33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

A Concept to Standardize Raw Biosignal Transmission for Brain-Computer Interfaces

Christian Breitwieser, Christa Neuper, Gernot R. Müller-Putz

***Abstract*— With this concept we introduced the attempt of a standardized interface called TiA to transmit raw biosignals. TiA is able to deal with multirate and block-oriented data transmission. Data is distinguished by different signal types (e.g., EEG, EOG, NIRS,...), whereby those signals can be acquired at the same time from different acquisition devices. TiA is built as a client-server model. Multiple clients can connect to one server. Information is exchanged via a control- and a separated data connection. Control commands and meta information are transmitted over the control connection. Raw biosignal data is delivered using the data connection in a unidirectional way. For this purpose a standardized handshaking protocol and raw data packet have been developed. Thus, an abstraction layer between hardware devices and data processing was evolved facilitating standardization.**

I. Introduction

Various brain-computer interface (BCI) systems have been built since 1973 when the idea of a BCI was mentioned the first time by Vidal [1]. All those BCIs have the similarity to deal with brain signals, a small subset of biosignals. To compare and summarize commonalities in BCI systems, Mason and Birch presented a common BCI structure in 2003 [2], shown in Fig. 1. The BCI was divided into distinct modules, each one with a specific responsibility inside the BCI processing chain. Those modules are connected with different interfaces, which can be seen as the key principle for standardization processes for BCI systems.

Tackling the first interface between "Amplifier" and "Feature Extractor", as shown in Fig. 1, commonalities like acquisition of various channels or a defined sampling rate can be found in different BCI system like OpenVibe [3], BCI2000 [4], rtsBCI [5], or xBCI [6]. Every one of those systems acquire data and transmit it for further processing. But as no standardized interface definition between acquisition and the first processing module is available, partly incompatible systems are the result. BCI systems dealing with other signal types than just brain signals like EEG (electroencephalogram) or NIRS (near infrared spectroscopy) are also mentioned in literature [7]–[11]. Such systems, called hybrid BCIs (hBCI), deal with different types of user inputs to form a more flexible BCI by also including assistive technology (joysticks, buttons,...). To deal with different kinds of signals in a standardized way, some kind of abstraction is needed. Additionally the type or manufacturer of a respective data acquisition source should be completely irrelevant for the following processing chain. Such a standardized abstraction layer would enhance flexibility and autonomy concerning used hardware.

Therefore an attempt of a standardized interface for raw biosignal transmission, especially for BCIs, called TiA (Tools for BCI – Interface A) was developed. With this interface it is possible to deal with different kinds of biosignals in a common way. It is a first step to decouple the data acquisition system from the BCI processing chain and provide ensured exchangeability.

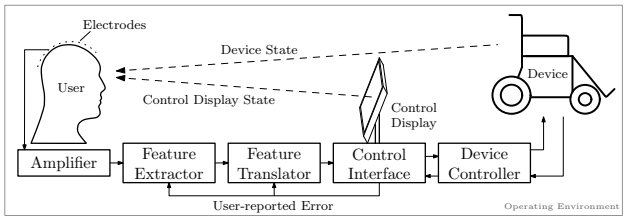

Fig. 1. Functional model from Mason and Birch (modified from [2]).

II. Requirement Analysis and Design

Different BCI systems have already been built using pure brain control as well as hybrid combinations. EEG [12], magnetoencephalogram (MEG) [13], the NIRS signal [14], or the blood oxygen level dependent (BOLD) signal [15] have already been utilized to control a BCI just using mentioned brain signals. Developing hybrid BCI systems, the number of potential kinds of possible signals further increases. Signals like the electromyogram (EMG) [10] and the electrocardiogram (ECG) [16] have already been successfully combined with EEG to control a hybrid BCI. But also other signals like electrooculogram (EOG) or information delivered by various assistive devices (e.g. buttons or joysticks) or sensors could be used in combination with an arbitrary brain signal to form an hBCI system.

TiA evolves an abstraction layer between data acquisition and data processing. Therefore, a standardized possibility to distinguish between different kinds of signals beyond this abstraction is an important issue. For that purpose so-called "Signal Types" were introduced, allocating every different kind of signal a unique identifier.

When analyzing different kinds of signals transmitted be-

This work is supported by the European ICT Programme Project FP7-224631. This paper only reflects the authors' views and funding agencies are not liable for any use that may be made of the information contained herein.

C. Breitwieser and G.R. Müller-Putz are with the Institute for Knowledge Discovery, BCI Lab, Graz University of Technology, Austria, 8010 Graz, Krenngasse 37, e-mail: c.breitwieser@tugraz.at, e-mail: gernot.mueller@tugraz.at

C. Neuper is with the Institute for Knowledge Discovery, BCI Lab, Graz University of Technology, Austria, 8010 Graz, Krenngasse 37, and the Department of Psychology, University of Graz, Austria, 8010 Graz, Universitätsplatz 2/III, e-mail: neuper@tugraz.at

tween data acquisition and data processing, various commonalities can be found. Signals are, or can be, divided into channels with related channel names and a defined scaling. Those channels can have a position or location and are acquired with a defined sampling rate. Single samples can be grouped together, forming blocks of samples.
Mason and Birch [2] showed a unidirectional transmission (see Fig. 1) from data acquisition (amplifier) to the first processing module (feature extraction). Using a client–server architecture is one possibility to address such a principle. In this case the data acquisition plays the server role and processing modules are the respective clients. Applying the client–server principle in this case easily facilitates the usage of multiple and potentially distributed processing chains (Mason and Birch merely show one processing chain in their models).

A. Design Principles

Information transmitted via TiA can be distinguished into two categories: (i) mutable and (ii) immutable information. Therefore, the data distribution is also split into two parts, initial meta information transmission to transmit immutable information and a continuous data stream to transmit mutable data to the client. Control messages using a defined handshaking protocol are used to transmit meta information. Mutable data (e.g., recorded voltage from an EEG channel) is delivered using a unidirectional binary data stream from the server to the client. Using the client–server principle similar or individual data streams can be established to multiple clients, only depending on the transmitted meta information. It is possible for an arbitrary number of clients to attach to the server at runtime. The client–server principle used for TiA is illustrated in more detail in Fig. 2.
A whitepaper concerning design and implementation of TiA (e.g., signal type flags) is available for download at arXiv.org [17].

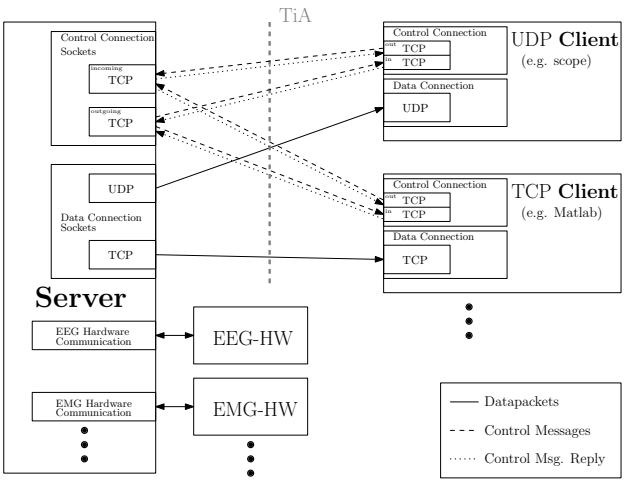

Fig. 2. TiA principle showing one server and two clients. An arbitrary number of clients can connect to the server choosing TCP or UDP for raw data transmission.

B. Software Design

A single data acquisition system, implementing the TiA interface, acquires data from different hardware devices. To set up a connection, HTTP-like (hyper text transfer protocol [18]) control messages are sent using two TCP (transmission control protocol [19]) connections. Messages sent over those connections are used for handshaking between client and server. Meta information can be optionally appended to this control messages. For mutable raw data transmission, a TCP or UDP (user datagram protocol [20]) connection can be chosen during the handshaking process.
The handshaking process is handled with two separated connections, whereby one connection is client–server oriented and the other one has a server–client orientation. The client–server connection is a mandatory requirement in TiA, supporting the server–client connection is optional. Incoming messages are always answered using the same connection on which the message was received.

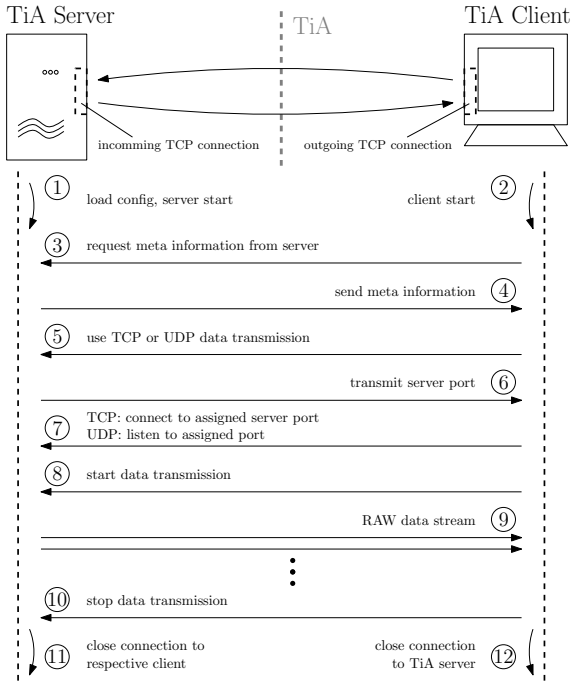

Fig. 3. TiA – client–server handshake. Steps 3–8 and 10 are done using the client–server control connection. During step 9 information is transmitted over the data connection; it represents the raw data stream from the server to the client.

1) Handshaking Process: Fig. 3 illustrates the steps between client–server communication. Information exchange is represented using arrows. Every message is transmitted in a standardized way using defined control messages and data packets. During steps 1 and 2 the client and server are started. In case of an error the startup is interrupted. In step 3 the client requests meta information from the server, the server responds with the meta information in step 4. For raw data transmission the client can choose whether to receive data via TCP or UDP. The desired raw data transmission protocol is sent to the server during step 5.

The server responds in step 6 with the respective port the client has to connect to (in case of TCP) or to listen to (in case of UDP). Subsequently in step 7 the client establishes a connection to (TCP) or starts listening (UDP) for data packets on the assigned port. To start data transmission, the client sends a message to the server during step 8. Starting in step 9, the server starts transmitting data packets to the client (the data packet is the same for TCP and UDP). In case of UDP, packets are broadcasted. The first connected client requesting UDP starts the broadcast and the last client disconnecting from the server stops the broadcast. If the client does not want to receive any more messages, a stop command is sent to the server during step 10. In case of TCP no more packets are delivered using the respective data connection. In case of UDP the broadcast is only stopped if the respective client was the last one requesting UDP. Otherwise UDP packets are further broadcasted. During step 11 the respective client is removed from the servers list of connected clients. Subsequently, in step 12, the client closes both the data and control connections to the server.
This handshaking procedure is mandatory to establish a connection. In case of an error during this handshaking procedure no connection is created.

2) Data Transmission: Mutable raw data is transmitted via TiA data packets using a binary data stream. Acquired data is encapsulated within those data packets. An exemplary data packet is visible in Fig. 4.

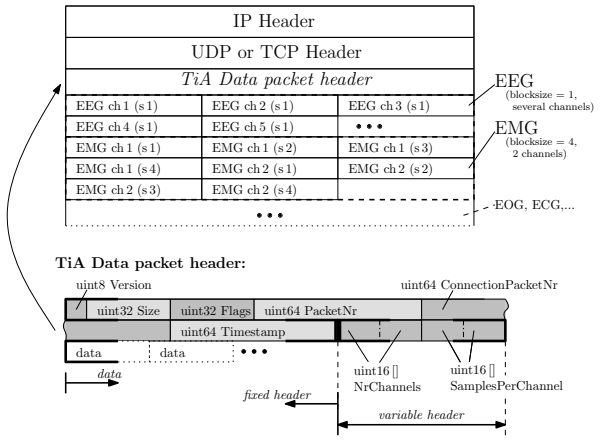

Fig. 4. Graphical representation of a TiA data packet. EEG and EMG content is shown as an example. EEG ch 1 (s 1) is an abbreviation for the signal type EEG, the first channel and the first sample. The block size for EEG is also one. For EMG an exemplary block size of four is used.

a) TiA Data Packet: The TiA data packet consists of three parts: (i) a fixed header; (ii) a variable header; and (iii) the raw data. By interpreting information stored in the fixed and variable header it is possible to correctly parse and read the whole data packet. The packet is equipped with a timestamp, a packet number, and a unique identifier per connection to facilitate proper timing and detect lost packets using UDP data transmission. Every signal type inside the data packet is identified with a unique flag. The number of channels and the block size can vary from packet to packet, but within TiA, a constant number of channels over time is assumed. Within the data packet raw data is stored as a 32 bit binary single precision floating point number (IEEE 754-2008 [21]). As a distinction between different signal types is possible within the data packet, data acquisition of multiple signal types at the same time and transmission within one data packet is possible. Different signal types can have different sampling rates and different block sizes, but within on signal type the sampling rate and the block size must be the same. Furthermore a single hardware device can acquire just one or also multiple signal types, as far as prior requirements are fulfilled. Detailed information concerning the data packet is available at arXiv.org [17].

b) Data Stream: A client has to perform the TiA handshaking process before receiving any data packets and has to choose either TCP or UDP for data transmission. Potential lost packets in case of UDP are not re-sent. If guaranteed data transmission is required, TCP has to be chosen. Using TCP for data transmission, a separate TCP connection from the client to the server is established and data packets are sent via this connection. The TiA data transmission is also restricted in some sense: for a single signal type only one block size and one sampling rate is allowed. But different signal types may have different sampling rates and block sizes.

III. Implementation and Testing

A. Implementation

A library and a first prototype called "signal server" using this TiA library, both written in C++, have been implemented and are available for download (http://bci.tugraz.at/downloads.html). The implementation is cross platform (Windows and Linux). During implementation a main focus was performance and stability. Up to now various hardware devices like generic joysticks and amplifiers from g.tec [Guger Technologies OG, Graz, Austria] and Brain Products [Brain Products GmbH, Gilching, Germany)] are supported by the signal server. To use those devices for BCIs, different clients using TiA have been written for Matlab [The MathWorks Inc., Natick, USA], Matlab Simulink, and BCI2000 [4]. Thus it is possible to stream data into these systems using TiA.

B. Testing

Testing measurements were accomplished using common personal computers (HP dc7700 workstation, Intel Core2Duo 6300@1.86 GHz, 4 GB Ram, Nvidia GeForce 9500 GT, Western Digital WD1002FAEX) using Windows Xp 32 bit and Debian unstable 64 bit.

1) Stability and Memory Consumption: The signal server was tested with a TiA client, both running on the same machine or on two different PCs connected via Ethernet. Long-term tests, lasting at least 10 hours, were performed under Linux (Debian unstable) and Windows XP to check for stability problems. In Linux additional memory leak tests

using valgrind [22] were conducted. The tested version of the signal server and the TiA client showed no increasing memory consumption in ten tests on both operating systems. The required memory was constant also after a continuous operation longer than ten hours. This was achieved in both operating systems. No memory leak was detected by valgrind under Linux when closing the server or the client. The memory consumption of the server was always below 1 MB, the client required less than 15 MB. The higher memory consumption of the client is caused by its receive buffer for incoming data packets. This value has been increased in comparison with the server to prevent the client from loosing data packets in case of reading delays.

2) Processing Time: A low processing time during data acquisition is essential. Acquired data has to be delivered as fast as possible to the clients. The processing time was measured from creation of a data packet (nearly the moment when data is read out from the respective data acquisition driver) until it's handover to the operating systems networking library functions. The timestamp stored inside the data packet was utilized to measure the delay. Packets were created with 10 kHz and 128 channels to simulate a high workload and sent using TCP over the loopback device (a virtual local network interface). Statistical values were computed over five minutes (resulting in $3 \cdot 10^6$ packets), the computer was idle except the signal server and client processes. According to the results, shown in Tab. I, the maximum packet rate for the signal server would be roughly 40 kHz (with a mean processing time of 25 µs per data packet). Using a higher sampling rate, new data would be available before older one was completely processed.

TABLE I

PROCESSING TIME OF A SINGLE TIA DATA PACKET.

	mean	std	median	min	max
Debian	19 µs	±3 µs	16 µs	7 µs	2817 µs
Windows	23 µs	±18 µs	18 µs	9 µs	3741 µs

IV. DISCUSSION

We have shown that it is possible to introduce an attempt of a standardized layer between the data acquisition module (Amplifier) and the first data processing module (Feature Extractor) into the functional model described by Mason and Birch [2]. A library and a data acquisition software have been written for this purpose and have been successfully integrated into different programs used for BCI purposes nowadays (BCI2000, Matlab). Performance and stability of the current implementation has been analyzed. Furthermore, using TiA, it is a simple process to add additional signal types.
By using TiA it becomes possible to decouple a BCI system from the used data acquisition hardware and make one step towards Masons standardized BCI model.

ACKNOWLEDGMENT

Special thanks to Christoph Eibel and Andreas Schuller from Graz University of Technology and Francesco Leotta from Fondazione Santa Lucia for their contributions.

REFERENCES

[1] J. J. Vidal, "Toward direct brain-computer communication." *Annu Rev Biophys Bioeng*, vol. 2, pp. 157–180, 1973.
[2] S. G. Mason and G. E. Birch, "A general framework for brain-computer interface design." *IEEE Trans Neural Syst Rehabil Eng*, vol. 11, no. 1, pp. 70–85, Mar 2003.
[3] Y. Renard, F. Lotte, G. Gibert, M. Congedo, E. Maby, V. Delannoy, O. Bertrand, and A. Lécuyer, "Openvibe: An open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 19, no. 1, pp. 35–53, 2010.
[4] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "Bci2000: a general-purpose brain-computer interface (bci) system." *IEEE Trans Biomed Eng*, vol. 51, no. 6, pp. 1034–1043, Jun 2004.
[5] R. Scherer, A. Schlögl, G. Müller-Putz, and G. Pfurtscheller, "Inside the graz-bci: rtsbci," in *Proceedings of the 2nd International Brain-Computer Interface Workshop and Training Course 2004*, 2004.
[6] I. P. Susila, S. Kanoh, K.-i. Miyamoto, and T. Yoshinobu, "xBCI: A generic platform for development of an online bci system," *IEEJ Trans Elec Electron Eng*, vol. 5, no. 4, pp. 467–473, 2010.
[7] J. D. R. Millán, R. Rupp, G. R. Müller-Putz, R. Murray-Smith, C. Giugliemma, M. Tangermann, C. Vidaurre, F. Cincotti, A. Kübler, R. Leeb, C. Neuper, K.-R. Müller, and D. Mattia, "Combining brain-computer interfaces and assistive technologies: State-of-the-art and challenges." *Front Neurosci*, vol. 4, 2010.
[8] G. Pfurtscheller, B. Z. Allison, C. Brunner, G. Bauernfeind, T. Solis-Escalante, R. Scherer, T. O. Zander, G. Müller-Putz, C. Neuper, and N. Birbaumer, "The hybrid bci." *Front Neurosci*, vol. 4, p. 42, 2010.
[9] G. Pfurtscheller, T. Solis-Escalante, R. Ortner, P. Linortner, and G. R. Müller-Putz, "Self-paced operation of an ssvep-based orthosis with and without an imagery-based "brain switch:" a feasibility study towards a hybrid bci." *IEEE Trans Neural Syst Rehabil Eng*, vol. 18, no. 4, pp. 409–414, Aug 2010.
[10] R. Leeb, H. Sagha, R. Chavarriaga, and J. D. R. Millan, "Multimodal fusion of muscle and brain signals for a hybrid-bci." *Conf Proc IEEE Eng Med Biol Soc*, vol. 1, pp. 4343–4346, 2010.
[11] C. Brunner, B. Z. Allison, D. J. Krusienski, V. Kaiser, G. R. Müller-Putz, G. Pfurtscheller, and C. Neuper, "Improved signal processing approaches in an offline simulation of a hybrid brain-computer interface." *J Neurosci Methods*, vol. 188, no. 1, pp. 165–173, Apr 2010.
[12] G. Müller-Putz, B. C., F. Cincotti, R. Leeb, M. Schreuder, F. Leotta, M. Tavella, L. Bianchi, A. Kreilinger, A. Ramsay, M. Rohm, M. Sagebaum, L. Tonin, C. Neuper, and J. Millán, "Tools for Brain-Computer Interaction: A new concept for a hybrid BCI," *IEEE Trans Biomed Eng*, submitted 2011.
[13] J. Mellinger, G. Schalk, C. Braun, H. Preissl, W. Rosenstiel, N. Birbaumer, and A. Kübler", "An meg-based brain-computer interface (bci)," *NeuroImage*, vol. 36, no. 3, pp. 581 – 593, 2007.
[14] S. M. Coyle, T. E. Ward, and C. M. Markham, "Brain–computer interface using a simplified functional near-infrared spectroscopy system," *Journal of Neural Engineering*, vol. 4, no. 3, p. 219, 2007.
[15] J.-H. Lee, J. Ryu, F. A. Jolesz, Z.-H. Cho, and S.-S. Yoo, "Brain-machine interface via real-time fmri: Preliminary study on thought-controlled robotic arm," *Neuroscience Letters*, vol. 450, no. 1, pp. 1 – 6, 2009.
[16] R. Scherer, G. R. Mller-Putz, and G. Pfurtscheller, "Self-initiation of EEG-based brain-computer communication using the heart rate response," *Journal of Neural Engineering*, vol. 4, pp. L23–L29, 2007.
[17] C. Breitwieser and C. Eibel, "TiA – Documentation of TOBI Interface A," *ArXiv e-prints*, Mar. 2011.
[18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – http/1.1," United States, 1999.
[19] V. Cerf and R. Kahn, "A protocol for packet network intercommunication," vol. 22, no. 5, pp. 637–648, 1974.
[20] J. Postel, "User datagram protocol," Internet Engineering Task Force, RFC 768, August 1980.
[21] "Ieee standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1 –58, 29 2008.
[22] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," *SIGPLAN Not.*, vol. 42, pp. 89–100, June 2007.