ATLANTIS PRESS

Advances in Intelligent Systems Research (AISR), volume 155

2018 International Conference on Advanced Control, Automation and Artificial Intelligence (ACAAI 2018)

A Medical Device Interaction System based on IEEE 11073

Zhiwei Wen[1,3], Jinzhong Cui[1,3] and Leiting Chen[1,2,3]

[1]Digital Media Key Lab of Sichuan Province, University of Electronic Science and Technology of China, Chengdu, China

[2]Institute of Electronic and Information Engineering in Guangdong Province, University of Electronic Science and Technology of China, Chengdu, China

[3]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

***Abstract*—One of the most significant achievements in modern healthcare systems is the release of the IEEE 11073 family of standards, which formalize the compatibility and interoperability of medical devices and external computer systems. However, IEEE 11073 does not define standards and modes of communication between medical devices. This paper designs a medical device interaction system based on IEEE 11073 that enables communication between medical devices. The system includes a smart gateway that serves as the IEEE 11073 manager and middleware of the system to connect one medical device to another. Specifically, the paper extends the IEEE 11073 stack for more efficient interaction between medical devices.**

Keywords—Medical devices; interaction; IEEE 11073; communication; data

I. INTRODUCTION

As the average age of the population increases, a growing number of people are requesting in-home healthcare. Accordingly, the popularization of personal medical devices has become inevitable. The IEEE 11073 family of standards (X73 standards) was developed to define communication standards between medical devices and external computer systems. It aims to allow different medical devices from different manufacturers to work together, benefiting both users and manufacturers. Although the IEEE 11073 family of standards defines the communication standards between medical devices and external computer systems, it does not define the communication standards between medical devices.

The healthcare system includes a variety of medical devices such as ECG monitors, heart rate monitors, oximeters, sphygmomanometers, and so on. These medical devices contain sensors that collect the user's physiological data and send it to a medical data center for analysis and processing. In order to make the healthcare system more convenient and useful, it is necessary to connect medical devices to each other through a network. Since there are no communication standards for interactions between medical devices, inter-device communication can be difficult and degrade users' experience of the healthcare system. Currently, there are related studies on the integration of medical devices into the healthcare system so that the devices can communicate with each other [1] [2]. In order to achieve more efficient interactions between medical devices, this paper designs a medical device interaction system based on IEEE 11073. Specifically, the IEEE 11073 stack (X73 stack) has been extended by this paper, providing more efficient interaction between medical devices. As the Android operating system is now commonly used in mobile devices, Android mobile telephones were selected to act as a smart gateway to enable communications between medical devices.

II. TECHNIQUE OVERVIEW

A. IEEE 11073 Family of Standards

The objective of the IEEE 11073 family of standards [3] is to provide real-time plug-and-play interoperability for medical devices. The standards enable communication between medical devices and external computer systems. The standards can be divided into two parts: the IEEE 11073-20601 Optimized Exchange Protocol and the IEEE 11073-104zz (zz ranges from 01 to 99) device specializations, which define details for medical devices such as blood glucose and blood pressure monitors. The IEEE 11073-20601 Optimized Exchange Protocol is a core part of the IEEE 11073 family of standards. It defines a framework that contains three models: the domain information model (DIM), the service model, and the communications model. The IEEE 11073 family of standards defines two device types: *agents* and *managers*. Agents are data producers, while managers are data collectors. The DIM model is defined using an object-oriented model. It describes a set of objects from an agent, and one or more attributes for each object. Attributes are used to describe the status of an agent and to control its behavior. The service model provides data-access commands, such as GET and SET, to allow managers to access data from an agent. The communication model defines the session state machine and the state-driven rules between agent and manager. These three models work together to allow medical devices to effectively interact with external computer systems. Figure 1 provides an overview of the IEEE 11073 family of standards.

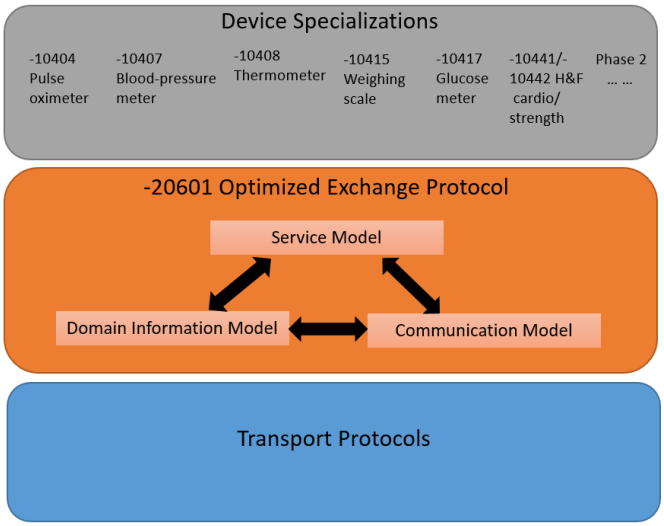

FIGURE I. OVERVIEW OF THE IEEE 11073 FAMILY OF STANDARDS.

B. *Java Native Interface*

The Java Native Interface (JNI) [4] is part of the Java programming platform. It is a very important feature that allows Java programs to call functions written in a native language, generally C/C++. The JNI is regarded as an adaptor for communications between Java and native languages. The emergence of JNI enables developers to take advantage of both the rich class libraries of the Java language and the high performance of native languages. The JNI code in Android is mainly located in the application and application framework layers. The application layer is developed by using a standard JNI programming model, while the application framework layer is developed using a custom JNI programming model in Android. The custom JNI programming model makes up for the shortcomings of the standard JNI programming model. The JNI can call native language and interact with the application and application framework layers through the Dalvik virtual machine. Figure 2 shows the JNI in the Android framework.

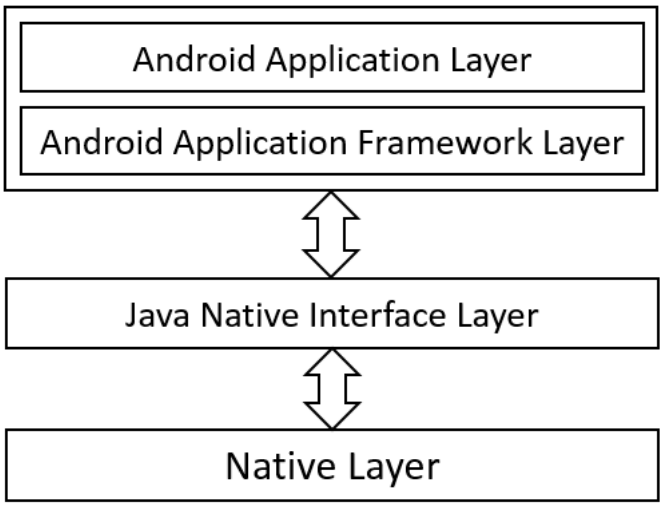

FIGURE II. THE JNI IN THE ANDROID FRAMEWORK.

III. SYSTEM MODEL

Figure 3 shows the system model. The model includes three parts, as follows:

- Master devices: The master devices are used to gather users' physiological data, which are sent to the smart gateway as X73 messages.
- Smart gateway: The smart gateway acts as middleware to enable communications between medical devices. The smart gateway and devices are connected via Bluetooth, because the Bluetooth Health Device Profile is mainly designed for the IEEE 11073 family of standards and X73 messages can be transmitted via Bluetooth.
- Slave devices: The slave devices are used to receive and process the X73 messages sent by the smart gateway.

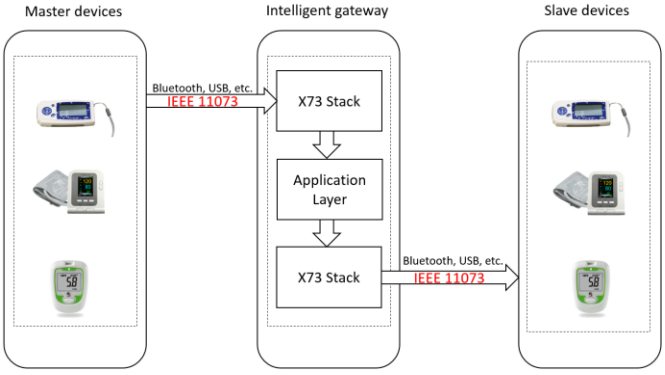

FIGURE III. SYSTEM MODEL

A. *Master and Slave Devices*

The X73 standards define the *agent* and *manager*. Agents are the sources of medical data, while managers are responsible for receiving data sent by agents. Data transmission between agents and managers can be initiated by either side. In this system, a piece of medical equipment serves as an agent. An agent that takes the initiative to send data to a manager is the *master device*, and agents that passively receive data sent by managers are *slave devices*.

B. *Smart Gateway*

The *smart gateway* serves as the X73 manager and middleware of the system that connects the master and slave devices. The main job of the smart gateway is to set events and response events appropriately. The smart gateway is composed of a X73 stack and an application layer. The X73 stack is used to set the X73 messages sent from the master devices to the events. Typically, the system extends the X73 stack to quickly match X73 messages to an event. The application layer is used to set the response event. When the X73 stack matches the X73 messages to an event, it notifies the response event. Then, the application layer generates X73 messages according to the response event and sends them to the slave devices.

IV. IMPLEMENTATION

Figure 4 shows the system architecture. The architecture is divided into three layers: the *device layer*, *X73 stack*, and *application layer*. The smart gateway is composed of the X73 stack and application layer. In order to take advantage of the high performance of native language, the X73 stack is developed using the C language, and JNI allows the Java programs to interact with the X73 stack.

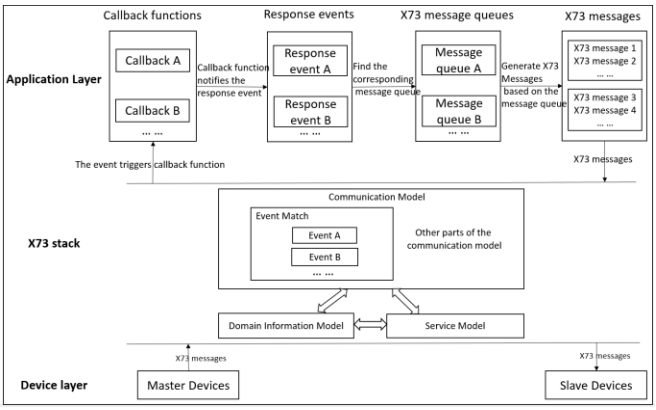

FIGURE IV. SYSTEM ARCHITECTURE

A. Data Transmission Modes

The device layer is composed of master devices and slave devices. The X73 standards define two data transmission modes: the *agent-initiated mode*, where an agent takes the initiative to send data to the manager; and the *manager-initiated mode*, where a manager requests data stored by an agent. The master devices adopt the agent-initiated mode and take the initiative to send data to a manager. The slave devices adopt the manager-initiated mode and passively receive data sent by a manager. Figure 5 shows the data transmission modes.

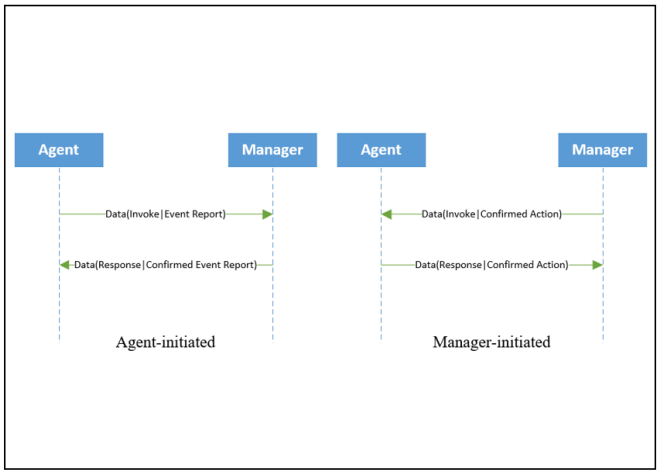

FIGURE V. DATA TRANSMISSION MODES

B. X73 Stack

The X73 stack, which consists of a service model, a domain information model, and a communication model, is used to control communications between medical devices and smartphones. In particular, in order to quickly match X73 messages to an event, this paper extends the X73 stack by adding event matching capabilities to its communication model. Figure 6 shows session states of an extended X73 stack. The message processing state is an extended part of the X73 stack's communication model. If the data transmission occurs in the agent-initiated mode, the smart gateway receives the X73 messages from the master devices and the X73 stack decides whether they match to an event. If the X73 stack matches X73 messages to an event, the event will trigger the callback function of the application layer. For example, the ECG monitor based on the X73-10406 standards uses Asist-Lim, Brady-Lim, Taqui-Lim, and QRS-Number attributes to automatically diagnose cardiovascular disease. Then, we can set these four attributes as a cardiovascular event. When the X73 stack matches X73 messages to a cardiovascular event, the event will triggers the callback function of the application layer. If the X73 stack does not match X73 messages to an event, it continues to receive messages from the master until it does so. In addition, if the data transmission occurs in manager-initiated mode, the X73 stack receives X73 messages from the application and sends the messages to the slave devices.

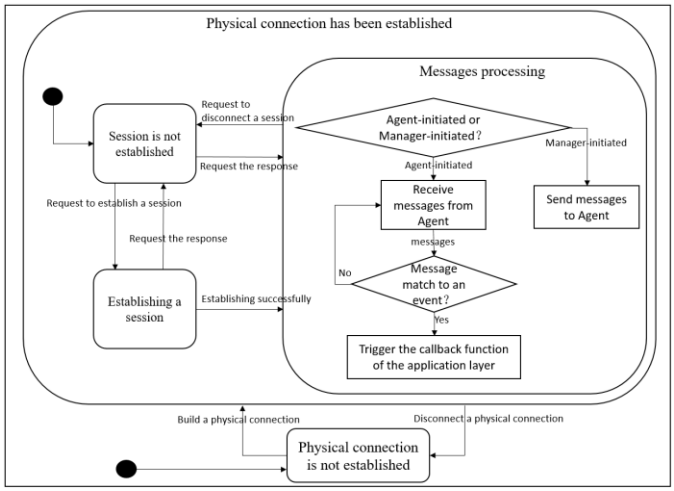

FIGURE VI. SESSION STATES OF AN EXTENDED X73 STACK

C. Application Layer

The application layer serves as the management studio for response events. Each response event corresponds to a message queue, and a message queue is composed of many X73 messages. When the X73 stack matches X73 messages sent by the master devices to an event, it triggers the callback function of the application layer to notify a response event. It then finds the message queue corresponding to the response event and, finally, the application layer generates X73 messages that are sent to the slave device based on the message queue.

In order to better explain how the application layer sets response events, we will use the Sleep Apnea Breathing Therapy Equipment (SABTE) device [5] [6] as an example. SABTE sends and receives X73 messages according to the X73-10424 standard. The DIM of the SABTE is shown in Figure 7.

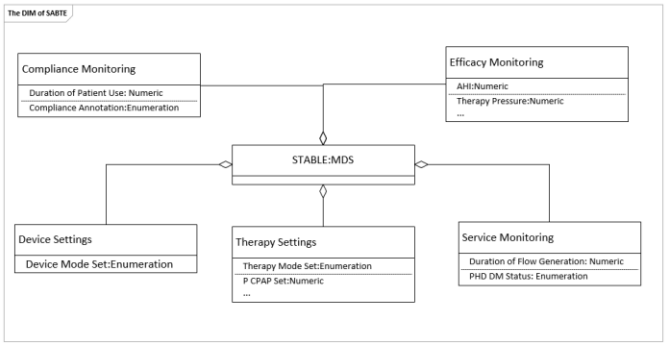

FIGURE VII. THE DIM OF SABTE

The DIM of SABTE has five components: compliance monitoring, efficacy monitoring, device settings, therapy settings, and service monitoring. Each component contains at least one attribute. Managers can use attributes to control the behavior of agents. The application layer sets the attributes that control a certain behavior to a message queue. For example, SABTE has the ability to apply different types of therapy, such as continuous positive airway pressure (CPAP), auto-CPAP and Bi-Level positive airway pressure (Bi-Level PAP). Table 1 shows the message queues that control behaviors. The application layer should associate the response events properly to the message queue. In order to facilitate the transmission, it is a good choice to encode the message queue into JSON format.

TABLE I. MESSAGE QUEUES

Response Event	Message Queue	Behavior
CPAP Event	CPAP / hPa, PAP Waveform / %.	Switch to CPAP mode
Bi-Level-PAP Event	IPAP / hPa, EPAP / hPa, Inspiration Trigger Sensitivity / % or Self-Adjusting, I:E Ratio / % or Self-Adjusting, Minimum Respiratory Frequency /bpm or Self-Adjusting.	Switch to Bi-Level-PAP Event mode
Auto-Bi-Level-PAP Event	Min IPAP / hPa, Max IPAP / hPa, Min EPAP / hPa, Max EPAP /hPa.	Switch to Auto-Bi-Level-PAP Event mode

D. Flowchart of the System

Figure 8 shows the flowchart of the system. The main steps are as follows:

a) The master device sends messages to the X73 stack.

b) The X73 stack receives messages.

c) Can the X73 stack match X73 messages to an event? If yes, go to step d; if not, go to step b.

d) The event triggers the callback function to notify a response event and find the corresponding message queue.

e) The application layer generates X73 messages based on the message queue and sends them to the X73 stack.

f) The X73 stack sends messages to control the slave devices.

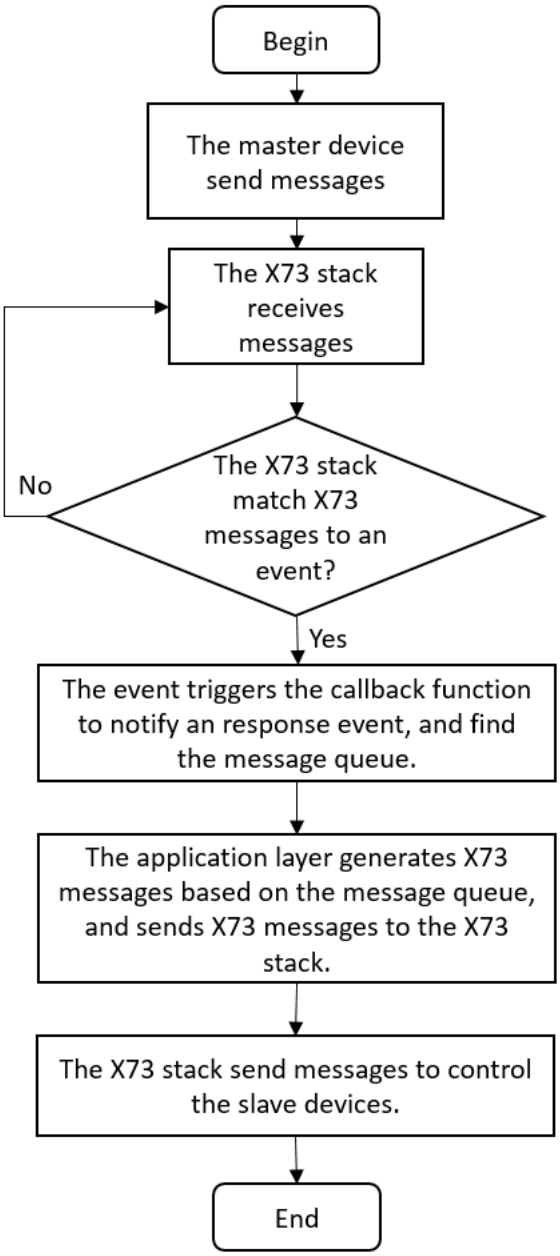

FIGURE VIII. FLOWCHART OF THE SYSTEM

V. CONCLUSIONS

The paper proposes a medical device interaction system based on IEEE11073. The system includes three parts: master devices, a smart gateway, and slave devices. The system puts most of the burden on the smart gateway, which is used to set events and response events, and acts as middleware to enable communications between medical devices. In order to allow more efficient interaction between medical devices and avoid dealing directly with the X73 messages which are complex, the paper extends the X73 stack and encodes the X73 messages of

the application layer into the JSON format. The system can significantly improve the development of electronic healthcare systems.

ACKNOWLEDGMENT

This research was supported by Application Science and Technology Planning Project of Guangdong Province (No.2015B010131002), The National High Technology Research and Development Program ("863"Program) of China (No.2015AA016010), Major Science and Technology Projects of Dongguan (No.2015215102).

REFERENCES

[1] AF Martins, DFS Santos, A Perkusich, and HO Almeida, "UPnP and IEEE 11073: Integrating personal health devices in home networks," Consumer Communications and NETWORKING Conference IEEE, pp. 1-6, 2014.

[2] J Song, XY Kong, HC Wu, and Y Wei, "Integrate Personal Healthcare Devices into Smart Home with ISO/IEEE 11073 PHD Standard," Applied Mechanics & Materials, 394, 487-491, 2013.

[3] "IEEE Health informatics - Personal health device communication - Part 20601: Application profile- Optimized Exchange Protocol," IEEE Std 11073-20601, pp. 1-253, 2014.

[4] Lee, Yann Hang, P. Chandrian, and B. Li. "Efficient Java Native Interface for Android Based Mobile Devices," IEEE, International Conference on Trust, Security and Privacy in Computing and Communications IEEE Computer Society, pp. 1202-1209, 2011.

[5] HG Barrón-González, M Martínez-Espronceda, JD Trigo, S Led, and L Serrano, "Recent Advances in mHealth: An Update to Personal Health Device Interoperability Based on ISO/IEEE11073," Adibi S. (eds) Mobile Health. Springer Series in Bio-/Neuroinformatics, vol 5. Springer, Cham, 2015.

[6] "IEEE Health informatics - Personal health device communication - Part 10424: Device Specialization–Sleep Apnoea Breathing Therapy Equipment (SABTE) - Corrigendum 1," IEEE Std 11073-10424, pp. 1-29, 2017.