Dynamic Scheduling of Real-time Multi-core Subtask Based on Traffic Prediction

Yi Song
National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
Email: songy@dsp.ac.cn

Wu Zhang, Hong Ni and Xiuyan Guo
National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing, China
Email: {zhangw, nih, guoxy}@dsp.ac.cn

***Abstract*—Subtask scheduling on multi-core is an important direction in the world today. There is little research of dynamic scheduling subtask according to the network traffic prediction. This paper proposes a method of dynamic scheduling of real-time multi-core subtask based on network traffic prediction. According to the prediction result of network traffics, the larger tasks scheduled dynamically into some subtasks. In the simulation, we scheduled the subtasks in the multi-core network software pipeline architecture, according to the prediction of Markov Model of media data streams. Eventually reach in the case of packet drop rate is acceptable, and CPU utilization rate has fallen.**

***Index Terms*—Multi-core Software Architecture, Network Traffic Prediction, Markov, Subtask Scheduling**

I. INTRODUCTION

Multi-core network processors with those numerous cores and all kinds of hardware acceleration for packet processing are widely used in the field of network data stream processing [1]. In order to adapt to multi-core network processor and network data flow characteristics, running tasks on the multi-core network processors are different from normal processors. They should be divided into subtasks, assigned to the different cores, and coordinate the communication among those cores. Nowadays, Task scheduling is a research hotspot. Chen and Wang introduced WRR to the latest Hadoop platform, and proposed a new task scheduling algorithm [2]. The subtask scheduling of multi-core network software architecture can be described by DAG graphs. But the actual task scheduling is not as complicated as a lot of theory in the study of DAG graph. The tasks usually divided into subtasks of pipeline form finished step by step. Then they scheduled to multi-core network software architecture. There are three common multi-core network software architecture, namely all parallel RTC (Run-to-Complete), all serial S-SPL (Single-Software PipeLine), and in combination with parallel and serial P-SPL (Parallel-Software PipeLine). In order to take full advantage of multi-core performance, we usually use P-SPL as multi-core software architecture. The multi-core network software architecture described below in this paper is P-SPL. The traditional study of subtasks is that the subtask is pre-scheduled, and doesn't change during the life of whole system. But static subtasks scheduling methods cannot adapt to changing network traffic load. If the network traffic is not heavy, the multi-core network processor needs not to open all the cores, just make some of its core running. It's necessary to develop a method for dynamic scheduling of real-time multi-core, according to the network traffic status.

II. RELATED RESEARCH

It needs to work in two areas for scheduling real-time multi-core subtasks dynamically according to the network traffic status, namely prediction of network traffic, and determination of subtask division, and also determine which core to schedule to.

IP network traffic prediction began in 1994, by Groschwitz N.K;Polyzos G.C [3]. In the nearly two decades of development, a variety of random process model is used to predict the network traffic, for example Poisson Process, Gaussian Process. WEI Duo and LYU Guanghong propose a method of IP traffic matrix estimation based on ant colony optimization [4]. For MPEG-4 traffic such specific traffic in the IP network, some studies have pointed out that the MPEG-4 has a short correlation characteristics [5]. Some other studies pointed out that the MPEG-4 traffic has a long correlation characteristic from the point of view of statistical multiplexing [6]. The MPEG-4 traffic also has self-similiar characteristics [7], and MPEG-4 is multifractal [8]. More, more detailed study showed that not a generally applicable flow model can describe the characteristic of the media data stream [9]. So there is not a particularly good traffic model can predict the flow characteristics of the media data stream. TIAN Zhichao analysed the data stream traffic characteristics using wavelet joint estimation method [10]. Latest research shows that Markov Model is a better traffic prediction model [11]. As a Markov Process, Markov Model has been a hot direction, and has long been used to describe the network traffic model. Markov Model considers the basic characteristics of the network traffic, rather than some uncertain, complex features, and thus has better predictive accuracy. This paper uses Markov Model as media data stream traffic prediction model.

A lot of research based on the predicted results of media stream. Yonghua Xiong proposed an adaptive transmission method based on media traffic prediction method [12]. In the scheduling of real-time multi-core subtask, existing studies use the DAG describing multi-core tasks model. Han and Liu proposed a task scheduling algorithm for multiprocessor systems. It gives different priorities to different tasks according to the out degree and software execution time [13]. Previous studies raised a lot of task scheduling algorithms based on DAG [14]. But usually multi-core network software architecture is RTC, S-SPL or P-SPL, the three DAG's special cases [15]. Deciding which cores need to be turned on or off, WANG Tao proposed multiprocessor task allocation algorithms according to the characteristics of task set, aiming to use as little as possible processors to achieve optimal scheduling results [16]. This paper uses prediction result of network traffic to determine how to divide subtasks as well as schedule subtasks to which core in P-SPL software architecture. The work of this paper was based on the widely used Cavium OCTEON platform [17, 18].

III. Scheduling Method

This paper used the Markov Model to predict network traffic. According to the prediction result, we change the multi-core software architecture dynamically. We predict the network traffic in the next time segment based on the historical characteristics. The network traffic is task load. We decide the network software architecture on the basic of task load in the future. If the task load getting lighter, we can turn off some running cores; if the task load getting heavier, we can turn on some sleep cores, so they can processing tasks. It also should be considered that changing multi-core software architecture is not completed immediately, but need some time to convert. So it is necessary to predict a period of time from now on, and not only to predict changes in a short period of time. It has several advantages. The first is when the task load is light, fewer cores cost less power, or the unused cores can be scheduled to the other tasks. The second is when the task load is heavy, system knows it in advance and handles it in time. The third advantage is all the changing process is autonomic without manual intervention.

Set there are m data streams in the network. There are all media data streams. The average bitrate of each stream is v. m is describe by Markov Model. That is the probability random variable m can be found at m_1, m_2, m_3, ... comply with Markov process.

$$P(m_{n+1} = m \mid m_0, m_1, m_2, \ldots, m_n) = P(m_{n+1} = m \mid m_n) \tag{1}$$

Set pipeline software architecture is P-SPL. The pipeline has 3 stages, packet input I, packet process P and packet output O respectively, as shown in Figure 1, expressed in DAG as in Figure 2. The multi-core processor has N cores. Among the N cores, there are n cores in the running state. The other cores are in the sleep state. $n \le$ N. Each core in the running state belongs to a specific stage of I, P, and O. There are n_S cores in the S stage (S=I,P,O)。$n_I+n_P+n_O=n$. In Figure 1 and Figure 2, $n_I=3$, $n_P=2$, $n_O=3$.

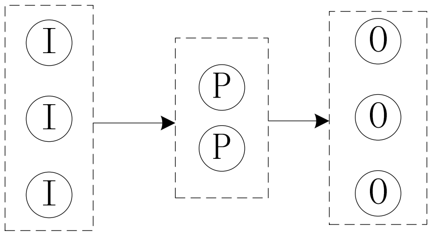

Figure 1 3 stages multi-core network software architecture P-SPL pipeline sketch

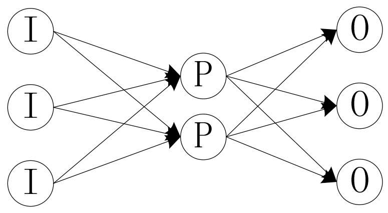

Figure 2 Multi-core network software architecture DAG Graph

The CPU utilization in the system can be shown as

$$util_{CPU} = \frac{1}{t_1 - t_0}\int_{t_0}^{t_1} \frac{n}{N} dt$$

(2)All of the cores must be open, if we don't predict the network traffic. That is n=N，$util_{CPU}=1$.

Each architecture in the system has 4 properties, how many cores in each stages $\{n_I, n_p, n_o\}$, number of streams the system can process m_T, how long it can transit to this architecture t_T, Quality of Service QoS_T. Let there are 3 software architectures when there are n cores in the running state. That is the architectures can be T_{n1}, T_{n2}, T_{n3}, and the number of processing data streams is m_T

$$m_T = f(T)$$

(3)f(T) is the relationship between the software architecture and T. In the P-SPL pipeline architecture, to keep the packet throughput maximization, shall ensure that each stage of packet processing capacity approximately the same. So the packet can't stay in the system, achieving a complete streaming process. Let the workload ratio of the 3 pipeline I, P and O is 3:2:3 approximately. That is in the case of a certain total number of cores, n_I:n_P:n_O = 3:2:3, the processing capacity of data streams is maximum. When n_I:n_P:$n_O \ne$ 3:2:3, the contribution of cores more the ratio in 3 stages to the processing capacity is a, b, and c respectively. That is the contribution of cores more than the ratio is

$a\left[\frac{1}{3}n_I - \min\left(\frac{1}{3}n_I, \frac{1}{2}n_P, \frac{1}{3}n_O\right)\right]$. So the *f*(T) is:

$$f(T) = a\left[\frac{1}{3}n_I - \min\left(\frac{1}{3}n_I, \frac{1}{2}n_P, \frac{1}{3}n_O\right)\right]$$
$$+b\left[\frac{1}{2}n_P - \min\left(\frac{1}{3}n_I, \frac{1}{2}n_P, \frac{1}{3}n_O\right)\right]$$
$$+c\left[\frac{1}{3}n_O - \min\left(\frac{1}{3}n_I, \frac{1}{2}n_P, \frac{1}{3}n_O\right)\right]$$
$$+d\min\left(\frac{1}{3}n_I, \frac{1}{2}n_P, \frac{1}{3}n_O\right)$$

(4)The *a*, *b*, *c*, *d* above are fixed factors. Simplify $+d\min\left(\frac{1}{3}n_I, \frac{1}{2}n_P, \frac{1}{3}n_O\right)$ formula (4), we can get

$$f(T) = \frac{a}{3}n_I + \frac{b}{2}n_P + \frac{c}{2}n_O$$
$$-\left(\frac{a}{3} + \frac{b}{2} + \frac{c}{3} - d\right)\min\left(\frac{1}{3}n_I, \frac{1}{2}n_P, \frac{1}{3}n_O\right)$$

(5)The drop rate during t_0 and t_1 in the system is

$$drop = \int_{t_0}^{t_1}(m - m_T)v\,dt$$

(6)The target of prediction of media data streams is to minimize the $util_{CPU}$, when the *drop* is in the acceptable range.

Changing the cores state, no matter from running to sleep or from sleep to running, is not complete immediately. It takes time *t*. That is *m* is changing continuously, but *n* can't transit from n_1 to n_2 immediately, it takes time *t*. So the prediction algorithm must predict the trend of *m* before the actual change happed. So we can adjust *n* in time. Scheduling a subtask to a new core in the multi-core network architecture, we need to take a core from sleep to running. We can schedule it to I, P, or O, one of them. It will form 3 different software architectures. Namely is the upper right, the right and the lower right in Figure 3. Take a core from running to sleep, in order to remove a subtask from a core in the multi-core network software architecture. We can take the core from I stage, P stage or O stage, any stage will be OK. It forms 3 different software architectures. That is the upper right, the right, and the lower right of Figure 4.

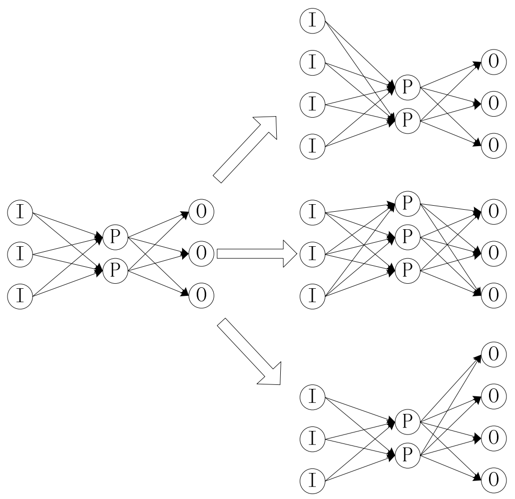

Figure 3 Schedule subtask to a core in multi-core network software architecture

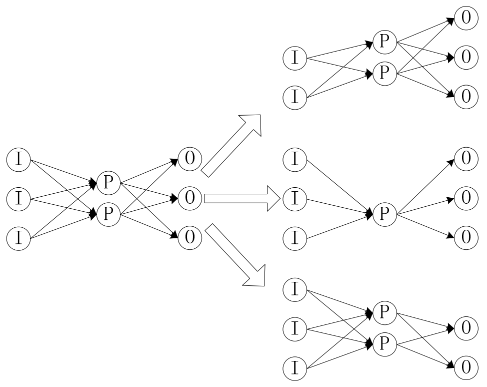

Figure 4 Schedule subtask from a core in multi-core network software architecture to sleep

Every *t* time in the system, we should predict the media data stream network traffic, in order to adjust the software architecture. We predict *m* in the next *t* and 2*t* time. Following is pseudo code:

```
while (The system is running)
{
    predict $m$ in the t time, according to the Markov Model
    if ($m_T < m$ and $n < N$)
    {
        take a core from sleep to running
        choose a software architecture in the 3 different architectures that has $n$+1 running cores following the method below
        if (there are more than one architecture that $m_T \geq m$ and $t_T < t$)
        {
            if (there are more than one architecture that $m_T \geq m$ in the next 2$t$ time)
            {
                choose the architecture that the $QoS_T$ is maximum, put the coming core to the correct pipeline stage
            }
            else
            {
                choose the architecture that the $QoS_T$ is maximum, put the coming core to the correct pipeline
stage
                }
            }
            else
            {
                The existing software architecture remains the same
            }
        }
        else if ($m_T > m+m_\Delta$ and $n > 0$) /* $m_\Delta$ is set to avoid the frequent change of software architecture */
        {
            put a core from running to sleep
            choose a software architecture in the 3 different architectures that has $n$-1 running cores following the method below
            if (there are more than one architecture that $m_T \geq m$ and $t_T < t$)
            {
                if (there are more than one architecture that $m_T \geq m$ in the next 2$t$ time)
                {
                    choose the architecture that the QoS$_T$ is maximum, put the core to the sleep state
                }
                else
                {
                    choose the architecture that the QoS$_T$ is maximum, put the core to the sleep state
                }
            }
        }
}
```

IV. Simulation

We used Matlab for simulation. We use Star Wars IV MPEG4 high quality frame size trace file, verbose file [9]. The trace file contains a movie fragment of 3599840 ms, 89998 frames. We use these frames size as network traffic. The 89998 data are divided to sample spaces and make Markov property test. The sequence conform Markov Characteristic. Establish transition probability matrix. Predict the network traffic according to the transition probability matrix. Plus a margin to the prediction result. So the system can always process all the network traffic. The predict result is shown in Figure 5. We can see the difference between the predict network traffic and actual network traffic is little, and the actual traffic is slightly greater than the predict traffic. The simulation data is from the Cavium OCTEON CN5860, 16 cores 800MHz MIPS SoC network processor. The multi-core network software architecture data is from a streaming encryption software in the Octeon Simple Executive environment. The software divides the packet processing into input, processing and output 3 stages. There is at least one core in one stage. According to the method this paper proposed, the number of cores in the running state is shown in Figure 6. It can be seen that the number of cores in the running state is changing according to the state of network traffic. The cores in the running state are less when the network traffic is less, and vice versa.

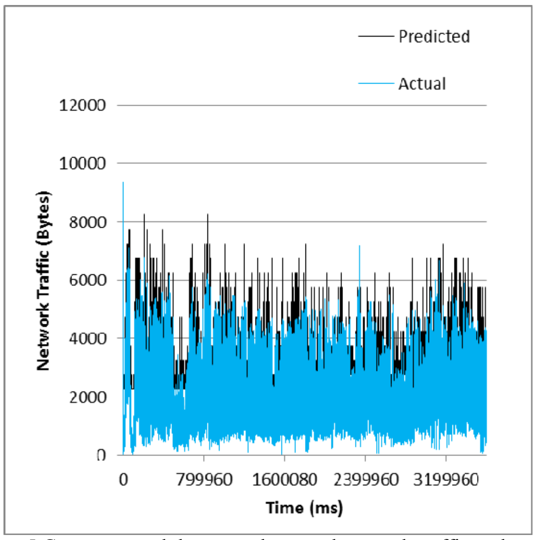

Figure 5 Compare result between the actual network traffic and predict network traffic

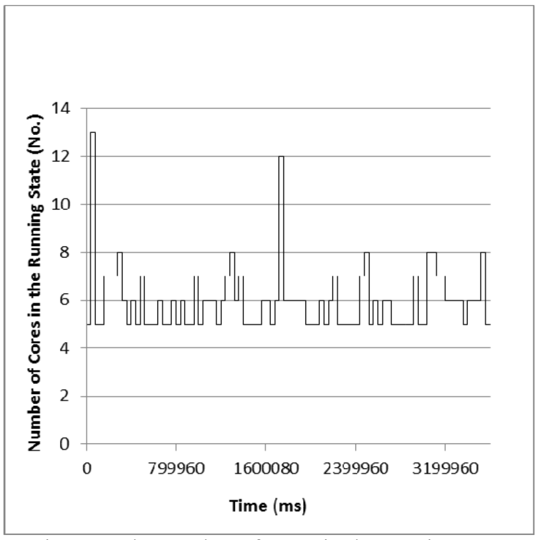

Figure 6 The number of cores in the running state

Figure 7 is the comparison char of network traffic of system can process and actual traffic. It can be seen, system can process all the traffic in most cases. If system cannot process all the traffic, there will be packet drop. The packet drop can be seen from the difference between actual traffic and system can process traffic. The packet loss is not serious from Figure 8.

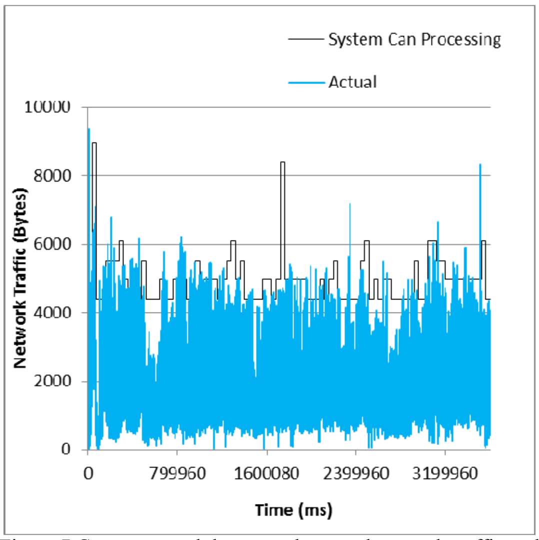

Figure 7 Compare result between the actual network traffic and network traffic system can process

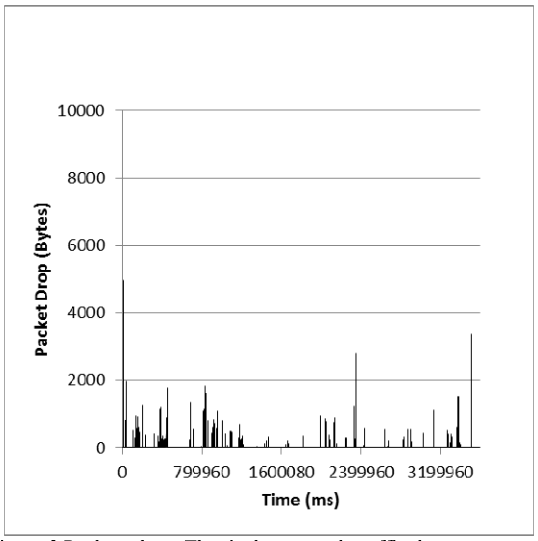

Figure 8 Packets drop. That is the network traffic the system can't process

In the system run time, the average CPU utilization is 30%. If we use a constant multi-core software architecture, not the predicted method proposed in this paper, which the number of running core is not change, the CPU utilization at a given value unchanged. According to the simulation data, 8 running core can handle all the network traffic all the time. That was the CPU utilization is 50%. This showed that the method proposed in the paper can reduce the CPU utilization by 20%, so the Dynamic Scheduling of Real-time Multi-core Subtask Based on Traffic Prediction is effective.

V. Conclusions

In this paper, we predict the media data stream traffic, in order to schedule the real-time multi-core subtask dynamically. This can be used to turn off some cores if traffic is not heavy. We use Markov Model for media data network traffic prediction. We used P-SPL as multi-core network software architecture. We took 3 stages pipeline as example, made a theoretical derivation. And then algorithm pseudo-code was put forward. The conclusion which come from simulation is in the condition of the drop rate is acceptable, the CPU utilization has fallen.

Acknowledgement

The authors wish to thank the reviewers for their valuable comments on this paper. This work was supported by the Key Technology Research and Verification of the NGB Intelligent Access Gateway, Key Deployment Project, Chinese Academy of Sciences under number KGZD-EW-103-4, Integration of Network Service System Development, 863 Project under number 2011AA01A102 and Key Technology Research of 3D TV System and Integrated Broadcast Platform, Science and Technology Support Program under number 2011BAH08B01.

References

[1] X. Chu, K. Zhao and M. Wang, Accelerating network coding on many-core GPUs and multi-core CPUs. Journal of Communications, 2009. 4(11): p. 902-909.

[2] D. Wang, J. Chen and W. Zhao, A Task Scheduling Algorithm for Hadoop Platform. Journal of Computers, 2013. 8(4): p. 929-936.

[3] N.K. Groschwitz and G.C. Polyzos. A time series model of long-term NSFNET backbone traffic. in Communications, 1994. ICC '94, SUPERCOMM/ICC '94, Conference Record, 'Serving Humanity Through Communications.' IEEE International Conference on. 1994.

[4] W. Duo and L. Guanghong, IP Traffic Matrix Estimation Based on Ant Colony Optimization. Journal of Computer Applications, 2013(1): p. 92-95.

[5] W. Peng, D. Zhimin and W. Weiling. A simple and efficient MPEG-4 video traffic model for wireless network performance evaluation. in Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE. 2004.

[6] M. Krunz and S.K. Tripathi, On the characterization of VBR MPEG streams. Performance Evaluation Review, 1997. 25(1): p. 192-202.

[7] D. Arifler and B.L. Evans. Modeling the self-similar behavior of packetized MPEG-4 video using wavelet-based methods. in Image Processing. 2002. Proceedings. 2002 International Conference on. 2002.

[8] F. Hong and Z. Wu. Multifractal analysis and model of the MPEG-4 video traffic. in Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International. 2003.

[9] F.H.P. Fitzek and M. Reisslein, MPEG-4 and H.263 video traces for network performance evaluation. Network, IEEE, 2001. 15(6): p. 40-54.

[10] T. Zhichao, Analysis of the Characteristics of Multimedia Traffic. 2009, Southwest Jiaotong University Master Degree Thesis.

[11] S. Hongtao, Traffic Prediction Model Based on Wavelet and Markov Chain. 2010, Ocean University of China.

[12] Y. Xiong, M. Wu and W. Jia, Delay prediction for real-time video adaptive transmisson over TCP. Journal of Multimedia, 2010. 5(3): p. 216-223.

[13] H. Han, et al., Efficient algorithm for hardware/software partitioning and scheduling on MPSoC. Journal of Computers (Finland), 2013. 8(1): p. 61-68.

[14] T. Mingwang and L. Hui, A Task Scheduling Algorithm Based on DAG. Journal of Wuhan University of Technology, 1999(5): p. 42-45.
[15] J. Verdu, et al. MultiLayer Processing - An execution model for parallel stateful packet processing. in 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS '08, November 6, 2008 - November 7, 2008. 2008. San Jose, CA, United states: Association for Computing Machinery.
[16] W. Tao and L. Daxin, The Performance Evaluation of Rate Monotonic Tasks Assignment Algorithms on Multiprocessor. Computer Science, 2007(1): p. 272-277.
[17] Z. Wan, G. Liang and T. Li, Multi-core processors based network intrusion detection method. Journal of Networks, 2012. 7(9): p. 1327-1333.
[18] Y. Song, et al., BENCHMARKING APACHE ON MULTI-CORE NETWORK PROCESSOR PLATFORM. 2011 3RD INTERNATIONAL CONFERENCE ON COMPUTER TECHNOLOGY AND DEVELOPMENT (ICCTD 2011), VOL 1, 2012: p. 293-297.

Yi Song is a PhD student in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing. He received the B.S. degree from the Information Engineering School at University of Science & Technology Beijing. His research interest is network new media technology.

Wu Zhang is a Research Associate in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing. He received the PhD degree from Institute of Acoustics, Chinese Academy of Sciences. His research interest is computer network.

Hong Ni is a Research Fellow in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing. He received the master degree from Institute of Acoustics, Chinese Academy of Sciences. His research interest is multimedia network technology.

Xiuyan Guo is an Assistant Researcher in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing. He received the PhD degree from the University of Science and Technology of China. His research interest is multimedia network technology.