UKACC International Conference on Control 2012
Cardiff, UK, 3-5 September 2012

Video transmission on JXTA

Zhan-Yu Wang
School of Astronautics
Harbin Institute of Technology
Harbin, China
wangzhanyu@hit.edu.cn

Guo-Ping Liu[1,2]
1.School of Astronautics
Harbin Institute of Technology
Harbin, China
2.Faculty of Advanced Technology
University of Glamorgan
Pontypridd, U.K.
gpliu@hit.edu.cn

Zhen-fu Cui
School of Computer Science and Technology
Beijing Institute of Technology
Beijing, China

***Abstract*—By analyzing the peer classification, structure, and configuration of juxtapose framework (JXTA), a mixed topology in a campus network environment is designed and implemented for peer-to-peer instant communication. A novel method of transmitting video data on JXTA overlap network is proposed, by which video can be included in JXTA applications. The feasibility is proved, furthermore, the performance is tested.**

Keywords- P2P; video transmision; JXTA

I. INTRODUCTION

In recent decades, P2P technology is increasingly used in instant message applications and file sharing systems based on internet. There has appeared many open source or private P2P frameworks provided for developers, by which users can implement their own communication softwares, such as JXTA[1], Self-Chord[2] and [3]. All above frameworks facilitate our developing of transmitting text messages, but they are not convenient to transmitting video. However, Red5[4], another powerful P2P framework, provides both text and video transmission interfaces, it is not widely used restricted by its size of hundreds Mbs.

Considering the raising performance and increasing bandwidth of internet devices, video transmission is not problem any longer for common local PC or even remote PC on wide area network scale. So, a cross-platform P2P instant messaging system with video transmission is quite significant for users.

JXTA, pronounced 'juxta', comes from the word juxtapose, that reflects the operations by which peers establish temporary associations to form a P2P network. Reference [5] shows a method which sends a stream media file via JXTA pipe, it denotes that we can also receive real time video package on JXTA overlap network.

In this paper, we describe how we design and implement the transmission with JXTA version 2.6 comprehensively. It is also applied for version 2.7.

It is quite noticable for JXTA beginners that the URI which is offered by official containing Relay list and Rendezvous list have not worked any more after version 2.5[1]. We should establish our own Rendezvous seed peer and Relay seed peer (if neccessary) on a known IP address, so that our JXTA peer can contact each other via the services provided by them.

This work was supported in part by the National Natural Science Foundation of China under Grant 61021002.

II. PEERS OF JXTA

There are three types of nodes in JXTA framework, edge peer, relay peer and rendezvous peer, they play different roles respectively when JXTA is running.

A. Edge Peer

An Edge peer is nearly the most frequently used peer in JXTA. As a basic JXTA peer, it tries to connect and to remain the connection to one and only one Rendezvous peer. An Edge peer can use both unicasting and multicasting to contact with other JXTA peers. Each communicate mode can be implemented as TCP or HTTP by modifying the configuration file before Edge peer starts. If multicasting is enabled, an Edge peer can communicate with other Edge peers directly on LAN.

B. Relay Peer

A Relay peer is a special Edge peer who has routing ability. We can compare a Relay peer to a router on network. It is needed when a peer with private IP attempts to contact peers from WAN. Typically, a remote peer wants to establish a connection to NATed peer or peer behind firewall, it should try to find a route which can go "through" NAT server or firewall device under the help of Relay peer. It provides means for peers having private IP on a LAN to become reachable. A Relay peer must have a public IP, namely, a private IP is useless for Relay peer.

C. Rendezvous Peer

Rendezvous peers are Edge peers those offer services of JXTA rendezvous for a peer group. A peer group can assimilate to a JXTA LAN, and Rendezvous peer can assimilate to a gateway of this JXTA LAN. So, a peer group should have at least one Rendezvous peer. Edge peers can know each other via Rendezvous peer who performances as the leader of the group. Rendezvous peers accept lease requests from Edge peers and propagate these messages through the peer group to Edge peers.

III. PEER CONFIRGURATION AND DEPLOYMENT

JXTA beginners are always confused for constructing their first JXTA environment, because of their lack understanding of the roles those JXTA peers play. Similarly, we can see different

type of peers as different network devices with the help of network knowledge. Edge peers work as the common PC in LAN, rendezvous peer can be considered as the gateway PC of this LAN, and we can regard relay peer as the router that makes local peers contact other peers out of the LAN.

According version 2.5 of JXTA and former versions, there was URI http://rdv.jxtahosts.net/cgi-bin/rendezvous.cgi?3 plays the role of default Rendezvous seed. People can use it as a static Rendezvous peer in their own systems. But now, it is not valid any longer. Users need to configure different kinds of peers and deploy them on internet.

If you want to start a peer, at first you should set type of peer, such as Rendezvous or Relay peer, even Edge peer, these codes are given in Figure1.

```
1 new NetworkManager(NetworkManager.ConfigMode.type);
2 NetworkConfigurator.addSeedRendezvous(rendezvousURI);
3 NetworkConfigurator.addSeedRelay(relayURI);
4 NetworkManager.startNetwork();
```

Figure 1. Code of peer's configuration

After the codes on line 1, the seed should be added to the peer you implement, then your peer can communicate with other LAN peers via Rendezvous, even WAN peers via Relay.

These configure operations must be done before you start the JXTA. Once *startNetwork()* method runs, the JXTA framework works. It will cost few seconds to initiate the low layer program.

As mentioned before, different type peers play different roles respectively on JXTA network. For a JXTA application running on LAN, the organization is as simple as shows in Figure 2. All peers on LAN compose the overlap network of JXTA. At least one Rendezvous peer is required to this peer group, and other peers can communicate with each other, for example, using pipes.

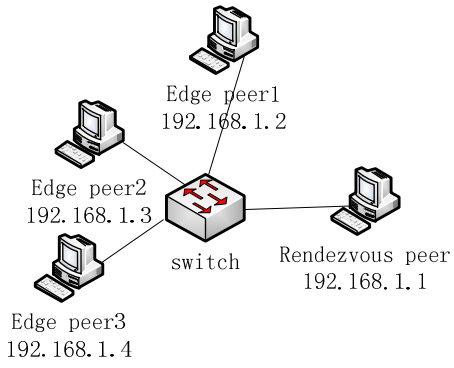

Figure 2. Topology of JXTA overlap network on LAN

But in actual case, only LAN scope is not enough for a more complex application, which peers are required to send and receive data through NAT devices or firewall devices even both of them. In this case, Relay peer works as router to help peers become visible to others, and Rendezvous peer plays a role like gateway to help Edge peers, which connect to it directly, to communicate with other any peers those connect indirectly.

Figure 3 demonstrates a typical topology of JXTA wide area network. A classic mixed structure is adopted to fulfill our design, which contains both peers and server that runs as the register server to let peers logon the JXTA network. Meanwhile, the server is a special configured peer that is not only the Rendezvous, but also a Relay peer.

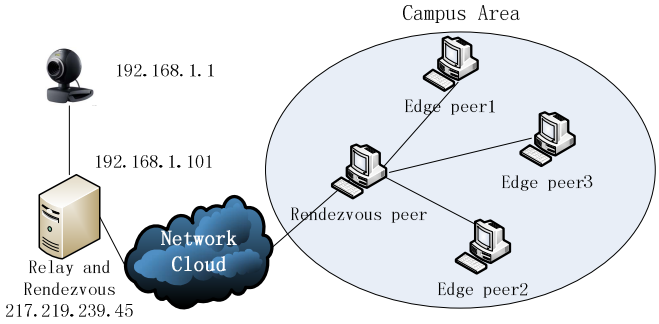

Figure 3. Topology of JXTA overlap network on WAN

IV. COMMUNICATION AMONG PEERS

JXTA provides three kinds of data communication methods, singlepipe, bidirectionpipe and multicast socket, whose services are implemented based on TCP socket, and these functions are the most important of JXTA communication.

A singlepipe is a single direction socket pipe. Two participants of singlepipe should possess the same PipeAdvertisement, which is the unique identifier of JXTA pipe in a JXTA network. It means that the later joined peer must know the PipeAdvertisement created by the former peer, namely the PipeAdvertisement needs to be hardcoded in the later joined peer, and data can only be sent from the former to the later.It is quite inconvenient to develop a temporary pipe in the practical programs.

Unlike singlepipe, a bidirectionpipe can send and receive data from either side of the pipe. Contributed by bidirectionpipe, we do not need to create two simple pipes for both side of a data interaction.

Multicastsocket plays important role in our software, because in many cases data needs to be sent from one peer to many other peers who have the advertisement of this multicastsocket. It is low-performing and inflexible to create many pipes for the data transmission, Instead, multicastsocket is quite appropriate for handling these jobs.

V. VIDEO TRANSMISSION

The working principle of IP camera that we use is to create 30 or less jpeg pictures and to send them each per second by a self-contained web server to browsers of clients via TCP/IP or HTTP links. And our purpose is to transmit these jpeg pictures on JXTA network by sender module below. So, jpeg pictures must be captured first.

A. *Video capture*

According to the CGI document provided by IP camera producer, we establish a HTTP connection with IP camera and get the byte stream of jpeg pictures. The code below, in Figure 4 shows how we get data from IP camera.

```
1 MediaLocator ml = new MediaLocator ( "192.168.1.101 ");
2 MyDataSource datasource = new DataSource(ml);
3 DataInputStream dis = null;
4 @override
5 public void connect(){
6      java.net.URL url = new java.net.URL(
7                          getLocator().toString());
8      url.openConnection().connect();
9      dis = new DataInputStream(
10           url.openConnection().getInputStream());
11     stream = new MJpegBufferStream(
12              getLocator(), dis);
13     }
14
```

Figure 4. Code of capturing video data from IP camera

As shown above, *ml*, the instance of class MediaLocator, will be set as the media locator in constructed function of class MyDataSource which extends PushBufferDataSource. In the overrided *connect()* method, *getLocator()* method will get the media locator and attach it to url, an instance of class URL. After connecting with url, variant *stream* is filled with class MJpegBufferStream object.

An instance of class MyDataSource is used in class MJpegBufferStream which implements PushBufferStream interface, and *read(Buffer buffer)* method is overrided in MJpegBufferStream class to push data into the parameter—buffer which is used be sent out by JXTA pipe.

B. *Sending and Receiving*

After capturing the Jpeg pictures from IP Camera, each Jpeg picture, namely the data in buffer, will be encapsulated as a DatagramPacket object, and sent out via JXTA pipe or JXTAMultiSocket. This sending process is implemented by a single class HttpSender. Then, the peer creates an advertisement that announces its resource, publishes and remote publishes this advertisement on JXTA network, so that other peers can discover it.

Instance of class HttpReceiver takes charge of receiving data of DatagramPacket type one by one from JXTA pipe or JXTA multicastsocket.

C. *Display*

One received DatagramPacket data is a MJpeg picture, it will be conducted by JPanelPlayer class which extends JPanel and responses to create a JPanel style player. This player is initiated in class CameraTest and once a picture is retrieved *updateScreen()* method in CameraTest would recall *repaint()* method to repaint the current picture on the JPanel instance. The relationship among all the classes is illustrated as UML diagram below, in Figure 5.

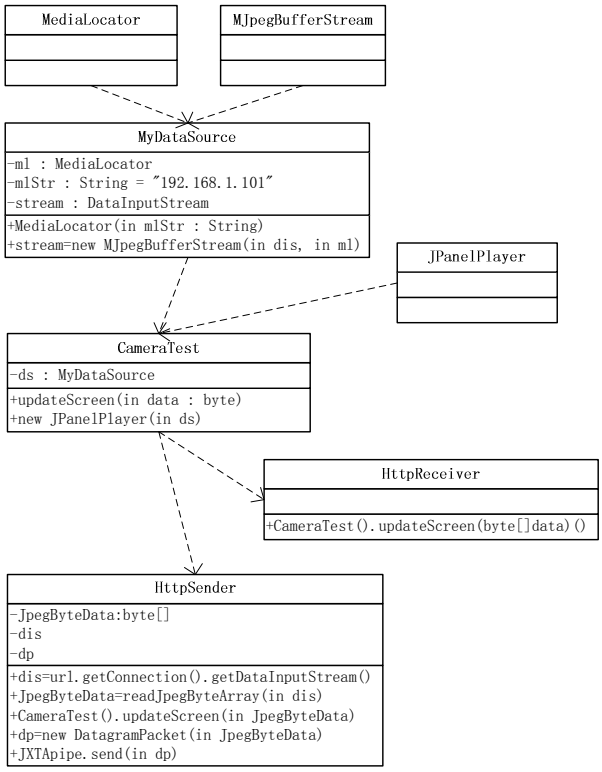

Figure 5. UML diagram of classes

VI. EXAMPLE

A. *Enviroment*

Table I shows both system and hardware devices information those participate in our test.

TABLE I. ENVIRONMENT OF TEST

Device	*Info*	*Peer mode*	*IP Address*
Server	CPU: i7 930 2.8Ghz Memmory:3.25G System:Win XP 32bit BandWidth of Enthernet:100M	RDV and Rendez-vous	217.219.118.45
IP camera		none	192.168.1.101
PC1	CPU: AMD 640x4 3.0GHz Memmory:3.25G System:Win XP 32bit BandWidth of Enthernet:100M	Edge	219.217.242.18
PC2	CPU: AMD 640x4 3.0GHz Memmory:3.25G System:Win7 32bit BandWidth of Enthernet:100M	Edge	172.17.52.83
PC3	CPU: Pentium D 3.0GHz Memmory:1G System:Win XP 32bit BandWidth of Enthernet:100M	Edge	202.118.233.57

Environment of test

A server accesses in internet with a public IP 217.219.239.45, and connects to an IP camera with private IP 192.168.1.101 directly. PC1, PC2 and PC3 locate on different network environments that we do not know clearly, and they are all the receivers which are used to test the performance of this video

transmission based on JXTA framework. The topology of this example is illustrated by Figure 6.

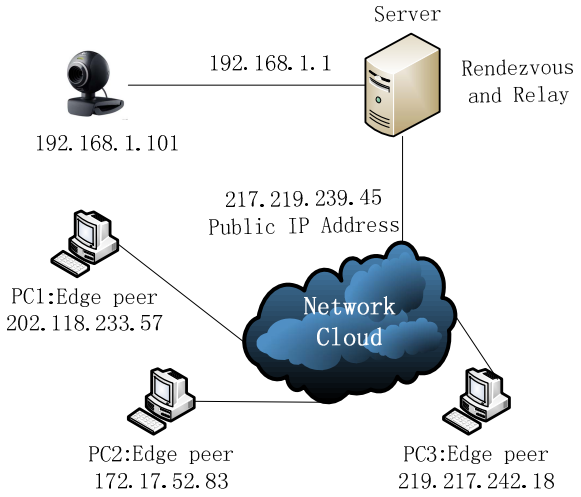

Figure 6. Test topology

B. Result

This software is tested in three different places. It firstly runs on an IP 219.217.242.18 that connects to server directly, After connection established, video is captured by the receiriver, and displayed in a JPanel style window that shows in Figure 7.

Figure 7. Captured video on IP 219.217.242.18

It is a direct connection between client and the server as shown in Figure 8.

Figure 8. Path between 219.217.242.18 and 219.217.239.45

Secondly, we run our application in another laboratory with IP 172.17.52.83, seemingly not in the same network segment to server's IP. The software received video data from server, and showed it by the player, as Figure9.

Figure 9. Captured video on IP 172.17.52.83

Figure 10 shows that there are four hops between this receiver and the server.

Figure 10. Path between 172.17.52.83 and 217.219.239.45

Finally, we test it in another campus of our university, and the IP is 202.118. 233.57, seemingly not in the same network segment to the IP of server either, and the data is received, as we can see in a display window of Figure11.

Figure 11. Captured video on IP 202.118.233.57

We find the three hops path from the local PC to server. That denotes the data can be delivered crossing different network segment, as shown in Figure 12.

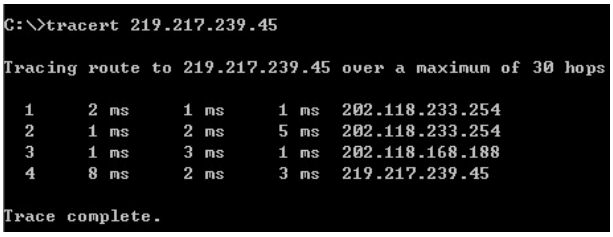

Figure 12. Path between 202.118.233.57 to 217.219.239.45

That denotes the data can be delivered crossing different network segment.

VII. CONCLUSION

The proposed video transmission system achieves sending and receiving video data via internet based on JXTA. It may be quite useful for developers who want to develop their own P2P instant message systems containing real time video image. By analyzing source code of JXT and CGI document of IP camera, we capture the video data from HTTP and divide to individual byte array for each MJpeg format picture. Then JXTA pipe or JXTA multicast socket is created to send them out. Finally, these video data can be received by other JXTA terminals on internet those run as the receiver. For original edition JXTA, for example version 2.7 and earlier releases, only text messages and XML format messages can be used by provided interfaces.

In this paper, we get a simple improvement of transmitting video via JXTA without any optimizing. So, the performance depends on network environment absolutely, this can be inferred indirectly in the result section of this article. As a large scaled application, P2P software should have an optimized structure to enhance its processing ability and increase it efficiency[6][7]. It is the target of our future research.

REFERENCES

[1] Jerome Verstrynge, Practical JXTA II, 2nd ed, Lulu Enterprises, Inc. (www.lulu.com), July, 2010, p 166.

[2] Agostino Forestiero, Emilio Leonardi, Carlo Mastroianni, "Self-Chord: A Bio-Inspired P2P Framework for Self-Organizing Distributed Systems", IEEE/ACM Transactions Networking, vol.18, p1651-1664, October 2010.

[3] Silvia Rueda, Pedro Morillo, Juan M. Orduna, "A Peer-To-Peer Platform for Simulating Distributed Virtual Environments," Proceedings of the 13th International Conference on Parallel and Distributed Systems – ICPADS, vol 2, 2007.

[4] Wang Dongjin, Xu Ke, "Red5 Flash Server Analysis and Video Call Service Implementation," Proceedings - 2010 IEEE 2nd Symposium on Web Society, Beijing, China, pp 397-400, August 2010.

[5] Sébastien Vénot, Lu Yan, "On-demand mobile peer-to-peer streaming over the JXTA overlay," Proceedings - International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM. Turku, Finland, vol. 16, pp. 131–136, 2007.

[6] Chang Le, Liu Yangyang, Wei Zhonghua, Pan Jianping , "Optimizing BitTorrent-like peer-to-peer systems in the presence of network address translation devices," Peer-to-Peer Networking and Applications, vol. 4, num. 3, pp. 247–288, 11. 2011.

[7] Gaspare Giuliano E. Bruno, Daniel R. Figueiredo, "Optimal Multi-Tree Peer-to-Peer Video Streaming," Proceedings of 19th International Conference on Computer Communications and Networks, ICCCN 2010. Zurich, Switzerland