Sense World Data Network
http://sensestage.hexagram.ca

Marije Baalman

July 13, 2011

Abstract

The data network framework is meant to make sharing of data (from sensors or internal processes) between collaborators in an interactive media art work easier, faster and more flexible. There is a central host, which receives all data, and manages the client connections. Each client can subscribe to data *nodes*, to use that data in its own internal processes; and each client can publish data onto the network, by creating a node. A new client can query the network which nodes are present and is informed when new nodes appear after the client has been registered.

1 Data Network Elements

The data network is built up from different elements:

DataNetwork the network itself

DataNode a node is a collection of slots, usually based upon a device or another common source (e.g. result from a function).

DataSlot a slot is a single data stream

Data on the network is set by calling the function method `setData` with as arguments the node ID and an array of data values (either numbers (floats) or strings). The ID is an unique identifier (an integer). The function can be called for example by a class instance that parses serial data.

Each **DataNode** and each **DataSlot** can be given a label, so that their functionality becomes more human understandable.

2 OSC interface

There is an OSC interface to the network, which allows clients to become part of the data network and access its data, and also create its own data nodes on the network.

The network will announce itself to the broadcast address of the network, to a number of ports (default: range 6000-6009, and 57120-57129), so that clients can automatically configure to connect to the network, as soon as it is in the air.

A textfile with the network's OSC port can be found in the file `http://hostip/SenseWorldDataNetwork`[1], which can be retrieved by clients, so they know where to send the registration message.

The general setup is that an OSC client first sends a register message to the data network server. Then it will start receiving ping messages, to which it has to reply with pong messages. The client has to query which nodes and slots are present on the network after registering, so it will receive info messages on each node and slot. Then it can subscribe to nodes and slots, and will receive data from the nodes and slots it is subscribed to via the data messages.

The client can supply a new node to the network, by using the /set/data message; it can also label the nodes and slots thus created. Whenever a new node or slot is added (or changed, e.g. when it gets a label), the client will receive a new info message. If there occurs an error in the communication, then an error message is sent. The unregister message only needs to be sent, if for example the client crashed and is trying to reconnect on the same port.

All messages to the server now have a reply, which is either the requested info, a confirmation message, or a warning or error.

See table 1 for an overview of commands.

3   Interaction with the MiniBees

You can send queries to the network to get information about the minibees that are present. Info messages of minibees that appear after a client has joined the network, will automatically be sent.

You can map datanodes to send data to a MiniBee (see table 2):

output The DataNode that is used to map from, has to have an equal amount of slots as the MiniBee has outputs. The PWM outputs come first, then the digital outputs.

custom The DataNode that is used to map from, has to have as many slots as the custom message received in the MiniBee has data bytes.

A client can also provide a hive of MiniBees, by using the special hive register message. It has to request a certain number of nodes that it will create (the amount of MiniBees), and the DataNetwork host will reply with the minimum and maximum id that are reserved for the hive's nodes. Mapping messages for the minibees of the hive are then forwarded by the DataNetwork to the hive client.

---

[1]e.g.   for a host with IP 192.168.1.7 the url is: `http://192.168.1.7/SenseWorldDataNetwork`

Address	Types	Arguments	Description
/datanetwork/announce	si	host, port no.	announce the network with its coordinates
/datanetwork/quit	si	host, port no.	inform that the host has quit
/register	is	port no., name	register to the network as a client, the name is used as an identifier
/registered	i	port no.	reply to register to the network as a client
/unregister	is	port no., name	unregister to the network as a client
/unregistered	is	port no., name	reply to unregister to the network as a client
/ping	is	port no., name	message to check if client is still there
/pong	is	port no., name	expected reply to the /ping message
/error	ssi	cause, error message, error ID	error occurred upon request (indicated by cause)
/warn	ssi	cause, warn message, error ID	non fatal error occurred upon request
/query/all	is	port no., name	do all queries
/query/expected	is	port no., name	query which nodes are expected in the network (reply /info/expected)
/query/nodes	is	port no., name	query which nodes are in the network (reply /info/node)
/query/slots	is	port no., name	query which slots are in the network (reply /info/slot)
/query/clients	is	port no., name	query which clients are in the network (reply /info/client)
/query/setters	is	port no., name	query which nodes the client is the setter of (reply /info/setter)
/query/subscriptions	is	port no., name	query which subscriptions the client has (reply /subscribed/node, /s...
/info/expected	i(s)	node ID, node label	info about an expected node
/info/node	isii	node ID, node label, number of slots, node type	info about a node
/info/slot	iisi	node ID, slot ID, slot label, slot type	info about a slot
/info/client	sis	ip, port no., name	info about a client
/info/setter	isii	node ID, node label, number of slots, node type	info about a node the client is setting
/subscribe/all	is	port no., name	subscribe to receive data from all nodes
/unsubscribe/all	is	port no., name	unsubscribe from all nodes
/subscribe/node	isi	port no., name, node ID	subscribe to receive data from a node
/subscribed/node	isi	port no., name, node ID	reply to subscribe to receive data from a node
/unsubscribe/node	isi	port no., name, node ID	unsubscribe to receive data from a node
/unsubscribed/node	isi	port no., name, node ID	reply to unsubscribe to receive data from a node
/subscribe/slot	isii	port no., name, node ID, slot ID	subscribe to receive data from a slot
/subscribed/slot	isii	port no., name, node ID, slot ID	reply to subscribe to receive data from a slot
/unsubscribe/slot	isii	port no., name, node ID, slot ID	subscribe to receive data from a slot
/unsubscribed/slot	isii	port no., name, node ID, slot ID	reply to unsubscribe to receive data from a slot
/data/node	iff..f	node ID, data values	node data
/data/node	iss..s	node ID, string data values	node data
/get/node	isi	port no., name, node ID	get data from a node (reply /data/node)
/data/slot	iif	node ID, slot ID, data value	slot data
/data/slot	iis	node ID, slot ID, string data value	slot data
/get/slot	isii	port no., name, node ID, slot ID	get data from a slot (reply /data/slot)
/set/data	isif..f	port no., name, node ID, data values	set data to a node (reply /data/node)
/set/data	isis..s	port no., name, node ID, string data values	set data to a node
/label/node	isis	port no., name, node ID, node label	set label to a node
/label/slot	isiis	port no., name, node ID, slot ID, slot label	set label to a slot
/remove/node	isi	port no., name, node ID	remove a node (only possible if client is setter)
/removed/node	i	node ID	reply to remove a node
/remove/all	is	port no., name	remove all nodes the client is a setter of (generates /removed/node ...
/add/expected	isi(isi)	port no., name, node ID, node size, node label, node type	add an expected node to the network (reply /info/expected)
			if node size is given, the node is created as well (and generates a /in...
			node type is 0: float, 1: string (default is 0)

Table 1: OSC namespace for the Data Network

3

Message	Type	Arguments	Description
/unregister/hive	is	port no., name	unregister ...
/unregistered/hive	is	port no., name	reply to un...
/query/minibees	is	port no., name	query which ...
/info/minibee	isiiis	node ID, number of slots (inputs), number of outputs, config id, serial number	info about ...
/info/minibee	isisiiis	port no., name, node ID, number of slots (inputs), number of outputs, config id, serial number	info about ...
/remove/minibee	isi	port no., name, node ID	remove a n...
/removed/minibee	i	node ID	reply to rem...
/query/configurations	is	port no., name	query whic...
/query/configurations			query whic...
/info/configuration	is	port no., name	query whic...
/info/configuration			query whic...
/map/minibee/output	isii	port no., name, node ID, minibee ID	map node o...
/map/minibee/custom	isii	port no., name, node ID, minibee ID	map node o...
/map/minibee/output	ii	node ID, minibee ID	map node o...
/map/minibee/custom	ii	node ID, minibee ID	map node o...
/mapped/minibee/output	ii	node ID, minibee ID	reply to /m...
/mapped/minibee/custom	ii	node ID, minibee ID	reply to /m...
/mapped/minibee/output	isii	port no., name, node ID, minibee ID	reply to /m...
/mapped/minibee/custom	isii	port no., name, node ID, minibee ID	reply to /m...
/unmap/minibee/output	isii	port no., name, node ID, minibee ID	unmap nod...
/unmap/minibee/custom	isii	port no., name, node ID, minibee ID	unmap nod...
/unmap/minibee/output	ii	node ID, minibee ID	unmap nod...
/unmap/minibee/custom	ii	node ID, minibee ID	unmap nod...
/unmapped/minibee/output	ii	node ID, minibee ID	reply to /u...
/unmapped/minibee/custom	ii	node ID, minibee ID	reply to /u...
/unmapped/minibee/output	isii	port no., name, node ID, minibee ID	reply to /u...
/unmapped/minibee/custom	isii	port no., name, node ID, minibee ID	reply to /u...
/configure/minibee	isii(s)	port no., name, minibee ID, config ID (, serial number)	configure m...
/configured/minibee	ii(s)	minibee ID, config ID (, serial number)	reply to /co...
/configure/minibee	ii(s)	minibee ID, config ID (, serial number)	configure m...
/configured/minibee	is ii(s)	port no., name, minibee ID, config ID (, serial number)	reply to /co...
/configured/minibee/error	is ii(s)	port no., name, minibee ID, config ID (, serial number)	reply to /co...
/minihive/configuration/create	is isii ii..i	port no., name, config id, configname, samples per message / message interval, number of pins config (N), number of twi devices (M) / N times pin config, M times twi config	create a co...
or	is isii ss..s ss..s		client -¿ ho...
/minihive/configuration/created	isii ii.i ii..i	config id, configname, samples per message / message interval, number of pins (N), number of twi devices (M) / N times pin config, M times twi config	reply to /m...
or	isii ss..s ss..s		
/minihive/configuration/create	isii ii.i ii..i	config id, configname, samples per message / message interval, number of pins (N), number of twi devices (M) / N times pin config, M times twi config	create a co...
or	isii ss..s ss..s		host -¿ clie...
/minihive/configuration/created	is isii ii.i ii..i	port no., name, config id, configname, samples per message / message interval, number of pins (N), number of twi devices (M) / N times pin config, M times twi config	create a co...
or	is isiii ss..s ss..s		- host -¿ hi... / reply to /m...
/minihive/configuration/delete	is i	port no., name, config id	- hiveclient / delete a co...
/minihive/configuration/deleted	i	config id	reply to /m...
/minihive/configuration/delete	i	config id	delete a co...
/minihive/configuration/deleted	is i	port no., name, config id	reply to /m...
/minihive/configuration/save	is s	port no., name, filename	save config...
/minihive/configuration/saved	s	filename	reply to /m...
/minihive/configuration/save	s	filename	save config...
/minihive/configuration/saved	is s	port no., name, filename	reply to /m...
/minihive/configuration/load	is s	port no., name, filename	load config...
/minihive/configuration/loaded	s	filename	reply to /m...
/minihive/configuration/load	s	filename	load config...

1	"Client with IP"+addr.ip+"and port"+addr.port+"is not registered. Please register first"
2	"Client with IP"+addr.ip+"and port"+addr.port+"is already registered. Please unregister first"
3	"Client with IP"+addr.ip+"and port"+addr.port+"was not registered"
4	"Client with IP"+addr.ip+"and port"+addr.port+"is not the setter of node with id"+..
5	"Node with id"+..+"is not part of the network"
6	"Node with id"+..+"is not expected to be part of the network"
7	"There are no expected nodes in the network"
8	"There are no nodes in the network"
9	"There are no clients in the network"
10	"Client with IP"+addr.ip+"and port"+addr.port+"has no setters"
11	"Client with IP"+addr.ip+"and port"+addr.port+"has no subscriptions"
12	"Node with id"+...+"does not have"+..+"slots"
12	"Node with id"+...+"does not have"+..+"slots"
13	"Node with id"+...+"has wrong type"+...
14	"Client with IP"+addr.ip+"and port"+addr.port+"was not registered under name"+name
15	"Client with IP"+addr.ip+"and port"+addr.port+"and name" + name + "is not registered. Please register firs

Table 3: Error codes and strings

4 Max implementation (by Harry Smoak, Joseph Malloch and Brett Bergmann)

In the Max implementation, there is a data *sink*, which manages the connection to the network (registering, subscriptions, etc.), and gives the received data. There is a data *source*, which can send data into the network. The subscriptions are handled by textfiles, as are the published data nodes, so they can be easily restored upon opening a max patch. The objects react to the announce message from the network to set the right host IP and port.

The Max patch dn.node can receive data from any number of nodes, as it takes multiple arguments (i.e. the object "dn.node 10 12 15" would subscribe to data from nodes 10, 12 and 15)

5 Processing implementation (by Vincent de Belleval and Brett Bergmann)

The Processing client implementation is done as a Processing library, using JavaOSC for OSC communication. It comes with two example files and an HTML reference documentation.

6 C++ implementation (by Marije Baalman)

The C++ client implementation comes as a library, with a doxygen file to generate documentation, and an example client.

7 SuperCollider implementation

The SuperCollider implementation is done in a set of classes.
Documentation for these is available in HTML format.

8 Installation

8.1 SuperCollider Quark

The DataNetwork can be most easily installed from SuperCollider's Quarks
extension management system. This also includes the client patches for other
software environments.
To install the classes, do the following inside SuperCollider:

```
Within SuperCollider do the following:
// check out all quarks:
Quarks.checkoutAll;

// or update them all to the latest version:
Quarks.update;

// install the SenseWorld DataNetwork quark
// - this will install all other quarks that are needed
Quarks.install("SenseWorld DataNetwork");

// recompile the library

//---------------- host ----------------------
// On OSX, add this to the startup file (or execute it each time)
SWDataNetworkOSC.httppath = "/Library/WebServer/Documents/";

// create a network:
x = SWDataNetwork.new;
// add the OSC interface
x.addOSCInterface;

//---------------- client ----------------------

// create a network client:
y = SWDataNetworkClient.new( ~hostip, "myname" );
// where ~hostip is an IP address of the datanetwork host like: "192.168.0.104",
// and "myname" is the name by which you (as a client) will be identified in the network:
// so it becomes:
// y = SWDataNetworkClient.new( "192.168.0.104", "myname" );

// to show a GUI:
```

```
y.makeGui;

// For more help, access the helpfile:
SenseWorldDataNetwork
```

8.2 Apache

You need to install a webserver such as Apache on the host system.

(package apache2 on Debian/Ubuntu; usually available on OSX)

The general files will be put in `/var/www`. You have to make this directory writable by the user by executing (as root)

```
cd /var/www
chmod 775 .
chgrp netdev .
```

Assuming that the user running SuperCollider is member of the group `netdev`. You can check this by:

```
groups
```

To add yourself to the group, execute as root (with instead of "nescivi" your username):

```
adduser nescivi netdev
```

You may need to logout and log back in for this to take effect.

On OSX the default path for http files is: `/Library/WebServer/Documents/`. You may have to enable "Web Sharing" in the "Sharing" settings in the System Preferences.

9   Detailed message description

- `/datanetwork/announce` - si - host, port no.

  sent by host When the host becomes active, or manual. It is sent both as a broadcast message to the network, and to any clients that may be known already by the host by their IPs.

  response clients should note the possible change in port and reregister if not registered

- `/datanetwork/quit` - si - host, port no.

  sent by host When the host shuts down (in a proper way, not when it crashes).

  response clients should check if this is from their host, and note that they are no longer registered with the datanetwork

- **/register** - is - port no., name

    sent by client In order to register

    response If it is a new client, the host will register the client and reply with the **/registered** message. If it is a client at the same IP, port and name that was already registered, the host will reply with **/registered**. If a client with that IP and port, but with a different name was already registered, the host will return an error message 2.

- **/registered** - is - port no., name

    sent by host Upon a succesful registration of the client

    response The client should note that it has been registered with the host.

- **/unregister** - is - port no., name

    sent by client In order to unregister

    response If the client was registered under that port and name, the host will return the **/unregistered** message. If the client name does not match, it will return error 14; if the client was not registered, the host returns error 3.

- **/unregistered** - is - port no., name

    sent by host Upon a succesful unregistration of the client, or before a shutdown of the host, or when the maximum number of pings are left without reply.

    response The client should note that it has been unregistered with the host.

- **/ping** - is - port no., name

    sent by host Every second to check that the client is still there

    response The client should send a **/pong** message back, if it has indeed this name.

- **/pong** - is - port no., name

    sent by client In response to the **/ping** message of the host.

    response none

- **/error** - ssi - cause, error message, error ID

    sent by host when an error occurs. See the error message table for descriptions of errors.

    response client could do attempts to fix the error.

- **/warn** - ssi - cause, error message, error ID

sent by host When a request from the client does not have any results. See the error message table for descriptions of warnings.

response client can display the warning to the user, but do not need to take action.

Acknowledgments

This software was created by

SuperCollider classes, C++ client, Python client & hive client: Marije Baalman (nescivi)

Processing & Java library: Vincent de Belleval, Brett Bergmann

Max objects: Harry Smoak, Joseph Malloch, Brett Bergmann

PureData objects: Joseph Thibodeau, Marije Baalman

Developed as part of the "Sense/Stage" project and the "Papyrus" project between

Design and Computation Arts, Fine Arts, Concordia University

and

Input Devices and Music Interaction Lab, Music Technology, McGill University

This work was support by grants from the Social Sciences and Humanities Research Council of Canada and the Hexagram Institute for Research/Creation in Media Arts and Sciences, Montréal, QC, Canada.

Development since 2010 by nescivi/sensestage in Amsterdam (http://www.sensestage.eu)

(c) 2008-11 by the authors
Released under the GNU/GPL (see COPYING file)
C++ library under GNU/LGPL

ChangeLog

- 6/4/2010 - Added /map/minibee messages to the network.

- ————- v 0.5 ————

- 12/12/2009 - all messages from clients need to have port no and client name. This breaks compatibility with previous versions.

- ————- v 0.3 ————

- 2/12/2009 - lots of bugfixes. New clients: processing and C++. Better assertion of argument types now in the sending of data by the host.

- ————- v 0.3 ————

- 9/7/2009 - improved data logging and playback support. Added string data nodes (adds a type tag to some osc messages). Logging now saves the spec with labels.

- ————— v 0.2 —————

- 27/5/2009 - added remove all message. error codes for error and warn messages.

- 19/5/2009 - performance improvement, fix bugs in gui, logging option for osc communication, logging option for update times, clients are now removed after a certain amount of missed pongs. Protection for non-numerical data coming in.

- 18/5/2009 - added subscribe all and unsubscribe all messages.

- 4/4/2009 - added client gui, and updated the client in SC.

- 2/4/2009 - added gui for connected clients

- 1/4/2009 - added help files and wii mote support, improved main gui

- 12/3/2009 - added pattern support

- 12/3/2009 - create a bridge from GeneralHID, including some other bug-fixes

- 12/3/2009 - added a size argument to expected nodes; if set, this will create the node already with the given size, with data values 0, so that properties of the node and slots can be set. (to fix the todo: create "virtual nodes" for nodes that are expected but not there yet, so some settings can already be set)

- 12/3/2009 - implemented the port storage in a file mechanism

- 21/11/2008 - implemented backup mechanism for reconnection of any clients that were connected before a restart and the SC client version

- 21/11/2008 - added warn message for some actions

- 06/10/2008 - added announce message

- 06/10/2008 - added acknowledgement messages for actions that do not have an immediate reply otherwise

- 06/10/2008 - changed so that nodeID's and slotID's now are always integers.