PAPER

Improving Recovery Rate for Packet Loss in Large-Scale Telecom Smart TV Systems

Xiuyan JIANG[†], ***Nonmember***, **Dejian YE**[†a)], ***Member***, **Yiming CHEN**[†], ***Nonmember***, ***and*** **Xuejun TIAN**[††], ***Member***

SUMMARY Smart TVs are expected to play a leading role in the future networked intelligent screen market. Currently, many operators are planning to deploy it in large scale in a few years. Therefore, it is necessary for smart TVs to provide high quality services for users. Packet loss is one critical reason that decreases the QoS in smart TVs. Even a very small amount of packet loss (1–2%) can decrease the QoS and affect users' experience seriously. This paper applies stochastic differential equations to analyzing the queue in the buffer of access points in smart TV multicast systems, demonstrates the reason for packet loss, and then proposes an end-to-end error recovery scheme (short as OPRSFEC) whose core algorithm is based on Reed-Solomon theory, and optimizes four aspects in finite fields: 1) Using Cauchy matrix instead of Vandermonde matrix to code and decode; 2) generating inverse matrix by table look-up; 3) changing the matrix multiplication into the table look-up; 4) originally dividing the matrix multiplication. This paper implements the scheme on the application layer, which screens the heterogeneity of terminals and servers, corrects 100% packet loss (loss rate is 1%–2%) in multicast systems, and brings very little effect on real-time users experience. Simulations demonstrate that the proposed scheme has good performances, successfully runs on Sigma and Mstar Moca TV terminals, and increases the QoS of smart TVs. Recently, OPRSFEC middleware has become a part of IPTV2.0 standard in Shanghai Telecom and has been running on the Mstar boards of Haier Moca TVs properly.

key words: *smart TV, packet loss, QoS, error recovery*

1. Introduction

Following PCs (Personal Computer) and personal intelligent mobile devices, smart TVs are most likely to play a leading role in the next-generation networked intelligent screens. QoS (Quality of Service) is a critical problem that hinders the development of smart TVs while packet loss is a critical reason that affects the QoS, which causes unpleasant viewing experience for users of Smart TVs. Figure 1 shows the random packet loss (1%–2%, less than 5%) existing in the operating Shanghai Telecom Smart TV multicast system, which results in a decrease in QoS.

Some references demonstrate that even a very small packet loss in streaming system can affect users' experience seriously [1]–[3]. Usually, it is acceptable that less than one video frame with minor quality problems occurs every four hours. More strictly speaking, this kind of video frame occurs less than one time within several days or weeks. So packet loss in Shanghai Telecom smart TV multicast system affects users' experience seriously. There are many approaches to solve the packet loss problem, mainly falling into two categories: ARQ (Automatic Repeat Request) [4]–[6] and FEC (Forward Error Correction) [7]–[10]. ARQ is simple and consumes less computing resources on both the client side and the server side, but there will be congestion in the network if request number is too great. In a large scale smart TV system whose live streaming utilizes multicast, the server to clients is a one-to-many, so too much random packet loss causes a lot of retransmission which will lead to network congestion and force clients to receive too much redundant packets; also, at least one RTT (Round Trip Time) is needed for clients to receive the retransmitted packets because ARQ requires terminals to check the loss and send the message to the server. So ARQ is not appropriate for packet loss in smart TV multicast system.

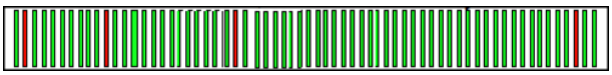

Fig. 1 Diagram of random packet loss in smart TV multicast system.

Since FEC does not check errors and does not retransmit packets, its delay is small with good real-time performance and it has been widely applied in live streaming systems. But FEC increases the bandwidth load because of redundant codes. Reference [11] adjusts the schedule to alleviate the congestion and decreases packet loss, but it does not take other possible reasons for packet loss into account. Reference [12], [13] combines ARQ and FEC to decrease the bandwidth cost and sacrifices some FEC effect and some real-time performance. In LT theory, if the original data consists of k input symbols then each encoding symbol can be generated, independently of all other encoding symbols, on average by $O(\ln(k/\delta))$ symbol operations, and the k original input symbols can be recovered from any $k+O(\sqrt{k}\ln 2(k/\delta))$ of the encoding symbols with probability $1-\delta$ by on average $O(k\ln(k/\delta))$ symbol operations [18]. Reference [14] proposed a method to speed up channel switching and to decrease the packet loss caused by channel switching. Reference [15] proposed a FEC method based on the Reed-Solomon (RS) [16] coding theory but it considers little about the weak computing capability of clients, so it cannot run on the weak clients such as STB (Set Top Box) properly.

Manuscript received October 9, 2012.
Manuscript revised May 28, 2013.

[†]The authors are with the School of Computer Science, Fudan University, China.
[††]The author is with the Faculty of Information Science and Technology, Aichi Prefectural University, Nagakute-shi, 480–1198 Japan.
a) E-mail: dejianye@fudan.edu.cn

DOI: 10.1587/transinf.E96.D.2365

Reference [17], [18] applies the LT coding theory to correct the errors and saves the decoding time, but its cost is not stable and can't ensure the real-time performance. Reference [19] proposes the Raptor code technology to ensure the stability of cost but sacrifices part of the error recovery capability, and it is suitable in satellite communication systems on the physical layer. In summary, the above FEC methods have the following problems: 1) not able to correct the errors 100% in smart TV multicast systems; 2) not considering completely the clients' weak computing capability and the requirement of real-time performance, so they are not suitable for Shanghai Telecom Smart TV multicast system. This paper analyzes packet loss in smart TV multicast systems by stochastic differential equations and proposes an error recovery scheme which can recover the packet loss effectively without sacrificing the real-time performance and improve QoS of Shanghai Telecom smart TV multicast system substantially.

2. Packet Loss Analysis

2.1 Analysis of Smart TV Multicast System Architecture

In this section, to understand the situation of packet loss in Shanghai Telecom smart TV multicast system, we carry out the analysis in order to give improvement proposal. As shown in Fig. 2, the backbone network topology of Shanghai Telecom smart TV multicast system consists of three layers: core node, regional nodes and edge nodes.

The distribution network is private and the distribution platform provides qualified QoS mechanism to prevent packet loss. Then where does packet loss happen? Detection results show that packet loss happens at the PoP AP (Access Point) on edge nodes. We analyze the captured packets both at APs and at client sides, find that the lost packets on the client side are the same as the lost packets at AP, which demonstrates that packet loss just happens at POP AP. Figure 3 shows that packets enter into the buffer at AP, form queues and then are distributed through the internet.

Suppose the bandwidth of the distributed network is C, the video rate is R, usually C > R in the distribution design, so it is impossible to lose packets due to the bandwidth limit. Some researches [20], [21] show that VBR (Variable Bit Rate) streaming has heavy burst which is caused by scenes variation. VBR's burst and random noise can cause the dynamic variation of the queue length and leads to buffer overflow, which leads to packet loss. In this section, we will apply the stochastic differential equation to expressing the queue length in AP buffers explicitly, analyze VBR burst and random noise influence on the queue length in the buffer of AP.

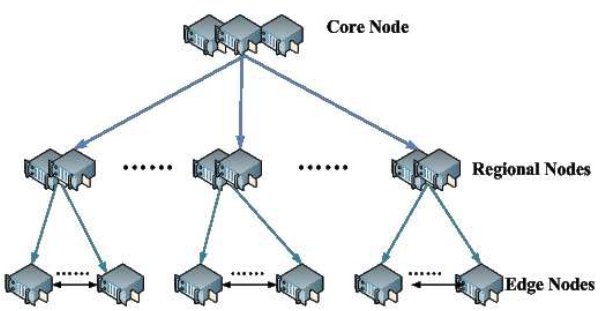

Fig. 2 Topology of smart TV multicast system.

2.2 Analysis of Packet Loss Using Stochastic Differential Equation Model

2.2.1 Independent Noise with Two-State Model for Traffic Flow

We use the two-state Markov Process as the process of a video traffic model to simulate the scene changes based on a DC component, which is to prevent distortion on negative noise.

$$dx = (1 - x)dN_b - xdN_a, x(0) \in \{0, 1\} \quad (1)$$

$$z(t) = bx(t) \quad (2)$$

$$dv = -cI(v)dt + z(t)dt + sd\omega + mdt \quad (3)$$

Where, $v(t)$ is the queue length at time t, $z(t)$ is a function of the mean and smoothed arrival process, x is two-state Markov On-Off process, b is a magnification factor, m is the DC component of video traffic flow, ω is the standard Wiener Process and are coefficient.

2.2.2 Model Analysis

If the jumping rate of Markov On-Off process is λ and μ respectively, then the autocorrelation is:

$$R_{xx}(\tau) = \frac{\lambda}{\lambda + \mu}\mu e^{(\lambda+\mu)\tau+\lambda}$$

It meets the format of autocorrelation of video traffic flow properly. Consequently the model can represent the statistical characteristics of the video traffic flow extremely well. From (2) we get:

$$\mathrm{E}[z] = b\mathrm{E}[x] = \frac{\lambda b}{\lambda + \mu}$$

Write the equation for v^2 as:

$$dv^2 = -2cvI(v)dt + 2bvxdt + 2vsd\omega + 2vmdt + s^2dt$$

We notice that the last term s^2dt is $It\hat{O}$ item. Taking expectation, we have:

$$\frac{d}{dt}\mathrm{E}[v^2] = -2c\mathrm{E}[vI(v)] + 2b\mathrm{E}[vx] + 2m\mathrm{E}[v] + s^2 \quad (4)$$

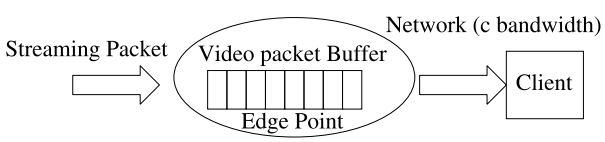

Fig. 3 Traffic queue in Edge Points in smart TV multicast system.

In the heavy traffic setting and the steady state, we can obtain

$$(c-m)\mathrm{E}[\nu] = b\mathrm{E}[\nu x] + \frac{s^2}{2} \tag{5}$$

$$\begin{aligned} d\nu x &= \nu dx + x d\nu \\ &= -\nu x dN_a + \nu(1-x)dN_b - cxI(\nu)dt \\ &\quad + bx^2 dt + mxdt + sxd\omega \end{aligned}$$

Taking expectations and again assuming stability, then:

$$\mathrm{E}[\nu x] = \frac{\lambda}{\lambda+\mu}\mathrm{E}[\nu] - c\mathrm{E}[xI(\nu)] + b\mathrm{E}[x^2] + m\mathrm{E}[x]$$

Considering that $\mathrm{E}[xI(\nu)] = \mathrm{E}[x]$ and $\mathrm{E}[x^2] = \mathrm{E}[x]$, we have:

$$\mathrm{E}[\nu x] = \frac{\lambda}{\lambda+\mu}\mathrm{E}[\nu] + \frac{b-c+m}{\lambda+\mu}\mathrm{E}[x]$$

Substitute this for Eq. (5), then

$$\mathrm{E}[\nu] = \frac{\mathrm{E}[z](b-c+m)}{(\lambda+\mu)(c-\mathrm{E}[z]-m)} + \frac{s^2}{2(c-\mathrm{E}[z]-m)} \tag{6}$$

Equation (6) for $\mathrm{E}[\nu]$ is an explicit expression in heavy traffic. The latter item including s^2 is caused by $It\hat{O}$ item calculus.

2.3 Analysis of Network Video Traffic Queueing Model in Smart TV Systems

In a real environment with complicated traffic flows, we give the general model for video traffic queuing flow:

$$\begin{cases} dx &= (1-x)dN_b - xdN_a, x \in \{0,1\} \\ z(t) &= bx(t) \\ d\nu &= -cI(\nu)dt + \sum_{i=1}^{n} b_i x_i dt + sd\omega + mdt \end{cases} \tag{7}$$

This model reflects network video transmission process of the dynamic buffer on the server and the input stream can reflect various video encoding statistical characteristics. Item $\sum_{i=1}^{n} b_i x_i dt$ represents various scenes in a specific film, which at most denotes 2^n scenes with different traffic rates. s_i gives the traffic rate of various scenes.

We get the following explicit expression via complex computation.

$$\begin{aligned} \mathrm{E}[\nu] \;=\; & \frac{\sum_{i=1}^{n}\tau_i(b_i+m-c)\mathrm{E}[z_i]}{c-m-\sum_{i=1}^{n}\mathrm{E}[z_i]} \\ & + \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\tau_i\mathrm{E}[z_i]\mathrm{E}[z_j]}{c-m-\sum_{i=1}^{n}\mathrm{E}[z_i]} \\ & + \frac{s^2/2}{c-m-\sum_{i=1}^{n}\mathrm{E}[z_i]} \end{aligned} \tag{8}$$

From (8), we can see that the former two terms in $\mathrm{E}[\nu]$ is caused by continuous scenes while the last is caused by the $It\hat{O}$ calculus with noise items. The calculus item on the expectation of queue length cannot be neglected. The above is a more accurate estimation of expectation of the entire queue length in a heavy traffic environment. The scenes parameters of b_i, m and the noise parameter s affect the queue length heavily, and their values are random and make the queue length unpredictable, which is the reason why the buffer in POP AP overflow and lose packets in smart TV multicast system. From the above analysis, we can see that the reason for packet loss are VBR's burst and random noise and it can happen randomly. Packet loss cannot be recoverd on lower layer of network (such as physical layer).

3. Design of Error Recovery Scheme (OPRSFEC)

3.1 Principles of Design

- In the view of possible adjustment of network service in the future, the scheme should be independent from the network and a server-to-client, end-to-end scheme is suitable.
- Since the heterogeneity of STBs, the scheme should screen the differences of all types of STBs, and have a good universal property.
- Since the clients such as STBs' computing resources are very weak and the server has a heavy load, the scheme should cost less computing resources both on the client side and the server side.
- Meeting the requirement for real-time performance
- Having good error correction capability, achieving perfect error correcting, 100% error correction.
- Low redundancy to decrease the extra bandwidth.

3.2 Architecture of OPRSFEC Scheme

Figure 4 describes OPRSFEC (Optimized Forward Error Correction Based On Reed-Solomon Theory) which is implemented in application layer. The coding scheme's procedure and decoding process base on RTP packets. The procedure includes the following steps:

- The server codes the RTP packets and generates redundant RTP packets;
- After adding heads, the server sends redundancy RTP packets and original RTP packets to the client; packet loss will probably happen;
- After receiving all RTP packets from the server, the client decodes and recovers lost packets to form complete packets same as the original ones.

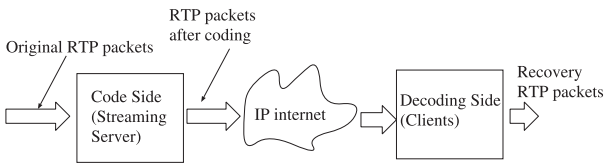

Fig. 4 Achitecture of OPRSFEC Scheme.

3.3 Related Work: RSFEC (FEC Algorithm Based on Reed-Solomon Theory)

RSFEC is composed of coding part and decoding part as the following:

3.3.1 Coding Part

Suppose the server codes the packets by group, every group includes k packets. Assuming that, the number of redundant packets is ω, $n = k + \omega$, S is the k original source packets matrix, E is the coded packets matrix including redundant packets, F is $n \times k$ matrix which is Vandermonde matrix generated in $GF(2^s)$ Galois Field, then

$$E = F \times S \tag{9}$$

As shown in Fig. 4, we hope the matrix generated by coding includes original packets and redundant packets, so we need to change the submatrix which is composed of the front k lines of F matrix into an identity matrix. The coding procedure is described as the following two steps.

a) Generating Vandermonde F

$$F = V_{kk}^{-1} \times V_{kn} \tag{10}$$

Since the front part of F is identity submatrix, we only need to compute the back $n - k$ columns, that is $V_{kk}^{-1} \times V_{k(n-k)}$. Inverting matrix V_{kk}, $V_{k(n-k)}$ and V_{kk} can be acquired by Vandermonde matrix definition of $GF(2^8)$ Galois Field. Every element in F relates to k and n only.

b) Computing E

It can be given by (9).

3.3.2 Decoding Part

According to Reed-Solomon theory, when $k < n \leq 255$ and $c \geq k$, it can correct 100% errors. So once the client receives k packets, RSFEC start decoding. Suppose E is the matrix which is composed of received complete packets, then decoding algorithm works in the following steps:

(1) Computing original coding matrix by (10);

(2) Computing decoding F matrix

$$F' = {V'_{kk}}^{-1} \tag{11}$$

According to the sequence numbers of packets in the coding matrix, we select the corresponding columns in F to form $k \times k$ Vandermonde matrix V'_{kk}, then invert it.

(3) Computing decoded matrix S

$$S = F' \times E' \tag{12}$$

3.4 OPRSFEC Algorithm (Optimized RSFEC)

3.4.1 RSFEC Matrix Invertion in Finite Field

According to Reed-Solomon theory, only when $k < n \leq 255$ and $c \geq k$, RSFEC can recover all the errors, so it is necessary to divide streaming packets into groups which consists of k packets. Each time when we start RSFEC, a coding matrix F is generated, which includes inverting V_{kk} and its computing complexity is $O(N^3)$. It is noticed that F only relates to k and n. OPRSFEC computes the matrix F one time and save the value. When k and n value are fixed, we can use the value of F repeatedly, thus OPRSFEC can save time greatly. In practice, k is basically fixed, while n varies with network and will be fixed only when the packet loss rate changes little.

```
Generate GF256 exp()&log() table
1   Q←0xFF ;
2   ALPHA←0x2;
3   MODULUS←0x1d;
4   x←y←1;
5   m_exp[Q]←1;
6   For i←0 to Q
        m_exp[i]←x
        y move 1 bit to the left
        if (y and 0x100 is equal to 1)
            y←y exclusive or MODULUS
        y←y/0x100
        x←y
7   For i←0 to Q+1
        m_log[m_exp[i]]← i
8   m_log[0]←0xFF
9                                   m_log[1]←0
```

Fig. 5 Generation of exp()&log() table.

3.4.2 Operation of Look-Up Table in Finite Field

When OPRSFEC works, $V_{k(n-k)}$, and V'_{kk} will be generated. In order to save operation time, a $GF(2^8)$ exp()&log() table shown in Fig. 5 will be generated before generating these three matrixes. Then OPRSFEC looks up in it to find appropriate elements to form the above three matrixes.

3.4.3 RSFEC Inteligent Matrix Division in Finite Field

In the heads of the source and redundant packets, there are group sequence numbers and packet sequence numbers of the group, by which OPRSFEC can deduce the lost packet number. The following process will be: In order to recover lost packets, OPRSFEC generates a decoding matrix *WilldecodeV* composed of the appropriate value where packets are lost, by which OPRSFEC generates matrix *RecoverV*. Then OPRSFEC generates Matrix *OriginalV* using received original packets and merges *RecoverV* and *OriginalV* into the complete corrected packets matrix. OPRSFEC can save the time of matrix multiplication because it only needs to recover the lost packets rather than for every received packet.

RSFEC intelligent matrix division in finite field is described as the following:

L: the length of packet (byte)

Set $i(0 < i)$: sequence number of lost packets in source packets,

$R[L][k]$: Sent packets matrix

$R_y[L][\omega]$: Received redundant packets matrix

$R_r[L][k]$: Decoded packets matrix including received source packets and decoded packets

$R'[L][k]$: If there is packet loss, the matrix is composed of received source packets and received redundant packets with k packets. Then,

$$R' = \begin{pmatrix} R_{1,1} & \dots & R_{1,i-1} & R_{Y[1,j]} & R_{1,i+1} & \dots & R_{1,k} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{L,1} & \dots & R_{L,i-1} & R_{Y[L,j]} & R_{L,i+1} & \dots & R_{L,k} \end{pmatrix} \quad (13)$$

$$R'' = R' \times \begin{pmatrix} V'_{1,i} \\ . \\ . \\ . \\ V'_{L,i} \end{pmatrix} \quad (14)$$

Where, R'' is the recovered loss packet matrix. Then the complete correct packets matrix after decoding can obtained as follows:

$$R_r = \begin{pmatrix} R_{1,1} & \dots & R_{1,i-1} & R''_{1,i} & R_{1,i+1} & \dots & R_{1,k} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ R_{L,1} & \dots & R_{L,i-1} & R''_{L,i} & R_{L,i+1} & \dots & R_{L,k} \end{pmatrix} \quad (15)$$

3.4.4 Progressive Optimization Strategy

The coding computing of FEC strategies based on RS coding is not complicated, and the procession capability of existing streaming media server is sufficient for RS coding. We found through experiments that coding based on RS which is group coding has following characteristics: (1) CPU usage on the coding side fluctuates largely during group RS coding; the CPU utilization is higher at the moment of group coding; the memory usage is also higher because the group coding needs to catch a group of packets in the cache. (2) Using the Gaussian Elimination matrix inversion with the complexity $O(N^3)$, some set-top boxes cannot perform RS decoding in time when receiving high-definition video streaming.

Since each digital television channel requires a separate FEC coding, so the performance of FEC coding is directly related to the maximum number of channels that the streaming media server can support. Therefore, progressive optimization strategy optimizes the RSFEC performance in three aspects: (1) to reduce the CPU peak usage; (2) to make the CPU usage fluctuates steadily; (3) to reduce the memory usage. This progressive optimization strategy is divided into progressive encoding and progressive decoding.

a) Progressive Encoding Strategy

Unfold (9) into a matrix:

$$\begin{pmatrix} E_{0,0} & E_{0,1} & \dots & E_{0,s-1} \\ E_{1,0} & E_{1,1} & \dots & E_{1,s-1} \\ \vdots & \vdots & \ddots & \vdots \\ E_{n-1,0} & E_{n-1,1} & \dots & E_{n-1,s-1} \end{pmatrix} =$$

$$F \times \begin{pmatrix} S_{0,0} & S_{0,1} & \dots & S_{0,s-1} \\ S_{1,0} & S_{1,1} & \dots & S_{1,s-1} \\ \vdots & \vdots & \ddots & \vdots \\ S_{k-1,0} & S_{k-1,1} & \dots & S_{k-1,s-1} \end{pmatrix} \quad (16)$$

We found that the nature of the coding package of FEC algorithm based on RS coding is coefficients of the linear combination of the original packet group.

$$E_i = \sum_{j=0}^{k-1} a_{i,j} S_j \quad (17)$$

So, we can decompose the group coding into k separate encoding process. Each time the coding side receives a new package of raw data, and multiplies its corresponding coefficient in F, then add to the data packet generated from the coding:

$$C'_i = C_i + a_{i,j} S_j \quad (18)$$

Since a total of n coded packets is needed, the process need to repeat n times. Work on the generator matrix F, making the coding module to generate the system code, namely the first k packets are the original packets. The $n \times k$ -order of F is as follows:

$$\begin{pmatrix} & & I_k & & \\ \frac{1}{k+0} & \frac{1}{k+1} & \frac{1}{k+2} & \cdots & \frac{1}{k+(k-1)} \\ \frac{1}{(k+1)+0} & \frac{1}{(k+1)+1} & \frac{1}{(k+1)+2} & \cdots & \frac{1}{(k+1)+(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{(n-1)+0} & \frac{1}{(n-1)+1} & \frac{1}{(n-1)+2} & \cdots & \frac{1}{(n-1)+(k-1)} \end{pmatrix} \quad (19)$$

b) Progressive Deocding Strategy

Suppose, during decoding, it receives k data packets in which s packets are redundant. Make the data packets correspond to line subscripts $\{r_s, r_{s+1}, \dots, r_{k-1}\}$ in F redundant packets to $\{t_0, t_1, \dots, t_{s-1}\}$, lost packets to F_k. Then after adjustment, the submatrix F_k becomes

$$\begin{pmatrix} I_{k-s} & 0 \\ A & B \end{pmatrix} \quad (20)$$

Where, A is a $S \times (k-s)$ Cauchy matrix, B is a $S \times S$ Cauchy matrix.

LU decomposition of F_k:

$$F_k = \begin{pmatrix} I_{k-s} & 0 \\ A & I_s \end{pmatrix} \begin{pmatrix} I_{k-S} & 0 \\ 0 & B \end{pmatrix} \quad (21)$$

So the inverse matrix of F_k is:

$$\begin{aligned} F_k^{-1} &= \begin{pmatrix} I_{k-s} & 0 \\ 0 & B^{-1} \end{pmatrix} \begin{pmatrix} I_{k-S} & 0 \\ -A & I_s \end{pmatrix} \\ &= \begin{pmatrix} I_{k-S} & 0 \\ B^{-1}(-A) & B^{-1} \end{pmatrix} \end{aligned} \quad (22)$$

While the determinant of B is:

$$\det(B) = \frac{\prod_{i<j}(t_i - t_j) \prod_{i<j}(r_i - r_j)}{\prod_{i,j=0}^{s-1}(t_i + r_j)} \quad (23)$$

Because any submatrix of B is a Cauchy matrix, using its cofactor matrix C, we can get the inverse matrix of B:

$$B^{-1} = \frac{1}{\det(B)} C^T \tag{24}$$

After simplified, it becomes

$$B^{-1} = \left(d_{ij}\right)_{i,j=0,\ldots,s-1} \tag{25}$$

where,

$$d_{ij} = (-1)^{i+j} \frac{e_j f_i}{a_j b_i (t_j + r_i)} \tag{26}$$

$$a_k = \prod_{i<k}(t_i - t_k) \prod_{k<j}(t_k - t_j) \tag{27}$$

$$b_k = \prod_{i<k}(r_i - r_k) \prod_{k<j}(r_k - r_j) \tag{28}$$

$$e_k = \prod_{i}(t_k + r_i) \tag{29}$$

$$f_k = \prod_{i}(t_i + r_k) \tag{30}$$

The complexity for the above B^{-1} computational process is $O(s^2)$, and the complexity for the Gaussian Elimination method is $O(s^3)$, even in the case $s > 5$, the decoding complexity can be reduced substantially. The complexity for F_k^{-1} is $O(ks^2)$ after the optimization of the algorithm. Compared with $O(k^3)$ before the optimization, the decoding complexity is reduced considering that S is generally lower than k an order of magnitude.

4. OPRSFEC&RSFEC Performance Analysis

OPRSFEC is the optimization of RSFEC. This section will analyze and compare their performance.

4.1 Comparison of Operation Times at the Coding Side

Coding side is composed of (9) and (10). As shown in Table 1, coding side has 4 steps. In OPRSFEC, F only relates to k and n. If k and ω are fixed, F is needed to be computed only once; if ω varies, F is needed to be computed only one time when the first new group is coded, which results in time saving. In table 1, L is the length of the number of one source RTP packet ($k \le n \le 255, L \gg k$). Suppose that k and n are fixed when coding, the number of group is M. Table 1 shows that step 1 to step 3 save a lot of operation times, so the operation times of OPRSFEC is much less than those of RSFEC.

4.2 Comparison of Operation Times at the Decoding Side

Decoding side composed of (5), (6) and (7), includes four steps in Table 2. Since every time V'_{kk} is different in step 3 and step 4, OPRSFEC does not save computation times. But since OPRSFEC applies intelligent matrix division in finite field, only recovers lost packets (only 1%-2% of all the packets). Steps 4 actually save time greatly. Table 2 shows the comparison of operation times at decoding side. M' is decodin times (if all sourc packets are received, decoding operation does not work, so $M' \le M$). Table 2 and the above analysis shows that step 1, step 2, step 4 is optimized in OPRSFEC, so OPRSFEC saves a lot of time.

Table 1 OPRSFEC & RSFEC coding perpormance.

Sequence	Step	Computation Times (RSFEC)	Computation Times (OPRSFEC)
1	V_{kk}, V_{kn}	ML	1
2	V_{kk}^{-1}	ML	1
3	$V_{kk}^{-1} \times V_{kn}$	ML	1
4	$F \times S$	ML	ML

Table 2 OPRSFEC & RSFEC decoding perpormance.

Sequence	Step	Computation Times (RSFEC)	Computation Times (OPRSFEC)
1	V_{kk}, V_{kn}	$M'L$	1
2	$V_{kk}^{-1} \times V_{kn}$	$M'L$	1
3	$V'_{kk}{}^{-1}$	$M'L$	$M'L$
4	$S = F \times E$	$M'L$	$M'L$

5. Simulation

5.1 Setting

Clients: PC (win7 CPU: Intel Core i3, 2.93GHz) / Sigma STB (OS: embedded linux, CPU:500Mkz, Memory:128M) / Mstar Moca board (OS: embedded Linux, CPU:447 Mip, Memory:128M).

Server: virtual machine with 2.10 GHz, dual core, Red Hat Linux 5.

Network configuration: In Shanghai Telecom, the bandwidth from the server to the client:2Mbps, streaming rate: 1.3Mbps

Packet loss: happened randomly, with 5% packet loss rate or less.

Number of executives: 10 times and get the average value.

5.1.1 Simulation of OPRSFEC

With $k = 100, \omega = 5$, from Fig. 6 and Fig. 7 we can see that when packet loss happens, OPRSFEC can recover the errors and the streaming played correctly without mosaic, which means QoS of the smart TV multicast system is enhanced.

5.1.2 Comparing OPRSFEC Decoding Speed with RSFEC Decoding Speed

With PC clients, under the condition of $k = 100, \omega = 5$, that is, every one hundred packets forms one coding matrix. Figure 8 shows the computing time of RSFEC and OPRSFEC in every key step.

From Fig. 8 we can see that OPRSFEC cuts the time

Fig. 6 OPRSFEC effect on packet loss in multicast system (Packet loss with OPRSFEC).

Fig. 7 OPRSFEC effect on packet loss in multicast system (Packet loss without OPRSFEC).

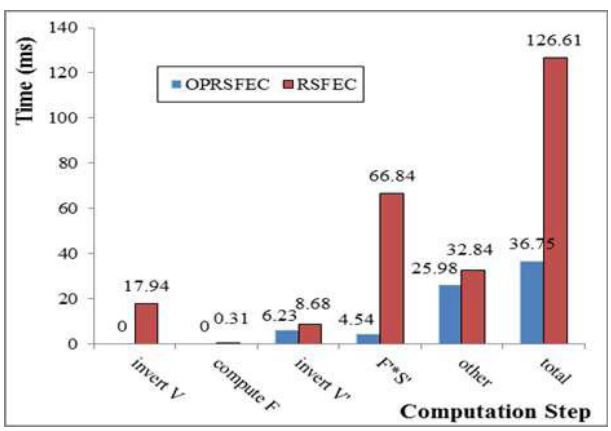

Fig. 8 Decoding time comparison tween OPRSFEC and RSFEC.

of "compute F", "invert V" steps, and decreases the time of $F' \times S'$ steps more than 10 times. The total time of OPRSFEC is about 36 ms, which is less than 1/3 that of RSFEC, so OPRSFEC can save the computation time greatly. Usually, 1000 ms is required to ensure real-time performance in Shanghai Telecom Smart TV system.without OPRSFEC, the total time for delay from the request to the the start time of play at the client is from 0 to 300 ms in Shanghai Telecom Smart TV system.

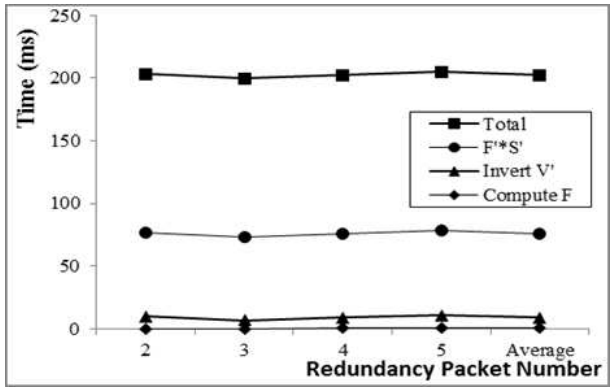

Fig. 9 Group's redundant packet number influence on decoding time.

Table 3 OPRSFEC DECODING performance comparison.

Client Type	CPU	Memory (M)	Server to Client Delay
sigma Design	50%	42.4 (33.5%)	300ms
Mstar	36%	42.7 (34.1%)	290ms

As shown in Fig. 9, under the condition of $k = 100$ and that ω varies from 2 to 5, the decoding time remains almost the same while redundant packet number changes, so redundant packet number have little influence on decoding time.

5.1.3 OPRSFEC Performance Analysis on Different Weak Clients

Set $k = 100, \omega = 5$. Table 3 shows OPRSFEC performance on sigma design STB (OS: embedded linux, CPU: 500Mkz, Memory: 128M), Mstar Moca TV Terminal (OS: embedded Linux, CPU:447 Mip, Memory:128M). The server-to-client delays on Sigma STB and MStar Moca TV terminal are both less than 300ms, within users' acceptable scope. On Sigma Design STB and Mstar Moca TV terminal, the memory consumptions is within 35%, and CPU consumption is within 50%. Since these clients' CPU and memory consumption is usually small, OPRSFEC API (see Sect. 6.1) can work on these two clients properly. The delays between the server and each clients (STB, and TV) are different, because each client's computation resource are different.

5.1.4 Analysis of the Influence of Redundant Packet Number on Decoding Speed

Set $\omega = 5$, with the Sigma Design STB client. Figure 10 shows that the bigger the group size, the more decoding time and coding time. But the decoding time varies faster than coding time.

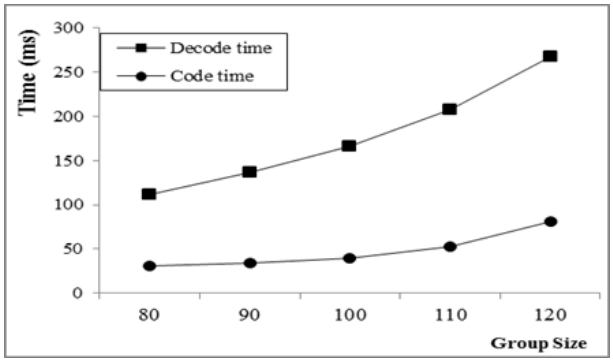

Fig. 10 Group size influence on OPRSFEC performance.

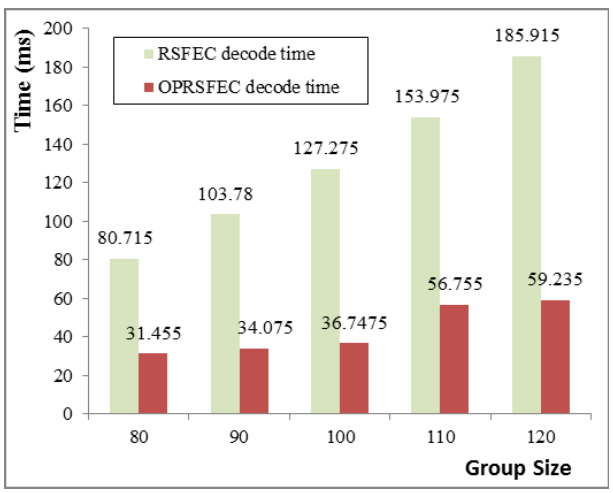

Fig. 11 Group size influence on RSFEC&OPRSFEC decoding time.

Set $\omega = 5$, with the PC client. Figure 11 tells that the group size has some influence on the decoding time. When k is less than 100, the group size has little influence on OPRSFEC decoding time, but much influence on RSFEC. It means that with the growth of the group size, the advantage of OPRSFEC is becoming more obvious. When the group size is more than 90, OPRSFEC decoding time is more stable, only 1/3 that of RSFEC. Since Sigma Design STB used in the case of Fig. 10 has weaker computation resource than PC used in the case of Fig. 11, so the decode times of OPRSFEC shown in Fig. 10 is longer than that in Fig. 11.

5.1.5 Experiment Results Analysis of Progressive Optimization Strategies

a) Coding Performance Analysis

From Fig. 12, we can see CPU usage fluctuates in a very small range when using progressive encoding and its peak is far less than without the progressive encoding. CPU usage of the progressive encoding converges to the range from 2% to 3%, with very small fluctuations, while about 20% CPU usage is from 5% to 6% for a group coding. Therefore, the progressive coding strategy compared with

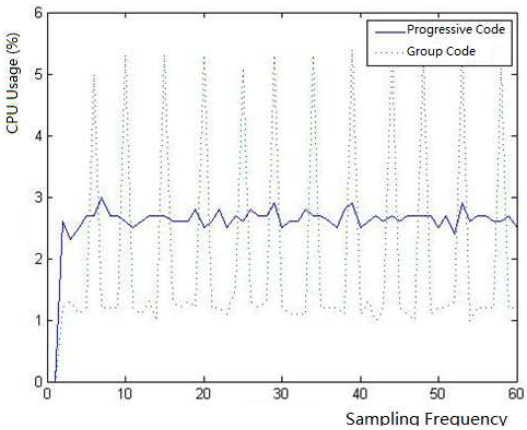

Fig. 12 CPU change between progressive encoding and group encoding.

Table 4 The mean and standard deviation of CPU utiliation before and after the progressive coding.

Experiment Types	Average CPU useage	CPU useage variance
Group Coding	1.9817%	1.6665
Progressive Coding	2.6083%	0.3642

the group coding strategy can double the quality multicast channels supported by the streaming media server under the same hardware conditions.

From Table 4, we found that the average CPU utilization of the group coding is lower than the progressive encoding. It is because the number of group encoding is far less than that of the progressive encoding, so the average CPU usage is lower. By the contrast between the two variances, we can see that the fluctuations of the group encoding rate are far higher than those of the incremental coding.

Because of the UNIX-like system memory management mechanism, we cannot directly obtain the actual memory occupied by the processes from the top command. Therefore, we, in the coding module, cache the original data packets, and indirectly measure the memory usage by real-time monitoring of the original packet queue length changes. Figure 13 shows the queue length changes of the two strategies. Considering every element is a packet with a fixed length, we can compare the cache changes of the two strategies by comparing the length. Figure 13 tells that: (1) the grouping coding needs to cache the entire group of data packets firstly (in this experiment, $k = 50$). (2) Upon receiving one packet, it codes without waiting for other packets and then forwards it, so the queue length is not more than 1. Based on the above two points, we confirm that the progressive encoding significantly improves the memory usage efficiency of the coding side.

We use CPU utilization and decoding time to evaluate the performance enhancement effects of the optimal decoding algorithm based on Cauchy matrix. Figure 14 is a CPU usage of the client before and after optimization. We find that the optimization algorithm enables the CPU usage to lower an average of 25% than before the optimization.

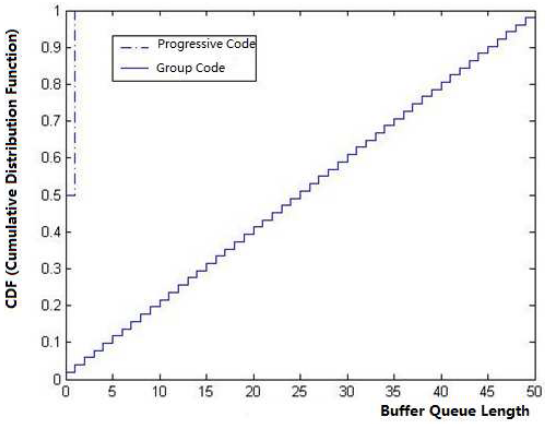

Fig. 13 Buffer queue length CDF of progressive coding and group coding.

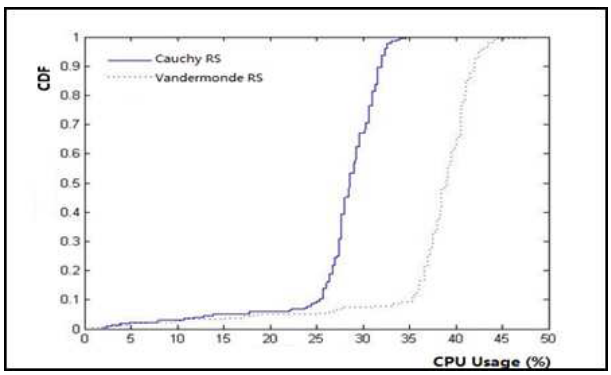

Fig. 14 Usage after RS decoding optimization.

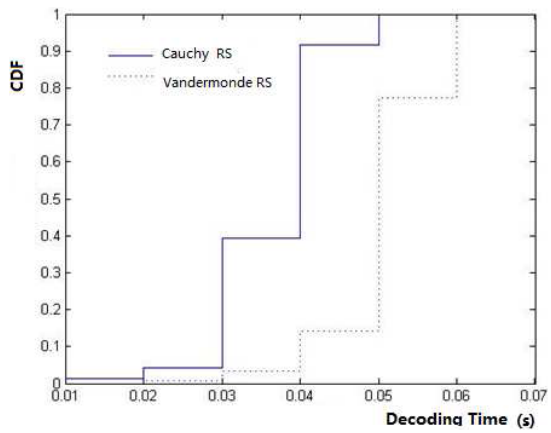

Fig. 15 Decoding time contrast before and after the decoding optimization.

Figure 15 shows the cumulative distribution function (CDF) comparison of the decoding time before and after optimization. We can see that the decoding time with optimization converges to the narrow range from 0.03 seconds to 0.04 seconds. In the contrast, it is in the range from 0.05 to 0.06 seconds in the case without optimization.

Table 5 Optimization decoding strategy for performance improvements.

Experiment Types	Average CPU useage	Average Decoding Time
Vandermonde RS decoding	37.4157%	0.0505s
Cauchy RS decoding	27.7633%	0.0364s
Comparison before and after optimization		
Performance improvement	34.77%	38.8%

Table 5 shows the specific performance improvement of our proposed optimization strategy. It can be seen that the optimized decoding strategy simplifies the complexity of the FEC decoding algorithm and improves the algorithm performance, also reduces the hardware requirements on the decoding side.

5.1.6 Extra Bandwidth Analysis of OPRSFEC

Under the same condition of 5.3, for the 1.3Mbps original video, we get the average bandwidth cost is 1.37Mbps with OPRSFEC, so OPRSFEC caused about 5% extra bandwith.

6. Implementation of OPRSFEC Middleware

Because of the multiple providers and the heterogeneity of STBs, OPRSFEC needs to be independent on other functions of STBs and Servers to avoid other applications' influence, and adapts to all kinds of smart TV platforms, so we apply middleware to realizing OPRSFEC. OPRSFEC includes APIs at both the coding and the decoding side, running on the server and the client side separately, as shown in Fig. 16.

6.1 Implementtion of Coding API

Coding API is implemented at the server side as shown in Fig. 16. Firstly the server sends original RTP packets to the source packets buffer then sends UDP packets by port A to the clients after adding heads. When the number of packets in source packet buffer is equal or more than k, the server starts to code and sends the redundant packets to the redundant packet buffer at the server side, then sends the redundant packets by port B to the client side.

6.2 Implementation of Decoding API

Decoding API is implemented at the client side (such as STB) as shown in Fig. 16. When the source packet receiving buffer receiving the source UDP packets by port C from the server side, it generates the original RTP packets and sends them to the complete and correct packet buffer. The redundant packet receiving buffer receives redundant packets by port D from the server side; while the received packet number is equal to the Threshold, it starts to decode and sends the recovered packets to the complete and correct buffer.

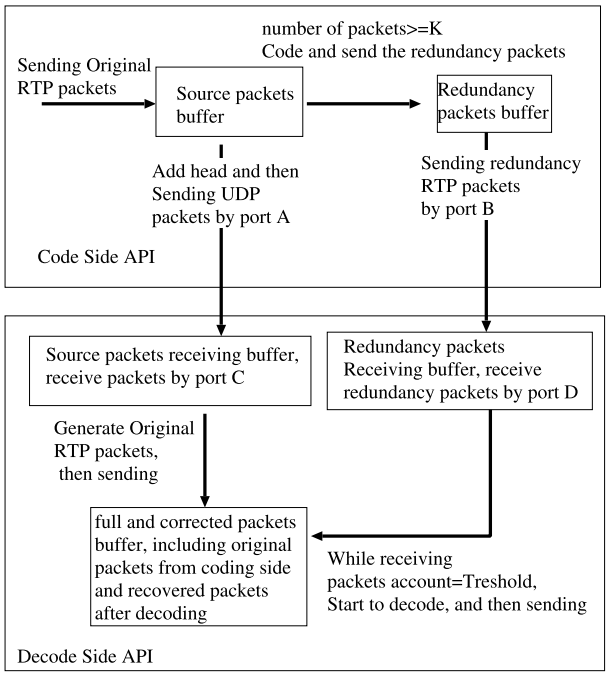

Fig. 16 Implement of OPRSFEC middleware.

7. Deployment Cases

7.1 Deployment in Shanghai Telecom Smart TV System

OPRSFEC middleware has been deployed in Smart TV multicast system in Shanghai Telecom Company who is the largest IPTV operator in the world, it has been running with good performance both on the server side and on different clients such as Huawei STB (with Sigma chip), ZTE STB (with Sigma chip) and other STB terminals (with Mstar Moca board); it can correct 1%–2% packet loss completely without sacrificing users' real time experience. Currently it is applied in IPTV 2.0 standard in Shanghai Telecom Company which requires all STB providers provide OPRSFEC API interface.

7.2 Deployment in Haier Moca TV Terminal

OPRSFEC middleware is deployed in national Hegaoji key project "Moca Digital TV Streaming Middleware Development and Industrialization". It becomes a part of the QoS module and runs well with good performance on Mstar Moca TV of Haier Company who is one of top three Home appliance manufacturers and wins approval from all the related authoritative experts.

8. Conclusions

This paper applies stochastic differential equations to analyzing the queue in the buffer of access points in smart TV multicast systems after demonstrating the reason for packet loss, then proposes an end-to-end error recovery scheme (short as OPRSFEC) whose core algorithm is based on Reed-Solomon theory. Our proposal optimizes four aspects in finite fields: 1) Using Cauchy matrix instead of Vandermonde matrix to code and decode; 2) generating inverse matrix by table look-up; 3) changing the matrix multiplication into the table look-up; 4) originally dividing the matrix multiplication. We implemented the scheme on the application layer, which screens the heterogeneity of terminals and servers, corrects 100% packet loss (loss rate is 1%-2%) in multicast systems, and brings very little effect on real-time users experience. Simulations results prove that the proposed scheme has good performances and can successfully run on Sigma and Mstar Moca TV terminals, which leads to higher QoS of smart TVs. Recently, OPRSFEC middleware has become a part of IPTV2.0 standard in Shanghai Telecom and has been running on the Mstar boards of Haier Moca TVs properly.

References

[1] ETSI. Digital Video Broadcasting (DVB); Guidelines for DVB-IP Phase 1 Handbook[R], 2007.
[2] N. Degrande, K. Laevens, D. De Vleeschauweret al., "Increasing the user perceived quality for IPTV services," IEEE Commun. Mag., vol.46, no.2, pp.94–100, 2008.
[3] O. Hohlfeld, R. Geib, G. Haßlinger, "Packet loss in real-time services: Markovian models generating QoE impairments," IEEE, pp.239–248, 2008.
[4] J.G. Kim, M.M. Krunz, "Bandwidth allocation in wireless networks with guaranteed packet-loss performance," IEEE/ACM Trans. Netw. (TON), vol.8, no.3, pp.337–349, 2000.
[5] C. Hoene, A. Gunther, and A. Wolisz, "Measuring the impact of slow user motion on packet loss and delay over IEEE 802.11 b wireless links," Proc. 28th Annual IEEE Int. Conf. on Local Computer Networks, 2003. LCN '03, pp.652–662, 2003.
[6] T. Zhang and Y. Xu, "Unequal packet loss protection for layered video transmission," IEEE Trans. Broadcast., vol.45, no.2, pp.243–252, 1990.
[7] J. Nonnenmacher and E.W. Biersack, "Reliable multicast: Where to use FEC," 1996.
[8] W.T. Tan and A. Zakhor, "Video multicast using layered FEC and scalable compression," IEEE Trans. Circuits Syst. Video Technol., vol.11, no.3, pp.373–386, 2001.
[9] L. Rizzo, L. Vicisano, "A Reliable Multicast data Distribution Protocol based on software FEC techniques," Citeseer, pp.23–25, 1997.
[10] R. Puri, K. Ramchandran, K.W. Lee, et al., "Forward error correction (FEC) codes based multiple description coding for Internet video streaming and multicast," Signal Processing: Image Communication, vol.16, no.8, pp.745–762, 2001.
[11] S. Zare and A.G. Rahbar, "An FEC scheme combined with weighted scheduling to reduce multicast packet loss in IPTV over PON," J. Network and Computer Applications, 2011.
[12] M. Wu, S. Makharia, H. Liu, et al., "IPTV multicast over wireless LAN using merged hybrid ARQ with staggered adaptive FEC," IEEE Trans. Broadcast., vol.55, no.2, pp.363–374, 2009.
[13] P.A. Chou, A. Mohr, A. Wang, et al., "FEC and pseudo-ARQ for receiver-driven layered multicast of audio and video," Proc. DCC 2000, pp.440–449, 2000.
[14] A.C. Begen, N. Glazebrook, and W. Ver Steeg, "A unified approach for repairing packet loss and accelerating channel changes in multicast IPTV," 6th IEEE Consumer Commun. Netw. Conf., 2009. CCNC 2009, pp.1–6, 2009.
[15] M. Johanson, "Adaptive forward error correction for real-time internet video," Proc. 13th Packet Video Workshop, Nantes, France,

2003.
[16] J. Lacan, V. Roca, J. Peltotalo, et al., "Reed-solomon forward error correction (FEC) schemes," Institute of Electrical and Electronics Engineers, 2009.
[17] M. Luby, L. Vicisano, J. Gemmell, et al., "The use of forward error correction (FEC) in reliable multicas," RFC 3453, Dec. 2002.
[18] M. Luby, "LT codes," pp.271–280, 2002.
[19] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, et al., "Practical loss-resilient codes," Proc. 29th Annual ACM Symposium on Theory of Computing, pp.150–159, 1997.
[20] Z. Gu and K.G. Shin, "Algorithms for effective variable bit rate traffic smoothing," Performance, Computing, and Communications Conference, 2003. Conference Proc. 2003 IEEE International, pp.387–394, 2003.
[21] D. Ye, J.C. Barker, Z. Xiong, et al., "Wavelet-based VBR video traffic smoothing," IEEE Trans. Multimedia, vol.6, no.4, pp.611–623, 2004.

Xiuyan Jiang Ph.D, is currently an assistant professor in Computer Science School of Fudan University. She received the B.S. and M.S. from Harbin Institute of Technology, China, in 1997, 1999 respectively. Her research field includes computer network and multimedia.

Dejian Ye is currently an associate professor in Computer Science School of Fudan University. He received the B.S., M.S., and Dr. degrees in engineering from Zhejiang Univeristy, Harbin Institute of Technology, Tsinghua University (Ph.D), China, in 1997, 1999 and, 2003 respectively. He worked as a visiting professor at University of Massachusetts Boston from 2003 to 2004. His research interests include QoS, computer network, multimedia and automation control. He is a member of IEEE and IEICE.

Yiming Chen is master student in computer Science School of Fudan University, his research field includes IPTV and multimedia.

Xuejun Tian graduated from Hebei University, China in 1985. He received his M.S. degree from Department of Electrical and Mechanical Engineering, Tianjin Institute of Technology, China in 1991, and Ph.D. degree from Department of Intelligence and Computer Science, Nagoya Institute of Technology, Japan, in 1998, respectively. Since 1998, he was an Assistant Professor and Associate Professor from 2008 in the Department of Information Systems, Faculty of Information Science and Technology, Aichi Prefectural University, Japan. From Jul. 2003 to Jun. 2004, he was a Visiting Assistant Professor in Department of Electrical and Computer Engineering at University of Florida, Gainesville, FL. Dr. Tian is a member of the IEICE (The Institute of Electronics, Information and Communication Engineers), the IEEJ (the Institute of Electrical Engineers of Japan), IPSJ (Information Processing Society of Japan) respectively. His research interests includes QoS, wireless networks, mobile communications and ubiquitous computing.