Design and Implementation of H.264 Video Transmission Network Simulation

Weimin Lei, Hao Zheng, and Wei Zhang

***Abstract*—To design and implement the network simulation scenario of H.264 video service based on OMNeT++ simulation platform can provide testing, validation and evaluation method for the key technologies, such as to evaluate the QoE (Quality of Experience) of network video service, to evaluate multipath transport control mechanism. Based on H.264 codec standard and RTP payload structure, real-time transmission mechanisms for H.264 video transmission frame is studied. Based on standard host structure of INET Framework, three logical entities are designed, including user agent, relay controller and relay server. The topology structure of simulation networks and logical entities are described by network description language (NED). The key is to design and implement RTP node to support H.264 service. RTP node realizes sending and receiving module in INET Framework. By importing video files and relevant configuration files, simulation of H.264 service based on OMNeT++ is conducted. The effect of delay and packet loss on quality of H.264 video session is evaluated by different network configuration parameters.**

***Index Terms*—H.264 codec, video transmission, RTP, OMNeT++.**

I. Introduction

The transmission of network video data has several features [1], including high bandwidth requirement, tolerance of a certain range packet loss, and delay sensitivity, etc. H.264 codec standard has high performance in data compression. Compared with MPEG-4, the bit rate can be reduced by 50%. In the case of the same bit rate, signal-to-noise ratio is improved obviously [2]. Therefore, H.264 codec standard is widely used in multimedia communication. In addition, in order to ensure continuity of video data, it must choose an appropriate video transmission protocol to guarantee the quality of video transmission. RTP is used for the transmission of multimedia data streams on Internet. RTP provides end-to-end delivery service for multimedia data with real-time characteristics, and realizes the streams synchronization [3]-[5].

Accurate and efficient simulation platform can save manpower and material resources, thus speed up the research progress. Simulation is an important method to verify the correctness of design schemes and evaluate quality of service (QoS). Simulation of real-time video transmission is an important way to study the key research technologies, such as to evaluate the quality of experience (QoE) of video service transmitted by the Internet, to evaluate control mechanism and transport protocol. It can also provide testing, validation and evaluation method for related research work.

The goal of this paper is to design and implement the network simulation scenario of H.264 video service based on OMNet++ simulation platform, and to analyze RTP and the characteristics of H.264 codec standard in detail. Secondary development based on OMNeT++ simulation platform is also discussed in detail in this paper. And the simulation of H.264 video transmission is also validated.

The rest of this paper is organized as followss. First, Section II introduces OMNeT++ simulation tool and RTP protocol stack. Then Section III discusses the design and implementation of H.264 video session network simulation based on OMNeT++ platform. Next, Section IV discusses validation and analysis of simulation results. Finally, the conclusions and future prospects are given in Section V.

Manuscript received January 28, 2015; revised October 23, 2015.

Weimin Lei, Hao Zheng, and Wei Zhang are with the Institute of Communication and Information System, College of Information Science and Engineering, Northeastern University, Shenyang, China (e-mail: leiweimin@ise.neu.edu.cn, zhenghao92a@163.com, zhangwei1@ise.neu.edu.cn).

II. OMNeT++ and RTP Protocol Stack

A. OMNeT++ Simulation Platform

OMNeT++ [6] is an object-oriented modular discrete event network simulation framework. It has a generic architecture, so it can be used in various problem domains. Its core ideas are module-based structures and message mechanisms. OMNeT++ simulations can be run under various user interfaces. Graphical, animating user interfaces are highly useful for demonstrations and debugging purposes, and command-line user interfaces are best for batch execution. It also provides an embedded simulation kernel, so it is compatible with Windows, Linux and other operating system. Due to the universal and flexible architecture, OMNeT++ has been successfully applied to many fields, such as protocol modeling, modeling of queuing networks, evaluating performance aspects of complex software systems, and distributed hardware systems. In general, modeling and simulation of any system where the discrete event approach in suitable, and can be conveniently mapped into entities communicating by exchanging messages.

Modules can be connected with each other via gates, and combined to form compound modules. The depth of modules nesting is not limited. Modules communicate with message passings, where messages may carry arbitrary data structures. Massages can be sent either via connections that span modules or directly to other modules. The user can configure the properties of connections, including bandwidth, delay and bit error rate. Massages can be sent from source modules to objective modules via gates according to the connection configuration parameters. And the direct transmission can send the messages to the destination module through the simulation kernel.

doi: 10.18178/jacn.2015.3.4.180

OMNeT++ uses NED (Network Description) topology description language to define network structure. Namely, the connections between modules, simple module, compound modules and network topology can be defined using NED. It can greatly simplify the process of simulation. Module behaviors uses the C++ code to define. And network simulation environment and the input or output data can be set in the INI files (usually named omnetpp.ini).

OMNeT++ itself is not a simulator of anything concrete, but rather provides infrastructure and tools for writing simulations. But it has developed a lot of modules or frameworks to solve the problems. For example, INET Framework is an open communication network simulation package based on the OMNeT++ platform, which contains many modules and supports of wired and wireless network protocols, such as UDP, TCP, SCTP, IP, Ethernet, PPP, 802.11 etc. We can carry on the secondary development in the modules or frameworks according to the demand. It will greatly reduce the workload of simulation and speed up the research progress [7].

B. RTP Protocol Stack in OMNeT++

To realize real-time transmission of video sessions is based on the RTP. RTP payload data packages are the original video data transported from the application layer to transport layer, and finally reach the network layer. If data packages are not divided before the application layer and loaded into the RTP packets, it will produce packets larger than the maximum transmission unit (MTU) of IP network. It will cause the loss of data packages. However, in the corresponding standard, there is different payload format. Therefore, the network layer will split big blocks into several smaller packets than the IP MTU, and then the smaller packets will be transported by RTP protocol avoiding severe packer loss rate.

The simulation of video data transmission service uses the RTP protocol stack in INET Framework. Through the above analysis, this function of video data transmission involves several specific simulation modules in OMNeT++.

Module1: (RTPApp module) The RTPApp module is a simple application module using RTP protocol. The module is used for processing the RTP data packets from the transport layer, or sending the media data to the transport layer. There are several important parameters in this module, such fileName, profileName and payloadType. “fileName” parameter determines that the RTP applications are used as the sender or receiver. When fileName is set, RTPApp acts as the receiver. If the fileName is empty, then RTPApp is used as the sender. “profileName” is a profile created by RTP module, usually it is named RTPAVProfile. “payloadType” is the type of sending module created by the RTPProfile, which is used to mark the specific type of transmission files.

Module2: (RTP module) The RTP module is an important function entity of the design. It is the core part of video stream transmission. RTPApp requires RTP module to generate a profile module of the transmission and to initialize it. The RTP module is responsible for communication with application and UDP layer.

Module3: (RTPProfile module) The RTPProfile module is used to control and generate sending and receiving module. According to the payloadType and profileName, RTPProfile generates sending and receiving module dynamically. They are named:

RTP<profileName>Payload<payloadType>Sender

and

RTP<profileName>Payload<payloadType>Receiver

respectively.

Module4: (RTPPayloadSender module) The RTPPayloadSender module is used for getting RTP data packets. It contains RTP packet features, such as opening and closing data files, choosing the initial sequence number and time stamp value, and reading the parameter MTU (its initial value is from the profile module). For a specific codec, the packing process of packets is completed in this module.

Module5: (RTPPayloadReceiver module) The RTPPayloadReceiver module is used for processing packets sent by the sending module. At initialization time, it opens an output file and generates a queue to store receiving packets. RTPInnerPackets only have a packaged RTP data packet in the processing time.

The functions and relationships of above several modules are introduced as depicted in Fig. 1.

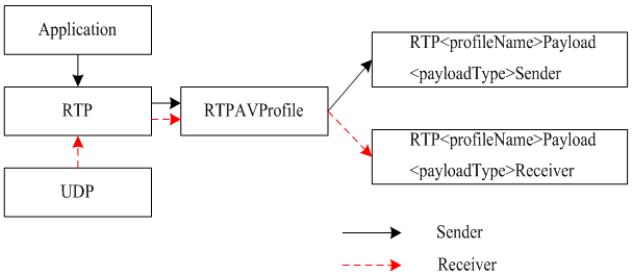


Fig. 1. Modules related to RTP-based real-time transmission and their connection relationship.

At the sending end, application layer module require RTP module to create the sender module of specific codec according to ProfileName and PayloadType. And the above two parameters are defined in the configuration file omnetpp.ini. RTP module will deliver this task to RTPProfile to accomplish. The sender module inherits the RTPPayloadSender module and overrides the data organization method of packaging, namely sendPacket().

At the receiving end, RTP module will send received RTP data packets to RTPProfile module. RTPProfile module will create data receiving Profile based on payloadType in the RTP Packet Header and ProfileName defined in the omnetpp.ini file. The receiving module inherits the RTPPayloadReceiver module and overrides the data organization method of unpacking, namely processPacket().

III. Design and Implementation of H.264 Video Transmission Network Simulation

A. Overview

The simulation of H.264 transmission service uses the RTP protocol stack in INET Framework. In order to support the H.264 service, the RTP module creates H.264 sending and

receiving module dynamically. RTPAVProfilePayload264Receiver module and RTPAVProfilePayload264Sender module are defined. The RTPAVProfilePayload264Sender module completes the extraction and packing of H.264 video data, and the RTPAVProfilePayload264Receiver module completes the extraction and unpacking of H.264 video data according to the RFC3984 [8]. Fig. 2 shows the process of creating sending and receiving Profile by the user agent.

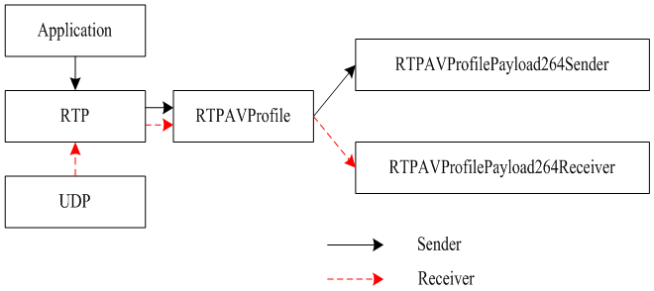


Fig. 2. Modules related to H.264 RTP-based real-time transmission and their connection relationship.

The H.264 video data transmission over the Internet must go through application, transport and network layer. The original H.264 video data is as the RTP payload. In the application layer, H.264 NAL unit is obtained. And RTP adds a RTP header to the NALU to make it to be a RTP data package carrying the H.264 video data. In the transport layer, UDP adds a UDP header to the RTP header, and make it to be a UDP package. In the network layer, UDP package is added a IP header and transmitted through the underlying network. The process of packing and transmitting of H.264 video data is described as Fig. 3.

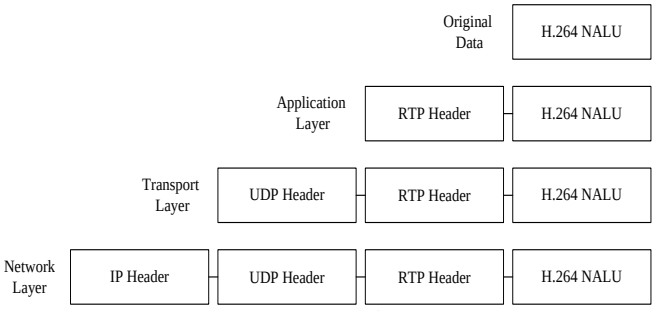


Fig. 3. Packet encapsulation of H.264 video data.

And in the receiving end, unpacking process is absolutely opposite. The packages go through network, transport and application layer. Finally, H.264 NAL units are obtained.

B. RTPAVProfilePayload264Sender Module

In the RTPAVProfilePayload264Sender module, H.264 video data is packed to be one or more RTP packages, which is determined by the size of H.264 video data.

According to RFC3984, an original NAL unit is composed of three parts. They are [Start Code], [NALU Header] and [NALU Payload]. [Start Code] is used to mark the start of a NAL unit, and it must be 0×00000001 or 0×000001. [NALU Header] is only one byte. All of the following is content of NAL unit. Therefore, in order to pack H.264 video data into the RTP packages, [Start Code] must be removed. For the NALU output stream, firstly, read the first 3 bytes to determine whether it is start code 0×00000001 or 0×000001, and then continue to read the data. Finally extract the data between two [Start Code] into a NALU. Then simulation should continue to search down. After the above process, the NAL units can be obtained. Then the NAL unit will be sent to the packing module.

Because of the huge data of H.264 video, the extracted NALU data has two ways to be packed. We need to determine whether the length of NALU is larger than MTU (this paper set MTU 1400 bytes). When the data length is less than 1400 bytes, we take single NALU package model. For H.264 video data, it is more likely that NALU length exceeds the preset threshold value. When the length of NALU is more than 1400 bytes, it has to carry on patch processing, which is suitable for network transmission, namely patch package model. So the implementation process of H.264 sending module (RTPAVProfilePayload264Sender) is shown in Fig. 4.

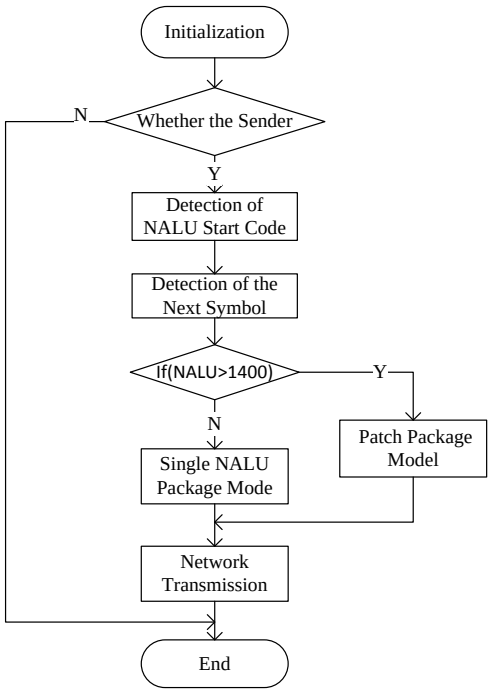


Fig. 4. Processing flow chart of RTPAVProfilePayload264Sender module.

In OMNeT++, *_m.h and *_m.cc files are created to support message member operation at compile time. The two files consist of declarations and definitions for message members. In order to define the behavior of sending module, this design creates a C++ file to describe it. Using include method, message operation methods in *_m.h file can be used in the C++ file. So this design creates the sending C++ files, RTPAVProfilePayload264Sender.h and RTPAVProfilePayload264Sender.cc to define the behavior of sending module.

Sending C++ files sends the messages to the destination module from the gate that is specified by the gatename and gateindex, using "send (cMessage *msg, const char *gatename, int gateindex=-1)" methods.

C. RTPAVProfilePayload264Receiver Module

After the above process, we have already got H.264 NAL unit. And using the single NALU package model or patch package model packs the H.264 NALU to be a RTP package.

According to the type of RTP data packages,

RTPAVProfilePayload264Receiver module chooses single NALU package or patch package model to unpack. If it is single NALU package model, RTP payload is a entire H.264 NAL unit without [Start Code]. The unpacking method, processPacket(), adds a header of NAL unit to restore to original H.264 video data. If it is patch package model, processPacket() determines the order of packages through the flag bits of fragmentation units, flag e and flag s. The receiving process of H.264 video data is shown in Fig. 5.

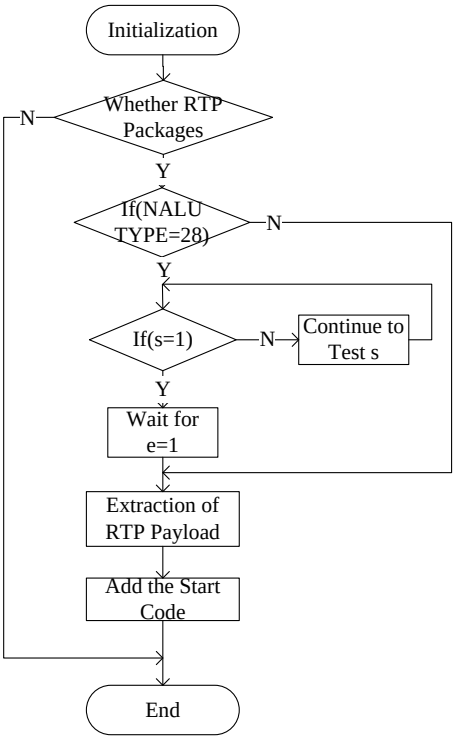


Fig. 5. Processing flow chart of RTPAVProfilePayload264Receiver module.

In the RTPAVProfilePayload264Receiver module, receiving C++ files are created. They are RTPAVProfilePayload264Receiver.h and RTPAVProfilePayload264Receiver.cc. Receiving C++ files solve self messages and the messages which are from other modules. When solving the messages from other modules, receiving C++ files use handleMessage (cMessage *msg) method. In addition, IsSelfMessage () and getArrivalGateId () method are used to judge the sources, then select the different processing functions to send the messages out.

D. Logical Entity Structure and Simulation Environment Configuration

At the beginning of the simulation, initialize() method reads configuration parameters from omnetpp.ini and *.ned files. So the first step is to create logical entity structure and configure simulation environment in the omnetpp.ini and *.ned files.

Compound modules mainly consist of three parts, namely parameter, submodule and connection. Parameter defines the parameters of module, which can be assigned in these NED files and also can be assigned in the omnetpp.ini files. Submodule defines the sub modules which are belong to compound modules. Connection defines the connections between sub modules of composite modules.

Based on the standard host structure of INET Framework, this simulation creates a specific host as the sending and receiving module. They are H1 and H2. The purpose of this design is to realize that H.264 video data can be transmitted between these two nodes. Their working principle is described as Fig. 3. H1 and H2 is responsible for sending and receiving H.264 video data. The packing and unpacking works are finished in these two modules. The structure of H1 and H2 is as shown in Fig. 6.

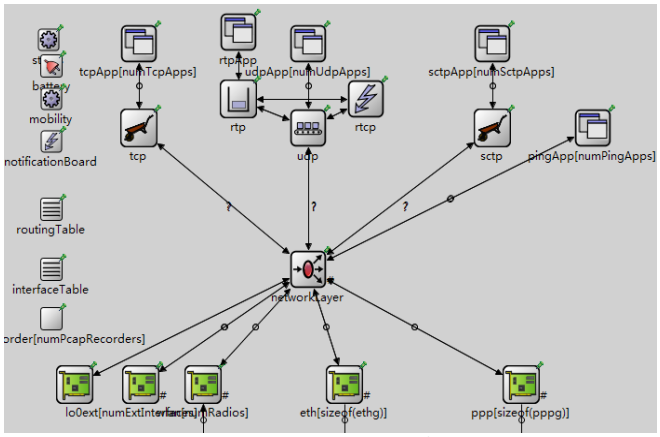


Fig. 6. The simulation structure of RTP host.

In the NED file of H1 and H2, this network adds application, rtp, rtcp and udp sub module. And using NED language defines the connection between each gate of sub modules. For example, the definition method of rtpapp and rtp is described as follows:

rtpapp.rtpOut --> rtp.appIn;

rtpapp.rtpIn <-- rtp.appOut;

Besides above gate parameters in the NED file, parameters of sending and receiving module are configured in the omnetpp.ini file, including fileName, destinationAddress, portNumber, bandWidth and profileName. “fileName” is used to mark the name of sending file. If it is receiving module, fileName is empty. “destinationAddress” is used to mark the destination of sending or receiving module. “profileName” can link the omnetpp.ini file to the RTPAVProfile. For the sender H1, fileName is “send.h264”; destinationAddress is H2; profileName is inet.transport.rtp.RTPAVProfile; portNumber is 5004; bandwidth is 8000. For the receiver H2, fileName is empty; destinationAddress is H1; profileName is inet.transport.rtp.RTPAVProfile; portNumber is 5004; bandwidth is 8000; outputFileName is rcv.h264. By configuring like this, the network topology can implement the communication of sending and receiving module.

IV. Simulation Instance and Result Analysis

To realize the transmission of H.264 video data in OMNeT++, this paper designs a kind of feasible network topology structure according to above methods of connecting. Each function entity and network topology of the RTP package transmission of H.264 video data are as shown in Fig. 7.

After configuring the network environment, it can be run in the OMNeT++ simulation platform. The simulation process is shown in Fig. 8. Fig. 8 is the Tkenv interface, which shows the details of specific events, including time, number, action modules and the module of triggering the

action. It is obvious to find the modules which are sending or receiving data packets in the Tkenv interface. It verifies the validation of the network topology which is designed in this paper.

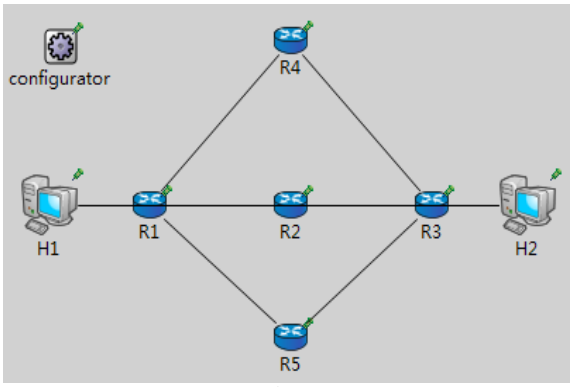


Fig. 7. A simulation instance of H.264-based video transmission.

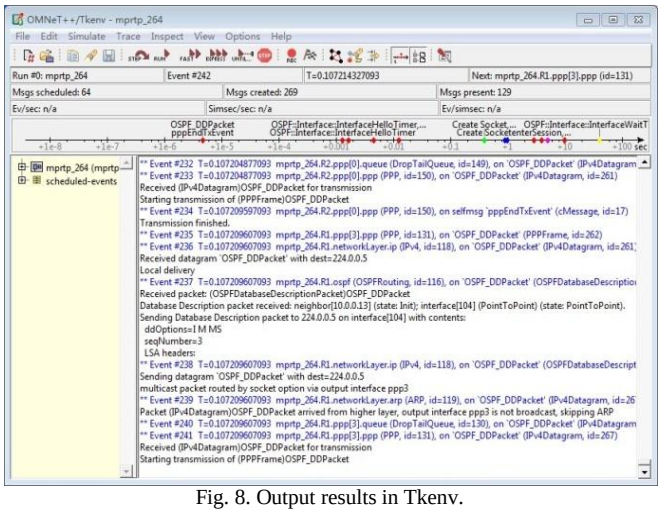


Fig. 8. Output results in Tkenv.

The receiving module receives the H.264 packages transmitted by RTP and recombines data packages. Then the data packages are transmitted to the upper RTP module, and finally recovered to the original H.264 coded video stream by the receiving Profile (RTPAVProfilePayload264Receiver).

Next, the correctness of this design scheme and the error rate effects on the quality of video images are discussed in this paper. Namely, the time delay and other variables are fixed, and the error rate changes. In this situation, the fiftieth frame and ninetieth image from the sending end and receiving end video stream respectively are got and compared. The simulation comparison is depicted as shown in Fig. 9.

Through comparing of (a) and (b), it is not sufficient to affect the quality of experience and find the differences of image quality in the appropriate circumstances. These two images show that it has no impact on video image quality transmitted through the network topology designed in the simulation to implement RTP transmission of H.264 video data packages. The conclusion is using the RTP protocol can accurately pack and transmit H.264 video data.

Through comparing of (a) and (c), it is obvious to find that image quality has changed a lot when time delay is same, and error rate changes. It indicates that the H.264 video data packages are transmitted through the designed simulation network. And it also clearly shows the effect of bit error rate on image quality. And by comparing the size of sending and receiving files, we find the receiving file is smaller than the sending, which is indicating that the error rate affecting on image quality, mainly reflects in the loss of a portion of the data. So it affects the difference of visual effects and image quality.

delay = 0.1us;
ber=0;
(a) sending end

delay = 0.1us;
ber = 0;
(b) receiving end

delay = 0.1us;
ber = 0.00001;
(c) receiving end

Fig. 9. Image comparison.

V. Conclusions and Prospect

As the new generation efficient video codec standard, H.264 is more and more mature in the network compatibility and codec efficiency. To further promote the development of real-time multimedia communication, we must choose an appropriate transmission protocol to guarantee the transmission quality of video streams on the Internet. Combining the H.264 video codec standard and RTP protocol, this paper researches NALU of H.264 and RTP payload format for H.264. In addition, single NALU package model and patch package model are also discussed.

Based on the above theory, the simulation scenario of H.264 video session is designed and implemented by using

OMNeT++ simulation tool. It simulates transmission link status. This paper design that the RTP node supports H.264 video service, thus the sending and receiving module are realized. By simulating the transmission link statement, such as changing the bit error rate and delay, different H.264 video image qualities of RTP transmission are obtained. By comparing the video image effects of sending and receiving end, it verifies the validation of this design and infers the possible factors affecting the quality of video images. This work provides a basis for the error recovery method of H.264 video coding standard in further research.

Although this paper has already achieved the expected goal, there are still a lot of works to finish:

1) The design is based on a default route to realize the transmission of H.264 video data, which can be extended to multipath network to realize the transmission of video data;
2) Error control and flow control have no testing and implementation in RTP transmission;
3) Video data encryption and decryption can be considered in this design.

In a word, this paper realizes the packing and unpacking process of video data combining the H.264 video codec standard and RTP based on OMNeT++ simulation tools. The further researches can also study as the extension and expansion of this design.

ACKNOWLEDGEMENTS

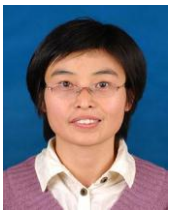
This work was supported by the National Natural Science Foundation of China (No. 61401081).

REFERENCE

[1] H. J. Bi and T. Li, "Latest progress about video communications," *World Telecommunications*, vol. 4, pp. 32-36, 2004.
[2] L. Qin, H. J. Wang, D. X. Zha, and Z. Y. Wu, "Main technological characteristics of video coding standard H.264 and its application expectation," *Microcomputer Applications*, vol. 25, pp. 449-455, July 2004.
[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC 3550: RTP: A transport protocol for real-time applications," *IETF*, July 2003.
[4] K. Shi, J. T. Luo, and Z. Z. Zhang, "Transmission of H.264 video over wireless networks based on RTP and QoS control," *Microcomputer Applications*, vol. 25, pp. 162-164, July 2009.
[5] H. R. Xu and M. S. Li, "Real-time video transmission subsystem based on RTP," *Computer Engineering and Design*, vol. 26, pp. 876-878, April 2005.
[6] A. Varga. (July 2010). OMNET++ discrete event simulation system version 4.3 user manual. [Online]. Available: http://www.omnetpp.org
[7] INET framework for the OMNeT++ discrete event simulator. [Online]. Available: http://github.com/inet-framework/inet
[8] S. Wenger, M. M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RFC 3984: RTP payload format for H.264 video," *IETF*, February 2005.

Weimin Lei was born in Shanxi, China, in 1969. He received the B.S. degree from Nankai University, Tianjin, China, in 1992, the M.S. degree from Institute of Computing Technology, Chinese Academy of Sciences in 1995, and the Ph.D. degree from Dalian University of Technology, China, in 1999. From 1999 to 2009, he was a professor in the Institute of Computing Technology, Chinese Academy of Sciences. Since 2009, he has been a full professor with the Institute of Communication and Information System, College of Information Science and Engineering, Northeastern University, China. He has published more than fifty research papers in IEEE journals and international conferences. He is the inventor of several Chinese patents and pending applications. His research interests include: protocols and services in IP multimedia system, multi-path transmission, overlay network, SDN and ad-hoc network.

Hao Zheng was born in Liaoning, China, in 1992. She received the B.S. degree from Northeastern University, Shenyang, China, in 2014. She is currently pursuing the M.S. degree in information science and engineering at Northeastern University, Shenyang, China. Now she is studying following Prof. Lei. Her research interests include protocols and services in IP multimedia system, multi-path transmission, overlay network.



Wei Zhang was born in Shandong, China, in 1980. She received the B.S. degree in computer science and technology from Ocean University of China in 2002, M.S. degree from Institute of Computing Technology, Chinese Academy of Sciences in 2005, and Ph.D. degree from Northeastern University, China, in 2014. From 2005 to 2009, she was a research assistant of Institute of Computing Technology, Chinese Academy of Sciences. Since 2009, she has been a lecturer of Institute of Communication and Information System, College of Information Science and Engineering, Northeastern University, China. She was published more than five research papers in IEEE journals, and submitted three IETF drafts. She is also the inventor of two Chinese patents and two pending applications. Her research interests include multimedia communication, protocols and services in next generation network architecture of cloud computing.