TOBI - Hybrid BCI – Data Transmission
Design Document

**(for TOBI internal use only,
must not be published or distributed)**

TUG, Graz, April 13, 2010

1 Reqirements

- **Combined signal data server (EEG, EMG, other signal types)**

- Data transmission over network (or localhost) using NW-sockets
- Software should be platform independent (portable framework and libraries)
- Multiple clients possible
- Possibility against package leakage
- Hardware requirements low
- Configuration communication over Network
- Variable number of EEG/EMG/. . . channels
- Variable sampling frequency
- Variety of data acquisition hardware possible
  (EEG: at the beginning only g-tec USBAmp, EMG amplifier, joysticks,. . .)
- Delivery of data data "in time"

- Possibility to store data with higher sampling rate (internal downsampling before transmission)
- Connection oriented transfer (e.g. using TCP) with modified sampling rate

2 Client - Server Architecture

- One data server
- "Many" clients
- Clients can be attached to the server at any time

- Communication seperated into “configuration communication” and data transmission
- Server provides one TCP socket for “configuration communication”
- “Configuration communication” done in xml-style
- Server provides one UDP socket for data broadcast
  (packet loss acceptable)
- Server provides one TCP socket for connection oriented data transmission
  (packet loss **un**acceptable)

- Clients create connections to respective server socket
- Data transmission to multiple clients done by broad- or multicast

3 Startup and Connection-Setup – an idea

Before any network activity:

- Server starts with HW configuration and UDP/TCP ports specified in .xml file
- Client starts with configuration in .xml file
  (server-IP and ports are familiar to the client)

All network-messages are in xml-style, except data
UDP packets will be broadcasted into the whole subnet.

Connection scheme shown in Figure 1.
Client-Server handshake shown in Figure 2.

- Client connects to server and requests configuration
  (sampling rate, nr. of channels, stored signal data type (int, float,. . .))
- Server sends configuration
- Client requests start of UDP or TCP data transmission
  (data loss unacceptable → data over TCP connection
  if TCP connection: connect to server TCP socket)
- Server starts data transmission

- Optional: Other Clients connect to the server

- Client requests to stop data stream

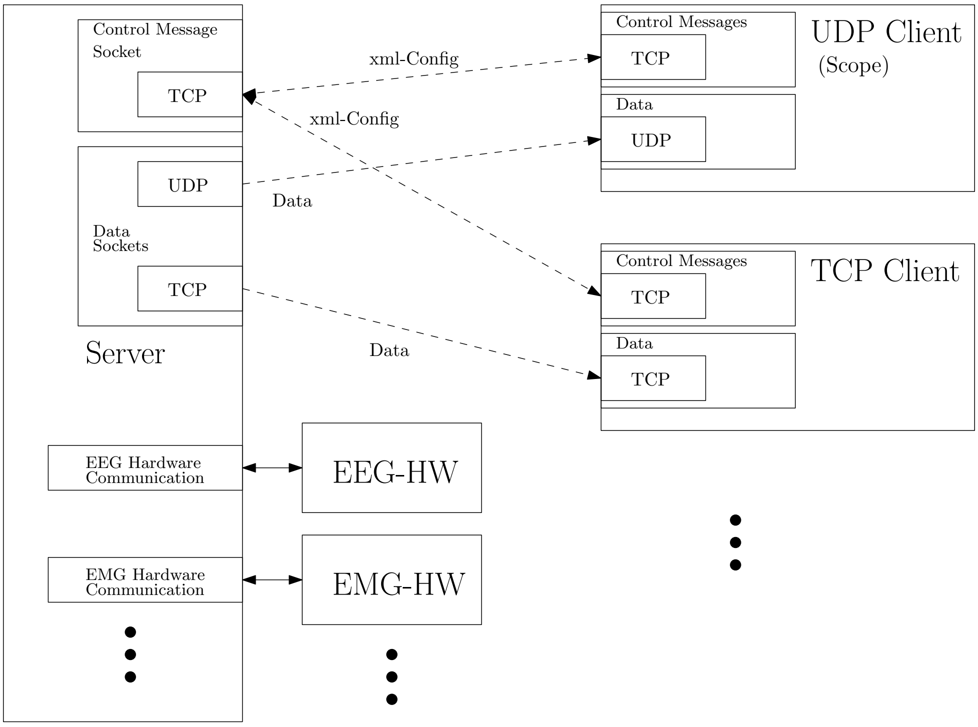

Figure 1: Connection Scheme Client-Server

- UDP: if other clients available: keep transmitting data
  TCP: stop transmitting data over TCP connection

- Client closes connection to server (client specific shutdown)

- Last clients requests to stop data stream

- Server closes connection to last client and stops transmitting data

- Client specific shutdown

4 The Connection – TCP, UDP, creation, termination, . . .

- Protocol application dependent
  (UDP for scope, TCP for critical apps . . . data loss unacceptable)

- Data stream can't stop while clients are attached

- Configuration done by xml files

- Communication between client and server done in xml-style

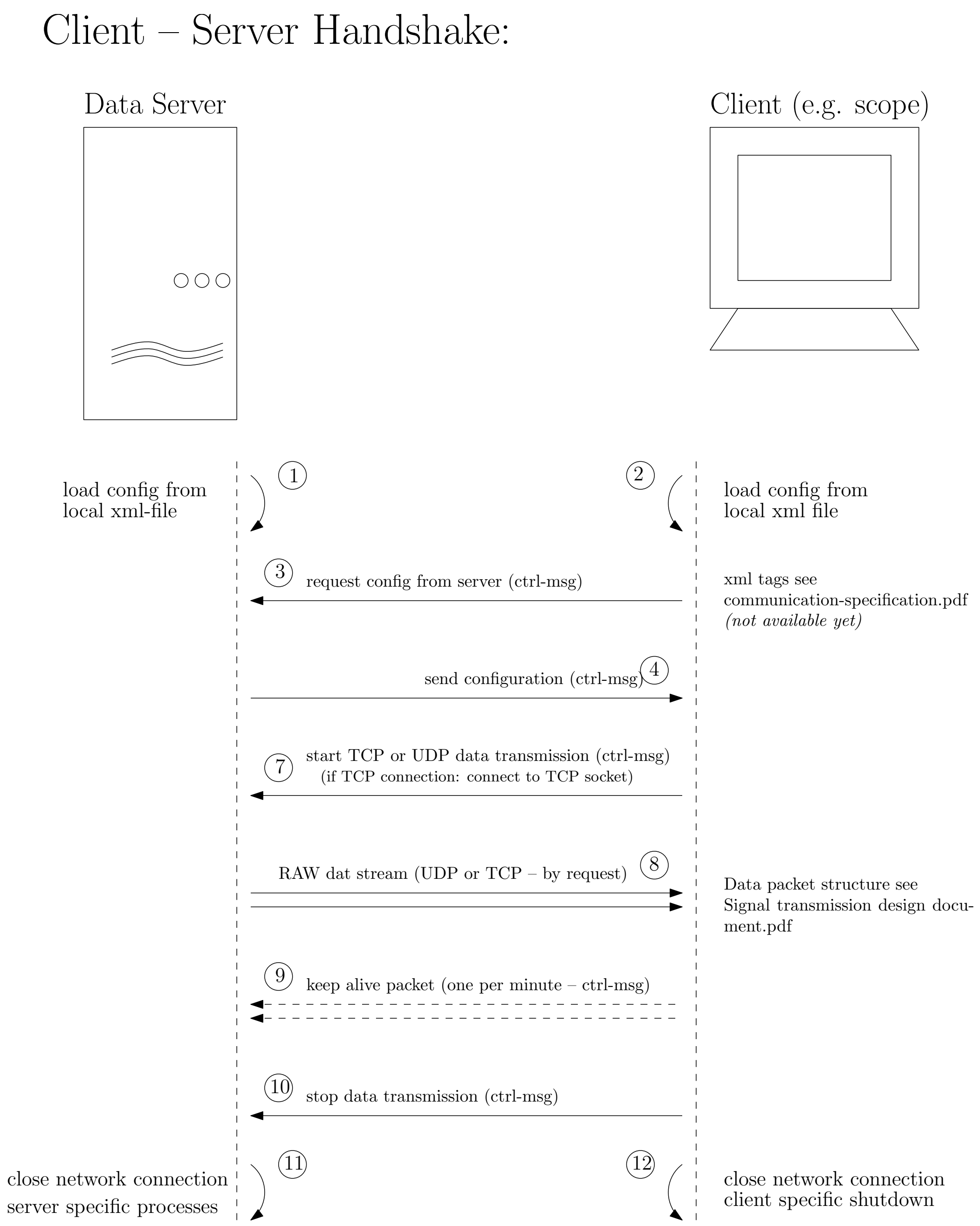
Client – Server Handshake:
Data Server
Client (e.g. scope)
1
load config from local xml-file
2
load config from local xml file
3
request config from server (ctrl-msg)
xml tags see communication-specification.pdf *(not available yet)*
send configuration (ctrl-msg)
4
7
start TCP or UDP data transmission (ctrl-msg)
(if TCP connection: connect to TCP socket)
RAW dat stream (UDP or TCP – by request)
8
Data packet structure see Signal transmission design document.pdf
9
keep alive packet (one per minute – ctrl-msg)
10
stop data transmission (ctrl-msg)
11
close network connection
server specific processes
12
close network connection
client specific shutdown

Figure 2: Client-Server Handshake

5 The Packet – what's inside

Header:

- Flags
- Running Packet Number → loss of data can be recognized
- Running Sample Number
  → if transmitting with different sampling rates, related
  samples can be clearly identified
- Number of different signal types → loss of data can be recognized
- Offset of signal types in the packet
- Number of channels per signal type

Data:

- EEG (RAW values – type EEG amplifier specific)
- EMG (RAW values – type EEG amplifier specific)
- Other Signal types (ECG, manual input e.g. button)

- if needed also Events
- Additional Information ?? (again Channels, Frequency,. . .)

Data will be sent either as floats (4byte) or doubles (8 byte). The biggest type, dependent on the data acquisition system will be chosen for all transmitted data.
For scaling purposes, all equal signals have to scaled to the same range (e.g. µV, mV,. . .).

Defined Flags and data order:
→ Signals in predefined order:
EEG | EMG | EOG | ECG | HR | BP | Buttons | Joystick | Sensors | . . .

32 bit flags:

01 . . . EEG (0x0001)

02 . . . EMG (0x0002)

03 . . . EOG (0x0004)

04 . . . ECG (0x0008)

05 . . . HR (0x0010)

06 . . . BP (0x0020)

07 . . . Buttons (0x0040)

08 . . . Joystick (0x0080)

09 . . . Sensors (0x0100)

10 . . . NIRS (0x0200)

11 . . . FMRI (0x0400)

12

13

14

15

16

17 . . . User Defined I (0x010000)

18 . . . User Defined II (0x020000)

19 . . . User Defined III (0x040000)

20 . . . User Defined IV (0x080000)

21 . . . UNDEFINED (0x100000)

22 . . . Events (0x200000)

23

24

25

26

27 **RESERVED**: 6 bits for Packet Version

28 **RESERVED**: 6 bits for Packet Version

29 **RESERVED**: 6 bits for Packet Version

30 **RESERVED**: 6 bits for Packet Version

31 **RESERVED**: 6 bits for Packet Version

32 **RESERVED**: 6 bits for Packet Version

Data packet structure shown in Figure 3.

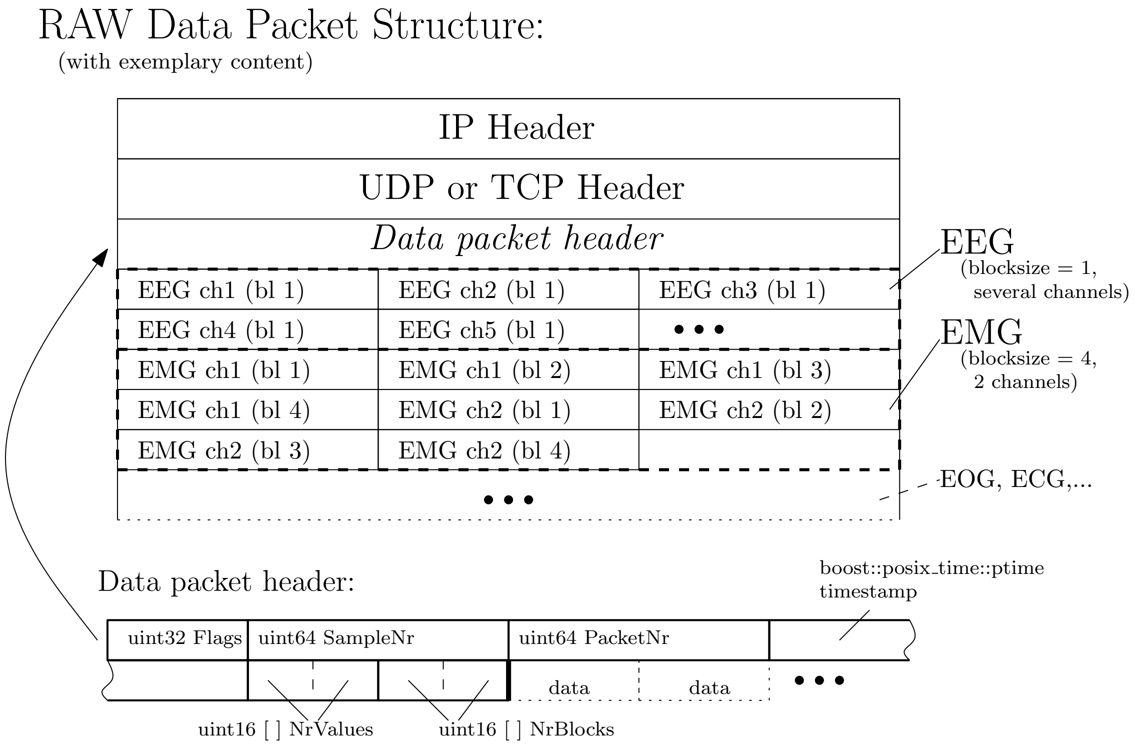

Figure 3: data packet structure

5.1 Estimation of needed bandwidth:

High values, overhead not included:

Sampling Rate f_s = 512 Hz
Nr. of channels: n = 128
Needed memory per channel and packet: s = 8 byte (double)

$$B = f_s \cdot (n+1) \cdot s \quad = \quad 516\,\frac{kbyte}{s}$$

Expected values, overhead not included:

Sampling Rate f_s = 256 Hz
Nr. of channels: n = 16 (USBAmp)
Needed memory per channel and packet: s = 8 byte (double)

$$B = f_s \cdot (n+1) \cdot s \quad = \quad 34\,\frac{kbyte}{s}$$

Within this calculations the packet overhead is not taken into account!

6 Outcome – WP5 and WP8 Meeting Berlin 7.7.09

- Broadcast over UDP
- No realtime network protocol
  (until a rationale of need)
- Header to describe package content
- Combined server
- Offline simulation possible – by loading an existing data file
- Downsampling for individual TCP connections done by the server
- Events transmitted by the data server
- Signal and event storage done by the server (use of .gdf format if possible)
- Different sapmling rates provided by the server
- Differing package content described by data packet header (flags, offsets,. . .)

- Storage with higher sampling rate than transmission possible

7 Discussion Points

Discussed:

- Additional requirements? (use of shared memory for communication)
- More than one client over UDP . . . broadcast or multicast?? (preferably broadcast into subnet)
- Realtime protocol required?
  (250 Hz . . . $\frac{1\,packet}{4\,ms}$, mean RTT in small subnet ca. 0.5 ms,
  maybe use of own 192.168.xxx.xxx subnet on a 2^{nd} network interface card)
- Additional information in data packet
- Offline simulation needed (file with events and different signals sent by the server)
- Support of different sampling rates by the data server needed?
  (Problem: Increases network load, higher complexity, data packet structure)
- Event transmission/creating – where/how?
- Data storage? – where, how (EEG synchronisation with events)
- Different sapmling rates

Still open:

- Definition of xml tags (partly done – 13.4.2010)
- Preprocessing: done by the server or elsewhere? (additional to downsampling?)
- How to handle broken TCP connections? (client hang-up,. . . → keep-alive packet?)