BCI2000 External Application Interface

Jürgen Mellinger, Gerwin Schalk

February 3, 2006

Contents

1 Introduction

The BCI2000 external application interface provides a bi-directional link to exchange information with external processes running on the same machine, or on a different machine over a local network.

Via the external application interface, read/write access to BCI2000 state vector information and control signals is possible. An external application may read the `ResultCode` state to access the classification result, set the `TargetCode` state to control the user's task, or get access to the control signal that is calculated by SignalProcessing so as to control an external output device (such as a robotic arm or a web browser). Multiple instances of BCI2000 running on separate machines may share sequencing and control signal information, allowing for interactive applications such as games.

The scope of this interface is to provide access to internal BCI2000 information in cases in which the generation of a full-fledged BCI2000 module is impractical. Such a case might be the control of external applications that practically do not allow full incorporation into the BCI2000 framework (such as the Dasher system for efficient low-bandwidth spelling). This interface is *NOT* intended to replace the existing BCI2000 framework for BCI2000 communication. The advantage of writing modules that are fully integrated into the BCI2000 framework are that their configuration is achieved through the same interface as other BCI2000 configuration, that this configuration is stored in the data file along with all other system parameters, and that the state of the module at any given time is encoded in event markers that are also stored in the data file. In contrast, control of an external device using the External Application Interface implies that the configuration of the external device has to be done outside of BCI2000, that this corresponding configuration is not stored along with the data file, and that the internal state of the output device is not saved together with the brain signals. This will make it more difficult to reconstruct what exactly was going on during an experimental session. It is thus important to keep this in mind when using this exciting new possibility.

The external application interface's design aims at simplicity, and at minimal interference with the timing of the signal flow through the BCI2000 system. With this in mind, a connection-less, UDP based transmission protocol was chosen rather than one based on TCP. This comes at the cost of a possible loss, or reordering of protocol messages. To keep the probability for such losses as low as possible, and their consequences as local as possible,

messages have been designed to be short, self-contained, and redundantly encoded in a human readable fashion.

For communication involving a large number of network nodes, or unreliable connections, we suggest using locally executed server processes that forward messages to a TCP connection.

2 Description

For each block of data processed by the BCI2000 system, two types of information are sent out and may be received from the external application interface:

- the BCI2000 internal state as defined by the values of the entries of its state vector data structure, and
- the BCI2000 control signals.

For the definition of and details about states and the control signal, please refer to the BCI2000 project outline document.

Sending data is done immediately after processing by the application module's task filter has taken place; receiving occurs immediately before the task filter. This makes sure that changes resulting from user choices are sent out immediately, and that received information will immediately be available to the task filter.

IP addresses and ports used are user-configurable. Sending and receiving need not use the same address and port.

3 Protocol

Messages consist of a name and a value, separated by white space and terminated with a single newline (`'\n'==0x0a`) character.

Names may identify

- BCI2000 states by name – then followed by an integer value in decimal ASCII representation;
- Signal elements in the form `Signal(<channel>,<element>)` – then followed by a float value in decimal ASCII representation.

Examples:

```
Running 0\n
ResultCode 2\n
Signal(1,2) 1e-8\n
```

Note that the first example will switch BCI2000 into a suspended state. While the system is in that state, no communication is possible over the application protocol.

4 Parameterization from within BCI2000

BCI2000 reads data from a local IP socket specified by the `ConnectorInputAddress` parameter, and writes data out into the socket specified by the `ConnectorOutputAddress` parameter. Sockets are specified by an address/port combination. Addresses may be host names, or numerical IP addresses. Address and port are separated by a colon as in

```
localhost:5000
134.2.103.151:20321
```

For incoming values, messages are filtered by name using a list of allowed names present in the `ConnectorInputFilter` parameter. To allow signal messages, allowed signal elements must be specified including their indices. To allow all names, enter an asterisk (*) as the only list entry.