A low latency quaternion-based web transmission system for augmented reality applications

Umer Ijaz, Arslan Dawood Butt*, Muhammad Husnain Khalid, Muhammad Fraz Anwar, Muhammad Malik
Department of Electrical Engineering, Government College University Faisalabad, Pakistan.
*e-mail: arslandawood@gcuf.edu.pk

Abstract

This work deals with the development and testing of a low latency Animation Transmission Technology (ATT) for augmented reality applications. This web-based transmission system based on AutoBahn Pyhton web server and WebSocket LibrarY (WSLAY) integrated clients has been studied to reduce packet header size and to simultaneously allow a large number of users to interact with each other. The transmission system in the proposed system architecture allows motion sensor's acquisition software to easily send compressed 3D quaternion-based data from transmitter to the rendering softwares at receiver side to generate real time 3D animation on an avatar. Furthermore, effects of lossy compression of quaternion data and server limitation have also been considered. Initial simulation test results with the Python server alongside WSLAY integrated clients with virtual motion sensors have been presented in this work. It has been evaluated that the network delay (ND) has a huge improvement from more than 300 ms to less than 25 ms at 100 Hz sensor sampling rate once lossy compression of 3D quaternion data is implemented. Furthermore, the effect of motion sensor sampling frequency and broadcast server limitation on maximum number of simultaneous users/sensors is also described in this work.

Keywords: Animation Transmission Technology, Body Sensor Network, Augmented reality network, Motion Acquisition, Quaternion compression.

Introduction

Recent advances in electronic sensors, wireless communication and mobile technologies has resulted in development of many 'Internet Of Things' (IOT) based systems covering a wide range of applications. In particular, the use of wearable motion sensors to form Body Sensor Networks (BSN) has gained much popularity. Such systems have been fundamental to capture human motion in applications like biomedical research [1-3] and augmented reality [4]. Such wearable inertial sensors based on MicroelEctroMechanical Systems (MEMS) are also used for motion tracking [5] and gesture recognition [6]. In addition to acquisition of human body motion and its analysis, Virtual/Augmented Reality (VR/AR) networks have also been devloped to share such information in runtime over a network [7]. Consequently, there is an increase in demand for real time transmission systems which could reduce the transmission delay when such information is communicated over the network [8]. This is typically achieved through implementation of a hybrid network where a dedicated local Body Sensor Network (BSN) with a gateway/sink node gathers motion information from individual sensor nodes and communicates with remote gateway nodes of other BSNs over the Internet Protocol (IP) [9]. This hybrid network allows utilization of cheaper sensor nodes with limited memory and computation capacity while still allows the BSN to be easily accessible over IP.

Recent Advancements in wireless communication technology with corresponding higher bandwidth has resulted in an overall reduction in transmission delays between nodes of a Wireless Sensor Network (WSN). However, a real time transmission system protocol, which can be easily integrated with both motion acquisition and rendering software at the transmitter and receiver to improve communication efficiency can also play a vital role in this regard. There are a wide range of binary, image-based and domain specific Animation Transmission Technologies (ATT) [10] available nowadays, however, they are limited by the bulky over headers which effect the highly interactive real-time applications and introduce Network Delay (ND). There is also a need of such ATTs which also facilitate the multiuser transmission by broadcasting data from a central server to multiple users. Additionally, use of animation compression algorithms in the ATTs to compress 3D data before transmission can also be utilized. This reduces the network delay and also improves the real time interactive behavior at the receiver side.

The system architecture proposed in this paper tries to tackle all of the above mentioned issues. It uses Web Socket LibrArY (WSLAY) [11] written in C, which can be easily integrated with any motion acquisition and rendering software at the transmitter and receiver respectively. It is also free from the bulky over headers and provides full duplex transmission at high data rates. Its network delay is minimal when compared to the other existing transmission technologies. It is also integrated with some animation compression algorithms in order to ensure fast data transmission. It not only provides efficient point to point communication but also can be used for multiuser communication when used in collaboration with the AutoBahn python broadcast server [12].

This work is aimed at quantitatively studying the use of Python server and WSLAY integration with clients transmitting and receiving 3D motion animation data over the network. In this scope, a generic system architecture compatible with the proposed server has been described. Here, the requirements of a wearable motion sensor and acquisition system, the role of the web broadcasting server and compatible 3D rendering system have been discussed in Section 2. Later, the integration of the three systems using WSLAY library functions and the data transmission/reception protocols needed for system operation have been explained in Section 3. Section 4 describes the experimental setup used to evaluate the performance of the proposed system architecture while Section 5 has been used to explain the obtained results pertaining to network performance. At the end, Section 5 concludes and summarizes this research activity.

http://dx.doi.org/10.24949njes.v11i2.405

System architecture

In the proposed system architecture, remote animation of a 3D avatar can be achieved in real time by transmitting animation data and other key parameters over the network using Internet Protocol (IP). The graphical parameters pertaining to movement are extracted from a set of motion sensors which can be worn by the users. To interpret this information, Web socket library has been integrated into the motion sensors' acquisition front-end on one side, and in renderer on the other side of the established communication network.

The proposed design can be described by a set of three subsystems based on their operation. These include the acquisition system, the transmission system and the rendering system. These subsystems are depicted in Fig. 1 and described in the following subsections.

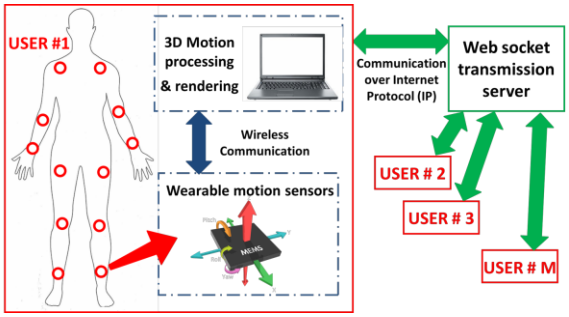

Fig. 1. Block diagram depicting the proposed system architecture and its individual components.

Motion sensing and animation acquisition

The subsystem acquires the information related to user motion and processes it so that it can later be passed to the renderer using the transmission subsystem. To acquire user's motion information, standard Micro-Electro-Mechanical-Systems (MEMS) devices can be used [13]. To appropriately capture the motion information of a user, a total of N number of wearable motion sensors have been considered in this work. Furthermore, a total of M number of users have been simultaneously interacting with each other as depicted in Fig. 1. This brings the total number of motion sensors in the system architecture to be $N \times M$ and greatly adds to the complexity of the transmission system.

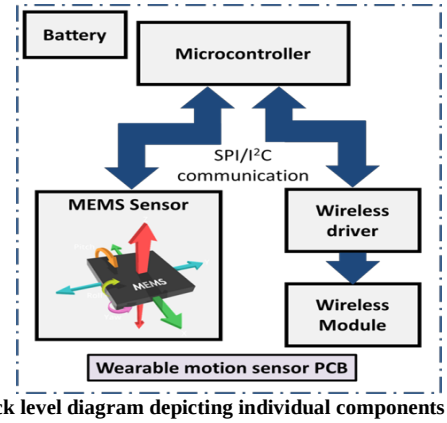

Fig. 2. Block level diagram depicting individual components in a generic wearable motion sensor.

The construction of a generic wearable motion sensor compatible with our system architecture can be seen in Fig. 2. Here, MEMS sensor chip, a micro-controller, compact rechargeable battery and a wireless sensor module and its driver have been depicted as blocks on a wearable motion sensor's Printed Circuit Board (PCB). Here, the micro-controller programs and receives/interprets the motion information provided by the MEMS sensor. It also converts this information into quaternions (if not already done so by the MEMS chip) and passes it to the acquisition system using the wireless module. Additionally, time-stamp information is also added by the micro-controller. The inter chip communication on the wearable sensor PCB can be achieved using either I2C and or SPI communication [14].

To appropriately capture and track motion of the user, commercial MEMS sensors with 9-DOF (Degrees Of Freedom) can be utilized. This can be achieved by using MEMS sensors having an integrated 3D Gyroscope, a 3D Magnetometer and a 3D Accelerometer. Alternatively, separate MEMS sensors with one or more of these motion sensing capabilities can be mounted on the wearable motion sensor PCB. To name a few 9-DOF MEMS chips, Inven-Sense MPU-9250 Motion processing unit [15] or STMicroelectronics 9-DOF LSM9DS1 [16] are compatible with the proposed system architecture.

Among the proposed MEMS chips, the 9-DOF MPU-9250 chip and others like it, also have a built-in mode where it provides a 32-bit 9-axis quaternion alongside heading accuracy thereby providing us with output in Attitude Heading Reference System (AHRS) [17]. This additional processing within a single chip greatly simplifies the later stages of the acquisition system which also assists in reducing system latency.

As far as the battery module is concerned, a standard 3.7 V 150 mAh rechargeable battery can be utilized. Considering the typical components of wearable motion sensor described above, the system is expected to operate continuously for around 7 hours. However, low power data transmission protocols and reduction of maximum sampling frequency to 100 Hz (sufficient for seamless user experience) can easily improve the battery life to several days.

Acquisition software runs on a PC where it communicates with each motion sensor node to acquire data and to program the firmware. This communication is established between the wearable motion sensor and the PC using the wireless modules available to both of them. This quaternion based 3D motion data contains information regarding motion of user's limbs and performed gestures and must be transmitted to other clients.

As there are a total of N sensors per user, to acquire quaternion information from all the sensors, the acquisition software must connect to each sensor's wireless module. This can be done both sequentially or in parallel. However, sequential data transfer adds to the latency of the overall system operation at User end, even before the network is considered. In both scenarios, once received, the information gathered from all motion sensors is arranged in a queue like data structure block as shown in Fig. 3. Each information

block comprises of four floating point quaternion data *qi*, followed by a floating-point time stamp *ti* of all the *N* motion sensors utilized by the User. Here, *i* represents the quaternion index while *j* represents sensor number.

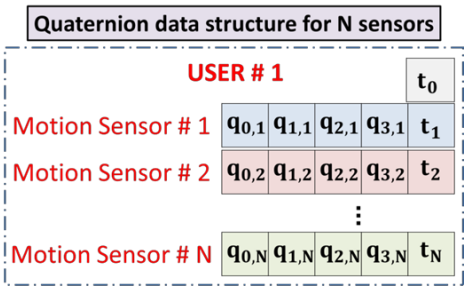

Fig. 3. Structure depicting quaternion-based motion sensor and time stamp information for a total of *N* sensors per user.

To enhance, data throughput of each user and consequently to reduce network delay, lossy quaternion compression of the data structure shown in Fig. 3 can be performed by the motion acquisition stage. In lossy compression, instead of four floating point numbers, three quaternions in short integer format are transmitted. As the quaternions have values between +1 and -1, this conversion is done simply by multiplying the floating-point quaternion by 2(16- = 32,768 and then storing it in short integer format. This way quaternion precision is limited to ±2-(16-1) and smaller values are lost to compression. As an advantage, the data size is reduced from 4 x 4 = 16 bytes in case of no compression to 3 x 2 = 6 bytes in case of compression which results in reduction of system latency. The missing fourth quaternion component for *jth* sensor can be evaluated using the quaternion property described in (1). This information is later communicated to the web socket transmission system to be shared with other users on the network.

$$1 = q_o^2 + q_1^2 + q_2^2 + q_3^2 \qquad (1)$$

Web socket transmission

To achieve real-time operation, transmission of motion sensor data to and from multiple clients over the network is handled by a Web socket server. In the proposed system architecture, the utilized web socket transmission system is based on a Python web server and WSLAY libraries running on multiple clients. The WSLAY libraries are used by the clients to send/receive quaternion-based motion sensor information to/from the web server. The Autobahn Python web server, on the other hand, is tasked with receiving this data from WSLAY clients and later to broadcast it to all the connected Users. While using the web socket protocol, C- based WSLAY can support up to 63 runtime users at a time [11]. The Web server maintains a separate session for each connection with a new user through a dedicated socket (assigned randomly). This allows server to avoid conflicts in case of simultaneous transmissions from multiple users. In addition, the proposed architecture is easily scalable and can be extended through various load-balancing techniques to handle multiple user requests simultaneously.

Furthermore, this WSLAY library provides us with both, an event-based and a frame- based Application Programming Interface (API) and can transfer User/Server data without Hyper Text Transfer Protocol (HTTP) handshake [18]. This helps in reducing the overhead which would otherwise be present in HTTP and can increase ND. In addition, external event looping and callback interface options of the WSLAY library also make them a suitable choice for low latency applications like the one described in this work.

Fig. 4 depicts how motion sensor information is transferred to the web server using the WSLAY native applications running on the clients. Here, a single quaternion corresponding to a single motion sensor has been depicted for simplicity. In a generic system, each User transmits up to *N* quaternions in the form of data structure shown in Fig. 3. For a single user, once the data corresponding to all sensors has been transmitted from the wearable motion sensors to the acquisition software, it is pushed by the local WSLAY client to the central Python web server node. This is done independently by all clients connected to this web server. Once the Python server receives the provided data, a callback function is immediately called to broadcast the received information to all the connected client nodes as depicted in Fig. 5.

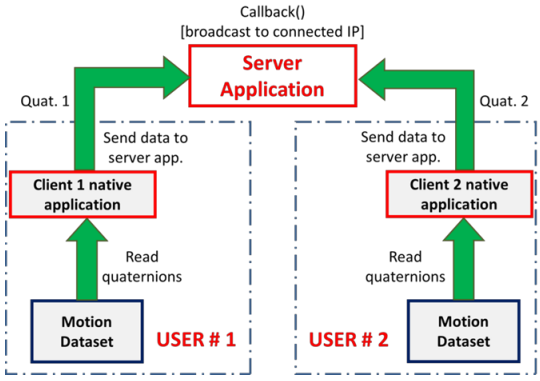

Fig. 4. Information diffusion from clients to server.

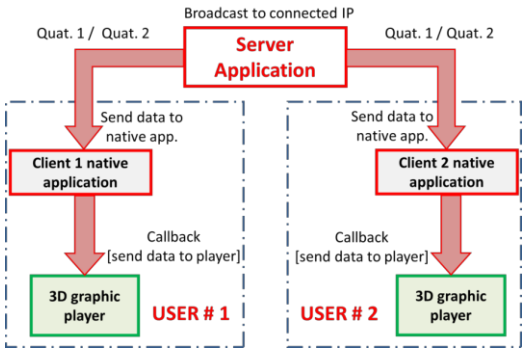

Fig. 5. Information broadcasting from server to clients.

Once each client receives this data, locally run WSLAY client generates a callback directed at the 3D graphics player to read the input WSLAY buffers. This allows the 3D renderer to process and animate the 3D avatar/model in run-time.

3D Rendering and animation

In the proposed system architecture, once a client receives the quaternion based motion data broadcasted from the Python web Socket node, it is passed on to the 3D graphics player for

animation as depicted in Fig. 5. These 3D rendering tools interpret the motion information into animation of Client Models/avatars to allow users to interact with each other in a 3D environment. These 3D rendering and animation platforms can be either standalone, web based or can even be a combination of the two. The compatibility of the system architecture described in this work includes, but is not limited to, the popular open source and licensed 3D graphic players described in Table 1.

Table 1. Compatible 3D rendering tools.

3D Rendering tools	Operating Systems	Application Platform
Orbisnap [19]	Windows, Mac & Linux	Standalone
Cortona VRML [20]	Windows	Plugin
X3DOM [21]	Windows	Plugin
Bit Management [22]	Windows & Linux	Plugin & Standalone
Flux player [23]	Windows	Plugin & Standalone
InstantReality [24]	Windows & Linux	Plugin & Standalone
View3DScee [25]	Windows, Mac & Linux	Plugin & Standalone

Here, Orbisnap [19] only supports standalone platform for visualizing 3D virtual environment while Cortona VRML [20] and X3DOM [21] only support Plugin application platforms. The remaining 3D rendering tools described in Table 1 can be used both as 1standalone and as web based plugins.

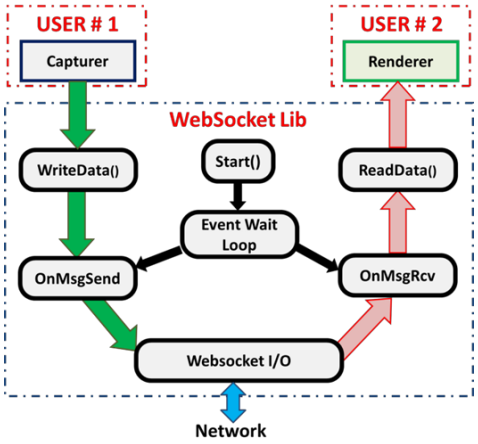

Fig. 6. WSLAY Library functionality. [11]

System integration and operation

The augmented reality applications considered in this work require a versatile and efficient web transmission system to have an overall reduced network delay. This is necessary to allow all users to experience smooth system operation. This system requirement means that integration of the transmission system with both the motion sensor side and the graphic player side needs to be optimized as well. This integration needs to be cost effective in terms of additional overhead bytes added to the motion sensor data to facilitate communication between the server node and the clients. In this section, the algorithms used for server-client communication have been described.

Multiple functions of the WSLAY used for system integration are depicted in Fig. 6. Here, two instances of the Web Socket library have been shown in a single figure. One instance is integrated with user #1 which is only acting as a Capturer i.e. it acquires the motion sensor data and has to transmit it to the Python server over the network using Web socket library functions. For this instance, no data is being received by the Renderer of User #1. The second instance is that of User #2 which is acting only as a Renderer i.e. it has to receive the motion sensor data provided by the Python server present on the network to be rendered by its 3D graphic player. Contrary to User #1, User #2 has nothing to transmit as shown in Fig. 6.

Once the WSLAY is initialized, it enters a loop function and waits for an event. Within this loop function, it continuously checks whether any data has been received from or needs to be transmitted to the network. Once the motion acquisition system in case of User #1 writes some data in the WSLAY library using WriteData() function, an OnMsgSend callback function is activated. This function is responsible for transmitting the available data onto the network using Websocket I/O port as shown in Fig. 6. The motion sensing and acquisition system has an embedded routine which calls this WriteData() function once quaternion-based structure shown in Fig. 3 has been received so that this information can be transmitted over the network to other users. This integration protocol is depicted in detail in Fig. 7.

In case of User #2, once the animation data has been received from the network by the Websocket I/O port, an OnMsgRcv callback function of WSLAY library is activated. This means that there is valid data available to be rendered by 3D graphic player of User #2. This animation data is copied into the 3D renderer software data structure using the WSLAY library function ReadData(). The event while loop inside the WebSocket libray, continues to receive/transmit data to/from the network as long as any valid data is available and moves back to the wait state afterwards. This integration protocol is depicted in detail in Fig. 8.

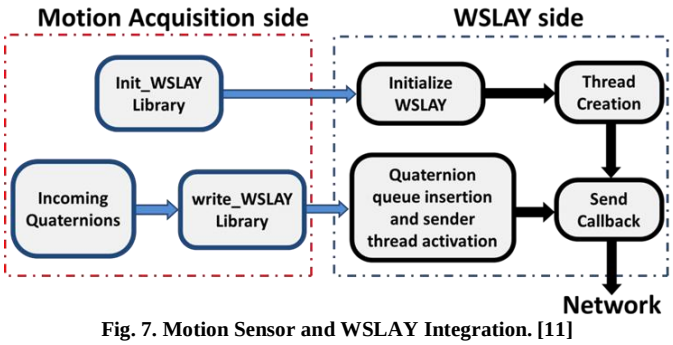

Fig. 7. Motion Sensor and WSLAY Integration. [11]

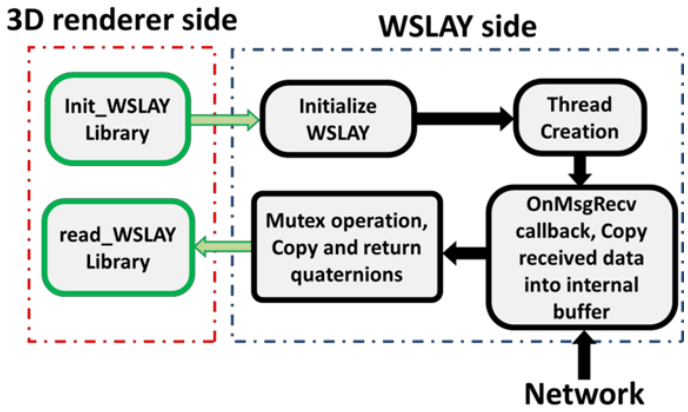

Fig. 8. 3D renderer and WSLAY Integration. [11]

Experimental Setup

In the proposed system architecture, the transmission system i.e. the python web server and its integration can be tested independent of the motion sensing/ acquisition and the 3D rendering system. Thereby, to analyze the performance of the web server and the WSLAY library integration on both ends with the capturer and the renderer, a simulation test has been performed.

The experiment has been performed with simulated wearable motion sensors. To achieve this, within the acquisition system, quaternion data has been added into the queue shown in Fig. 3 and this information is later read by the integrated WSLAY library.

Three Laptop devices have been utilized where one of them acts as a broadcast server while the other two are used as transmitter and receiver respectively. All Laptop devices (with Intel Core i3 i3-M350/2.27GHz processor and 4 GB RAM) have Linux operating system running on them. These Laptop devices are connected to public Internet network through an 802.11g Wi-Fi router. During the experimental tests, the network speed was monitored and was evaluated to have an average speed of 9 Mbps with maximum of 20 Mbps and minimum of 3.5 Mbps. These tests have been carried out at data frequencies of 25, 50 and 100 Hertz and with the simulated number of sensors *N* being 1, 5 and 10 to study the effects of change of data rate and network loading on the web transmission system.

The tests were designed to analyze the effect of increased number of virtual sensors, their sampling frequencies and quaternion data compression on the ND experienced by the users. In addition, maximum numbers of simultaneous users *M* and sensors *N* supported by the broadcast server were also evaluated as a function of sensor sampling frequency. In case of no compression scenario, four floating point numbers are transmitted per sensor sample, while with lossy compression enabled each quaternion is represented by means of three short integers respectively. A time-stamp is present with the data structure in all scenarios to calculate ND.

Simulation Results

A comparison of ND versus number of sensors at different frequencies under no compression and lossy compression scenarios can be seen in Fig. 9 and Fig. 10 respectively. In no compression scenario, ND at 25 or 50 Hz was nearly the same and less than 12 ms for up to 10 number of sensors. At 100 Hz, ND increases exponentially with the increase in number of sensors and approaches 150 ms and 300 ms for up to 5 and 10 number of sensors. Considering 100 ms as maximum acceptable delay [26], in particular for telepresence in gaming scenarios, the system showed poor performances during experiments at 100Hz.

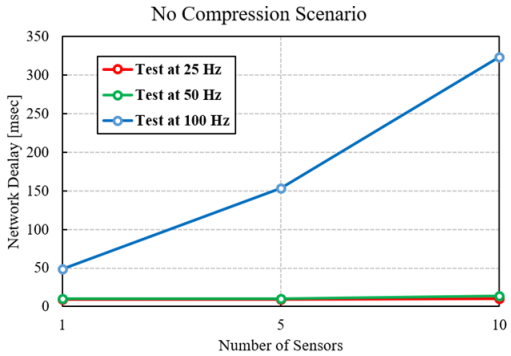

Fig. 9. Network Delay vs. number of sensors for a single user with no quaternion compression.

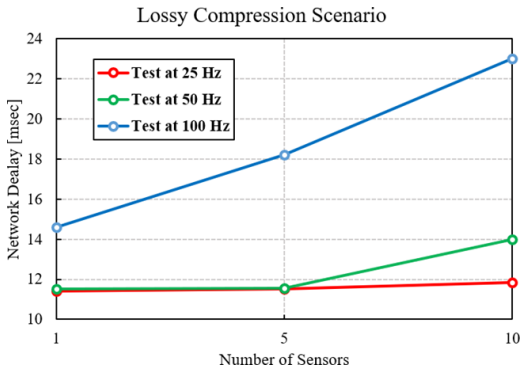

Fig. 10. Network Delay vs. number of sensors for a single user with lossy quaternion compression.

In lossy compression scenario, ND at 25 and 50 Hz remain almost same and less than 12 ms for up to 5 number of sensors. When number of sensors are increased to 10, the ND at 50 Hz increase linearly up to 14 ms compared to12 ms at 25 Hz. ND at 100 Hz increases abruptly with the number of sensors and approaches a maximum of 23 ms for 10 number of sensors. Results show that by adding compression the ND reduce under 100 ms even with 10 number of sensors.

In lossy compression scenario, ND at 25 and 50 Hz remain almost same and less than 12 ms for up to 5 number of sensors. When number of sensors are increased to 10, the ND at 50 Hz increase linearly up to 14 ms compared to12 ms at 25 Hz. ND at 100 Hz increases abruptly with the number of sensors and approaches a maximum of 23 ms for 10 number of sensors. Results show that by adding compression the ND reduce under 100 ms even with 10 number of sensors.

Fig. 11 depicts the limitations of broadcast server on the number of users or connections. Broadcast server allows a maximum of up to 20 users with 1 sensor each at 25 Hz. Number of users decrease with the increase in number of sensors at 25 Hz until it arrives at 7 for 10 number of sensors. Similarly maximum numbers of users turn out to be 14 and 12 at 50 and 100 Hz respectively. Number of users decrease with

increase in number of sensors and reach 6 and 5 at 50 and 100 Hz respectively. Results predict that number of users is a function of frequency and number of sensors and inversely proportional to both of them. The most ideal case is the one with 1 number of sensors at 25 Hz allowing maximum number of users to connect with broadcast server while the worst case is with 10 number of sensors at 100 Hz allowing 5 number of users to connect with the server. The broadcast server adopted does not properly scale in term of number of connections supported, and did not allow to properly test the system with large number of users.

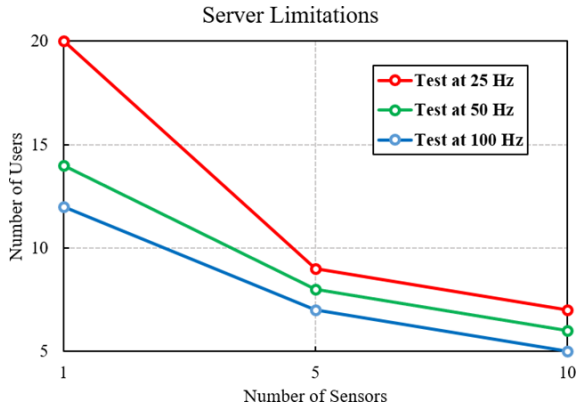

Fig. 11. Broadcast server limitation depicted as a relationship between no. of users and no. of sensors/user to ensure network delay < 100ms.

Conclusion

In this work a system architecture based on Autobahn Python web server and 3D motion sensors/Renderer with WSLAY integration has been proposed for low latency augmented reality applications. Additionally, first results regarding the performance of the server have also been presented. It has been evaluated through simulation that at 25 Hz sensor sampling frequency, the proposed system can handle up to 7 users with 10 sensors each while at 100 Hz up to 5 users with 10 sensors each can be supported by the network with acceptable network delay less than 100 ms [26]. These results can further be improved with algorithm optimization at both the server and the client end in future works.

Further tests with sensors and renderer software are to be performed to validate the presented simulation test results. Also effect of lossy compression, which significantly improves network delay, needs to be studied with the rendering software in full system tests.

Acknowledgements

The authors would like to thank Higher Education Commission (HEC), Pakistan for their support in the form of Grant no. 21- 1784 SRGP/R&D/HEC/2017 to facilitate research on this topic.

REFERENCES

1. S. Qiu, et al., “Using distributed wearable sensors to measure and evaluate human lower limb motions,” IEEE Transactions on Instrumentation and Measurement, vol. 65, no. 4, 2016, pp. 939-950.
2. C.L. Pulliam, et al., “Continuous assessment of levodopa response in parkinson's disease using wearable motion sensors,” IEEE Transactions on Biomedical Engineering, vol. 65, no. 1, 2017, pp. 159-164.
3. Y.-C. Kan and C.-K. Chen, “A wearable inertial sensor node for body motion analysis,” IEEE Sensors Journal, vol. 12, no. 3, 2011, pp. 651-657.
4. P. Jatesiktat and W.T. Ang, “Recovery of forearm occluded trajectory in kinect using a wrist-mounted inertial measurement unit,” Proc. 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2017, pp. 807-812.
5. D.L. Arsenault and A.D. Whitehead, “Quaternion based gesture recognition using worn inertial sensors in a motion tracking system,” Proc. 2014 IEEE Games Media Entertainment, IEEE, 2014, pp. 1-7.
6. D. Arsenault and A.D. Whitehead, “Gesture recognition using Markov Systems and wearable wireless inertial sensors,” IEEE Transactions on Consumer Electronics, vol. 61, no. 4, 2015, pp. 429-437.
7. X. Ge, et al., “Multipath cooperative communications networks for augmented and virtual reality transmission,” IEEE Transactions on Multimedia, vol. 19, no. 10, 2017, pp. 2345-2358.
8. E. Bastug, et al., “Toward interconnected virtual reality: Opportunities, challenges, and enablers,” IEEE Communications Magazine, vol. 55, no. 6, 2017, pp. 110-117.
9. B.K. Maharrey, et al., “Interconnection between IP networks and wireless sensor networks,” International Journal of Distributed Sensor Networks, vol. 8, no. 12, 2012, pp. 567687.
10. A.L. Ahire, et al., “Animation on the web: a survey,” Proc. Proceedings of the 20th International Conference on 3D Web Technology, ACM, 2015, pp. 249-257.
11. I. Fette and A. Melnikov, The websocket protocol, 2070-1721, 2011.
12. Autobahn python. [Online]. Available: http://autobahn.ws/python
13. K.D. Wise, “Integrated microelectromechanical systems: A perspective on MEMS in the 90s,” Proc. [1991] Proceedings. IEEE Micro Electro Mechanical Systems, IEEE, 1991, pp. 33-38.
14. F. Leens, “An introduction to I 2 C and SPI protocols,” IEEE Instrumentation & Measurement Magazine, vol. 12, no. 1, 2009, pp. 8-13.
15. InvenSense. ”MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS MotionTrackingTM Device”. [Online]. Available: https://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/
16. STMicroelectronics. ”iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer”. [Online]. Available: http://www.st.com/en/mems-and-sensors/lsm9ds1.html

17. J. Bartholomeycz, et al., “MEMS based inertial measurement unit for attitude and heading reference systems,” Proc. 2nd European Conference & Exhibition on Integration Issues of Miniaturized Systems-MOMS, MOEMS, ICS and Electronic Components, VDE, 2008, pp. 1-8.
18. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach et al., “Hypertext Transfer Protocol – HTTP/1.1,” United States, 1999.
19. ”Orbisnap”. [Online]. Available: http://www.orbisnap.com
20. ”Cortona 3D Viewers”. [Online]. Available: http://www.cortona3d.com/cortona3d-viewers
21. J. Behr, et al., “X3DOM: a DOM-based HTML5/X3D integration model,” Proc. Proceedings of the 14th International Conference on 3D Web Technology, ACM, 2009, pp. 127-135.
22. ”Bit Management”. [Online]. Available: http://www.bitmanagement.com/
23. ”Flux 3D Player”. [Online]. Available: http://fluxplayer.software.informer.com/
24. ”Instant Reality”. [Online]. Available: http://www.instantreality.org
25. ”View 3D Scene”. [Online]. Available: http://castleengine.sourceforge.net/view3dscene.php
26. C. Westphal, “Challenges in networking to support augmented reality and virtual reality,” IEEE ICNC, 2017.