Sense World Data Network

Marije Baalman

April 2, 2009

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. 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 Max implementation (by Harry Smoak)

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.

4 SuperCollider implementation

The SuperCollider implementation is done in a set of classes.

Documentation for these is available in HTML format.

5 Installation

5.1 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.

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

/datanetwork/announce	si	host, port no.	announce the network with its coordinates
/register	i	port no.	register to the network as a client
/registered	i	port no.	reply to register to the network as a client
/unregister	i	port no.	unregister to the network as a client
/unregistered	i	port no.	reply to unregister to the network as a client
/ping	i	port no.	message to check if client is still there
/pong	i	port no.	expected reply to the /ping message
/error	s	error message	error occurred upon request
/warn	s	warn message	non fatal error occurred upon request
/query/expected	i	port no.	query which nodes are expected in the network (reply /info/expected)
/query/nodes	i	port no.	query which nodes are in the network (reply /info/node)
/query/slots	i	port no.	query which slots are in the network (reply /info/slot)
/query/subscriptions	i	port no.	query which subscriptions the client has (reply /subscribed/node, /subscribed/slot)
/query/clients	i	port no.	query which clients are in the network (reply /info/client)
/query/setters	i	port no.	query which nodes the client is the setter of (reply /info/setter)
/info/expected	is	node ID, node label	info about an expected node
/info/node	isi	node ID, node label, number of slots	info about a node
/info/slot	iis	node ID, slot ID, slot label	info about a slot
/info/client	sis	ip, port no., hostname	info about a client
/info/setter	isi	node ID, node label, number of slots	info about a node the client is setting
/subscribe/node	ii	port no., node ID	subscribe to receive data from a node
/subscribed/node	ii	port no., node ID	reply to subscribe to receive data from a node
/subscribe/slot	iii	port no., node ID, slot ID	subscribe to receive data from a slot
/subscribed/slot	iii	port no., node ID, slot ID	reply to subscribe to receive data from a slot
/unsubscribe/node	ii	port no., node ID	unsubscribe to receive data from a node
/unsubscribed/node	ii	port no., node ID	reply to unsubscribe to receive data from a node
/unsubscribe/slot	iii	port no., node ID, slot ID	subscribe to receive data from a slot
/unsubscribed/slot	iii	port no., node ID, slot ID	reply to unsubscribe to receive data from a slot
/data/node	iff..f	node ID, data values	node data
/data/slot	iif	node ID, slot ID, data value	slot data
/get/node	ii	port no., node ID	get data from a node (reply /data/node)
/get/slot	iii	port no., node ID, slot ID	get data from a slot (reply /data/slot)
/set/data	iif..f	port no., node ID, data values	set data to a node (reply /data/node)
/label/node	iis	port no., node ID, node label	set label to a node
/label/slot	iiis	port no., node ID, slot ID, slot label	set label to a slot
/remove/node	ii	port no., node ID	remove a node (only possible if client is setter)
/removed/node	ii	port no., node ID	reply to remove a node
/add/expected	iisi	port no., node ID, node label, node size	add an expected node to the network (reply /info/expected) if node size is given, the node is created as well (and generate and /info/node message)

Table 1: OSC namespace for the Data Network

5.2 Apache

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

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
```

TODO (SuperCollider)

- add gui for connected clients
- update the client in SC
- (discard) change expectedNodes to become a security option

ChangeLog

- 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.