2.

Group C-1 is user group using streaming media service in subway. Subway is operated in subway line and operating time and stop time are always same. According to information site of Korean Railroad Cooperation (Korail) [79], average speed of Seoul metro subway is 50Km/h and maximum speed is up to 100Km/h. Because distance between stations is short and subway is stopped frequently at station. Since entry points and exit points are automated, we can collect how many passengers enter or out in a station at specific time. Equation 4-2 shows how to predict direction of clients in Group C-1 in station ***s*** at time ***t***.

$$Ptotal_{on}(l,t) = \frac{No.\ of\ passengers\ stay\ on\ subway\ at\ s}{No.\ of\ passenger\ at\ a}$$

$$Pindv_{on}(l,t) = \frac{No.\ of\ staying\ on\ subway\ at\ s}{No.\ of\ visiting\ at\ s}$$

$$Ptotal_{off}(l,t) = \frac{No.\ of\ passengers\ get\ off\ subway\ at\ s}{No.\ of\ passenger\ at\ a}$$

$$Pindv_{off}(l,t) = \frac{No.\ of\ getting\ off\ subway\ at\ s}{No.\ of\ visiting\ at\ s}$$

Equation 4-2

$$P_i(l,t) = Ptotal_i(l,t) \times 0.2 + Pindv_i(l,t) \times 0.8$$

$$Maximum(P_{on}(l,t), P_{off}(l,t))$$

$$(s = station,\ \ t = time, i = on \text{ or } off)$$

Group C-2 is streaming media service user group in vehicles in highway. The speed limit of Korean highway is 100Km/h or 110Km/h. According to Korean Highway Traffic Information site [80], usually 100Km/h is average speed in highway. Since clients in Group C-2 move fast, a handover direction prediction is important to provide stable QoS guaranteed streaming media service. Because traffic information of Korean highway is provided in Korean Highway Traffic Information

site, we can collect client's speed in highway. Since, movements of vehicles in highway are following highway path, we can predict client direction except out ramps, interchange areas, and rest places. However, because speed limits in out ramps, interchange areas, and rest places are less than 50Km/h, movements change to out ramps, interchange areas, and rest places can be predicted from moving speed. Furthermore, individual client's history should be considered like Group B, and Group C-1. Equation 4-3 shows how to predict direction of clients in Group C-2 in location ***l*** at time ***t***.

$$Ptotal_i(l,t) = \frac{No.\ of\ vehicles\ toward\ i_{th}\ direction\ at\ l}{No.\ of\ vehicles\ at\ l}$$

$$Pindv_i(l,t) = \frac{No.\ of\ moves\ toward\ i_{th}\ direction\ at\ l}{No.\ of\ visits\ at\ l}$$

$$P_i(l,t) = Ptotal_i(l,t) \times 0.2 + Pindv_i(l,t) \times 0.8 \qquad \text{Equation 4-3}$$

$$Maximum(P_1(l,t), P_2(l,t), \cdots, P_n(l,t))$$

$$(l = location,\ \ t = time,\ \ i = 1,2,\cdots,n)$$

3.2.4. Group D

Group D is streaming media service user group in high speed train. According to Korail Information Site [79], maximum speed and average speed of KTX are 300Km/h and 165Km/h individually, and maximum speed and average speed of MuGungHwa train are 120Km/h and 8~90Km/h individually. Since users in Group D are similar to Group C-1 but moving speed is much faster, handover probability is higher than Group C-1. However, because distances between stations are longer than subway, moving direction is predictable. Equation 4-4 shows how to predict

direction of clients in Group D in station ***s*** at time ***t***.

$$Ptotal_{on}(l,t) = \frac{No.\,of\,passengers\;stay\,on\,train\,at\,s}{No.\,of\,passenger\,at\,a}$$

$$Pindv_{on}(l,t) = \frac{No.\,of\,staying\,on\,train\;at\;s}{No.\,of\,visiting\,at\;s}$$

$$Ptotal_{off}(l,t) = \frac{No.\,of\,passengers\;get\;off\,train\,at\,s}{No.\,of\,passenger\,at\,a}$$

Equation 4-4

$$Pindv_{off}(l,t) = \frac{No.\,of\,getting\,off\;train\;at\;s}{No.\,of\,visiting\,at\;s}$$

$$P_i(l,t) = Ptotal_i(l,t) \times 0.2 + Pindv_i(l,t) \times 0.8$$

$$Maximum(P_{on}(l,t), P_{off}(l,t))$$

$$(s = station,\;\; t = time, i = on \text{ or } off)$$

3.3. Group Changing based on Mobility

In previous section, how to predict client's direction and speed of each group is explained. Clients in each group can be predicted moving direction and speed from equations of each group. However, clients are in only one group. For example, streaming madia service user B in subway, Group C-1. And then user B get off subway, Group A, and walks to bus stop. After that, user B gets on bus, Group B. In this case, user B should be able to change group (Group C-1 -> Group A -> Group B). In this section, a group changing method is proposed and explained.

A user in Group A can change to Group B, when the user gets on vehicles in city area. When the user uses subway or train, the user change to Group C-2 or Group D. A user in rest place of highway can change to C-2, when the user gets

on vehicle. Therefore, users in Group A can change to Group B, C-1, C-2, and D.

Users in Group B can change to Group A when users get off vehicle, and to Group C-2 when vehicle enters highway. Users in Group C-1 can change to only Group A, when users get off subway. Users in Group C-2 can change to Group A when users get off vehicle, and to Group B when vehicle enter city area via out ramp. Users in Group D can change to only Group A like Group C-1 users, when users get off. Figure 4-3 show group changing states of mobile clients as mentioned above.

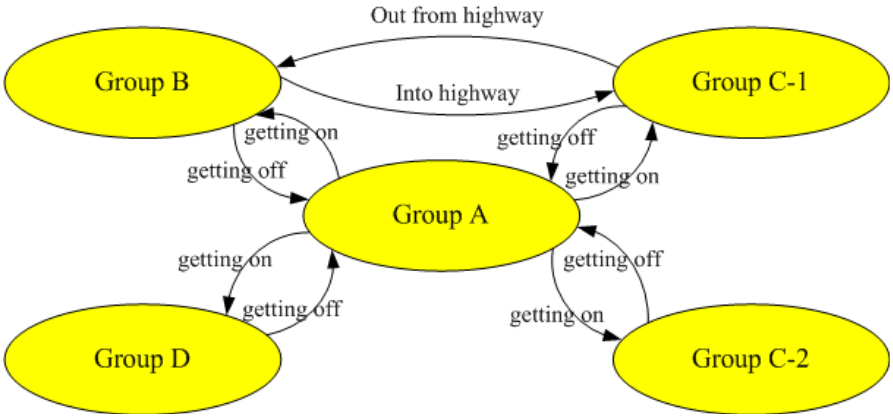

Figure 4-3. State Diagram of Mobile Clients Group.

4. Prefetching Method for Handover

Streaming media service for fast moving clients is hard to guarantee stable QoS. In each cell boundary, signal strength is weaker and bandwidth is lower than center area of cell. Therefore, mobile clients in cell boundary is possible to have low level of buffer. If handover is needed when buffer level is low, streaming media service could be stopped and waited until than buffer level is enough to resume

streaming media service.

In previous section, a detection method of moving direction and speed in each group is proposed. In this section, a prefetching method is proposed to keep stable streaming media service buffer state based on proposed detection method.

4.1. Prefetching before Handover

Since bandwidth reduction and no transmission are occurred when a MS moves from center area of cell to cell boundary, stream data for period until handover finished is pre-sent to the MS to prevent low level of buffer. Using proposed prediction method, time period from now to handover is estimated before that the MS enters cell boundary which is bandwidth reduction area. Based on time period, stream data amount can be calculated for prefetching. A streaming server sends calculated prefetching stream data to the MS when network bandwidth is stable. Equation 4-5 shows how to estimate prefetching stream data size. ***Distance(m)*** can be calculated with current speed and location of the MS and a radius of PBS cell.

$$SizeOfBefore(Kb) = \frac{Distance(m)}{\frac{Speed(Km/h)}{3600\text{sec}} \times 1000m} \times bitrate(Kbps)$$

$$Distance(m) = moving\ distance\ until\ Handover$$

Equation 4-5

4.2. Prefetching after Handover

After a handover is done, a MS is still in cell boundary of NBS. Therefore, the MS is provided lower bandwidth than center area of cell. Since the MS requests media stream data to server in low bandwidth area after the handover, media

stream data might be delayed to the MS. It causes low buffer level of the MS. To solve this problem, media streaming server sends stream data to the NBS before the handover to prevent low buffer lever of the MS after the handover. After the handover, the MS can receive stream data from NBA immediately to reduce delay of stream data retrieval. The amount of stream data can be calculated with time period from the handover finish time to the time of enter the NBS center area as shown as Equation 4-6. ***Distance(m)*** can be calculated with current speed and location of the MS and a radius of NBS cell.

$$SizeOfAfter(Kb) = \frac{Distance(m)}{\frac{Speed(Km/h)}{3600\text{sec}} \times 1000m} \times bitrate(Kbps) \qquad \text{Equation 4-6}$$

$$Distance(m) = moving\ distance\ into\ NBS\ core\ area\ after\ Handover$$

4.3. Prediction Failure Recovery

When a MS does not perform a handover to predicted cell, prefetched stream data is not useful anymore for the MS and the MS cannot receive prefetched stram data, because prefetched stream data is not located in NBS for handover. It causes low buffer level of the MS. However, the MS can be notified NBS before the handover in network initiated handover of Mobile WiMAX. Therefore, if FMIPv6 or HMIPv6 is used for handover, the MS knows NBS. When prefetching BS is not NBS, the PBS can send forwarding request to prefetching BS. When prefetching BS receives forwarding request, prefetching BS sends prefetched stream data to the NBS. Since the NBS is near prefetching BS and both are connected with back bone network, forwarding data is done shortly. Therefore, prefetched stream data can be forwarded before the handover is done, because the handover takes a few time.

5. Experimental Results and Analysis

5.1. Experimental Environment

We simulates proposed methods for buffer state of a high speed moving MS in streaming media service. A simulation program is developed with c language base on linux system. We assumes that buffer size of a MS is streaming data for 10 seconds, a cell radius is 1Km, a cell boundary is started from 0.8 of cell radius, a prefetching starting point is 0.7 of cell radius, and a handover is done at 0.9 of cell radius.

Table 4-2. Bandwidth variation according to client's speed.

	Speed	Bandwidth
High Mobility	over 120Km/h	under 144Kbps
Full Mobility	under 120Km/h	over 384Kbps
General Mobility	under 60Km/h	over 512Kbps
Limited Mobility	under 10Km/h	over 2Mbps

Table 4-3. Minimum bandwidth and average speed of each transportation.

	Group	Average Speed	Minimum Bandwidth
KTX	Group D	165Km/h	144Kbps
SaMaUl Train	Group D	110Km/h	384Kbps
Vehicles in Highway	Group C-1	100Km/h	384Kbps
MuGungHwa Train	Group D	70Km/h	384Kbps
Vehicles in City (Limited Speed)	Group B	60Km/h	512Kbps
Subway	Group C-2	50Km/h	512Kbps
Vehicles in City (Average Speed)	Group B	30Km/h	512Kbps
Pedestrian	Group A	4Km/h	2048Kbps

Table 4-4. Media bit-rate of experiment.

Media	1	2	3	4	5
bit-rate	56Kbps	200Kbps	400Kbps	800Kbps	1024Kbps

Table 4-2 shows bandwidth variation according to client's speed. Average speeds and minimum bandwidths of experimental mobility are shown as Table 4-3. Bit-rates of experimental media data are in Table 4-4.

5.2. Buffer State Analysis

Figure 4-4, 4-5, 4-6, 4-7, 4-8, 4-9, and 4-10 show buffer status of a MS using streaming media service in a KTX, a SaMaUl train, a vehicle in highway, a MuGungHwa train, a vehicle in uncongested city area, a subway, and a vehicle in congested city area respectively.

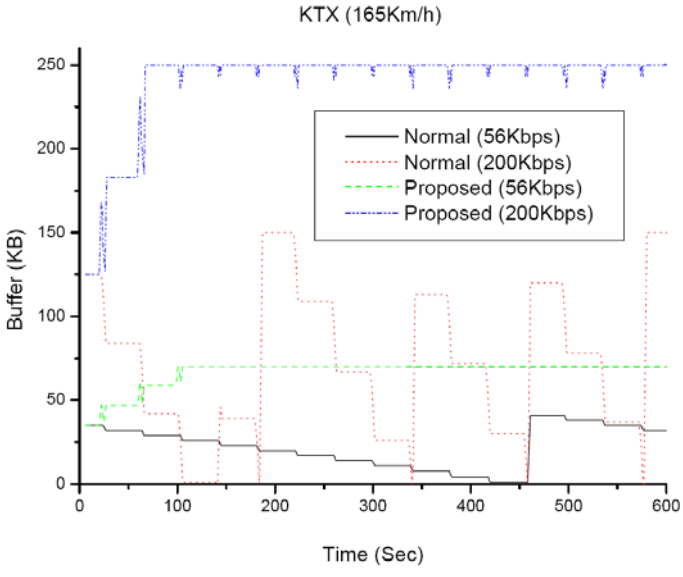

Figure 4-4. Buffer Status of Mobile Client in KTX.

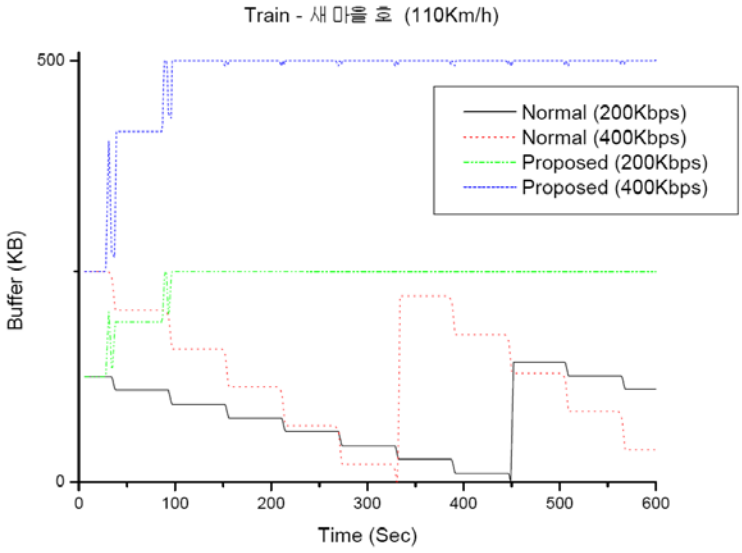

Figure 4-5. Buffer Status of Mobile Client in SeMaUl Train.

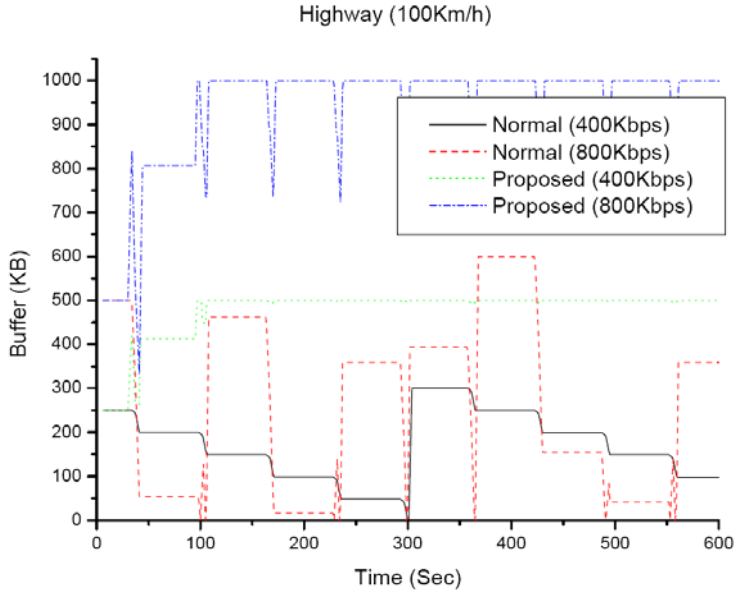

Figure 4-6. Buffer Status of Mobile Client in Vehicle on Highway.

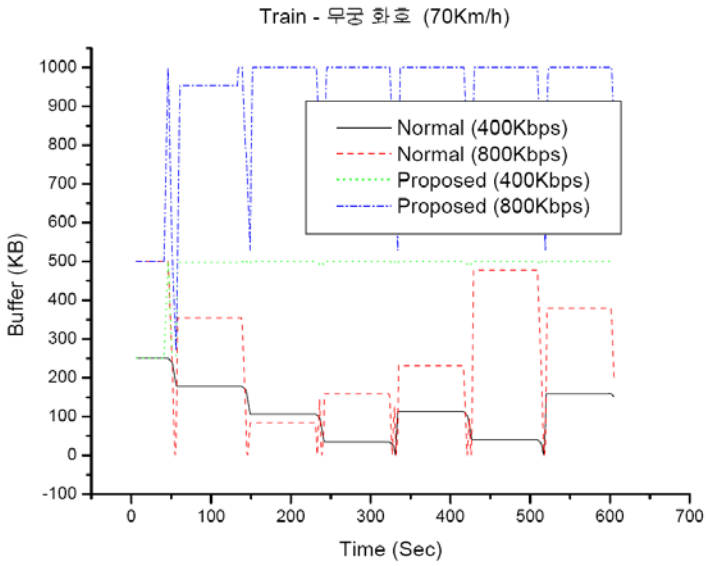

Figure 4-7. Buffer Status of Mobile Client in MuGungHwa Train.

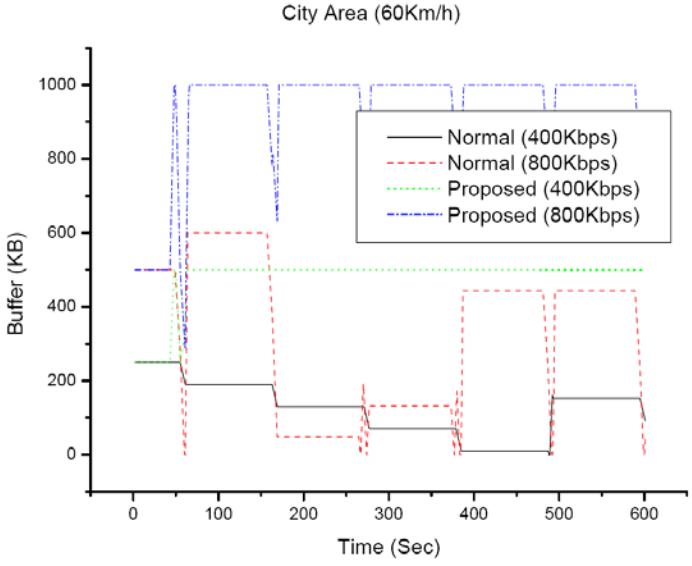

Figure 4-8. Buffer Status of Mobile Client in Vehicle on City Area.

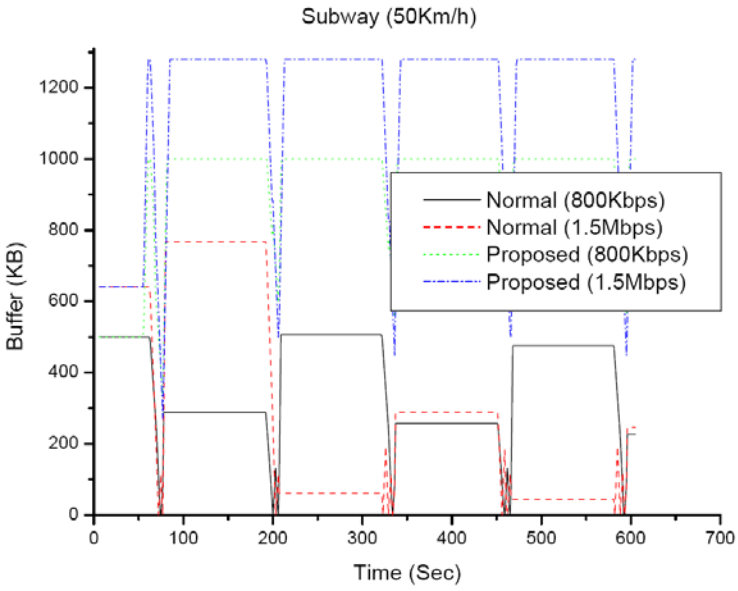

Figure 4-9. Buffer Status of Mobile Client in Subway.

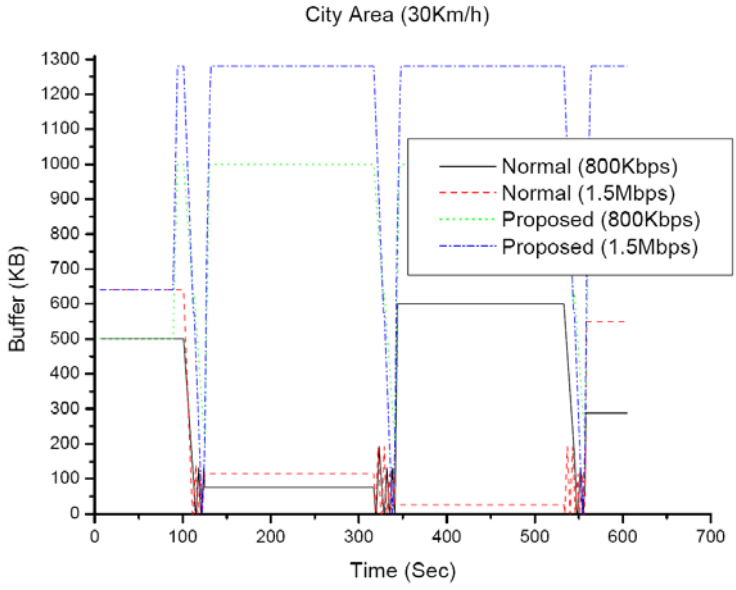

Figure 4-10. Buffer Status of Mobile Client in Vehicle on Congested City Area.

Figure 4-4 shows buffer status of a MS using 56Kbps and 200Kbps streaming media service in a KTX. A normal streaming media service with 56Kbps is stopped in around 450 seconds and a normal streaming media service with 200Kbps is stoped several times. However, a proposed method with 56Kbps and 200Kbps supports a seamless streaming media service without any problem and re-buffering. Figure 4-5 show buffer status of a MS using 200Kbps and 400Kbps streaming media service in a SeMaUl train. As same as figure 4-4, a proposed method provides stable buffer status of the MS.

Figure 4-6, 4-7, and 4-8 show buffer statuses of a MS using 400Kbps and 800Kbps streaming media service in a vehicle in highway, a MuGungHwa train, and a vehicle in uncongested city area, respectively. We can see that the proposed method provides stable buffer status and seamless streaming media service. Figure 4-9 and 4-10 show buffer statuses of a MS using 800Kbps and 1.5Mbps streaming media service in a subway and a vehicle in congested city area, respectively. We confirm that the proposed method provides stable buffer status and seamless streaming media service. When a pedestrian uses streaming media service, a buffer status of a MS is stable in every case, because a MS can be provided stable 2Mbps bandwidth.

6. Summary

As a result of improvements in mobile wireless internet technologies, high speed moving users are able to access a streaming media service using PDAs, laptops, mobile phones and so on. However, mobile wireless networks have variable bandwidths depending on the speed and location of clients. These characteristics

make it hard to support stable Quality of Service (QoS) streams for high speed mobile clients.

In this chapter, a client mobility-based media stream prefetching method to guarantee stable QoS in high speed internet environment like a Mobile WiMAX, was proposed. In proposed method, high speed moving clients is joint a group depending on their characteristics and provided streaming media service according to a joint group. Mobile clients can change their group depending on their situation. In each group, clients' direction and speed are predicted. Based on client grouping and prediction of client's direction and speed, media streams are prefetched against disconnect and latency caused by handover. From the experimental results, we confirm that the proposed method provides stable buffer status and seamless streaming media service in each group with various media bit-rates.

Chapter V

Conclusion and Future Work

In this dissertation, three issues in streaming media service for mobile clients was studied for a integrated transcoding media streaming service system. First issue was how to guarantee quality of service (QoS) over wireless network. Second issue was hot to guarantee QoS for various clients. Last issue was how to guarantee QoS for fast moving clients.

For first issue, the Network Adaptive Autonomic Transcoding Algorithm (NAATA) was proposed to support streaming media service for mobile clients. The proposed algorithm decides on target transcoding bit-rates in real-time according to the wireless network state. Since it protects continuous transmission failures for streaming media data, seamless and stable streaming media services are provided for mobile clients. In our experiments, the NAATA reduced transmission failures of 80% and 40% when compared with the legacy transcoding system and the ANAT system, respectively. We also established that the NAATA has less overhead and long time intervals between transmission failures. Based on these advantages, we confirmed that the NAATA provides a more seamless streaming media service for mobile clients, without jittering or ceasing phenomena.

For second issue, a new load distribution method was proposed for fair transcoding load distribution in the distributed transcoding servers. The proposed method controls and distributes transcoding requests based on transcoding time estimation for distributed transcoding environments. In our experimental distributed

transcoding servers, we evaluated the scalable performance of the RR, WRR, RWLD and TELD methods. Because the RR and WRR method did not consider the characteristic of transcoding jobs, it cannot satisfy the QoS requirement. Therefore, because some specific servers suffer from heavy transcoding load, it did not show good performance scalability. The RWLD method considers only three kinds of transcoding media grades, which are not enough to estimate the transcoding times for various mobile devices. As a result, even if the number of transcoding servers is increased, the RWLD should be able to limit the performance scalability. The TELD method distributes transcoding jobs based on the estimated transcoding time according to the characteristic of target media and transcoding servers. As a result, we confirmed that the proposed TELD method provided more linear performance scalability than other load distribution.

For third issue, a client mobility-based media stream prefetching method to guarantee stable QoS in high speed internet environment was proposed. In proposed method, high speed moving clients was joint a group depending on their characteristics and provided streaming media service according to a joint group. Mobile clients could change their group depending on their situation. In each group, clients' direction was predicted, and media streams were prefetched against disconnect and latency caused by handover. The proposed method was experimented and evaluated that buffer state was stable with handover for streaming media service.

In the future work, we plan to evaluate the client mobility-based media stream prefetching method in real environments. Also we plan to develop a integrated transcoding media streaming service system.

References

[1] Dinkar Sitaram, Asit Dan, Multimedia Servers: Applications, Environments, and Design, Morgan Kaufmann Publishers, 2000.

[2] Wu-chi Feng and Ming Liu, "Critical Bandwidth Allocation Techniques for Stored Video Delivery Across Best-Effort Networks," Proc. The 20th Intl. Conf. on Distributed Computing Systems, pp. 201-207, Apr. 2000.

[3] David H.C. Du and Yen-Jen Lee, "Scalable Server and Storage Architectures for Video Streaming," Proc. IEEE Intl. Conf. on Multimedia Computing and Systems, pp. 191-206, Jun. 1999.

[4] Florin Lahan, Irek Defee, Marius Vlad, Aurelian Pop, Prakash Sastry, "Integrated system for multimedia delivery over broadband ip networks," IEEE Transactions on Consumer Electronics, Vol. 48, No.3, pp.564~565, 2002.

[5] C. Li, G. Peng, K. Gopalan, and T. Chiueh, "Performance guarantees for cluster-based internet services", Proceedings of the 23rd International Conference on Distributed Computing Systems, pp378- 385, May 2003

[6] Sumit Roy, Michele Covell, John Ankcorn, Susie Wee, Takeshi Yoshimura, "A System Architecture for Managing Mobile Streaming Media Services," 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), pp.408-419, 2003.

[7] J. Guo, F. Chen, L. Bhuyan, and R. Kumar, "A cluster-based active router architecture supporting video/audio stream transcoding services", Proceedings of the 17th International Parallel and Distributed Processing Symposium, pp. 446- 453, Apr. 2003.

[8] S. Chandra, A. Gehani, C. S. Ellis, and A. Vahdat, "Transcoding characteristics of web images", Proceedings of the SPIE/ACM Multimedia Computing and Networking (MMCN2001), Jan. 2001.

[9] Dongmahn Seo, Joahyoung Lee, Yoon Kim, Changyeol Choi, Hwangkyu Choi, Inbum Jung, "Load Distribution Strategies in Cluster-based Transcoding Servers for

Mobile Clients," Lecture Notes in Computer Science, Vol. 3983, pp. 1156-1165, May 2006.

[10] Dongmahn Seo, Joahyoung Lee, Yoon Kim, Changyeol Choi, Manbae Kim, Inbum Jung, "Resource Consumption-Aware QoS in Cluster-based VOD Servers," Journal of Systems Architecture: the EUROMICRO Journal, Vol. 53 , Issue 1, pp. 39-52, Jan. 2007.

[11] Harini Bhradvaj, Anupam Joshi, Sansanee Auephanwiriyakul, "An Active Transcoding Proxy To Support Mobile Web Access," Proc. Intl. Conf. on Reliable Distributed System, pp 118-123, 1998.

[12] Anthony Vetro, Huifang Sun, "Media Conversions to Support Mobile Users," Proc. IEEE Canadian Conf. on Electrical and Computer Engineering, pp. 607-612, May. 2001.

[13] Behrouz A. Forouzan, Data Communications and Networking 2nd, Mc Graw Hill, 2001.

[14] Surendar Chandra, Carla Schlatter Ellis and Amin Vahdat, "Differentiated Multimedia Web Services Using Quality Aware Transcoding", Proc. IEEE INFOCOMM 2000, pp.961-969, 2000,

[15] Susie Wee, John Apostolopoulos, Wai-tian Tan, Sumit Roy, "Research and Design of a Mobile Streaming Media Content Delivery Network," Proc. IEEE ICME, July 2003, pp. I-5-8, 2003.

[16] Jacob Chakareski, Pascal Frossard, "Adaptive Systems for Improved Media Streaming Experience", IEEE Communications Magazine, Vol. 45, Issue 1, pp. 77-83, 2007.

[17] P. A. Chou, Zhourong Miao, "Rate-Distortion Optimized Streaming of Packetized Media", IEEE Trans. Multimedia, Vol. 8, No. 2, pp. 390-404, 2006.

[18] Jitendra Padhye, Jim Kurose, Don Towsley, Rajeev Koodli, "A Model Based TCP-Friendly Rate Control Protocol", NOSSDAV99, 1999.

[19] Seongho Cho, Heekyoung Woo, Jong-won Lee, "ATFRC: Adaptive TCP Friendly Rate Control Protocol", Lecture Notes in Computer Science, Vol. 2662, pp. 171-180, 2003.

[20] Eddie Kohler, Mark Handley, Sally Floyd, "Designing DCCP: Congestion Control Without Reliability", SIGCOMM'06, pp. 27-38, 2006.

[21] Ningning Hu, Peter Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques", IEEE JSAC Special Issue in Internet and WWW

measurement, Mapping and Modeling, Vol. 21(6), pp. 879-894, 2003.

[22] James F. Kurose, Keith W. Ross, Computer Networking - A Top-down Approach Featuring the Internet, Addison Wesley, 2004.

[23] Ningning Hu, Peter Steenkiste, "Estimating Available Bandwidth Using Packet Pair Probing", Technical Report CMU-CS-02-116, 2002.

[24] http://ffmpeg.org/, ffmpeg open source project homepage.

[25] http://www.mplayerhq.hu/, mplayer open source project homepage.

[26] Jiani Guo, Laxmi Narayan Bhuyan, "Load Balancing in a Cluster-Based Web Server for Multimedia Applications," IEEE Transaction on Parallel and Distributed Systems, Vol. 17, No. 11, pp. 1321-1334, Nov. 2006.

[27] Dongmahn Seo, Nansook Heo, Jongwoo Kim, Inbum Jung, "Transcoding Load Estimation Method for Load Balance on Distribution Transcoding Environments," Journal of KIISE : Computer Systems and Theory, Vol. 35, No. 10, pp. 466-475, Oct. 2008.

[28] http://www.mpeg.org/, mpeg homepage.

[29] "2008 Survey on the Wireless Internet Use," National Internet Development Agency of Korea, Dec. 2008.

[30] Brian K. Schmidt, Monica S. Lam, K. Duane Northcutt, "The Interactive Performance of SLIM: A Stateless, Thin-client Architecture," ACP SOSP' 99, pp.31-37, 1999.

[31] Monica Paolini, "Testing WiMAX Performance in the Clear Network in Portland", WiMAX Forum, Jan. 2009.

[32] IEEE 802.16 WG, "IEEE Standard for Local and Metropolitan Area Networks part 16", IEEE 802.16 Standard, Dec. 2005.

[33] WiMAX Forum, "WiMAX Forum Network Architecture - Architecture Tenets, Reference Model and Reference Points Base Specification", DRAFT-T32-001-R015v01-O, Sep. 2009.

[34] Sundan Bose, Arputharaj Kannan, "Adaptive Multipath Multimedia Streaming Architecture for Mobile Networks with Proactive Buffering Using Mobile Proxies", Journal of Computing and Information Technology, Vol. 15, pp.215-226, Mar. 2007.

[35] Minsik Shim, Hwasung Kim, "A Handover Mechanism for Preventing Out-of Sequence Problems in WiBro", Proc. of the KIISE Korea Computer Congress 2006, Vol. 35, No. 1(D), pp.184-186, 2006.

[36] ETRI, "The HPi Handover Specification", ETRI, 2003.

[37] TTA, "Specifications for 2.3GHz band Portable Internet Service Physical & Medium Access Control Layer", TTA Standard, TTAS.KO-06.0082, Jun. 2005.

[38] WiMAX Forum, "WiMAX Forum Network Architecture - Stage 3 - Annex: R6/R8 Anchored Mobility Scenarios", WMF-T33-0030R010v04, Feb. 2009.

[39] Hee-Jin Jang, Youn-Hee Han, Seung-Hee Hwang, "A Cross-layering Handover Scheme for IPv6 Mobile Station over WiBro Networks", Journal of KIISE : Information Networking, Vol. 34, No. 1, pp.48-61, Feb. 2007.

[40] IEEE 802.16 TGe WG draft, "Amendment for Physical and Medium Access Control Layers for Combined fixed and Mobile Operation in Licensed Bands", 802.16e/D9, Jun. 2005.

[41] WiMAX Network WG document, "End-to-End Network Systems Architecture", (Stage 2: Architecture Tenets, Reference Model and 12 Reference Points), Sep. 2005.

[42] Charles E. Perkins, "IP Mobility Support for IPv4". RFC 3344, Aug. 2002.

[43] David B. Johnson, Charles E. Perkins, Jari Arkko, "Mobility Support in IPv6", RFC 3775, 2004.

[44] Nicolas Montavont, Thomas Noel, "Handover Management for Mobile Nodes in IPv6 Networks", IEEE Communications Magazine, Vol. 40, No. 8, pp.38-43, Aug. 2002.

[45] Hesham Soliman, Mobile IPv6 : Mobility in a Wireless Internet, Addison Wesley, 2004.

[46] Lila Dimopoulou, Georgios Leoleis, Iakovos S. Venieris, "Fast Handover Support in a WLAN Environment: Challenges and Perspectives", IEEE Network, Vol. 19, No. 3, pp.14-20, Jun. 2005.

[47] Rajeev Koodli, "Fast Handovers for Mobile IPv6", RFC 4068, Jul. 2005.

[48] IEEE 802.21 WG document, IEEE Standard for Local and Metropolitan Area Networks: Media Independent Handover Services, P802.21/D00.01, Jul. 2005.

[49] Vivek G. Gupta, David Johnston, "IEEE 802.21, A Generalized Model for Link Layer Triggers", IEEE 802.21 WG, Mar. 2004.

[50] Xiaoyu Liu, Youn-Hee Han, "Interaction between L2 and Upper Layers in IEEE 802.21", IEEE 802.21 WG, Mar. 2004.

[51] Pete McCann, "Mobile IPv6 Fast Handovers for 802.11 Networks", RFC 4260, Nov. 2005.

[52] Fumio Teraoka, Kazutaka Gogo, Kochiro Mitsuya, Rie Shibui, Koki Mitani, "Unified Layer 2 (L2) Abstractions for Layer 3 (L3)-Driven Fast Handover". RFC 5184, May. 2008.

[53] Gyodu Koo, Youngsong Mun, "Improved Fast Handover Protocol using HMIPv6 based on IEEE 802,16e Network", The KIPS Transactions : Part C, Vol. 14, No. 6, pp.503-508, Oct. 2007.

[54] Hesham Soliman, Claude Castelluccia, Karim El Malki, Ludovic Bellier, "Hierarchical Mobile IPv6 Mobility Management", RFC 4140, Aug. 2005.

[55] KyoungHye Lee, Youngson Mun, "An Efficient Macro Mobility Scheme Supporting Fast Handover in Hierarchical Mobile IPv6", Lecture Notes in Computer Science, Vol. 3480, pp.408-417, May. 2005.

[56] Indra Vivaldi, Mohd Hadi Habaebit, Borhanuddin Mohd Ali, V. Prakash, "Fast Handover Algorithm for Hierarchical Mobile IPv6 Macro-Mobility Management", The 9th Asia-Pacific Conference on Communications, Vol. 2, pp.630-634, Sep. 2003.

[57] Andre E. Bergh, Neco Ventura, "PA-FMIP: a Mobility Prediction Assisted Fast Handover Protocol", IEEE Military Communications Conference, pp.1-7, Oct. 2006.

[58] George Lui, Gerald Maguire Jr., "A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communications", Mobile Networks and Applications, Vol 1, No. 2, pages 113-121, Jun. 1996.

[59] Gokhan Yavas, Dimitrios Katsaros, Ozgur Ulusoy, Yannis Manolopoulos, "A data mining approach for location prediction in mobile environments", Data and Knowledge Engineering, Vol. 54, No. 2, pp.121-146, Aug. 2005.

[60] Rakesh Agrawal, Ramakrishnan Srikant, "Fast Algorithms for Mining Association Rules. Proc. 20th. Conf Very Large Data Bases", Proceeding of 20th Internationl Conference on Very Large Data Bases, pp487-499, Sept 1994.

[61] Rakesh Agrawal, Ramakrishnan Srikant, "Mining Sequential Patterns", Proceeding of IEEE Conference on Data Engineering, pp.3-14, Mar. 1995.

[62] Alexandros Nanopoulos, Dimitrios Katsaros, Yannis Manolopoulos, "Effective Prediction of Web-user Accesses: A Data Mining Approach", Proceeding of the WebKDD Workshop, 2001.

[63] Alexandros Nanopoulos, Dimitrios Katsaros, Yannis Manolopoulos, "A data mining algorithm for generalized web prefetching", IEEE Transaction on Knowledge Data Engineering, Vol.15, No. 5, pp.1155-1169, Oct. 2005.

[64] Fang Feng, Douglas S. Reeves, "Explicit Proactive Handoff with Motion Prediction for Mobile IP", Proceeding of the Wireless Communications and Networking Conference, Vol. 2, pp.855-860, Mar. 2004.

[65] N. Van den Wijngaert, C. Blondia, "A Predictive Low Latency Handover Scheme for Mobile IP" Proceeding of ICMU'05, Apr. 2005.

[66] Ferenc Bodon, "A Trie-based APRIORI Implementation for Mining Frequent Item sequences", Proceeding of OSDM'05, pp.56-65, Aug. 2005.

[67] Shiow-yang Wu, Jungchu Hsu, Chieh-Ming Chen, "Headlight Prefetching and Dynamic Chaining for Cooperative Media Streaming in Mobile Environments", IEEE Transaction on Mobile Computing, Vol. 8, No. 2, pp.173-187, Feb. 2009.

[68] Minoru Etoh, Takeshi Yoshimura, “Wireless Video Applications in 3G and Beyond”, IEEE Wireless Communications, Vol. 12, No. 4, pp. 66-72, Aug. 2005.

[69] Frank H.P. Fitzek, Martin Reisslein, “A Prefetching Protocol for Continuous Media Streaming in Wireless Environments”, IEEE Journal on Selected Areas in Communications, Vol. 19, No. 10, pp.2015-2028, Oct. 2001.

[70] Baochun Li and Karen H. Wang, “NonStop: Continuous Multimedia Streaming in Wireless Ad Hoc Networks with Node Mobility”, IEEE Journal on Selected Areas in Communications, Vol. 21, No. 10, pp.1627-1641, Dec. 2003.

[71] Anna Kyriakidou, Nikos Karelos, Alex Delis, “Video-Streaming for Fast Moving Users in 3G Mobile Networks”, Proceeding of Fourth International Workshop on Data Engineering for Wireless and Mobile Access, pp. 65-72, 2005.

[72] Guang-Tao Xue, Zhao-Qing Jia, Jin-Yuan You, Ming-lu Li, “Group Mobility Model in Mobile Peer-to-Peer Media Streaming System”, Proceeding of 2004 IEEE International Conference on Services Computing, pp. 527-530, Sep. 2004.

[73] Min Qin, Roger Zimmermann, Leslie S. Liu, “Supporting Multimedia Streaming between Mobile Peers with Link Availability Prediction”, Proceeding of the 13th ACM International Conference on Multimedia, 2005.

[74] Meng Guo, Mostafa H. Ammar, Ellen W. Zengura, “V3: A Vehicle-to-Vehicle Live Video Streaming Architecture”, Proceeding of Third IEEE Internation Conference on Pervasive Computing and Communications, pp.171-180, Mar. 2005.

[75] Xiaoxin Wu, Bhargava Bharat, “AO2P: Ad Hoc On-Demand Position-Based Private Routing Protocol”, IEEE Transaction on Mobile Computing, Vol. 4, No. 4, pp.335-348, Aug. 2005.

[76] Ouri Wolfson, Bo Xu, Sam Chamberlain, Liqin Jiang, “Moving Objects Databases: Issues and Solutions”, Proceedings of 10th International Conference on Statistical and Scientific Database Management, pp.111-122, Jul. 1998.

[77] Ouri Wolfson, A. Prasad Sistla, Sam Chamberlain, Yelena Yesha, “Updating and Querying Databases That Track Mobile Units”, Distributed and Parallel Databases, Vol.7, No. 3, pp. 257-287, Jul. 1999.

[78] http://www.ktdb.go.kr, Korea Transport Database.

[79] http://info.korail.com/2007/eng/eng_index.jsp, Korea Railroad Information.

[80] http://www.ex.co.kr/portal/roa/tst/tst1/roa_tst01.jsp, Transport Information page at Korea Expressway Corporation,

이동 단말을 위한 스트리밍 미디어 서비스 연구

서 동 만

강원대학교 대학원 컴퓨터정보통신공학과

최근 무선통신 기술의 발전으로 PC뿐만 아니라 PDA, 노트북, 네비게이터, 무선 IP-TV 단말, 휴대전화 등 다양한 이동 단말 장치를 통하여 멀티미디어 서비스를 제공받을 수 있게 되었다. 스트리밍 미디어 서비스를 위한 영상 정보의 양이 텍스트 기반의 데이터 정보량에 비하여 매우 크다. 또한, 이동 단말 장치는 하드웨어의 성능 제약이 있으며, 낮은 네트워크 대역폭을 가지는 무선망에서 동작하기 때문에, 이동 단말을 위한 스트리밍 미디어 서비스에 대한 연구가 필요하다.

본 논문에서는 이동 단말을 위한 스트리밍 미디어 서비스를 제공하기위한 통합 트랜스코딩 미디어 스트리밍 서비스 시스템을 제안한다. 제안하는 시스템은 다양한 무선 네트워크에서 다양한 무선 단말을 통해 스트리밍 미디어 서비스를 이용하는 사용자들에게 안정적인 QoS를 제공하기 위해서는 무선 통신 망에서의 불안정한 대역폭에 알맞은 트랜스코딩 기법과 다양한 무선 단말에 QoS를 제공하기 위한 분산 트랜스코딩 서버에서의 부하 분산 기법 및 진입 제어 기법이 필요하다. 또한, 빠르게 이동하는 무선 단말에서 스트리밍 미디어 서비스를 제공하는 경우에 발생하는 빈번한 핸드오버와 셀의 바깥 영역에서의 낮은 대역

폭에서의 안정적인 QoS를 제공하기 위한 기법이 필요하다.

본 논문에서는 이러한 문제들을 해결하기 위한 세 가지 방안은 제안한다. 먼저, 이동 단말에게 안정적인 스트리밍 서비스를 제공하기 위한 Network Adaptive Autonomic Transcoding Algorithm (NAATA)를 제안한다. 제안한 알고리즘은 실시간으로 무선 네트워크의 상황에 맞추어 트랜스코딩 비트율을 변경한다. 연속적인 스트리밍 미디어의 전송 장애를 방지하게 때문에 안정적이고 끊임없는 스트리밍 미디어 서비스가 가능하다.

두 번째로, 분산 트랜스코딩 서버를 위한 부하분산 알고리즘을 제안한다. 제안한 알고리즘은 분산 트랜스코딩 서버의 트랜스코딩 시간을 예측하고 이를 기반으로 부하를 분산하고 진입을 제어한다. 제안한 알고리즘은 트랜스코딩 요구 특성과 스트리밍 미디어 데이터의 특성을 고려하기 때문에, 기존의 알고리즘에 비하여 높은 성능 확장성을 가짐을 실험을 통해 확인한다.

마지막으로, 와이브로와 같은 고속 무선 인터넷 환경에서의 안정적인 스트리밍 미디어 서비스를 위한 단말 이동성 기반의 미디어 스트림 선인출 기법을 제안한다. 제안하는 기법에서는 고속 이동 단말의 특성을 고려하여 그룹으로 분류하고, 그 특성에 따라 이동 방향과 속도를 예측한다. 이를 기반으로 하여 핸드오버 시에 발생할 수 있는 버퍼 고갈 상태와 전달 지연 상태를 예방한다. 제안한 방법은 실험을 통하여 고속 이동 단말 내의 스트리밍 미디어 버퍼의 상태를 안정적으로 유지함을 보인다.

Curriculum Vitae

Name : Dongmahn Seo
Date of Birth : October 7, 1976
email : sarum@kangwon.ac.kr

◎ Education

2004.03 ~ 2010.02	Dept. of Computer & Information Telecommunication Engineering (Ph.D) Kangwon National University
2002.03 ~ 2004.02	Dept. of Computer & Information Telecommunication Engineering (MS) Kangwon National University
1995.03 ~ 2002.02	Dept. of Computer Engineering (BS) Kangwon National University

◎ Publications

① International Journal

1. **Dongmahn Seo**, Inbum Jung, "Network-adaptive Autonomic Transcoding Algorithm for Seamless Streaming Media Service of Mobile Clients", Multimedia Tools and Applications, Published online, Oct. 2009.
2. **Dongmahn Seo**, Joahyoung Lee, Yoon Kim, Changyeol Choi, Manbae Kim, Inbum Jung, “Resource Consumption-Aware QoS in Cluster-based VOD Servers,” Journal of Systems Architecture: the EUROMICRO Journal, Volume 53 , Issue 1, pp. 39-52, Jan. 2007.
3. **Dongmahn Seo**, Joahyoung Lee, Yoon Kim, Changyeol Choi, Hwangkyu Choi, Inbum Jung, “Load Distribution Strategies in Cluster-based Transcoding

Servers for Mobile Clients," Lecture Notes in Computer Science, Vol. 3983, pp. 1156-1165, May 2006.

4. Joahyoung Lee, **Dongmahn Seo**, Yoon Kim, Changyeol Choi, Hwangkyu Choi, Inbum Jung, "Thin-Client Computing for Supporting the QoS of Streaming Media in Mobil Device," Lecture Notes in Computer Science, Vol. 3981, pp. 562-571, May 2006.
5. **Dongmahn Seo**, Joahyoung Lee, Dongkook Kim, Yoon Kim, and Inbum Jung "An Effective Failure Recovery Mechanism with Pipeline Computing in Clustered-Based VOD Servers", Lecture Notes in Computer Science, Vol. 3768, pp. 12-23, Nov. 2005.

② International Conferences

1. **Dongmahn Seo**, Heonguil Lee, and Inbum Jung "Transcoding Load Distribution Policy for Wireless Mobile Clients", The proceeding of the 2008 International Conference on Computer Design (CDES'08), Aug. 2008.
2. **Dongmahn Seo**, Hanmin Bang, Nansook Heo, Inbum Jung, and Yoon Kim "Experimental Evaluation for Scalability and QoS in a VOD System", Proceeding of the IEEE 2007 International Conference on Computational Science and its Applications, pp. 33-38, Sep. 2007.
3. Nansook Heo, Dongsun Lim, **Dongmahn Seo**, Inbum Jung, and Yoon Kim "Load Distribution Method and Admission Control for Streaming Media QoS in Distributed Transcoding Servers", Proceeding of the IEEE 2007 International Conference on Computational Science and its Applications, pp. 39-45, Sep. 2007.
4. Y. Kim, I.B. Jung, **D.M. Seo**, J.Y. Pyun, S.H. Park "Advanced Real-Time Rate Control for Low Bit Rate Video Communication", International Conference on Intelligent Computing, pp. 275-284, Aug. 2005.

③ Domestic Journal

1. 서동만, 허난숙, 김종우, 정인범, "분산 트랜스코딩 환경에서 부하 균형을 위한 트랜스코딩 부하 예측 기법", 한국정보과학회 논문지 제35권 9-10호, 466~475페이지 2008년 10월
2. 한우람, 허난숙, 박총명, 서동만, 정인범, "이동 단말에서 끊임없는 스트리

밍 미디어를 위한 오토노믹 멀티미디어 트랜스코딩 알고리즘”, 정보과학회 논문지, 제13권 제5호, 260~270페이지,2007년 10월

3. 서동만, 이좌형, 방철석, 임동선, 김윤, 정인범, “클러스터 VOD 서버에서 선호도 기반 세그먼트 버퍼 대체 기법”, 정보과학회논문지 제33권, 제11호, 797~809페이지, 2006년 12월.
4. 서동만, 이좌형, 최면욱, 김윤, 정인범, “효과적인 트랜스코딩 부하 분산을 위한 자원가중치 부하분산 정책”, 정보과학회논문지 제11권, 제5호, 401~415페이지, 2005년 10월.
5. 서동만, 방철석, 이좌형, 김병길, 정인범, “리눅스 기반의 클러스터 VOD 서버와 내장형에 클라이언트의 구현”, 정보과학회논문지 제10권, 제6호, 435~447페이지, 2004년 12월.

④ Domestic Conference

1. 방한민, 박총명, 서동만, 김학수, 정인범, "효율적인 부하 분산 정책을 위한 WMI기반 VOD서비스의 설계 및 구현", 한국정보처리학회 추계학술발표대회 논문집, 제15권, 제2호, pp. 1272~1275, 2008년 9월.
2. 김종우, 허난숙, 서동만, 정인범, "트랜스코딩 서버 간 부하 분산을 위한 트랜스코딩 부하 예측", 한국정보처리학회 춘계학술발표대회 논문집, 제15권, 제1호, pp. 909~912, 2008년 5월.
3. 한우람, 허난숙, 박총명, 서동만, 정인범, "디지털 전자액자를 위한 네트워크 적응적 스트리밍 미디어 서비스 설계 및 구현", 한국정보과학회 추계학술대회발표 논문집, 제34권, 제2호, pp. 477~481, 2007년 10월.
4. 한우람, 허난숙, 서동만, 정인범, "무선 네트워크상에서 전송장애 감지를 이용한 끊김 없는 스트리밍 미디어 서비스", 2007년 한국컴퓨터종합학술대회 논문집 제 34권 제 1호 121-122, 2007년 6월
5. 방한민, 한우람, 허난숙, 서동만, 정인범, "이동 사용자를 위한 네트워크 적응적 실시간 트랜스코더를 이용한 VOD 서비스의 구현", 2007년도 한국정보처리학회 춘계 학술발표논문집 제 14권 제 1호 1217-1220, 2007년 5월
6. 허난숙, 한우람, 이좌형, 서동만, 김윤, 정인범, "이동 사용자를 위한 적응적 트랜스코딩 서비스의 구현", 2006년도 한국정보과학회 가을 학술발표 논문집 제 33권 제 2호 183 ~ 188 페이지, 2006년 10월 20일.
7. 허난숙, 박총명, 서동만, 김윤, 정인범, "VOD 서비스에서 특산품 온라인

쇼핑 시스템의 구현", 정보처리학회 2005년 춘계 학술발표논문집 제 12권 제1호 1417 ~ 1420 페이지, 2005년 5월 13일.

8. 서동만, 박총명, 김동국, 김윤, 정인범, "이질적인 클라이언트 플랫폼을 위한 클러스터 VOD 시스템", 정보처리학회 2005년 춘계 학술발표논문집 제 12권 제1호 1413 ~ 1416 페이지, 2005년 5월 13일.

9. 박총명, 허난숙, 김동국, 서동만, 이좌형, 김윤, 정인범, "센서 네트워크를 이용한 교량 안전진단 시스템 구현", 정보처리학회 2005년 춘계 학술발표논문집 제 12권 제1호 1409 ~ 1412 페이지, 2005년 5월 13일.

10. 김동국, 박총명, 허난숙, 서동만, 이좌형, 김윤, 정인범, "무선 센서네트워크를 이용한 구조물 하중 감지 시스템", 정보처리학회 2005년 춘계 학술발표논문집 제 12권 제1호 1319 ~ 1322 페이지, 2005년 5월 13일.

11. 이좌형, 서동만, 방철석, 김병길, 박총명, 정인범, "클러스터형 VOD 서버에서 고가용성을 고려한 자체 복구 시스템", 한국정보처리학회 2003년11월 추계학술발표논문집, pp. 149 ~ 152, 제10권 2호, 2003년 11월 14일 ~ 15일.

12. 이좌형, 서동만, 방철석, 김병길, 박총명, 정인범, "클러스터형 VOD 서버에서 장애 복구의 설계 및 구현", 한국정보과학회 2003년10월 추계학술발표논문집, 제30권 2호, pp.427 ~ 429, 2003년 10월 24일 ~ 25일.

13. 서동만, 방철석, 이좌형, 김병길, 박총명, 정인범, "클러스터 VOD 시스템에서의 내장형 클라이언트 플랫폼 설계 및 구현", 한국정보처리학회 2003년도 춘계학술발표논문집, 제10권 제1호(중), pp.1153~1156, 2003년 5월 16일 ~ 17일.

14. 서동만, 방철석, 이좌형, 김병길, 정인범, "QoS를 지원하기 위한 리눅스 클러스터 VOD 서버의 성능 분석", 한국정보과학회 2003 봄 학술발표논문집, 제30권 제1호(C), pp.301~303, 2003년 4월 24일 ~ 25일.