International Journal of Engineering and Management Research e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-11, Issue-3 (June 2021)
www.ijemr.net https://doi.org/10.31033/ijemr.11.3.4

# Real-Time WebRTC based Mobile Surveillance System

Alistair Baretto[1], Noel Pudussery[2], Veerasai Subramaniam[3] and Amroz Siddiqui[4]
[1]Student, Department of Computer Engineering, Fr. C. Rodrigues Institute of Technology, Navi Mumbai, INDIA
[2]Student, Department of Computer Engineering, Fr. C. Rodrigues Institute of Technology, Navi Mumbai, INDIA
[3]Student, Department of Computer Engineering, Fr. C. Rodrigues Institute of Technology, Navi Mumbai, INDIA
[4]Assistant Professor, Department of Computer Engineering, Fr. C. Rodrigues Institute of Technology, Navi Mumbai, INDIA

[1]Corresponding Author: alistair.baretto@gmail.com

**ABSTRACT**

**The rapid growth that has taken place in Computer Vision has been instrumental in driving the advancement of Image processing techniques and drawing inferences from them. Combined with the enormous capabilities that Deep Neural networks bring to the table, computers can be efficiently trained to automate the tasks and yield accurate and robust results quickly thus optimizing the process. Technological growth has enabled us to bring such computationally intensive tasks to lighter and lower-end mobile devices thus opening up a wide range of possibilities. WebRTC-the open-source web standard enables us to send multimedia-based data from peer to peer paving the way for Real-time Communication over the Web. With this project, we aim to build on one such opportunity that can enable us to perform custom object detection through an android based application installed on our mobile phones. Therefore, our problem statement is to be able to capture real-time feeds, perform custom object detection, generate inference results, and appropriately send intruder alerts when needed. To implement this, we propose a mobile-based over-the-cloud solution that can capitalize on the enormous and encouraging features of the YOLO algorithm and incorporate the functionalities of OpenCV's DNN module for providing us with fast and correct inferences. Coupled with a good and intuitive UI, we can ensure ease of use of our application.**

***Keywords*—** Computer Vision, Deep learning, WebRTC, YOLO, Android Development, REST API

## I. INTRODUCTION

Security is of major concern in the 21st century. Security in corporate and personal spaces has become very important in recent times. It is a necessity in modern times in corporate sections. Physical security guards like watchmen are a feasible solution except when there is a restriction of location access like in parking spots, offices etc. In such conditions a remote surveillance would be a better option and much more efficient. Therefore, different security systems have been developed like intruder alarms, CCTV, PC based video footage systems etc. [8] These systems are mostly wired and require a dedicated complex setup to get them running. Also, these setups are expensive to set-up and keep running, moreover, once setup they are pretty much static.

With the bloom of 4G-5G technology, wireless bandwidth increased, which made it possible to develop applications for mobile phones to perform multimedia streaming. This gives us an edge in developing smart wireless surveillance systems. In order to overcome the limitations of traditional surveillance systems, this paper proposes using android devices as a surveillance device, which makes it possible to monitor and target any site, anytime and anywhere via an android smartphone under the coverage of wireless networks. The basic requirements for this proposed system are good internet connectivity and a smartphone with a rear camera.

Moreover, since most people are familiar with using a smartphone, it makes it easier for them to set it up whenever and wherever required and so such a service can be quite cost effective and user-friendly.

Through this paper, we propose a WebRTC based surveillance system to make the streaming and data transfer possible with the lowest possible latency.[5] Our mobile application will capture the live feed and save the images at the server end which can be viewed through a secondary android application. Additionally, the live inference can be viewed from another android device making it possible to remotely monitor events. To create an alerting mechanism that notifies the concerned authorities about the presence of an intruder in a restricted area, we have used a YOLO algorithm-based object detection module.[11]

The paper is organized into multiple sections: the first comprising of a brief introduction to the topic; the second summarizes some of the related work; the third states our approach, discusses the proposed system and the algorithm used for performing object detection i.e., the YOLO algorithm. The fourth illustrates our system architecture and the experimentally inferred results are presented in the fifth section. The sixth and final section concludes our report.

## II. RELATED WORKS

Andrei Costin [8] makes use of different data-sets to exemplify the various new and existent threats in real-time video surveillance, Closed-Circuit Television (CCTV) and IP-camera based systems. These insights were used in identifying the vulnerabilities associated while building and deploying such systems. A summary table has been created listing some of these novel threats alongside their tested or proposed countermeasures. This can then be used as a checklist in analyzing the level of security while developing new CCTV/monitoring systems. The publication further provides some recommendations on risk mitigation and management which can optimize security and privacy levels provided by the software, firmware, the network infrastructure and the operation of surveillance systems. This paper aims to shed light on the severity of the threat landscape that such set-ups are exposed to while encouraging and motivating future research to expand the scope of this field beyond its existing boundaries.

Limi Kalita [2] elucidates the fundamentals of socket programming. The paper discusses various sub-sections of network programming and the concepts required in developing socket-based network applications. The paper states the logic behind client-server communication while introducing various operations of sockets and ports. It illustrates the functionality of TCP and UDP based socket programming. It concludes by explaining how Java is preferred over other languages for using the socket function/methods in establishing client-server-based communication systems.

Santos-González I et al. [5] has performed a study comparing some of the widely used video streaming protocols: The Real Time Streaming Protocol (RTSP) and WebRTC (Real time communication for the Web). Additionally, based on these protocols, the paper proposes two android mobile based applications whose primary aim is to find out to what extent connection establishment time and the stream reception delay affect the streaming quality of the service. These applications are further compared with some of the very common video streaming applications for Android. The analysis concluded that the performance of those streaming applications with respect to the stream packet delay, improved considerably using the WebRTC based implementation.

Naveen Kumar et al. [10] states the basics of the cross-platform OpenCV library and all its associated functions used in image processing and video feed analysis. It comprises of various sub-libraries and functions that can aid in solving computer vision problems and containing both low level pre-processing functions and high-level algorithms for face detection, feature matching and tracing. The paper explains various applications where it can be deployed like Motion capture, Intruder systems, Authentication services and Edge detection programs. They've also briefly given an account of Image filtering and its transformation methodologies, Object tracking and feature detection.

Q. Mao et al. [3] state the challenges of running computationally intensive tasks on embedded devices and further propose a YOLO network that doesn't compromise much on detection accuracy and is much more lightweight. They considerably brought down the parameter size of the network by using depth separable and point-wise group convolutions that are based on Darknet-53. In the process they managed to create a backbone network for feature extraction that was only 16 percent of the conventional 53 layered Darknet. Since accuracy is a factor that cannot be degraded, a Multi-Scale Feature Pyramid Network based on an easier U-shaped structure was added that ultimately improved the performance of object detection on multiple scales and was called Mini-YOLOv3. This model is relatively lighter in size and contains a lesser number of trainable parameters and floating-point operations (FLOPs) in comparison to YOLOv3.

## III. PROPOSED SYSTEM

In this chapter we discuss our approach, implementation strategy and the technology stack that will be used to perform surveillance and obtain inferences. Additionally, we shed light on a few protocols which aid us in carrying out the above tasks including a number of recent and emerging technologies like WebRTC using PeerJS, REST API's etc.

### *A. WebRTC*

WebRTC stands for Web Real-Time Communication using which we can add real-time communication functionality to our application.[5] It supports data, voice and video transmission, allowing developers to build powerful video and voice-communication services. WebRTC can be used in any device with a JavaScript engine. This is because it is only available as regular JavaScript API's in major browsers like Chrome, Firefox etc.

With the help of WebRTC, we can create a data channel between two peers; in our case a server and a client through which we can send data to and fro and this is a secure channel.

WebRTC transports its data using the User Datagram Protocol (UDP) unlike other browser communication,[1] which use Transmission Control Protocol (TCP). Due to timeliness over reliability being the primary reason, UDP protocol is a preferred transport for delivery of real-time data. In order to meet all the requirements of WebRTC, the browser provides a cast of protocols and services to traverse the many layers of NATs and firewalls, negotiate the parameters for each stream,

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-11, Issue-3 (June 2021)
https://doi.org/10.31033/ijemr.11.3.4

provide encryption of user data, implement congestion and flow control, and more.

To do so, WebRTC uses something called a RTP stack summarized in Table 1.

**Table 1**: RTP STACK

| Protocol | Full-From |
|---|---|
| ICE | Interactive Connectivity Establishment |
| STUN | Session Traversal Utilities for NAT |
| TURN | Traversal Using Relays around NAT |
| SDP | Session Description Protocol |
| DTLS | Datagram Transport Layer Security |
| SCTP | Stream Control Transport Protocol |
| SRTP | Secure Real-Time Transport Protocol |

In order to establish and maintain a peer-to-peer connection over UDP; ICE, STUN, and TURN are necessary.[17] Encryption is a mandatory feature of WebRTC, hence DTLS is used to secure all data transfers between peers. SCTP and SRTP are the application protocols used to multiplex the different streams, provide congestion and flow control, and other additional services on top of UDP. The Session Description Protocol defines a standard for defining the parameters for the exchange of media between two peers. Refer Figure 1. [1]WebRTC data channels require no special infrastructure setup, the only things it need is a signaling server to coordinate the connection between peers, a STUN server to figure out public identity of the individual peer and optionally a TURN server to route messages between peers if a direct connection between peers cannot be established (for example when both peers are behind symmetric NATs or restricted NAT).

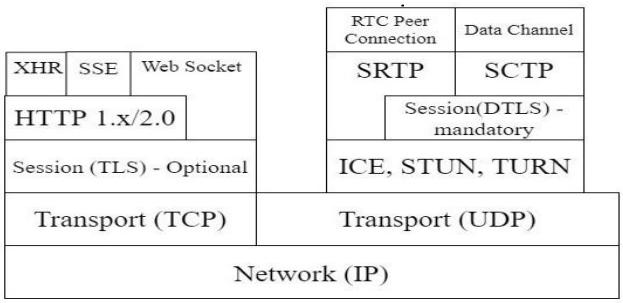


**Figure 1:** RTP Stack

***B. PeerJS***

PeerJS wraps the browser's WebRTC implementation to provide a complete, and easy-to-use connection API. [6] This API can help establish a peer-peer connection. Equipped with nothing but an ID, a peer can create a P2P data or media stream connection with a remote peer.

With PeerJS [21], identifying peers is simpler and every peer can identify using a unique ID. A string that the peer can choose itself, or have a server generate one. Although WebRTC promises peer-to-peer communication, we still need a server to act as a connection broker and handle signaling and this server is called the signaling server. PeerJS provides an open source implementation of this signaling server called as PeerJS Server. This server is written in Node.js. Refer Figure 2.

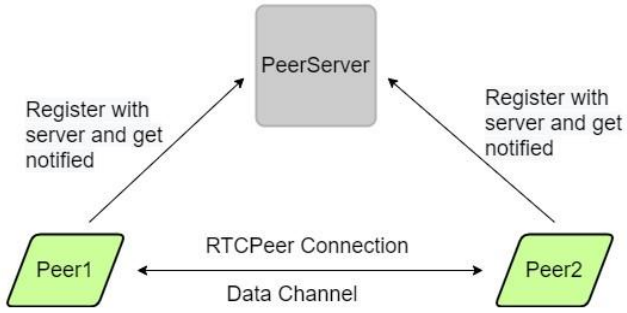


**Figure 2:** P2P PeerJS

***C. Android Application***

Android is an open source mobile operating system based on a modified version of the Linux kernel and it includes other open source software [19], designed primarily for touchscreen mobile devices such as tablets, smartwatches and smartphones. We have used Android Studio IDE for developing the android applications. It helps build and pack android files. Our android application can run on any android device with a version requirement of 6.0 and above. In order to create a connection between our android device and the server using PeerJS via WebRTC we need to leverage the chrome engine for JavaScript present in all android devices.

***D. Chrome Engine for JavaScript***

Android uses a JavaScript engine called Chrome V8 which is similar to a normal desktop JavaScript Chrome engine [18]. In order to use this, Android Studio provides us with an SDK under the name Chrome Webkit to evaluate and run JavaScript code directly in android. Thus, helping us leverage the power of WebRTC by calling the WebRTC API's from the chrome engine itself.

***E. NodeJS and Flask Server***

Our server runs on NodeJS which is a runtime environment for running JavaScript code outside of the browser [20]. So, our application connects to this server via a peer signaling server. This NodeJS server does the task of connecting to the android device and getting the frames and storing them. These frames are then sent over to the Flask server using the REST API [16] where inference is performed. The REST API is flask based, since our application makes use of OpenCV DNN module which is a python-based module.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-11, Issue-3 (June 2021)
https://doi.org/10.31033/ijemr.11.3.4

***F. OpenCV DNN Module***

In order to perform inference at a faster rate, we needed a framework which could provide us with high speed inference rates for object detection using the YOLO algorithm. OpenCV is built on C++ which itself is fast since it is a compiled language and it also provides us with python bindings for the same.[13] OpenCV can make use of CUDA cores present in Nvidia based GPU's to perform inference at a much faster rate.[12] Refer table to view the inference rates.

***G. CoTURN - A TURN Server***

For most WebRTC applications to function, a server is required for relaying the traffic between peers, since a direct peer to peer socket connection is not possible between the clients (unless they reside on the same local network). The common way to solve this is by using a TURN server.[22] Traversal Using Relays around NAT (TURN) is a protocol that assists in traversal of network address translators (NAT) or firewalls for multimedia applications. It is used to relay the traffic between peers even if they cannot connect directly. It can be thought as a workaround in-case a direct peer to peer connection fails. CoTURN is an open-source implementation of TURN (Traversal Using Relays around NAT) and STUN (session traversal utilities for NAT).

***H. YOLO Algorithm***

Yolo or the "You Look Only Once" is an effective real-time object detection algorithm.[11]

The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO uses a totally different approach than

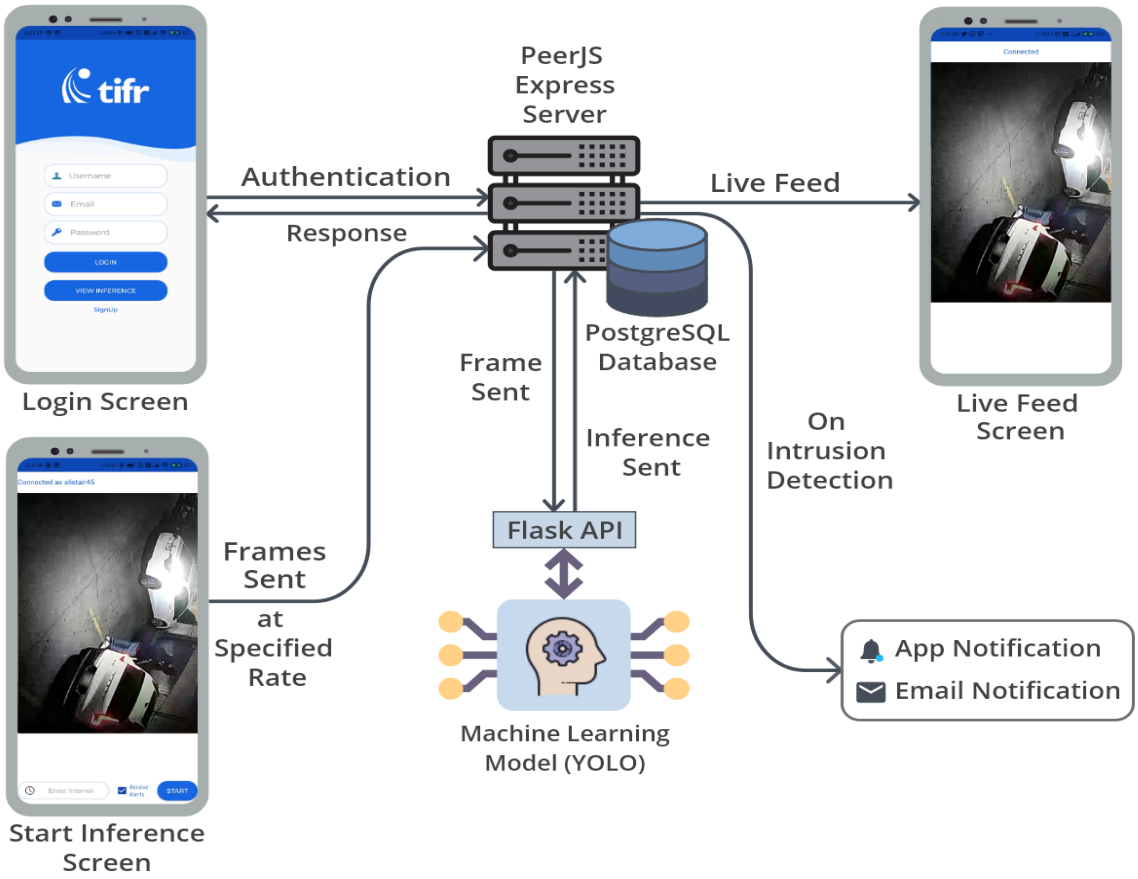


**Figure 3:** Architecture

other algorithms. YOLO is a clever convolutional neural network (CNN) for doing object detection in real- time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. The one we used, YOLOv3[14] is really powerful with a good tradeoff between accuracy and speed. There is also another version of YOLO known as 'Tiny-YOLO' which is a scaled down version of YOLO

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-11, Issue-3 (June 2021)
https://doi.org/10.31033/ijemr.11.3.4

which gives faster results on low-end systems with slightly lower results.[3] The benefits of YOLO have been applied to various applications due to the availability of pre-trained weights trained on the COCO dataset and reasonable training times. For instance, we have demonstrated the use of YOLO in counting traffic. This demonstrates the viability of YOLO as being a reliable object detection algorithm in diverse applications.

## IV. ARCHITECTURE

Our architecture mainly consists of two types of server, one being the peer signaling server which is needed for the initial handshake and connection for our mobile application and the inference server. The second server is the object detection server running YOLO as its object detection algorithm. This server does the work of receiving images from our mobile applications through PeerJS. Refer Figure 3. The application flow is as follows:

1) The user signs in to our mobile application.
2) He/she is then greeted into the main screen from where the user can start and stop the mobile surveillance operation.
3) On setting the interval of frame capture and after the clicking on start, the broker server receives the unique ID of our android device and when the android device requests for a connection to the server, the broker/signaling does so by creating a data channel between the two entities.
4) Once connection is established, the frames from the android device are sent to the second server, these images are received by the server in base64 format which are then converted into suitable format to be fed into the Yolo architecture for object detection.
5) If and when a person is detected the user is notified via an email alert which contains the image containing the detected person.
6) This android application also provides us with an option to view the feed of an android device if and when a session is active on the same User ID.

## V. RESULTS

This application has been tested on 6 devices connected to the server at the same time. Also, there are no data channel drops provided the internet connection is stable. As long as the object detection speed is concerned, refer to the Table 2 to get the comparative study of object detection speed on an image with OpenCV DNN module using YOLOv3 on GPU (Nvidia GTX 1050) and on CPU (Intel i5).[9] This study is done on images with 11 persons, 8 persons and 4 persons respectively in the image frame.

**Table 2**: INFERENCE TIMINGS

| Person Count | GPU/CPU | Inference Time |
|---|---|---|
| 11 | CPU | 440ms |
| | GPU | 8.98ms |
| 8 | CPU | 425ms |
| | GPU | 7.56ms |
| 4 | CPU | 412ms |
| | GPU | 6.98ms |

## VI. CONCLUSION

A mobile based surveillance system is not only the right step in technological advancement but also one that helps us to explore more possibilities given the resources at hand. Our application is designed to be intuitive and user-friendly. Being an android app helps our cause in tapping into the huge share held by Android devices in the market thereby making it compatible to be used on most mobile devices. Since the servers are hosted over the internet and can be deployed on any personal/organization's system, the process has become more efficient. Keeping in mind the limitations in the ability to perform computationally intensive tasks on the mobile itself, our system was built so as to keep it lightweight and free of any processing overheads. WebRTC, PeerJS and its associated technologies have enabled smooth peer to peer communication. Combined with the functionality of the OpenCV DNN module, obtaining inferences is now much more flexible with good accuracy. With security becoming a #NewNormal in our lives, surveillance systems have become the need of the hour. This is our attempt in this regard and we are confident enough that such systems with adequate optimizations are very much a part of our future and will help to create and sustain a reliable and secure environment for all of us.

## ACKNOWLEDGEMENT

We are grateful for the guidance given by our mentor Prof. Shashikant Dugad and Mr. Manish Manepalli for motivating us in developing an interest in the field of Computer Vision and Deep Learning. We wish to appreciate our guide Mr. Amroz Siddiqui for his constant inputs and support. We would also like to thank our project coordinator Ms. Rakhi Kalantri for providing us with timely inputs about documentation and project timeline.

## REFERENCES

[1] Santos-González I, Rivero-García A, Molina-Gil J, & Caballero-Gil P. (2017). Implementation and analysis of real-time streaming protocols. *Sensors (Basel), 17*(4), 846. DOI: 10.3390/s17040846.

[2] Limi Kalita. (2014). Socket programming. *International Journal of Computer Science and Information Technologies, 5*(3), 4802-4807.

[3] Q. Mao, H. Sun, Y. Liu, & R. Jia. (2019). Mini-YOLOv3: Real-time object detector for embedded applications. In: *IEEE Access, 7*, 133529-133538. DOI: 10.1109/ACCESS.2019.2941547.

[4] *Real Time Transport Protocol (RTP)*. Available at: https://www.geeksforgeeks.org/real-time-transport-protocol-rtp/.

[5] Real-time communication for the web. Available at: https://webrtc.org/.

[6] *PeerJS Docs*. Available at: https://peerjs.com/docs.html#start.

[7] Srujan Patel, Naeem Patel, Siddesh Deshpande, & Amroz Siddiqui. (2021). Ship intrusion detection system with YOLO algorithm. *International Research Journal of Engineering and Technology (IRJET), 8*(1).

[8] Andrei Costin. (2016). Security of CCTV and video surveillance systems: Threats, vulnerabilities, attacks, and mitigations. In: *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices (TrustED '16). Association for Computing Machinery, New York, NY, USA*, pp. 45–54. DOI: https://doi.org/10.1145/2995289.2995290.

[9] W. Thomas & R. D. Daruwala. (2014). Performance comparison of CPU and GPU on a discrete heterogeneous architecture. *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), Mumbai, India*, pp. 271-276. DOI: 10.1109/CSCITA.2014.6839271.

[10] Mahankali, Naveen Kumar & Ayyasamy, Vadivel. (2015). *OpenCV for computer vision applications*.

[11] Ahmad, Tanvir, MA, Yinglong, Yahya, Muhammad, Ahmad, Belal, Nazir, Shah, Haq, Amin, & Ali, Rahman. (2020). *Object detection through modified YOLO neural network. scientific programming*. DOI: 10.1155/2020/8403262.

[12] D. Steinkraus, I. Buck, & P. Simard. (2005). *Using gpus for machine learning algorithms*. Available at: https://hgpu.org/?p=1223.

[13] G. Bradski & A. Kaehler. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

[14] J. Redmon & A. Farhadi. (2018). *Yolov3: An incremental improvement*. arXiv preprint: arXiv:1804.02767.

[15] Neumann, Andy, Laranjeiro, Nuno, & Bernardino, Jorge. (2018). An analysis of public REST web service APIs. *IEEE Transactions on Services Computing*, pp. 1-1. DOI: 10.1109/TSC.2018.2847344.

[16] Chaitanya Mukund Kulkarni & Prof. M. S. Takalikar. (2018). Analysis of REST API implementation. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT). 3*(5).

[17] Peng Liang & Yang Shun. (2010). Research and implementation of voice transmission based on RTP protocol. *International Conference on Computational Problem-Solving, Li Jiang, China*, pp. 416-419.

[18] S. Delcev & D. Draskovic. (2018). Modern java script frameworks: A survey study. *Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia*, pp. 106-109. DOI: 10.1109/ZINC.2018.8448444.

[19] J. Liu & J. Yu. (2011). Research on development of android applications. In: *4th International Conference on Intelligent Networks and Intelligent Systems, Kuming, China*, pp. 69-72. DOI: 10.1109/ICINIS.2011.40.

[20] S. Tilkov & S. Vinoski. (2010). Node.js: Using JavaScript to build high-performance network programs. In: *IEEE Internet Computing, 14*(6), 80-83. DOI: 10.1109/MIC.2010.145.

[21] M. Kuhara, N. Amano, K. Watanabe, Y. Nogami, & M. Fukushi. (2014). A peer-to-peer communication function among web browsers for web-based volunteer computing. In: 14th *International Symposium on Communications and Information Technologies (ISCIT), Incheon, Korea (South),* pp. 383-387. DOI: 10.1109/ISCIT.2014.7011937.

[22] Rosenberg, J. (2010). *Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN)*. Available at: https://tools.ietf.org/id/draft-ietf-behave-turn-05.html.