Multimed Tools Appl
DOI 10.1007/s11042-009-0421-6

# Network-adaptive autonomic transcoding algorithm for seamless streaming media service of mobile clients

**Dongmahn Seo · Inbum Jung**



**Abstract** As a result of improvements in wireless communication technologies, a multimedia data streaming service can now be provided for mobile clients. Since mobile devices have low computing power and work on a low network bandwidth, a transcoding technology is needed to adapt the original streaming media for mobile environments. However, wireless networks have variable bandwidths depending on the movement of clients and the communication distance from Access Point (AP). These characteristics make it hard to support stable Quality of Service (QoS) streams for mobile clients. In this paper, a target transcoding bit-rate decision algorithm is proposed to provide stable QoS streams for mobile clients. In our experiments, the proposed algorithm provides seamless streaming media services based on the network adaptive bit rate control and reduces transmission failure.

**Keywords** Transcoding · Mobile client · Wireless · Network adaptive QoS · Streaming · Bit-rate

## 1 Introduction

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

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2008-D00424(I00901)).

D. Seo · I. Jung (✉)
Dept. of Computer Engineering, Kangwon National University,
Chuncheon, Gangwon, South Korea
e-mail: ibjung@kangwon.ac.kr

D. Seo
e-mail: sarum@kangwon.ac.kr

Published online: 31 October 2009

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 [6, 7, 17–19]. 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 [1, 17, 20]. 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 paper, 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.

The rest of this paper is organized as follows. Section 2 describes related work for our research. In Section 3, the NAATA to achieve the seamless streaming media service for mobile clients is proposed. In Section 4, the performance of the NAATA is evaluated. Section 5 provides a conclusion for the research.

## 2 Related work

### 2.1 Transcoding systems

There have been several approaches for transcoding systems, including source based static encoding system and static transcoding server systems [16, 17]. 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 specific 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 [3, 9]. Also previously proposed were the edge server strategy, packet transmission interval and datagram size control strategy [3, 21]. These methods, however, guarantee the QoS of streaming service with only the specific 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 [2]. 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 [2].

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 [2]. Automatic Repeat reQuest (ARQ) systems use combinations of time-outs 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 [2].

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 [2, 5].

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 [2, 4, 15]. The Datagram Congestion Control Protocol (DCCP) is UDP to support congestion control [12]. 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 s 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 [11, 16, 21].

### *2.3.1 Estimation of available network bandwidth*

Initial Gap Increasing (IGI) and Packet Transmission Rate (PTR) [11, 21] are algorithms for estimating the end-to-end available network bandwidth. Table 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

**Table 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;
}
```

**Table 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;
             }
        }
    }
}
```

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.

*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 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.

## 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.

As shown in Fig. 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.

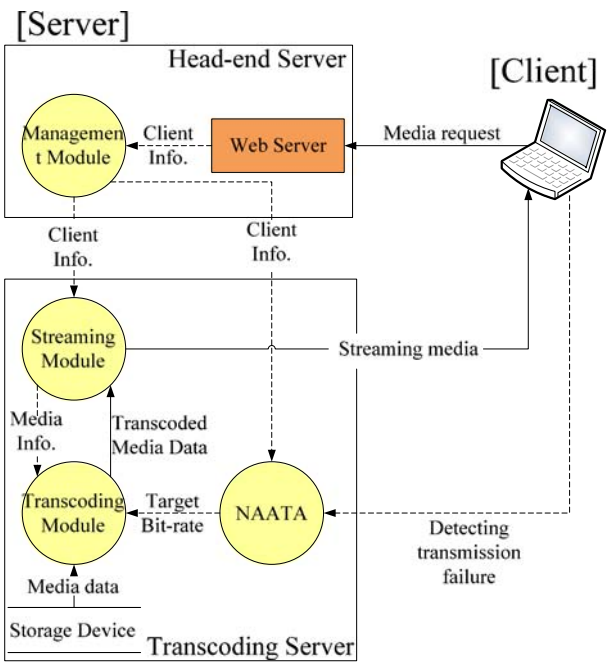


**Fig. 1** The NAATA concept using the AIMD

### 3.1 The AIMD congestion control algorithm in TCP

The AIMD is a part of the congestion control algorithm in TCP [10, 13]. 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 paper, 0 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 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

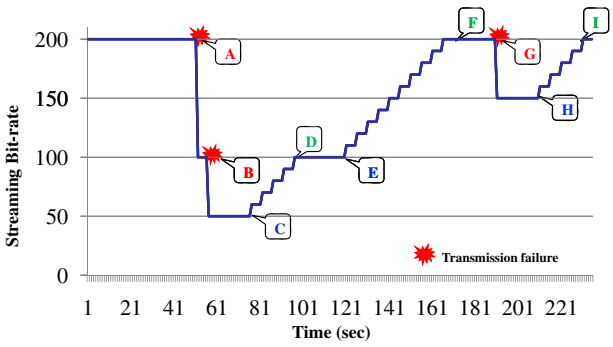


**Fig. 2** Activities in the NAATA

target transcoding bit-rate in the current stage as decided by the current state of the network. Point A of Fig. 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 Fig. 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 Fig. 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-defined 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 3 shows the pseudo-code for the NAATA. Part A of Table 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 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 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

**Table 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;                                          A-1
         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;                                                           B-2
         threshold_time += 10;
      } else {                           // when the other transmission failure occurs
         target_tr_bit = (max_bit+min_bit)/2;
         min_bit = target_tr_bit;                                               B-3
         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   C-1
         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   C-2
            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   C-3
            min_bit = max_bit;
            max_bit = init_bit;
            count = 0;
            threshold_time -= 10;
         }
      }
   }

   return target_tr_bit;
}
```

changed by *target_tr_bit*. Part B-2 of Table 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 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*.

Part C of Table 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 [11, 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 [8]. The mplayer is modified and applied to the client programs [14].

Table 4 shows the hardware specification for the server and the client. Table 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 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 | 1GB | 768MB | 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 5** Experiment media information

| | Bit-rate | Frame-rate | Resolution |
|---|---|---|---|
| Movie 1 | 871.7Kbps | 23.976fps | 576 × 256 |
| Movie 2 | 760.6Kbps | 23.976fps | 640 × 304 |

## 4.1 Performance of the NAATA

Figure 3 shows the streaming bit-rates of movie 1 as served by the ANATS and Figs. 4 and 5 show the streaming bit-rates of movies 1 and 2 as served by the NAATA. The streaming media bit-rate in Figs. 4 and 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 Figs. 3, 4 and 5, both initial bit-rates are 200 Kbps. When the first transmission failure occurs, the streaming bit-rate is changed to 100 Kbps in Figs. 4 and 5. However the streaming media bit-rate in Fig. 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 s in the Fig. 4, because target bit-rate is dropped from 200 Kbps to 100 Kbps or less. Instead of bit-rate dropping, re-buffing 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 s (approx. 11 min). It means that user should be patient every 2 or 3 min. However the NAATA supports seamless streaming media server without any re-buffering.

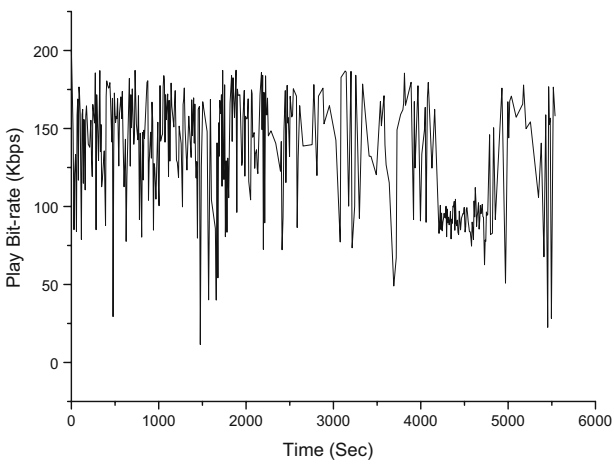


**Fig. 3** Streaming bit-rate of movie 1 with ANATS

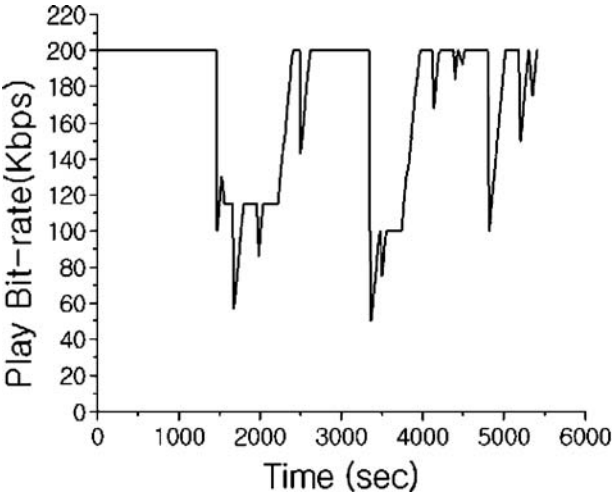


**Fig. 4** Streaming bit-rate of movie 1 with NAATA

As shown in Figs. 4 and 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 6 shows the accumulated number of transmission failures in the NAATA, the ANAT system and the legacy transcoding system. Various wireless network

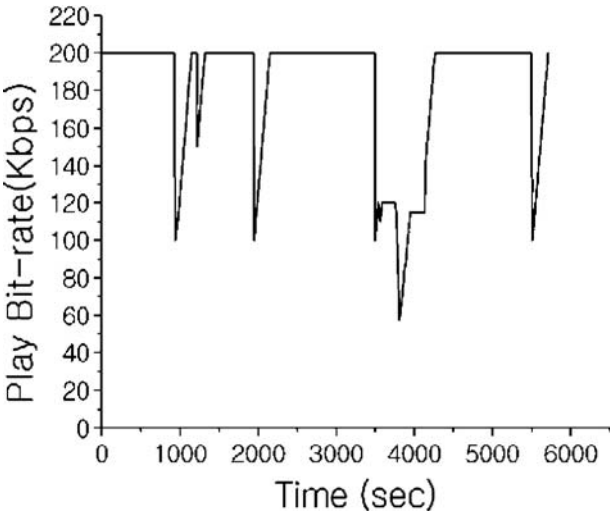


**Fig. 5** Streaming bit-rate of movie 2 with NAATA

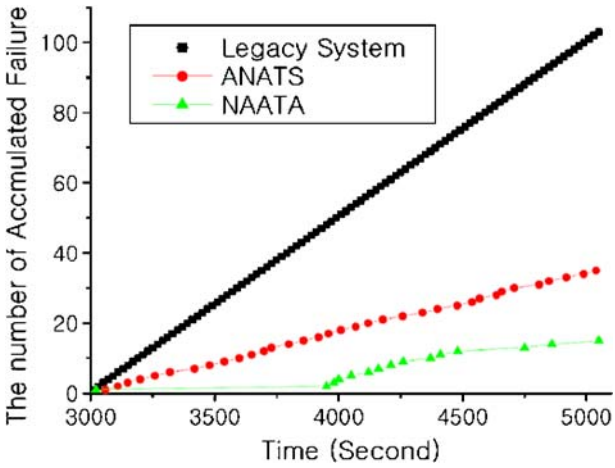


**Fig. 6** Accumulated number of transmission failures

states are created for near-real service environments as mobile clients change their locations.

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 Fig. 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 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 s, with a normal distribution and with a mean of 60 s. Thus, transmission failures occur every 1 min on mobile clients. This indicates that the jittering and ceasing events of streaming media occur every minute.

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 s at a maximum. In particular, the time intervals of less than 60 s 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.

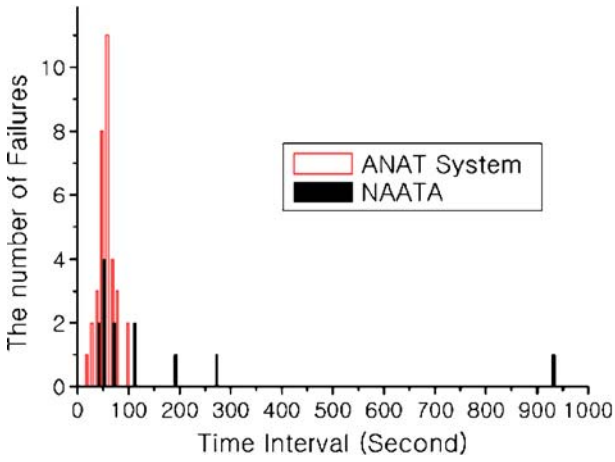


**Fig. 7** Histogram for time interval between transmission failures

### 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 [11, 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 s, the traffic overhead of 30 Kbytes occurs per 1 s. If a streaming service requires a network bandwidth between 50 Kbps and 200 Kbps, then the overhead for 1 s 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 s. 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 Conclusion

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 paper, 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.

## References


1. Bhradvaj H, Joshi A, Auephanwiriyakul S (1998) An active transcoding proxy to support mobile web access. In: Proc. intl. conf. on reliable distributed system, pp 118–123
2. Chakareski J, Frossard P (2007) Adaptive systems for improved media streaming experience. IEEE Commun Mag 45(1):77–83
3. Chandra S, Ellis CS, Vahdat A (2000) Differentiated multimedia web services using quality aware transcoding. In: Proc. IEEE INFOCOMM 2000, pp 961–969
4. Cho S, Woo H, Lee J-w (2003) ATFRC: adaptive TCP friendly rate control protocol. Lect Notes Comput Sci 2662:171–180
5. Chou PA, Miao Z (2006) Rate-distortion optimized streaming of packetized media. IEEE Trans Multimedia 8(2):390–404
6. Du DHC, Lee YJ (1999) Scalable server and storage architectures for video streaming. In: Proc. IEEE intl. conf. on multimedia computing and systems, pp 191–206
7. Feng WC, Lie M (2000) Critical bandwidth allocation techniques for stored video delivery across best-effort networks. In: Proc. the 20th intl. conf. on distributed computing systems, pp 201–207
8. ffmpeg.org (2009) ffmpeg open source project homepage. http://ffmpeg.org/
9. Forouzan BA (2001) Data communications and networking, 2nd edn. McGraw Hill, London
10. Hu N, Steenkiste P (2002) Estimating available bandwidth using packet pair probing. Technical report CMU-CS-02-116
11. Hu N, Steenkiste P (2003) Evaluation and characterization of available bandwidth probing techniques. IEEE JSAC (Special Issue in Internet and WWW measurement, Mapping and Modeling) 21(6):879–894
12. Kohler E, Handley M, Floyd S (2006) Designing DCCP: congestion control without reliability. In: SIGCOMM'06, pp 27–38
13. Kurose JF, Ross KW (2004) Computer networking—a top-down approach featuring the internet. Addison Wesley, Reading
14. mplayerhq.hu (2009) mplayer open source project homepage. http://www.mplayerhq.hu/
15. Padhye J, Kurose J, Towsley D, Koodli R (1999) A model based TCP-friendly rate control protocol. NOSSDAV99
16. Roy S, Covell M, Ankcorn J, Wee S, Yoshimura T (2003) A system architecture for managing mobile streaming media services. In: Proc. 23rd intl. conf. on distributed computing systems workshops, pp 408–413
17. Seo D, Lee J, Kim Y, Choi C, Choi H, Jung I (2006) Load distribution strategies in cluster-based transcoding servers for mobile clients. Lect Notes Comput Sci 3983:1156–1165
18. Seo D, Lee J, Kim Y, Choi C, Kim M, Jung I (2007) Resource consumption-aware QoS in cluster-based VOD servers. J Systems Archit EUROMICRO J 53(1):39–52
19. Sitaram D, Dan A (2000) Multimedia servers: applications, environments, and design. Morgan Kaufmann, San Francisco

20. Vetro A, Sun H (2001) Media conversions to support mobile users. In: Proc. IEEE Canadian conf. on electrical and computer engineering, pp 607–612
21. Wee S, Apostolopoulos J, Tan W-t, Roy S (2003) Research and design of a mobile streaming media content delivery network. In: Proc. IEEE ICME, pp I-5–I-8

**Dongmahn Seo** received his B.S. and M.S. degrees in Computer Engineering from Kangwon National Universiy in 2002 and 2004, respectively. From 2008 to 2009, he was a visiting scholar at University of Minnesota Duluth. He is currently a Ph.D. candidate in Computer Engineering at Kangwon National University. His research interests include multimedia system, parallel processing, embedded system and wireless sensor network.

**Inbum Jung** received his B.S. degree from Korea University, in 1985 and the M.S. and Ph.D. degrees in Computer Science from KAIST, in 1994 and 2000, respectively. From 1984 to 1995, he was with Samsung Electronics Co. Ltd., Korea. He is currently a faculty member at Kangwon National University. His research interests include operating system, parallel processing, streaming media and wireless sensor network.