DOI: 10.3939/j.issn.1673-5188.2014.01.007
http://www.cnki.net/kcms/detail/34.1294.TN.20131211.1242.001.html, published online December 11, 2013

Mobile Internet WebRTC and Related Technologies

Zhenjiang Dong, Congbing Li, Wei Wang, and Da Lyu
(ZTE Corporation, Shenzhen 518057, China)

Abstract

This paper describes an improved design for WebRTC technology. With this design, WebRTC communication at client side, server side, and between these two sides is improved. HTML5 WebSocket, media negotiation and synthesis, network address translator (NAT)/firewall traversal, Session Initiation Protocol (SIP) signaling interaction, and P2P communication security are all used in this improved design. This solution solves cross-browser running problem of WebRTC applications, reduces reliance on client-side processing capability, and reduces bandwidth consumption. With this design, WebRTC also become more scalable.

Keywords

WebRTC technology; application model; running across browsers; extension

1 Introduction

Web Real-Time Communication (WebRTC) [1] is a real-time audio and video communication technology based on Web browsers. It enables developers to use simple Web technologies to quickly and easily develop rich, real-time online multimedia applications without installing any plug-ins.

As network bandwidth and Web browsers continue to improve, WebRTC will have a huge impact on traditional real-time communication, and it is gradually becoming a mainstream real-time communication technology. Google has provided WebRTC support in its new version of Chrome on the Android platform. It is predicted that at least one billion terminals will support WebRTC in 2013. In an investigation results by CTI [2], a converged communications forum in China, 87% of the telecom companies in the survey are considering making WebRTC part of their product strategies; 86.9% of these companies say that WebRTC is a significant part of their overall product strategies; and 49% of these companies intend to deploy WebRTC solutions within a year.

WebRTC is developed by Global IP Solution Corporation, which was acquired by Google in 2010. After this acquisition, Google decided to open WebRTC technology to the public. Both the World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) have formed WebRTC standardization groups, and WebRTC standards are being supported by more and more manufacturers within the industry.

Compared with traditional communication technologies, WebRTC has the following advantages:

- It is easy to use, and plug-ins or different platform client applications do not need to be installed.
- It provides a consistent user experience.
- It can be quickly and easily upgraded at the server side.
- On the basis of WebRTC, web instant communication services can be developed with JavaScript or HTML.
- It can run across operating systems, which means that developers do not need to develop different application versions for different operating system.
- Developers can focus on services rather than media processing.

2 WebRTC Standard Progress

W3C and IETF are responsible for developing WebRTC standards, but these groups have a different target.

The W3C WebRTC working group aims at defining client-side Javascript APIs. With these APIs, Web applications can complete point-to-point or point-to-multipoint real-time communication between browsers. Key members of the W3C WebRTC working group are Web browser vendors, such as Google, Mozilla, and Opera.

The IETF RTC-Web working group defines real-time communication protocols and media formats between browsers. That is, it focuses on media codecs, network transmission, Network Address Translator (NAT)/firewall traversal, security, privacy, etc. Key members of IETF RTC-Web are Microsoft, Google, Skype, Yahoo!, Cisco, and FT (Orange).

Initially, WebRTC standards have three types of APIs: NetWork Stream, RTCPeerConnection, and DataChannel. NetWork Stream API is used for user media acquisition; RTCPeerConnection API is used to establish connectivity between browsers; and DataChannel API is used to transmit user data except video and audio stream from camera or microphone. Both NetWork Stream API and RTCPeerConnection API were developed earlier than DataChannel API and are used more widely. NetWork Stream API is independent from WebRTC standard in recent version.

As WebRTC standards mature, they are being supported by more and more modern browsers (**Table 1**).

This work is supported by the National Science and Technology Major Project of the Ministry of Science and Technology of China (2011ZX0300200201).

▼Table 1. Browsers that support WebRTC

Developer	Browser that Supports WebRTC
Google	Chrome 23+ for desktop, Chrome beta for Android
Mozilla	Firefox 22 beta+
Opera	Opera 12+

To verify the effectiveness of WebRTC in Web instant communication, browser vendors Google and Mozilla and third parties, such as Mobicents [3] (a VoIP middleware platform provider acquired by Red Hat), Livecome [4] (a communication service solution provider), and some Web developers, have developed their own WebRTC demos. Each of these WebRTC demos can be experienced with a camera and a microphone in WebRTC-supported browsers.

3 WebRTC Application Modes

3.1 Adding WebRTC Access to Traditional Audio and Video Services

Telecom operators and solution providers treat WebRTC as a new access mode for their traditional audio and video services. For example, users access a traditional call center based on web browser, and the browser becomes a new terminal of a traditional conference system.

The difficulty of this mode is ensuring compatibility between the WebRTC and traditional application architecture. A gateway device is added between WebRTC and the traditional application architecture in order to coordinate differences between them. For example, in ZTE's WebRTC-IMS gateway access solution, the WebRTC-IMS gateway is responsible for signaling conversion, media streaming relay, and NAT/firewall traversal mechanism conversion between WebRTC and traditional IMS networks.

The advantages of this solution are:

- A browser access mode is added in traditional audio/video service and no browser-side plug-ins need to be developed. This will greatly reduces client-side work.
- Media processing capabilities of WebRTC are strengthened, and user experience is improved by reusing Media Server in traditional architecture.
- Dependency on client-side processing capability is reduced and network bandwidth consumption are reduced when using WebRTC access traditional service.

3.2 Lightweight Real-Time Web Communication

The typical application scenario of WebRTC is to use WebRTC standard JavaScript APIs to develop lightweight Web audio/video calls and Web audio/video conference centers. Representative demos within the industry field are Mobicents SIP Servlet [5], Conversat.io[6] (now renamed as Talky.io), and Chatdemo [7],etc.

In this application scenario, almost all demos adopta decentralizedidea on the application architecture design. Its key points is decomposingmultiparty media negotiation process into multiple end-to-end negotiation processe, and functions like audio/video processing implemented in traditional media server are transferred to the client browser side.

Once obtained media formats and transmission protocol supported by both parties, two browsers can communicate with each other and media streams can be sent to the other side directly onther than relayed by the server. When a third party joins in, the third-party UA should initiate negotiations with the earlier two parties' UA. After this step ,the following process are the same as communication between two parties.

This solution is very easy and do not need to considering media processing. However, client-side media processing pressure will become increasingly heavier. In short, this solution largely dependend on client-side processing capabilities and therefore can't support more than six users in a video conference.

3.3 Comprehensive Web Real-Time Communication Service

Compared to the above one, the biggest difference is that we add a Multipoint Control Unit (MCU) in the server-side. Unlike the above solution, all media processing are concentrated in the MCU. Besides, MCU also provide some strengthen functions , such as audio/video mixing .

Because media processing are transferred to server-side, this solution significantly reduces WebRTC applications' dependence on client-side processing capability and extends the application range of WebRTC technology.

All in all, the last solution takes the advantages of the former two and can be applied in the large-scale application scenarios.

4 WebRTC Technical Solution: Problems and Improvement

4.1 Technical Problems in Current WebRTC Applications

As a developing technology and standard, WebRTC has many defects that represents a real challenge in its spread process. These defects are as follows:

- Compared with traditional video conferencing service, WebRTC has little control over user media. This characteristic means WebRTC is not adaptable to the requirements of complex conferencing.
- Considering media streams are sent to each other directly in WebRTC, client-side processing capability is very important. Less than five parties in a PC video conference is ok, but user experience suffers when this number number becomes larger. This problem is worse in the mobile terminal.
- Enormous amount of bandwidth consuming will Significant-

ly increase the cost of using WebRTC services.

- There are some differences in WebRTC APIs' name or HTML5 APIs' name in different browsers[8],[9]. WebRTC applications running at one browser successfully may encounter exceptions while running at another browser environment.
- WebRTC APIs can operate local media files and devices , this access mode may results in potential threats to the system [10].

4.2 Suggested Improvement

We propose an improvement solution in use of WebRTC to develop lightweight real-time Web communication services. This solution resolves the cross-browser running problem, reduces the dependencies of client-side processing capabilities and bandwidth consumption, and improves system scalability (**Fig. 1**).

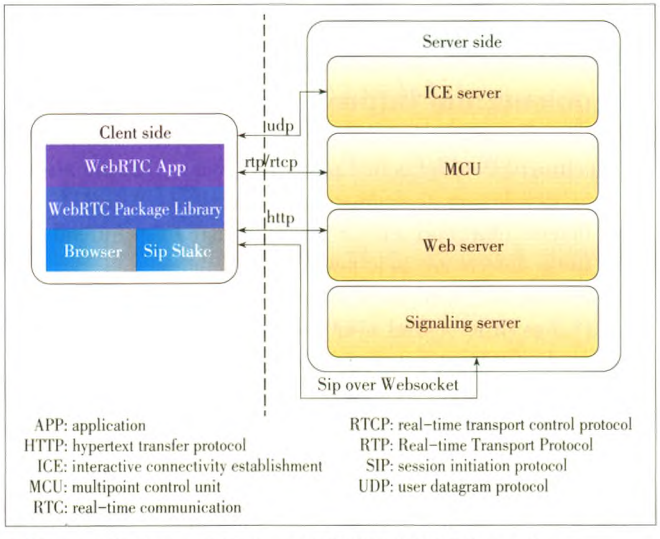
Fig. 1 shows the client side network elements (NEs), server side NEs, and how to communicate between these two sides.

4.2.1 Client Side

On the client side, the browser must support WebRTC standard interfaces, and the SIP Stack supports SIP signal processing. The WebRTC package library is based on WebRTC Javascript APIs and SIP Stack. Its implements: two-way communication between the client side and server side, media and firewall traversal negotiations between browsers, and running cross-browser support. The WebRTC APP completes the user interface display [11].

4.2.2 Server Side

The server side includes interactive connectivity establishment (ICE) [12] server, MCU, Web server, and signaling server. The ICE server interacts with the client-side browsers and returns iceCondidates to the client. The MCU synthesizes video streams transmitted from the browsers of various parties and sends the final stream back tothose browsers concerned in the session. The Web server provide the running environment for the WebRTC APP. The signaling server processes, routes, and distributes SIP requests from the client side and implements user authentication, user management, conference controlling and recording, and conference charging, etc. In an actual deployment scenario, the above four network elements can either be separately deployed or deployed on the same server or several servers.



▲Figure 1. System architecture of a lightweight Web real-time communication service.

4.2.3 Communication Between Client Side and Server Side

Communication between ICE client and ICE server complies with Simple Traversal of UDP over NATs (STUN) [13] and Traversal Using Relays around NAT (TURN) [14] protocols. UDP protocol is used for message transmission. SIP over WebSocket [15] technology is used for two-way SIP interaction between the client side and server side. Real-Time Transport Protocol/Real-Time Transport Control Protocol (RTP/RTCP) are used for transmission of the media stream between the client side and server side. HTTP is used for access to Web services.

Within this framework, a general multiparty video communication procedure can be described in the following steps:

1) Users access services provided by the Web server through client-side browsers.
2) Client browsers are registered to the signaling server and implement multiparty point-to-multipoint media negotiation. The MCU receives local audio and video streams transmitted by each party in the conference in the given format. It performs sound mixing, frequency mixing for these streams, then returns the synthesized stream to each party.
3) The local browser of each party receives and displays the media stream.
4) A multi-party communication process is established.

4.3 Key Technologies

4.3.1 HTML5 WebSocket

Traditional real-time communication is usually use polling or long-polling technology. Through increasing the frequency of client requests or extending server response time, developers can improve the user experience of real-time service to some extent, but an HTTP-based request-response mode cannot provide true real-time communication. In addition, polling or long polling technology increases client-side bandwidth consumption and server-side resource consumption.

To reduce delay in real-time communication and decrease bandwidth consumption, the HTML5 working group has developed a WebSocket communication standard. HTML5 WebSocket defines a full-duplex communication channel through a single socket on the Web and thus significantly improves real-time Web communication.

Before the client side and server side can communicate with each other, they need to complete a WebSocket handshake to establish a connection between them. Once the connection has been established, data can be transmitted between the two sides in both directions. In this way, when the server side updates data, this data can be directly pushed to the client side, and there is no need to wait for a request from the client side.

In very strong real-time applications, such as audio and video communication, HTML5 WebSocket can greatly improve performance and reduce bandwidth consumption by reducing transmission payload.

4.3.2 Media Negotiation and Synthesis

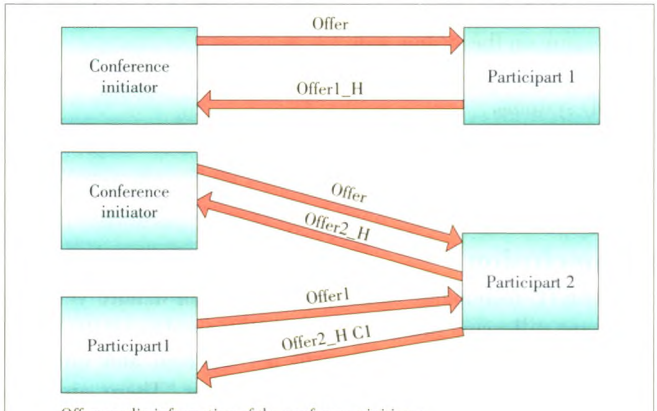
In this improvement, the media negotiation still occurs on the client sides. When there are multiple parties in a conference, the multiparty negotiation process needs to be decomposed into multiple end-to-end negotiation processes. Optional negotiation strategies are shown in **Fig. 2** and **Fig. 3**.

Theoretically, after a multiparty media negotiation has ended, multiparty media streams can be directly communicated to each other. However, this application mode creates significant issues. In a four-party conference, for example, each party needs to send out three-way local video streams at the same time and receive video streams from the other three parties. In this way, the client side can only support a very limited number of users, and a huge amount of network bandwidth is consumed.

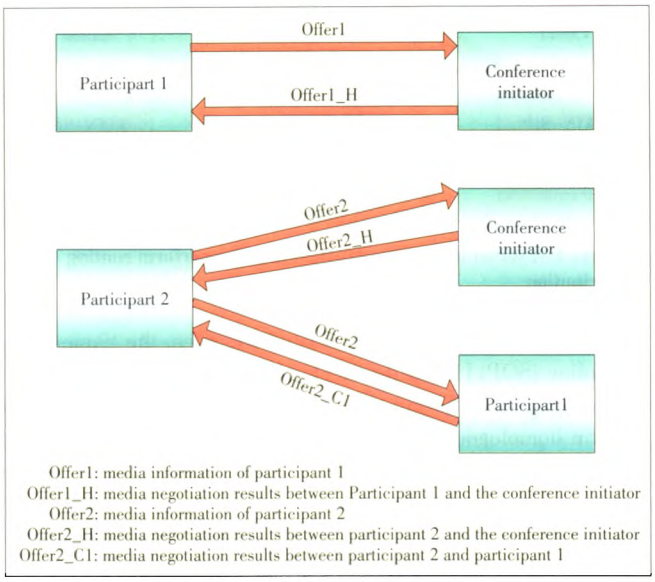
Therefore, an MCU is needed on the server side for media stream conversion, synthesis, and so. After the MCU has been introduced in this solution, each client only needs to send out one-way local video streams and receive one-way synthesized video streams from the MCU.

4.3.3 NAT/Firewall Traversal

For Web audio and video communication in a cross-LAN environment, NAT/firewall traversal equipment is used to route the transmission of media streams. The IETF-RTC Web standards stipulate that WebRTC browsers needs to support the ICE framework and the server side needs to provide the corresponding ICE server.



▲Figure 2. Negotiation process initiated by earlier entrants.



▲Figure 3. Negotiation process initiated by later entrants.

As for the selection of traversal schemes, STUN and TURN servers are supported at the same time. The STUN server completes non-symmetric NAT traversal, and the TURN server completes symmetric NAT traversal and firewall traversal. Through the synergy of the STUN and TURN servers, WebRTC media stream transmission is guaranteed.

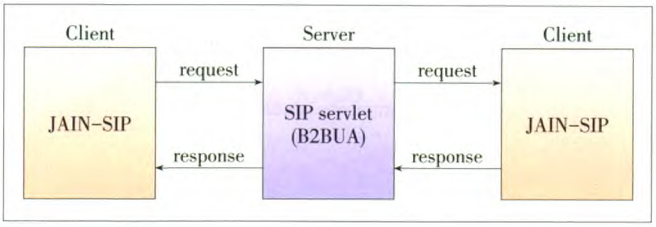
4.3.4 Signaling Interaction Technology Based on SIP Stack

In the W3C WebRTC standard, signaling between the client side and server side is not standardized. SIP is a simple, flexible, scalable protocol and become more and more popular in the industry filed. It should be said that SIP has become the virtual standard of next-generation communication.

To complete SIP signaling interaction between the client side and server side, we adopt the reference implementation of Mobicents JAIN-SIP on the client-side, and a B2BUA is developed on the server-side to process and distribute the requests from the client-side. **Fig. 4** shows the process for a server to act as the proxy and do routing distribution.

4.3.5 P2P Communication Security

To prevent browser programs from abusing local resources, we refer to [10] and introduce a sandbox into the WebRTC framework. The core idea of sandbox technology is to classify external websites as either trusted or untrusted. Trusted websites are considered safe and can access all local resources (or a limited number of the resources), and untrusted websites are put into the sandbox and cannot directly access local resources.



▲Figure 4. A server acting as a proxy in order to perform routing distribution.

To filter the trusted and untrusted websites, the Same-Origin Policy (SOP) [16] of the IETF is used. That is, the security of a webpage is determined by its origin, and it is necessary to perform homologous match on the protocol, host, and port that the webpage uses. A matching security access policy is assigned to each origin. For example, when a user accesses resources on a website B from website A, then the user is not allowed to access the encrypted local files on website B.

To prevent malicious websites from launching Distributed Denial of Service (DDOS) [17] attacks through Javascript API or simulating a domain name server (DNS) through a browser protocol packet and attacking a company's DNS, auxiliary defensive measures are introduced. These measures include regular scanning and checking the origins of visitors.

5 Enhancement and Optimization

To enhance efficiency and user experience in a lightweight WebRTC communication system, the following need to be optimized:

- video stream switching. In a traditional video-conferencing system, the video of a participant may be enlarged, and the screen may presents one big user interface and several smaller ones. In a WebRTC system, the <video> label of HTML5 needs to be used. A video stream is switched on demand by dynamically replacing the src attribute of the <video> label. Dynamically changing the src attribute is difficult because, unlike static video stream switching, the src attribute of each <video> label cannot be assigned in advance. In a WebRTC system, participants are dynamically added, and video streams are also dynamically obtained. To achieve video stream switching, video streams of the participants need to be stored, and the video streams have to correspond with their respective video container IDs in advance. When a user clicks a video, the application program determines which video stream needs to be enlarged and switched according to the video container ID.
- local echo elimination. In some WebRTC demo systems, media streams obtained from peripherals (such as cameras or microphones) are directly added to the local browser. This means that all participants can hear a local microphone echo before joining the conference. To eliminate this effect, two copies of the local media streams need to be saved before they are added to the browsers. Video streams from one of the copies (i.e. eliminating audio streams) needs to be extracted in order to add them to the local browsers. Then, the other complete copies of the media streams (including video and audio streams) are sent to the other participants through WebRTC interfaces. In this way, the other parties can hear the local voices, and the local audio echoes are eliminated.
- access support for non-WebRTC terminals. Such terminals include existing ZTE conference terminals, Polycom, Microsoft Lync, and IBM Sametime. On the one hand, various control protocols are supported; on the other hand, the system side needs to complete the conversion of video and audio streams.
- cross-browser support. The best browser that supports WebRTC is Google Chrome. Firefox22 comes in second place, and other browsers have weak support for WebRTC. There are interoperability problems between different browsers and currently, WebRTC is often restricted to a single browser. In order to run across browsers, we need to: 1) shield the differences between different browsers on the WebRTC interfaces in the WebRTC package library (including RTCPeerConnection, RTCSessionDescription, and RTCIceCandidate); 2) shield the differences between different browsers on implementation of HTML5 interfaces (including URL, MediaStream, and GetUserMedia) in the WebApp layer; and 3) shield the differences between different browsers on a webpage layout on the CSS layer.
- system-side flow reduction and linear expansion. Depending on the application scenario and access terminals, the system determines (automatically or according to policies) whether data streams between terminals are sent directly or sent through the central server. In a two-party or three-party scenario, if the terminals are the same type (video transcoding by the center is not required), they can be directly connected to each other, and data streams do not need to be forwarded through the center. This significantly reduces the pressure on the system side.
- integration with other systems, such as instant messaging (IM) systems.
- open system capabilities, open APIs (such as Rest), and support for third-party development.

6 Conclusion

WebRTC technology and standards are not mature yet, and there are still many problems in WebRTC implementation. For example, many functions are still not defined, and different browsers are incompatible with each other. There are also strict requirements on for client-side processing capability, and bandwidth consumption is high. This paper describes an improvement for WebRTC real-time communication. This improvement solves the main problems encountered in the applicationof WebRTC and is important to the development of We-

bRTC technology.

With the popularization of LTE, Wi-Fi and wired broadband; the extension of bandwidths; and the increased processing capabilities of PCs and mobile terminals, browsers on various terminals will support WebRTC so that real-time audio and video communication will become universal. Comprehensive Web communication is an ideal application mode.

HTML5 and WebRTC technologies and standards are maturing and spreading rapidly. Traditional IM,desktop sharing, electronic whiteboard, and audio and video conferencing will gradually be Web-based and will eventually be combined into a complete, unified communication system. Users will be able to communicate through browsers, and interpersonal interaction will become simpler and faster.

References

[1] *WebRTC 1.0: Real-time Communication Between Browsers* [Online]. Available: http://www.w3.org/TR/Webrtc/#rtcpeerconnection
[2] *Dialogic: 87% telecom leaders consider to put WebRTC into product strategy* [Online].Available: http://www.ctiforum.com/news/baogao/370610.html
[3] *Mobicents Project* [Online]. Available: http://hudson.jboss.org/hudson/view/All/job/Mobicents/
[4] *livecome Webrtc demo* [Online]. Available: http://lwork.hk:8086
[5] *Mobicents SIP Servlet* [Online]. Available: http://dev.telestax.com/sipservlets/wiki/HTML5WebRTCVideoApplication
[6] *talky.io* [Online]. Available: http://talky.io
[7] *chatdemo* [Online]. Available: http://ishare.iask.sina.com.cn/f/35083616.html
[8] *simpleWebrtc* [Online]. Available: https://github.com/HenrikJoreteg/SimpleWebRTC
[9] *Webrtc.io* [Online]. Available: https://github.com/WebRTC/Webrtc.io-client/blob/master/lib/ Webrtc.io.js.
[10] L. Li. "Security Analysis of Web Browser-Based P2P Real-Time Communication," *NetWork & Computer Security*, vol. 06, pp. 54-56, 2012.
[11] *Session Initiation Protocol* [Online]. Available: http://www.rfc-editor.org/rfc/rfc3261.txt
[12] *RFC5245-Interactive Connectivity Establishment* [Online]. Available: http://www.packetizer.com /rfc/rfc5245/
[13] *RFC3489-Simple Traversal of User Datagram Protocol* [Online]. Available: http:// www.ietf.org/rfc/rfc3489.txt
[14] *RFC5766-Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)* [Online]. Available: http://www.rfc-editor.org/info/rfc5766.
[15] *draft-ietf-sipcore-sip-Websocket-04* [Online]. Available: http://tools.ietf.org/html/draft-ietf-sipcore -sip-Websocket-04.
[16] *IETF-Security Considerations for RTC-Web draft-ietf-rtcWeb-security-02* [Online]. Available: https://datatracker.ietf.org/doc/draft-ietfrtcWeb-security.
[17] M. Wu, R.C. Wang, and Z.P. Wang. "Detection mechanism of DoS attacks in Peer-to-Peer networks based on support vector machine," *Computer Technology and Development*, vol. 19, no. 11, pp: 151-154, 2009.

Manuscript received: September 25, 2013

Biographies

Zhenjiang Dong (dong.zhenjiang@zte.com.cn) received his MS degree from Harbin Institute of Technology. He is the leader of Business Expert Team of ZTE Expert Committee for Strategy and Technology and the deputy president of ZTE Cloud Computing and IT Research Institute. His research interests include cloud computing, big data, new media, mobile Internet, and other technologies. He presides over 10 funded programs and has published more than 10 academic papers and one monograph.

Congbing Li (li.congbing@zte.com.cn) received his MS degree from Nanjing University of Science and Technology. He is currently a technical researcher in the field of mobile Internet at ZTE Service Research Institute. His research interests include WebRTC, node.js, HTML5, open platform, and other technologies. He is responsible for researching mobile Internet IM services and implementing related demos.

Wei Wang (wang.wei8@zte.com.cn) received her MS degree from Nanjing University of Aeronautics and Astronautics. She is a ZTE engineer and project manager in the field of mobile Internet at ZTE Cloud Computing and IT Research Institute. Her research interests include new mobile Internet services and applications, PaaS, terminal application development, and other technologies. She has authored five academic papers.

Da Lyu (lvda@zte.com.cn) received his MS degree from Nanjing University. He is deputy president of ZTE Cloud Computing and IT Research Institute. His research interests include Stored Program Control (SPC) exchanges, IP-based video solutions, and value-added services.

ZTE Launches the World's First Converged Intelligent Videoconferencing Solution at ISE 2014

February 4, 2014—ZTE Corporation has launched the world's first converged intelligent videoconferencing solution at Integrated Systems Europe, Amsterdam. The solution transforms existing video conferencing terminals from functional machines into smart machines.

Due to restrictions in system convergence, traditional videoconferencing can no longer meet the requirements of rapidly developing technologies and markets. The development of video codec technology and the extensive use of wireless technologies such as LTE are also raising people's expectations for higher video resolution and more flexible video communications. This converged intelligent videoconferencing solution uses an open video communication interface, provides a universal platform, and supports convergence with third-party applications, quickly meeting different user requirements. The system has pre-installed common conferencing software and supports plug and play USB flash disks and portable hard disks for easy sharing of electronic documents. With integrated functions such as E-whiteboard, data conferencing and multi-touch, the system supports simple and intuitive operations. (ZTE Corporation)