An mHealth Monitoring System for Telemedicine Based on WebSocket Wireless Communication

Weiping Zhang, Regina Stoll
Institute for Preventive Medicine, University of Rostock, Rostock, Germany
Email: {weiping.zhang, regina.stoll}@uni-rostock.de

Norbert Stoll and Kerstin Thurow
Center for Life Science Automation, University of Rostock, Rostock, Germany
Email: { norbert.stoll, kerstin.thurow }@celisca.de

***Abstract*—The telemedicine system is a new medical monitoring model which uses sensor communication technology, computer information technology and modern medical technology. This paper, based on real-time web communication technology HTML5-WebSocket, describes the development of a scalable, real-time, multi-parameter, remote monitoring system. The system consists of sensors, mobile front control side, and a module-based remote monitoring platform. It has a variety of network functions, such as transmission of multiple physiological parameters, search, and export. In addition, the system provides real-time monitoring, data review, waveform check, and real-time doctor-patient communication. Since no additional software needs to be installed at the client, maintenance cost is reduced while the versatility is enhanced.**

***Index Terms*—eHealth, telemedicine, WebSocket, mhealth, telematics platform**

I. Introduction

Increasing life expectancy, due to medical progress, and decreasing birth rates lead to a change in the population structure of Germany. As illustrated in Fig. 1, the percentage of people over the age of 65 is steadily increasing while the number of people under the age of 25 is declining [1]. This results in continuously rising costs for healthcare, especially for elderly people. One approach to this problem is the utilization of telemedicine systems to enable a relocation of the therapy to the patient's home, thus shortening the expensive periods of hospitalization. A real-time remote monitoring system can provide valuable information for diagnosis and therapy. Remote monitoring combines elements from different research areas, such as sensor communication technology, computer information technology, medical technology, and medical monitoring management [2]. Real-time data from wearable sensor devices are sent to smartphones over a short range wireless connection then forwarded to the remote monitoring platform via Wi-Fi, 2G or 3G. At the monitoring platform the medical data can be displayed or stored in a database. Conventional ECG monitoring, for instance, can provide important clues for the diagnosis of a patient's condition. A trend in biomedical engineering research is towards wireless systems that concurrently monitor multiple physiological parameters, such as ECG, respiration and acceleration.

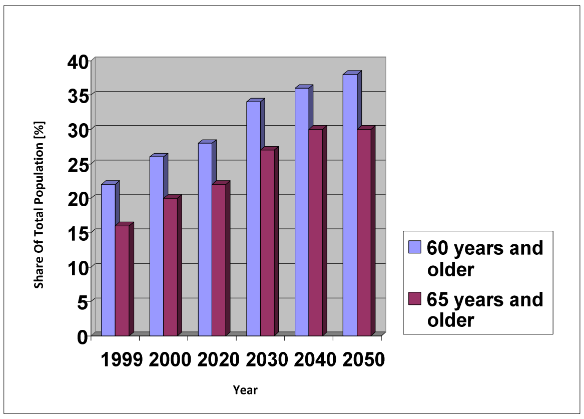


Figure 1. Projected Percentage of Elderly People in German Population (Source: German Federal Bureau of Statistics 2010)

Most of the existing remote multi-physiological parameter monitoring systems use Client/Server (C/S) structure in conjunction with a Browser/Server (B/S) structure. The C/S structure provides data collection while the B/S structure provides data management. The mobile client sends collected physiological parameters to a server. In a traditional B/S application, doctors or researchers send an HTTP request, via a web browser, the server receives, checks, and processes the request, returns the response, and, finally, the client browser displays the information. This mechanism might work for applications where information does not change frequently, however, for applications which require quick, real time response, such as remote monitoring, it cannot accurately realize real time results; the information displayed on the client side may be outdated on the server. In addition, the system doesn't support real time doctor-patient communication and data can only be transferred one way. In this new connection, keeping the client and server

This work was supported in part by eHealth-MV (Mecklenburg Vorpommern Germany).

synchronized is the challenge to the real-time web remote monitoring system and also a problem for Web developers. A new generation of real-time network technology is mandatory for the field of remote monitoring.

II. Related Work

Today, personal healthcare is one of emerging areas of research. Telehealth care has become an important issue for implementing novel medical services and has become a promising market for industry. Technology giants such as Google, Microsoft, and Intel have being developing their own eHealth systems and related applications in the last few years. The Intel health guide allows clinicians to remotely monitor patients and manage care on an individual basis [4]. To get full benefit of the system, the patient has to acquire the Intel health guide device, PHS600. Microsoft HealthVault is a web-based platform from Microsoft to store and maintain health and fitness information, but devices for vital signs collection should be compatible with Health Vault [5]. Google Health is a web-based system for organizing health information and sharing it with doctors as explained in [6]. The service allows Google users to volunteer their health records – either manually or by logging into their accounts at partnered health services providers – into the Google Health system, thereby merging potentially separate health records into one centralized Google Health profile. The import of data from vital sign devices is still a future development project for Google Health.

The implementation of different telemedical applications has been introduced in several papers. Promising results of stress monitoring, using distributed wireless intelligent sensor systems, have been presented in [7]. In [8] a simple fitness monitoring is presented. In the processing phase of reducing the variance within physiological information data (weight, blood pressure and heart beat), simple algorithms which do not take into account other facts are applied. A large body of literature suggests that Heart-Rate Variability (HRV) analysis can potentially be used for the assessment of stress. However, a practical problem that is, so far, not well addressed in literature is to derive some form of quantitative relationship between parameters of ANS activity and stress. The major difficulty in establishing these kinds of relations, is the difference in behavior among individuals, due to different body conditions, gender, age, physical fitness, emotional states, and so on. To handle such problems, the authors in [9] propose a fuzzy-filter based algorithm for the preprocessing of the physiological signal [10] .

III. System Structure

A. System Description

The overall system structure, illustrated in Fig. 2, is based on a client-server-architecture that is designed as a tiered distributed system. The system consists of three major tiers, the WBASN, the data processing center and the mHealth portal.

Three Tiers

- The bottom layer, WBASN, is responsible for collection, calculation, and transmission of vital data [11]. Different wearable medical body sensors such as pulse oximeter, heart rate monitor, blood pressure monitor, thermometer, weighing scale, glucose meter, etc., can be connected to the gateway via the short range wireless communications interface. The gateway functionality is provided by smartphones. The mHealth client, running on the smartphone, is responsible for the collection of sensor node data and for data transmission to a remote data processing center.
- The data processing center, based on a Hadoop server cluster, is responsible for data collection, analysis, processing, and data mining.
- The upper layer is the mHealth portal. It is in charge of the integration of heterogeneous modules and the interactions among them, providing a unified application service for users.

B. System Operating Principle

The development of an acquisition system allowing the subjects to move freely in their familiar work surroundings and, moreover, allowing the examiner to monitor their condition continuously from any location, offers optimal preconditions for research in occupational health [12]. The data acquisition is mostly related to noninvasive measurement of physiological parameters (i.e. HR, HRV, breath frequency, skin temperature....) and indirect load parameters by accelerometer sensors Ref. [13].

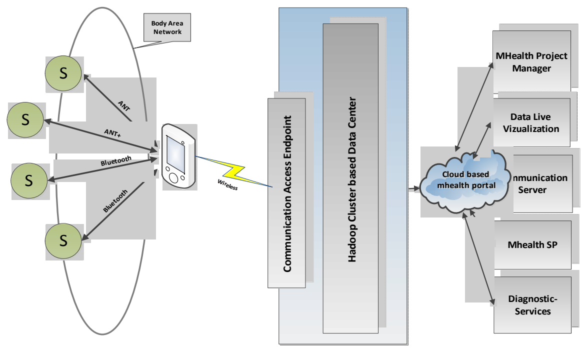


Figure 2. mHealth System Structure

A wireless connection is established between sensor and smartphone via near-field communication protocols such as Bluetooth and ANT/ANT+. A WebSocket connection and data interaction channel are established between mobile client and remote server. Through the introduction of WebSocket technology, researchers can use a web browser to acquire the data sent by the sensor collection terminal, extending the central monitoring platform to the computers of doctors and researchers, without the need for installing extra software [14].The system operating principles are as follows:

a) After the server system is initiated, a WebSocket server runs in background and monitors the connection requests.
b) The client system automatically sends requests of WebSocket connections to the server after being initiated.
c) The server agrees to connect, returns client-ID, checks client configuration information and synchronizes configuration information.
d) The user opens the smartphone client system and searches the sensors in visible range.
e) The system maintains a supported sensors list, automatically identifying the client-end and sensor communication protocols. As a Master Node, it can connect to multiple sensor nodes simultaneously.
f) Doctors and researchers can open the Telemedical Platform page, and acquire the Patient's Online Status, review the physiological parameters received by the server, and store the physiological parameters in the database.

The server connects via webservice with other functional modules such as the stress and fitness modules. After obtaining the physiological parameters, test results will be returned to the remote server monitoring platform and sent to the mobile client by the server.

The client-side can establish connections with multiple sensors, adopt the interface-orientated design, flexibly extending analysis methods for different sensors.

III. System Design

As web applications become more widespread and complex, some applications will demand new requirements for web applications, similar to real-time monitoring systems for stock market trading applications where instant web data communications is mandatory. However, the basic HTTP protocol of web applications strictly abides by the request-response models, where the client side sends a request to the server which sends a response to the client. Contrary to the full duplex connection of C/S applications, the HTTP connection is unidirectional. However, the C/S structure requires a specific client-side, which cannot satisfy the general requirements of telemedicine. As the web standard for the next generation, HTML5 has many compelling new features, including the WebSocket, called Web TCP, attracts developers' attention. The emergence of WebSocket makes it possible for the browser support socket, providing a two-way channel based on a TCP connection between browser and server. Web developers can easily use WebSocket to build real-time web applications.

The WebSocket protocol is essentially a TCP-based protocol. In order to establish a WebSocket connection, the client browser will initiate an HTTP request to the server. Unlike the usual HTTP request, this request contains some additional header information. The additional header information "Upgrade: Websocket" indicates that this is an HTTP request applying for protocol upgrade. The server-side resolves the additional header information and then generates response messages back to the client. In this way, the WebSocket connection between client and server is established and the information can be transmitted through this channel. This connection will serve continuously until either the client or the server side initiate closing the connection. Compared with the current web real-time technology, WebSocket can not only avoid the connection and portability issues in Comet, but can provide more efficient solutions than Ajax [15]. The WebSocket protocol provides a full-duplex, bidirectional communication channel that operates through a single socket over the Web and can help build scalable, real-time Web applications.

The WebSocket protocol has two parts. The handshake consists of a message from the client and a handshake response from the server. The second part is data transfer. Jetty's implementation of the WebSocket API is fully integrated into the Jetty HTTP server and servlet containers (see http://jetty.codehaus.org/jetty). Thus, a Jetty servlet can process and accept a request to upgrade an HTTP connection to a WebSocket connection. Further details on the WebSocket communication process are available in our prior work [16].

A. Server Software Design

The layered structure of the Telemedical-Portal is illustrated in Fig.3.

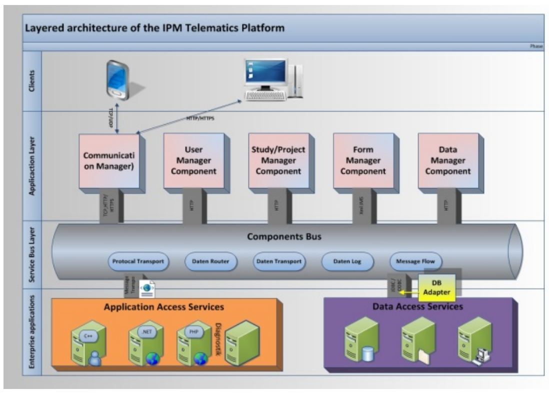


Figure.3: Layers of the Telematics Server

The module-based development approach has been used to develop each function, enhancing the flexibility and scalability of the entire system. The Telemedical-Portal [17] serves as a Components Bus and connects users to the implemented module. Different user modules such as Admin Manager, Data Communication Module, Live Visual Module, and Data Analysis module, constitute the core functions of the system as illustrated in Fig. 4.

The Data Communication Module uses multi-thread technology and works as follows:

A WebSocket monitoring thread would be created for each received client connection to handle events such as signal channel connection, suspension, disconnection and abnormality. The number of monitoring threads is only limited by system resources. The monitoring thread, after the connection is established, will be suspended and proceed to enter monitoring mode and establish a data processing sub-thread for specific communication between clients. The processing of each sub-thread is relatively independent. The functions of the server are not limited to data flow communication, but also include human-computer interaction, waveform drawing, data analysis and so on; it will not process data while monitoring the thread directly, but establish a separate data process thread for each connection.

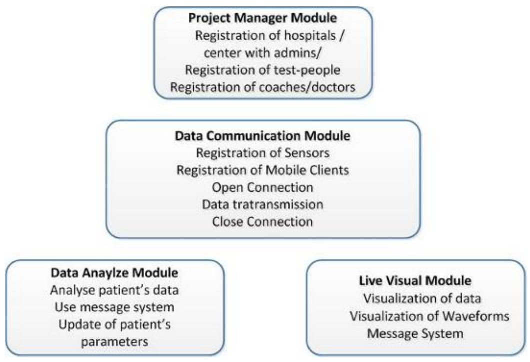

Project Manager Module
Registration of hospitals / center with admins/
Registration of test-people
Registration of coaches/doctors
Data Communication Module
Registration of Sensors
Registration of Mobile Clients
Open Connection
Data tratransmission
Close Connection
Data Anaylze Module
Analyse patient's data
Use message system
Update of patient's parameters
Live Visual Module
Visualization of data
Visualization of Waveforms
Message System

Figure.4. Mobule Structure in the of the Telematics Server

Between server and Client, the WebSocket connection is established by the WebSocket agreement during the first "handshake". It is also based on the underlying TCP/IP protocol. WebSocket agreement is relatively simple. The client, using an ordinary browser, sends a request for the handshake to the server through port 80 or 443. The server will identify whether it is a WebSocket request or not, according to HNP header. If so, it will upgrade the request to a WebSocket connection. After a successful handshake, it will enter the data transferring stage of two-way connection. WebSocket data transmission is a frame-based approach: 0x00 indicates that the data will begin. 0xFF indicates when the data transfer is finished. The data will be coded by UTF8. The following is an example script for a WebSocket connection with the communication module on the server Ref. [18].

```
var  wsServer = 'ws://localhost:8080/mhealth';
var  websocket = new WebSocket(wsServer);
websocket.onopen = function (evt) { onOpen(evt) };
websocket.onclose = function (evt) { onClose(evt) };
websocket.onmessage = function (evt){onMessage(evt)};
websocket.onerror = function (evt) { onError(evt) };
function onOpen(evt) {  ….. }
function onClose(evt) {  ….. }
function onMessage(evt) { ….. }
function onError(evt) {   …..  }
```

To establish a WebSocket connection, the client sends a WebSocket handshake request, and the server sends a WebSocket handshake response, as shown in the following example:

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Server response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

B. Middleware Design

Fig. 8 depicts the top view of the mHealth client middleware that is designed to connect different medical sensors and runs on a Sony Ericsson Xperia Arc Smartphone. The mHealth middleware consists of the Universal Sensor Interface Module, the Service Module, the Communication Module, the Security Module and a Gateway Proxy interface.

- *Universal Sensor Interface Module (USIM):* In terms of the Universal Sensor Interface, we define different objects which can be used to represent service and parameters for different sensor devices. It consists of configuration parameters for the medical sensor, the universal medical sensor data transmission format, and a Universal Sensor Transducer Interface (USTI). USTI is the basic object to enable the interaction between the mHealth middleware and different sensing devices. Here we use the medical device encoding standard, including network key, acquisition frequency, the sample array, the binary encoding rule (BER) and packet encoding rule (PER) [19].
- *Service Module (SM):* The Service Module is used to exchange data between sensor nodes and middleware. It has a data analysis interface which defines the general methods of analyzing sensor equipment. The system uses device specific analytical files to facilitate the integration of different sensors.
- *Communication Module (CM):* In order to enable the parallel connection of multi-sensor devices, we use the CM to define the main processes and sub-processes to realize the connection, combination and stop of the communication channel[23].
- *Security Module (SecM):* On the IPM-mHealth client side we implement the new Hummingbird-2 cipher to encrypt the sensor data with java code. On the server side, we use Secure Socket Layer (SSL) to encrypt vital data and utilize 3P-AAA-SPs [20], Authorization and Accounting Service Providers to authenticate the client access.

Hummingbird-2 is a new lightweight cryptographic algorithm targeted for resource-constrained devices like smart cards, and wireless sensor node.

The Hummingbird-2 cipher has a 128-bit secret key K and a 128-bit internal state R which is initialized using a 64-bit Initialization Vector IV. These variables are accessed as vectors of 16-bit words:

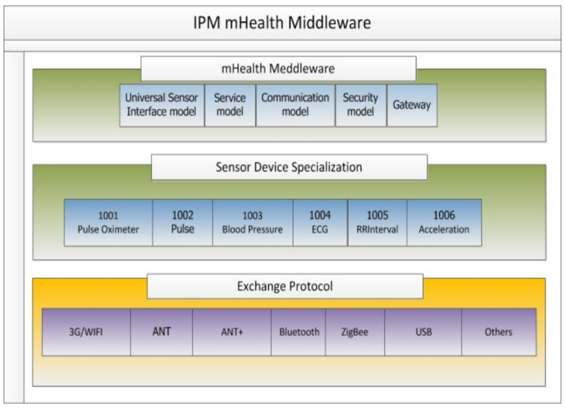


Figure 5. mHealth Middleware

$K = (K1, K2, K3, K4, K5, K6, K7, K8),$

$R = (R1, R2, R3, R4, R5, R6, R7, R8),$

$IV = (IV1, IV2, IV3, IV4),$

Hummingbird-2 is entirely built from operations on 16-bit words: the exclusiveor operation on words:⊕, addition modulo 65536:⊞,and a nonlinear mixing function f(x) .

Encryption of a single word of plaintext P_i to ciphertext word C_i requires four invocations of WD16 Ref. [22].

$t_1 = WD16\ (R_1^{(i)} \boxplus P_i\ , K_1, K_2, K_3, K_4)$

$t_2 = WD16(R_2^{(i)} \boxplus t_1, K_5 \oplus R_5^{(i)}, K_6 \oplus R_6^{(i)}, K_7 \oplus R_7^{(i)}, K_8 \oplus R_8^{(i)})$

$t_3 = WD16(R_3^{(i)} \boxplus t_2, K_1 \oplus R_5^{(i)}, K_2 \oplus R_6^{(i)}, K_3 \oplus R_7^{(i)}, K_4 \oplus R_8^{(i)})$

$c_i = WD16(R_4^{(i)} \boxplus t_3, K_5, K_6, K_7, K_8) \boxplus R_1^{(i)}$

Javacode for nonlinear mixing function f(x) :

```
private char Hummingbird_LinearTransformation(char word) {
    // forward operation = x XOR (x RotateLeft 6) XOR (x RotateLeft 10);
    char retval;
    retval = Hummingbird_EXOR(word,
            (char) (((word >>> 10) & 0x003F) | ((word << 6) & 0xFFC0)));
    retval = Hummingbird_EXOR(retval,
            (char) (((word >>> 6) & 0x03FF) | ((word << 10) & 0xFC00)));
    return (retval);
}
```

C. Client Software Design

The smartphone client is researched and developed on the Android v2.3.3 operating system. By adopting the interface-oriented design, the call sensors connection method, and its concrete realization, have been separated, which achieved the target of scalability. The core functions include Server Communication Modules and Sensor Communication Modules. It consists of the following classes and interfaces: singelton Class ConnectManager, Class SensorManager, Interface DataParser and Activity SensorActivity. It requires two class libraries support of ANT Radio Service API and Android Bluetooth API.

When the phone activates the sensor's communication module, there will be a singleton ConnectManager and listening system broadcast. When choosing to establish the connection, first the SensorManager object will be created and the sensor type and ID will be established and then start a new thread to establish a connection, waiting for a response from the sensor side. After receiving the sensor data, call DataParser interface to parse the data and sent it by Hanler to SensorUI and update the UI interface. In order to improve the response time of the program, sub-threads are used at different positions. The asynchronous processing of a variety of different events can prevent the machine from running between events needlessly. The flow chart for the sensor communication establishment is shown in Figure 6.

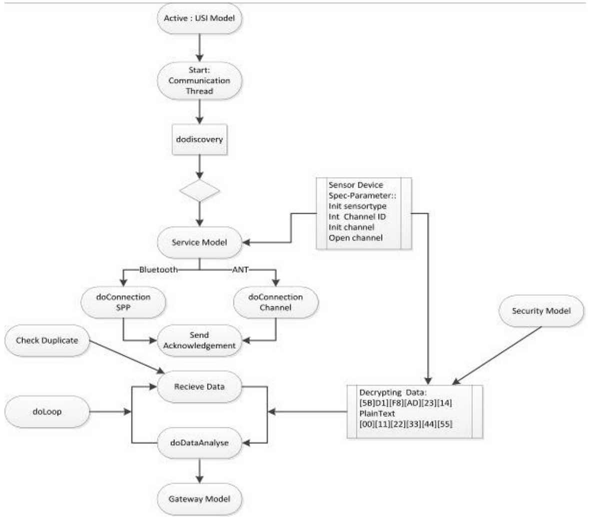

Active : USI Model
Start: Communication Thread
dodiscovery
Sensor Device Spec-Parameter:: Init sensortype Int Channel ID Init channel Open channel
Service Model
Bluetooth
ANT
doConnection SPP
doConnection Channel
Security Model
Check Duplicate
Send Acknowledgement
Recieve Data
doLoop
Decrypting Data: [5B]D1][F8][AD][23][14] PlainText [00][11][22][33][44][55]
doDataAnalyse
Gateway Model

Figure 6 : Flow of sensor communication creation

III. EXPERIMENTAL DESIGN

In these experiments, we used an mHealth server that accepts WebSocket connections and bounces every message it gets, back to the client. We wrote some javascript that establishes a connection, sends several messages to the server, then measures how long it takes for each message to be sent back. Once this data is gathered, we can produce a histogram. We did this in a way that resembles the situation where messages are sent periodically without first waiting for a reply.

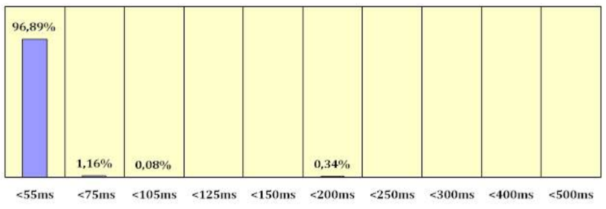
In the tests below, we used IPFW (Mac OS X's firewall/router tool) [21] to model different amounts of latency and packet loss on 250 samples. On OS X, it is easy to model latency and packet loss rate with the IPFW tool. We've simulated different packet loss rates on a low latency connection. First, we first ran these commands (as root):

```
$ ipfw add pipe 1 ip from any to any out
$ ipfw add pipe 2 ip from any to any in
$ ipfw pipe 1 config delay 12ms
$ ipfw pipe 2 config delay 12ms
```

This will result in a round trip time of about 50ms. (The packet is delayed by 12 ms twice in each direction, for a total of 48ms, which is pretty close to 50ms.) After we ran these commands, we measured the message latency at a variety of packet loss rates and produced histograms from the results. We then ran experiments with the following latency and packet loss combinations.

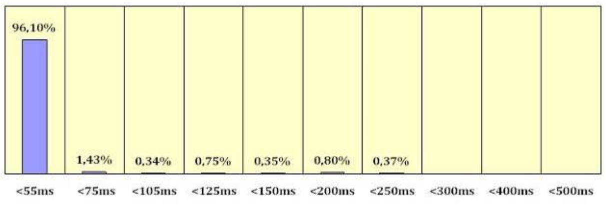
a. *50ms Latency, 0.1% Packet Loss* This will mean about 0.1% of messages will get dropped. It's 0.1% because they have a 0.05% chance of getting dropped on the way out and 0.05% on the way back. (It's actually 100 × (1 - .9995²) percent. 0.1% is close enough.)

```
$ ipfw pipe 2 config delay 12ms plr 0.0005
```



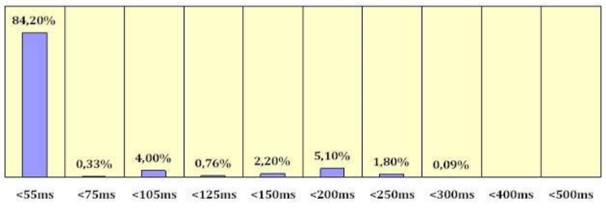
b. *50ms Latency, 1% Packet Loss*

```
$ ipfw pipe 2 config delay 12ms plr 0.005
```



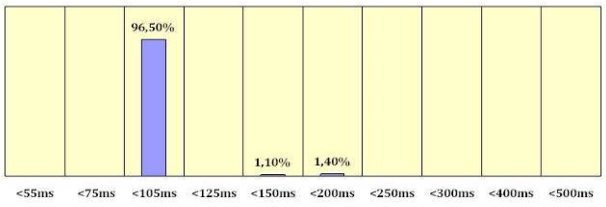
c. *50ms Latency, 5% Packet Loss*

```
$ ipfw pipe 2 config delay 12ms plr 0.025
```



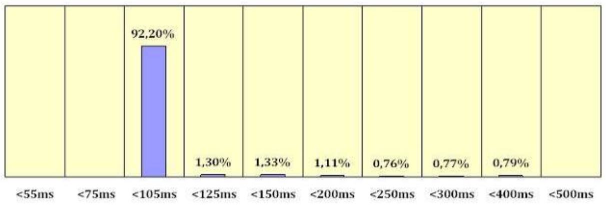
d. *100ms Latency, 0.1% Packet Loss*

```
$ ipfw pipe 2 config delay 25ms plr 0.0005
```



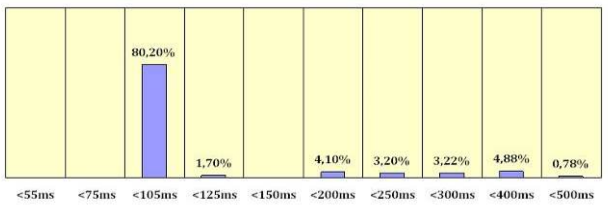
e. *100ms Latency, 1% Packet Loss*

```
$ ipfw pipe 2 config delay 25ms plr 0.005
```



f. *100ms Latency, 5% Packet Loss*

```
$ $ ipfw pipe 2 config delay 25ms plr 0.025
```



IV. Conclusion

The experimental environment is shown in Fig.6. We used a Sony Ericsson Xperia smartphone as the Client. The server runs outside the laboratory with independent IP and carries out various tests. Our results are based on a small sample of the possible Internet communication paths, and for a very limited duration, but are likely to be a good estimate of how WebSocket communication works for low-volume, continuous, real-time traffic. The stateful approach that the WebSocket protocol uses provides better latency (on average) for real-time Internet communication Ref. [15]. The header message consumes 570 bytes and 50 milliseconds during initial connection request and keeps the connection open to provide an exact, and efficient, data flow for the browser, ensuring minimal transmission delay and providing the end-user experience of desktop applications.

The multi-physiological parameter real-time monitoring system introduced in this paper, implements functions such as multi-physiological parameter data collection, testing, storage, network transmission and real-time online custody. The developed system makes up for the shortcomings of existing remote monitoring systems which only detect a single parameter and lacks any network interaction functionality. Meanwhile, it adopts a real-time remote monitoring system based on WebSocket B/S mode which requires only server side software management and maintenance. All clients use only browsers, requiring no maintenance, which provides new solutions for spreading remote monitoring systems.

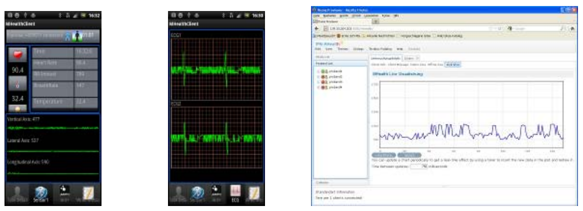


Figure.7: An example of collect physiological data and send data from smartphone to server

V. Acknowledgment

This work is part of the eHealth-MV (Mecklenburg Vorpommern) project which results from the cooperation between University of Rostock Institute for Preventive Medicine and Center for Life Science Automation Germany. The authors thank the ministry of economy, work and tourism in Mecklenburg-Vorpommern Germany for the financial support of this project.

REFERENCES

[1] H. Laitko, " Theoria cum praxi : Fünf Jahre Leibniz-Institut für interdisziplinäre Studien e.V. (LIFIS) Berlin," Trafo Verlag, pp.217-228, 2007.

[2] S. Holzmüller-Laue, B. Goede, R. Stoll, and Thurow. K, "A Highly Scalable Information System as Extendable Framework Solution for Medical R&D Projects," The XXIInd International Congress of the European Federation for Medical Informatics, vol. 150, pp. 101–105, 2009.

[3] P. Harvey, B. Woodward, S. Datta, and D. Mulvaney, "Data acquisition in a wireless diabetic and cardiac monitoring system ," Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 3154-3157, 2009.

[4] Intel-Health-Guide, online at http://www.intel.com/corporate/healthcare/emea/eng/healthguide/i ndex.htm [Accessed 06-12-2011].

[5] Microsoft-Health-Vault, online at http://www.healthvault.com [Accessed 06-12-2011].

[6] Google-Health, online at http://www.google.com/health [Accessed 06-12-2011].

[7] E. Jovanov, A. O. Lords, D. Raskovic, P. Reza-Adhami, and F. Andrasik, "Stress Monitoring Using Distributed Wireless Intelligent Sensors System," in IEEE Engineering in Medicine and Biology, pp. 49-55, 2003.

[8] D. Jea, J. Liu, T. Schmid, and M. Srivastava, "Hassle Free Fitness Monitoring," presented at Second International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments, Breckenridge, Colorado, 2008.

[9] M. Kumar, M. Weippert, D. Arndt, S. Kreuzfeld, K. Thurow, N. Stoll, and R. Stoll, "Fuzzy Filtering for Physiological Signal Analisis," IEEE Transactions on Fuzzy Systems, Vol. 18, pp. 208- 216, 2010.

[10] P.S. Pandian, K.P. Safeer, P. Gupta, D.T. Shakunthala, B. S. Sundersheshu, V. C. Padaki, "Wireless Sensor Network for Wearable Physiological Monitoring," Journal of Networks, Vol 3, pp.21-29, May 2008.

[11] R. Stoll, D. Arndt, M. Weippert, S. Kreuzfeld, K. Thurow, "Stress monitoring in hoch automatisierten Umgebungen der Life Sciences," BioSpektrum , pp. 258–261, 2008.

[12] S. Neubert, D. Arndt, K. Thurow, and R. Stoll, "Mobile real-time data acquisition system for application in preventive medicine," Telemedicine and e-Health, Vol. 16, pp. 504-509, 2012.

[13] Websocket online at http://www.websocket.org/ [Accessed 06-04-2012]

[14] F.Y. Jiang, H.C. Duan, "Application research of WebSocket technology on web tree component," Proceedings of 2012 International Symposium on Information Technologies in Medicine and Education, ITME, pp. 889-892, 2012.

[15] V. Pimentel, B.G. Nickerson, "Communicating and displaying real-time data with WebSocket " IEEE Internet Computing, 16 (4), pp. 45-53, 2012.

[16] Jwebsocket online at http://jwebsocket.org/ [Accessed 06-04-2012]

[17] R.D. Berndt, M.C. Takenga, S. Kuehn, P. Preik, N. Stoll, K. Thurow, M. Kumar, M. Weippert, A. Rieger, R. Stoll, "A scalable and secure telematics platform for the hosting of telemedical applications. Case study of a stress and fitness monitoring," 2011 IEEE 13th International Conference on e-Health Networking, Applications and Services, HEALTHCOM, pp. 118-121, 2011.

[18] J.T. Hsu, S.H. Hsieh, C.C. Lo, C.H. Hsu, P.H. Cheng, S.J. Chen, F.P. Lai, "Ubiquitous mobile personal health system based on cloud computing," IEEE Region 10 Annual International Conference, Proceedings/TENCON,pp. 1387-1390, 2011.

[19] Z. Ji, X. Zhang, I. Ganchev, M. Odroma, "A personalized middleware for ubiquitous mHealth services," IEEE 14th International Conference on e-Health Networking, Applications and Services, Healthcom , pp. 474-476, 2012.

[20] X.H. Le, M. Khalid, R. Sankar, S. Lee, "An Efficient Mutual Authentication and Access Control Scheme for Wireless Sensor Networks in Healthcare," Journal of Networks, pp.355-364, Mar 2011.

[21] http://blog.artillery.com/2012/06/websocketperformance.html [Accessed 11-03-2011]

[22] D. Engels, M.J.O. Saarinen, P. Schweitzer,E.M. Smith, "The hummingbird-2 lightweight authenticated encryption algorithm" Lecture Notes in Computer Science, pp. 19-31, 2012.

[23] G.Q. Liao, "Data Synchronization and Resynchronization for Heterogeneous Databases Replication in Middleware–based Architecture," Journal of Networks, Vol 7, 210-217, Jan 2012.

Weiping Zhang received the Diploma degree in Computer Science from Technical University of Dresden, Dresden, Germany, in 2009.

Since 2011, he has been a Research Scientist with the Institute of Preventive Medicine, University of Rostock, Rostock, Germany. His research interests include process information management systems and real-time mobile measurements of physiological parameters.

Regina Stoll received the Dip.-Med. degree in medicine, the Dr.med. degree in occupational medicine, and the Dr.med.habil. degree in occupational and sports medicine from the University of Rostock, Rostock, Germany, in 1980, 1984, and 2002, respectively.

She is currently the Head of the Institute of Preventive Medicine, University of Rostock. She is a Faculty Member with the Medicine Faculty and Faculty Associate with the College of Computer Science and Electrical Engineering, University of Rostock. She also holds the Adjunct Faculty Member position with the Department of Industrial Engineering, North Carolina State University, Raleigh. Her research interests include occupational physiology, preventive medicine, and cardiopulmonary diagnostics.

Norbert Stoll received the Diploma (Dip.-Ing.) degree in automation engineering and the Ph.D. degree in measurement technology from the University of Rostock, Rostock, Germany, in 1979 and 1985, respectively.

He was the Head of the Section Analytical Chemistry with the Academy of Sciences of GDR, Central Institute for Organic Chemistry, until 1991. From 1992 to 1994, he was the Associate Director of the Institute of Organic Catalysis, Rostock. Since 1994, he has been a Professor of measurement technology with the Engineering Faculty, University of Rostock, where he was the Director of the Institution of Automation from 1994 to 2000. Since 2003, he has also been the Vice President of the Center for Life Science Automation, Rostock. His research interests include medical process measurement, lab automation, and smart systems and devices.

Kerstin Thurow studied chemistry with the University of Rostock, Rostock, Germany, and received the Graduate degree. Afterward, she received the Ph.D. degree from the Ludwigs-

Maximilians-University, Munich, Germany (under the guidance of Prof. Lorenz), working on metal–organic sulphur compounds. In 1999, she received the Habilitation degree from the Department of Electrical Engineering, University of Rostock.

She became a Faculty Member with the veni legendi for “Measurement and Control.” In October 1999, she became Germany's Youngest University Professor and obtained the Worldwide Unique Professorship for “Laboratory Automation.” In December 2004, she was appointed novel professorship for “Life Science Automation” this chair is connected with the Center for Life Science Automation (CELISCA) Management Directorate, Rostock. She is currently the Managing Director of the Institute of Automation, University of Rostock. She is also the Managing Director for the CELISCA and the President of the Institute for Measurement and Sensorsystems. As a Founding Member and President of the Rostock-Raleigh e.V.—a Sister City Association—she is striving for cultural, sportive, and scientific and economical relations to one of the most important Life Science Regions in the United States.

Dr. Thurow received the highly renowned Joachim-Jungius-Award of Science in 2004, in addition to many awards, such as for the foundation of a start-up company Amplius—Screening Technologies & Analytical Measurement.