Sense/Stage - low cost, open source wireless sensor infrastructure for live performance and interactive, real-time environments

Marije A.J. Baalman, Vincent de Belleval, Harry Smoak

Brett Bergmann, Christopher L. Salter

Design and Computation Arts, Concordia University

Joseph Malloch, Joseph Thibodeau, Marcelo M. Wanderley

Input Devices and Music Interaction Laboratory (IDMIL), McGill University

Montréal, Québec, Canada

Overview

- Introduction and motivation
- SenseStage MiniBee
- SenseWorld DataNetwork
- Clients
- Integration
- Examples
- Conclusions

What is Sense/Stage?

Research-creation project to develop hardware and software for use in live performance (theater, music, dance) and interactive environments.

Components:

- small, battery powered wireless PCB's for sensing and actuation,
- software for real-time sharing of data, and
- modules to analyse datastreams and generate complex dynamics for output media.

Motivation

- Economic and technical constraints of live performance
- Lack of tools for artistic use
- Real world testing scenarios

Motivation

- Economic and technical constraints of live performance

  Limited time in rehearsals to use and test technology

- Lack of tools for artistic use

  Low cost, integration with common software environments

- Real world testing scenarios

  Application in professional productions, outside of the lab

SenseStage MiniBee

Design goals

» Overview

Introduction and motivation

SenseStage MiniBee
» Design goals
» SenseStage MiniBee
» Inputs and outputs
» Firmware

SenseWorld DataNetwork

Clients

Integration

Examples

Conclusion

- Low cost
- Small form factor
- Flexible sensor configuration
- Usable for control of motors, LEDs, and other actuators
- Operable in large groups (10+ nodes)
- Long battery life
- Ease of use
- Programmable, so that the board can take care of more logic and processing of data, if desired by the user

SenseStage MiniBee

» Overview

Introduction and motivation

SenseStage MiniBee

» Design goals

» SenseStage MiniBee

» Inputs and outputs

» Firmware

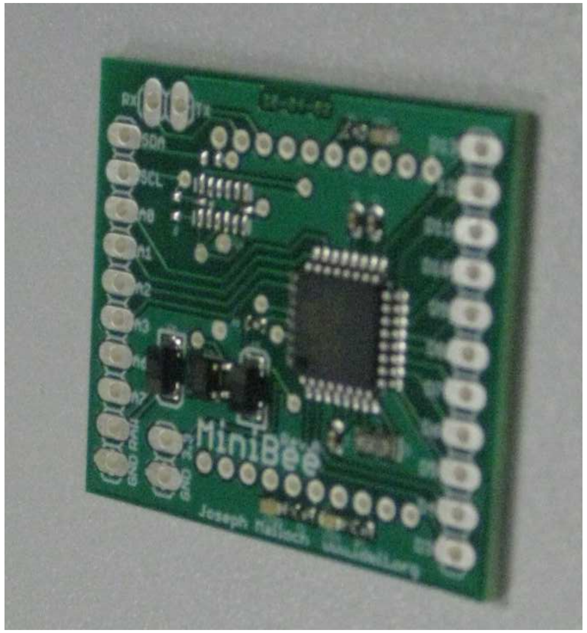
SenseWorld DataNetwork

Clients

Integration

Examples

Conclusion



- Small size: 42mm x 35mm
- Based on Arduino Mini Pro
- Useable with XBee radio (mesh network)
- Unit cost price 32 CAD (excluding XBee)

Inputs and outputs

- » Overview

Introduction and motivation

SenseStage MiniBee
- » Design goals
- » SenseStage MiniBee
- » Inputs and outputs
- » Firmware

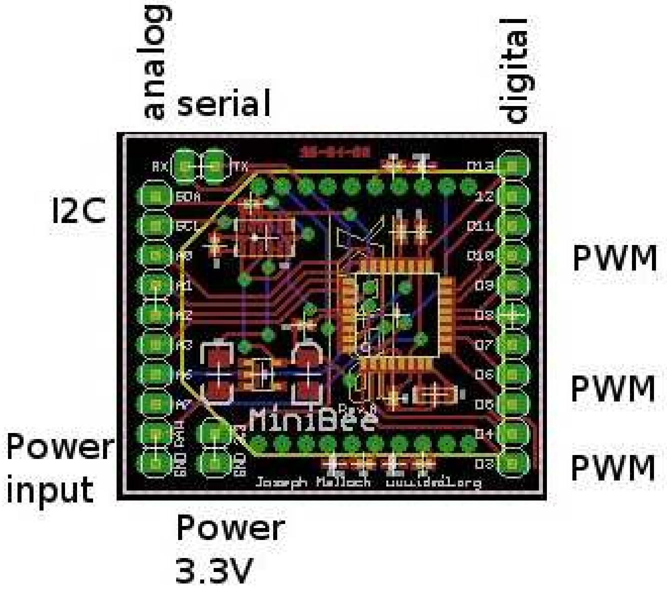
SenseWorld DataNetwork

Clients

Integration

Examples

Conclusion



- 8 analog inputs (left side): A4 (SDA), A5 (SCL), A0, A1, A2, A3, A6, A7
- 11 digital inputs or outputs (right side): D3 to D13
- Serial I/O (top left): RX, TX
- Power input (between 3.3V and 5V) (left side bottom): RAW, GND
- Regulated power output (left side bottom inside): 3.3V, GND
- I2C communication, (left side top): SDA, SCL
- PWM output, (right side), D3, D5, D6, D9, D10, D11

Firmware

Firmware library supports:

- Analog sensors (connected to the analog input pins, e.g. resistive sensors, analog accelerometers, infrared distance sensors)
- Digital sensors (on/off, e.g. buttons and switches)
- LIS302DL accelerometer, using I2C
- Relative humidity and temperature sensor
- Ultrasound sensors
- PWM output (e.g. dimmable LEDs, motors)
- Digital output (on/off)

Firmware

Wireless configuration of the firmware (compare to Firmata)

- Board reads the serial number of the XBee and sends it to the coordinator
- Coordinator assigns an ID and tells the board whether or not it will receive a new configuration
  - Board sends a message back that it is waiting
  - Coordinator sends configuration
- Board sends a summary of the current configuration (for the coordinator to verify)
- Board starts sending data

Firmware

Wireless configuration of the firmware (compare to Firmata)

- Board reads the serial number of the XBee and sends it to the coordinator
- Coordinator assigns an ID and tells the board whether or not it will receive a new configuration
  - Board sends a message back that it is waiting
  - Coordinator sends configuration
- Board sends a summary of the current configuration (for the coordinator to verify)
- Board starts sending data

Firmware can be customized for tasks not implemented in our library.

SenseWorld DataNetwork

SenseWorld DataNetwork

The aim of the SenseWorld DataNetwork is to facilitate

- sharing and
- manipulation of
- multiple data streams
- between collaborators
- in heterogeneous interactive performance environments

SenseWorld DataNetwork

The aim of the SenseWorld DataNetwork is to facilitate

- sharing and
- manipulation of
- multiple data streams
- between collaborators
- in heterogeneous interactive performance environments

Personal experience in interactive dance/theatre: sharing data between light, video, sound and sensing

SenseWorld DataNetwork

The aim of the SenseWorld DataNetwork is to facilitate

- sharing and
- manipulation of
- multiple data streams
- between collaborators
- in heterogeneous interactive performance environments

Personal experience in interactive dance/theatre: sharing data between light, video, sound and sensing

Other frameworks:

- Mapper of the Digital Orchestra project
- KeyWorx

Motivation

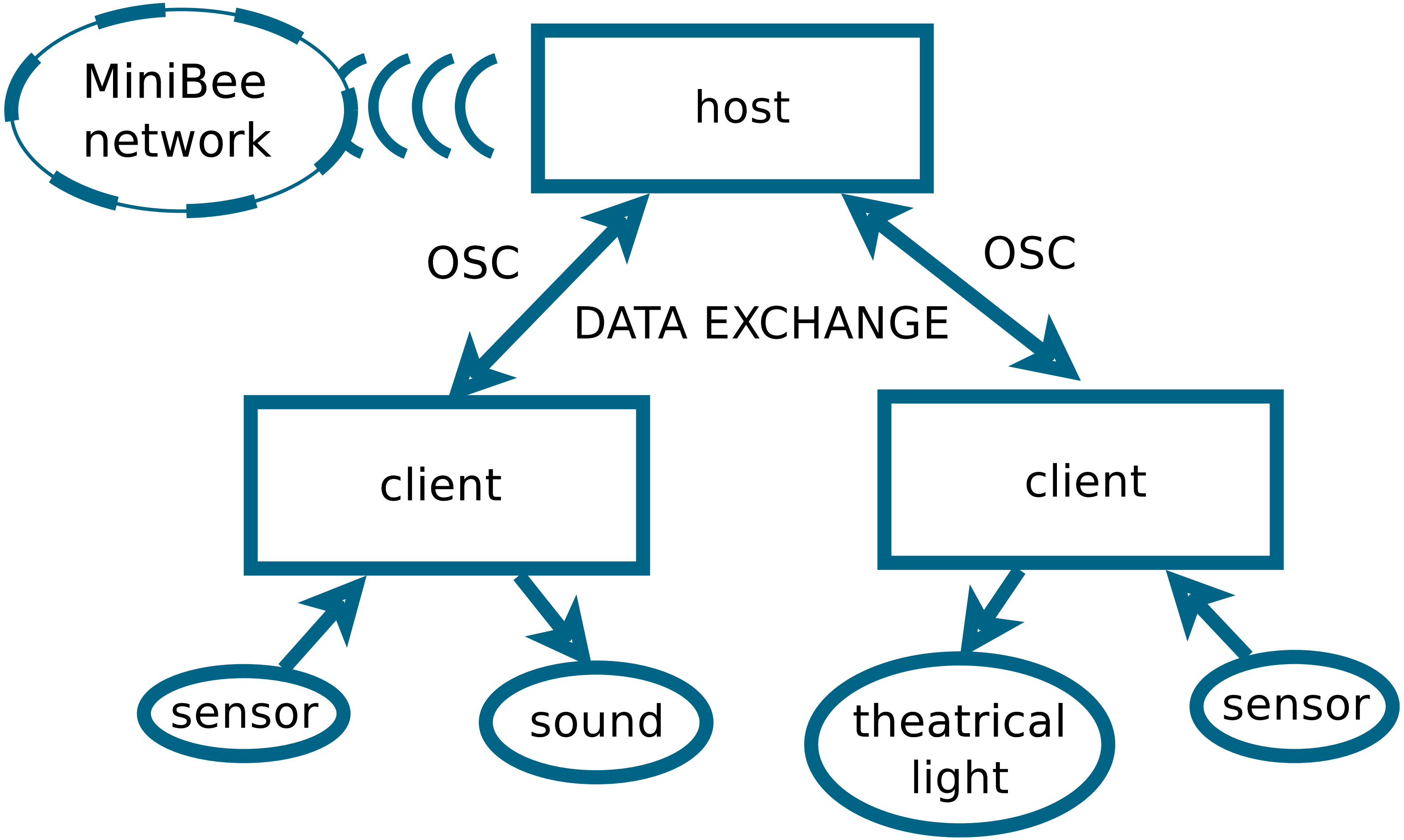
- OpenSoundControl provides a low level communication protocol
- and is used by many programs for interactive media (e.g. SuperCollider, Max/MSP, PureData, Processing, ...)
- But there is no higher level protocol to facilitate the communication

Motivation

- OpenSoundControl provides a low level communication protocol
- and is used by many programs for interactive media (e.g. SuperCollider, Max/MSP, PureData, Processing, ...)
- But there is no higher level protocol to facilitate the communication

Support *coordinated* collaboration with real time data within a live performance context, involving multiple types of media

Concept


MiniBee network
host
OSC
OSC
DATA EXCHANGE
client
client
sensor
sound
theatrical light
sensor

Design criteria

- Any client should be able to subscribe to data
- Any client should be able to supply data
- Restore configuration quickly
- Usable within heterogeneous media software environments
- Enable collaboration between different design practices
- Enable efficiency of collaboration within the limited timeframe of rehearsals
- Integration with the wireless sensor network

Design criteria

- Any client should be able to subscribe to data
- Any client should be able to supply data
- Restore configuration quickly
- Usable within heterogeneous media software environments
- Enable collaboration between different design practices
- Enable efficiency of collaboration within the limited timeframe of rehearsals

- Integration with the wireless sensor network

The framework defines an OSC-namespace for communication
between host and clients,
for exchanging and labeling data

Design criteria

The framework defines an OSC-namespace for communication between host and clients, for exchanging and labeling data

DataNetwork the network itself

DataNode a node is a collection of slots, usually based upon a device or another common source

DataSlot a slot is a single data stream

Client client program receiving and/or setting data

Client connection

- The host writes a textfile with the network's OSC port can be found in the file `http://192.168.1.7/SenseWorldDataNetwork`
- The host broadcasts an announce message

Client connection

- The host writes a textfile with the network's OSC port can be found in the file `http://192.168.1.7/SenseWorldDataNetwork`
- The host broadcasts an announce message

- The client sends a register message
- The client has to reply to regularly sent ping messages to confirm its presence
- The client queries which nodes and slots are present

Client connection

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

» SenseWorld DataNetwork

» Motivation

» Concept

» Design criteria

» Client connection

» Auto-recovery

Clients

Integration

Examples

Conclusion

- The host writes a textfile with the network's OSC port can be found in the file `http://192.168.1.7/SenseWorldDataNetwork`
- The host broadcasts an announce message

- The client sends a register message
- The client has to reply to regularly sent ping messages to confirm its presence
- The client queries which nodes and slots are present

- The client subscribes to nodes/slots of interest
- The client can set data to nodes
- The client is informed of new or changed nodes that are created after registration

Auto-recovery

Sometimes software and/or hardware is unexpectedly and fatally interrupted, even during rehearsals and performances.

Auto-recovery

Sometimes software and/or hardware is unexpectedly and fatally interrupted, even during rehearsals and performances.

- The host writes a textfile with the network's OSC port can be found in the file `http://192.168.1.7/SenseWorldDataNetwork`.
  -> The client looks this file up, upon startup

Auto-recovery

Sometimes software and/or hardware is unexpectedly and fatally interrupted, even during rehearsals and performances.

- The host writes a textfile with the network's OSC port can be found in the file `http://192.168.1.7/SenseWorldDataNetwork`.
  -> The client looks this file up, upon startup
- The host broadcasts an announce message
  -> The client reacts with register message

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

» SuperCollider host and client

» Processing client

» Processing client

» PureData client

» C++ client library

» Max/MSP client

Integration

Examples

Conclusion

Clients

SuperCollider host and client

Host:

```
x = SWDataNetwork.new; // define the network
x.createHost;
```

SuperCollider host and client

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

» SuperCollider host and client

» Processing client

» Processing client

» PureData client

» C++ client library

» Max/MSP client

Integration

Examples

Conclusion

Host:

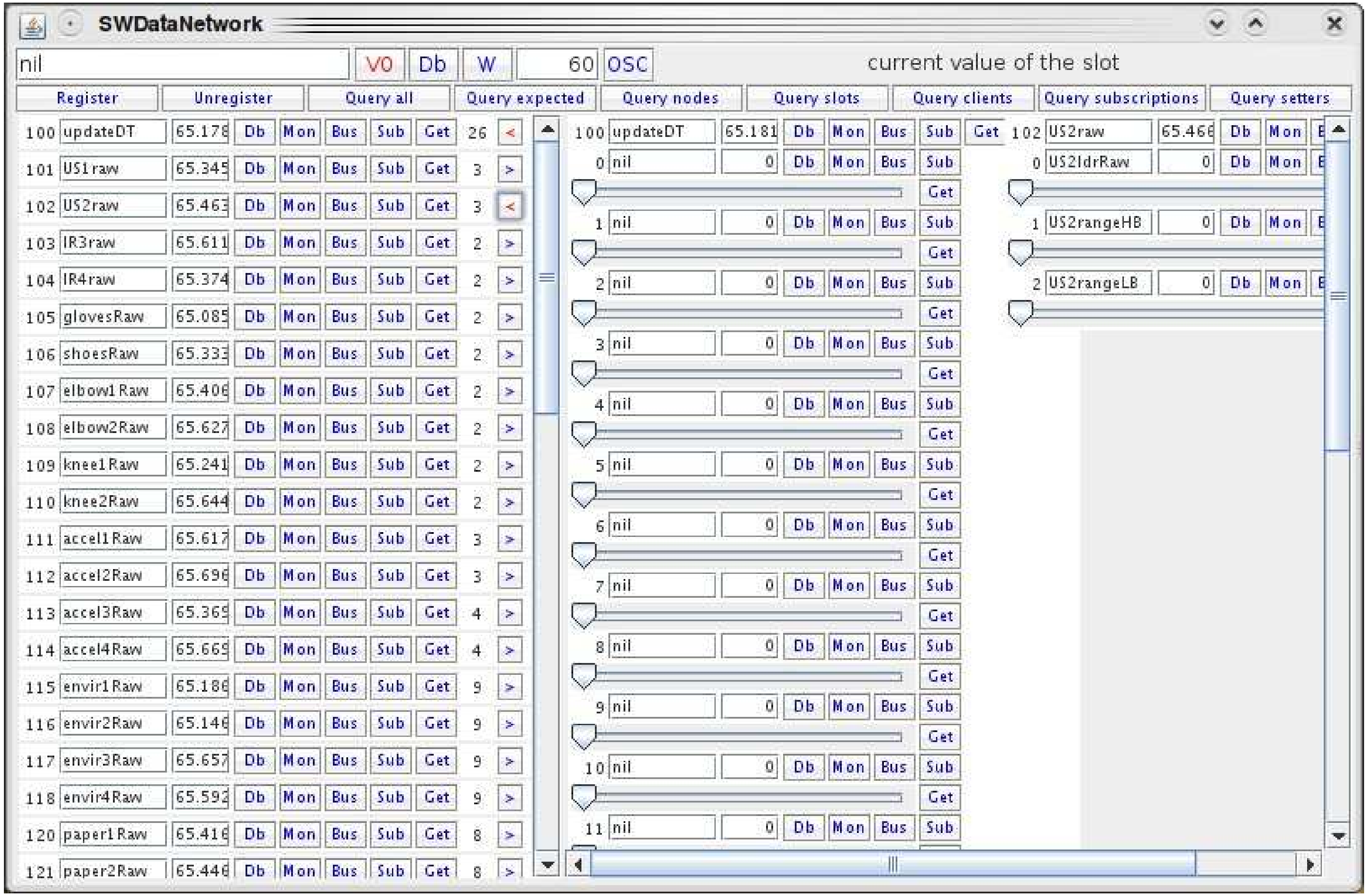
```
x = SWDataNetwork.new; // define the network
x.createHost;
```

Client:

```
y =
SWDataNetworkClient.new(“192.168.0.104”,“Marije”);
y.makeGui;
y.addExpected( 100, “testnode”, 5 );
y.setData( 100, [0,1,2,3,4] );
y.queryNodes; // query available nodes
y.subscribeNode( 101 ); // subscribe to node 101
```

SuperCollider host and client

- » Overview
- Introduction and motivation
- SenseStage MiniBee
- SenseWorld DataNetwork
- Clients
- » SuperCollider host and client
- » Processing client
- » Processing client
- » PureData client
- » C++ client library
- » Max/MSP client
- Integration
- Examples
- Conclusion



nil | 60 | current value of the slot

100	updateDT	65.178	26
101	US1raw	65.345	3
102	US2raw	65.463	3
103	IR3raw	65.611	2
104	IR4raw	65.374	2
105	glovesRaw	65.085	2
106	shoesRaw	65.333	2
107	elbow1Raw	65.406	2
108	elbow2Raw	65.627	2
109	knee1Raw	65.241	2
110	knee2Raw	65.644	2
111	accel1Raw	65.617	3
112	accel2Raw	65.696	3
113	accel3Raw	65.369	4
114	accel4Raw	65.669	4
115	envir1Raw	65.186	9
116	envir2Raw	65.146	9
117	envir3Raw	65.657	9
118	envir4Raw	65.592	9
120	paper1Raw	65.416	8
121	paper2Raw	65.446	8

100	updateDT	65.181
0	nil	0
1	nil	0
2	nil	0
3	nil	0
4	nil	0
5	nil	0
6	nil	0
7	nil	0
8	nil	0
9	nil	0
10	nil	0
11	nil	0

102	US2raw	65.466
0	US2IdrRaw	0
1	US2rangeHB	0
2	US2rangeLB	0

Processing client

```
import datanetwork.*;

DNConnection dn; //DNConnection instance
DNNode node; //DNNode instance

void setup() {
  dn = new DNConnection(this, "192.168.0.104",
           dn.getServerPort("192.168.0.104"),
           6009, "p5Client");
  node = new DNNode(2000, 5, 0, "p5Node");
}

void stop() {
  dn.unsubscribeAll();
  dn.removeAll();
  dn.close();
}
```

Processing client

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

» SuperCollider host and client

» Processing client

» Processing client

» PureData client

» C++ client library

» Max/MSP client

Integration

Examples

Conclusion

```
void keyPressed() {
    if(key == ’r’) dn.register();
    else if(key == ’q’) dn.queryAll();
    else if(key == ’f’) dn.subscribeNode(401);
    else if(key == ’d’) dn.setData(node,
      new float[] { 4.0, 2.0, 1.0, 2.3, 4.4 } );
}

void dnEvent(String addr, float[] args) {
    print("Float: " + addr);
    for(int i = 0;i < args.length;i++)
        print(" "+args[i]);
    println();
}
```

PureData client

- **dn.node** - get data from node(s)
- **dn.makenode** - supply data to a node

PureData client

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

» SuperCollider host and client

» Processing client

» Processing client

» PureData client

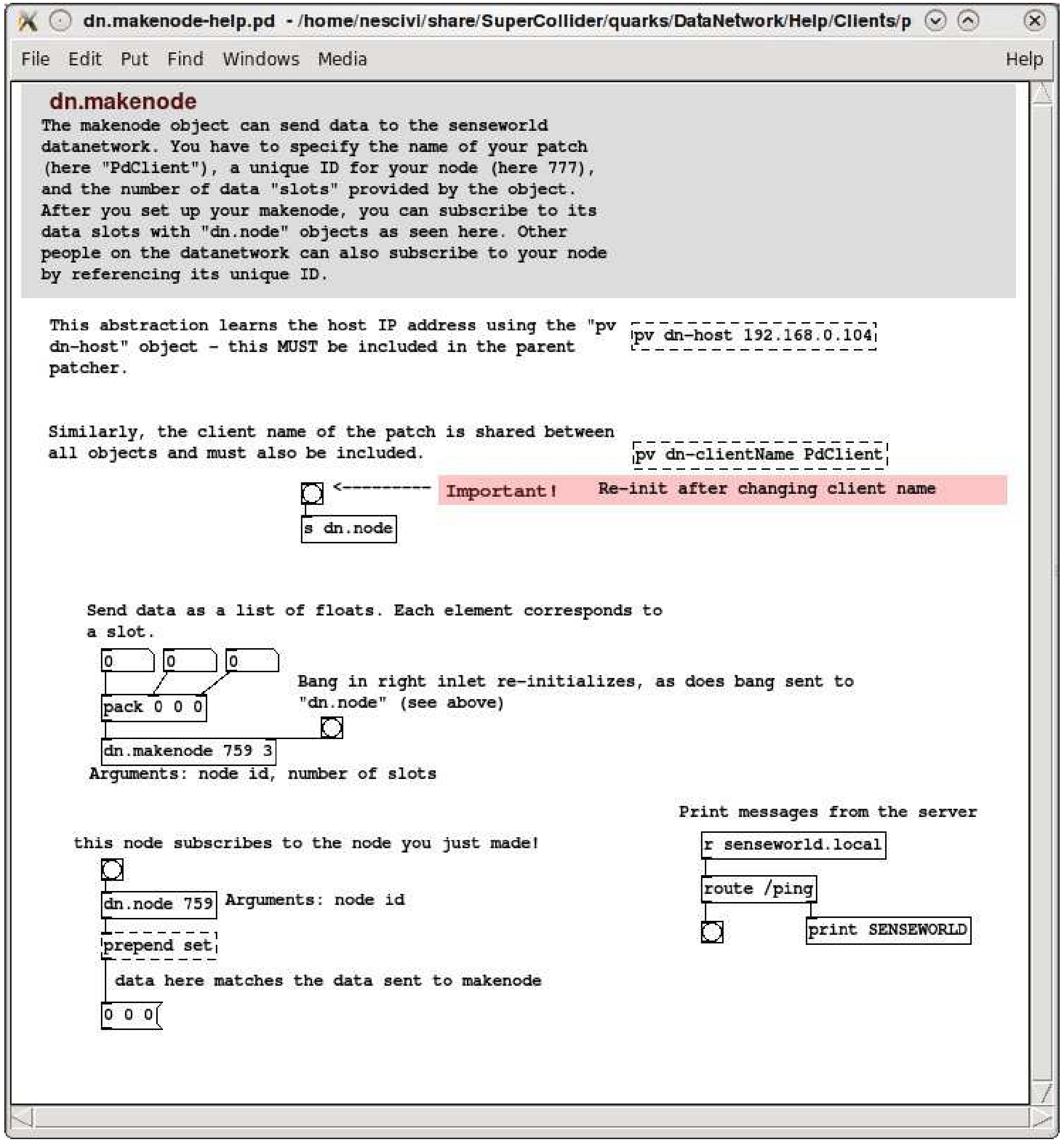
» C++ client library

» Max/MSP client

Integration

Examples

Conclusion



dn.makenode-help.pd - /home/nescivi/share/SuperCollider/quarks/DataNetwork/Help/Clients/p
File Edit Put Find Windows Media
Help

dn.makenode

The makenode object can send data to the senseworld datanetwork. You have to specify the name of your patch (here "PdClient"), a unique ID for your node (here 777), and the number of data "slots" provided by the object. After you set up your makenode, you can subscribe to its data slots with "dn.node" objects as seen here. Other people on the datanetwork can also subscribe to your node by referencing its unique ID.

This abstraction learns the host IP address using the "pv dn-host" object - this MUST be included in the parent patcher.

```
pv dn-host 192.168.0.104
```

Similarly, the client name of the patch is shared between all objects and must also be included.

```
pv dn-clientName PdClient
```

<---------- Important! Re-init after changing client name

```
s dn.node
```

Send data as a list of floats. Each element corresponds to a slot.

```
0    0    0
pack 0 0 0
dn.makenode 759 3
```

Bang in right inlet re-initializes, as does bang sent to "dn.node" (see above)

Arguments: node id, number of slots

Print messages from the server

```
r senseworld.local
route /ping
print SENSEWORLD
```

this node subscribes to the node you just made!

```
dn.node 759
```

Arguments: node id

```
prepend set
```

data here matches the data sent to makenode

```
0 0 0
```

C++ client library

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

» SuperCollider host and client

» Processing client

» Processing client

» PureData client

» C++ client library

» Max/MSP client

Integration

Examples

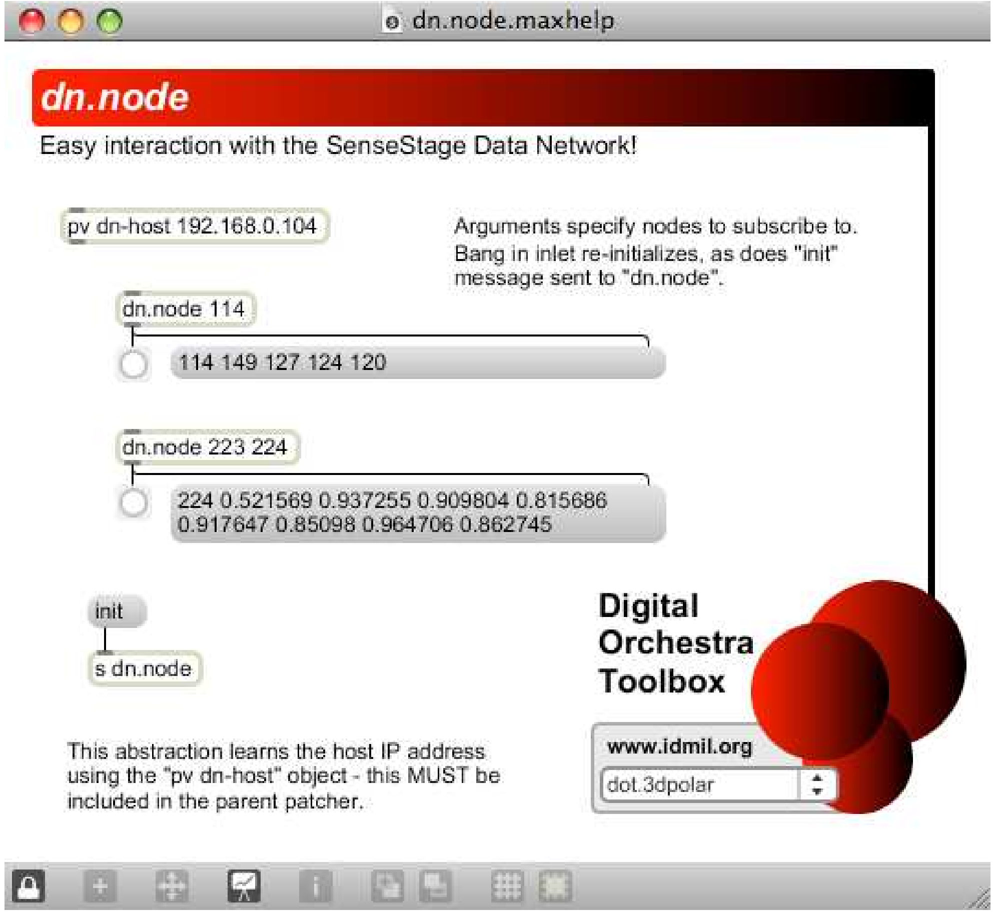
Conclusion

```
DataNode * node;
DataNetwork * dn;

dn = new DataNetwork();
dn->createOSC( "127.0.0.1", 7000, "libdn" );
dn->registerMe();
dn->query();

dn->subscribeNode( 5, true );

dn->createNode( 4, "world", 5, 0, true );

float dummydata[] = {0.1, 0.3, 0.4, 0.5, 0.6};
node = dn->getNode( 4 );
node->setData( 5, dummydata);
node->send( true );
```

Max/MSP client

- **dn.node** - get data from node(s)
- **dn.makenode** - supply data to a node
- **dn->dot** - bridge from the datanetwork to the digital orchestra mapping tools

Max/MSP client



dn.node.maxhelp

dn.node

Easy interaction with the SenseStage Data Network!

pv dn-host 192.168.0.104

Arguments specify nodes to subscribe to. Bang in inlet re-initializes, as does "init" message sent to "dn.node".

dn.node 114

114 149 127 124 120

dn.node 223 224

224 0.521569 0.937255 0.909804 0.815686 0.917647 0.85098 0.964706 0.862745

init

s dn.node

This abstraction learns the host IP address using the "pv dn-host" object - this MUST be included in the parent patcher.

Max/MSP client

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

» SuperCollider host and client

» Processing client

» Processing client

» PureData client

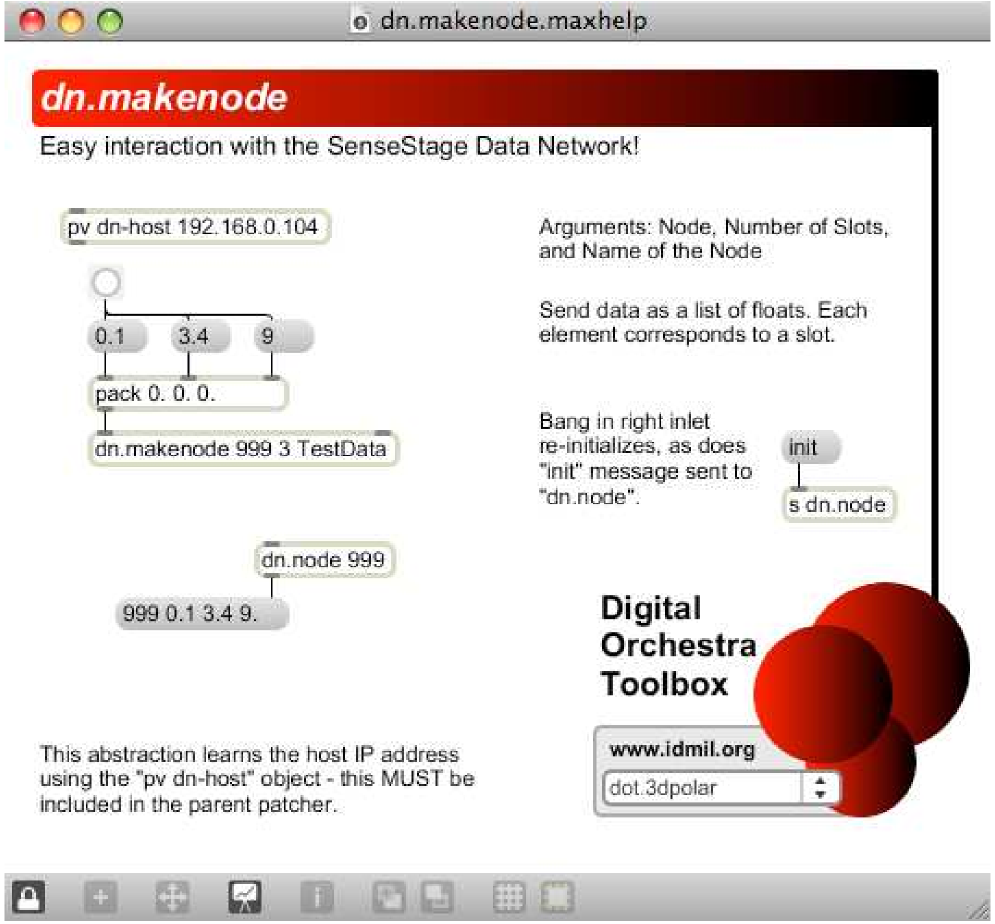
» C++ client library

» Max/MSP client

Integration

Examples

Conclusion



dn.makenode.maxhelp

dn.makenode

Easy interaction with the SenseStage Data Network!

```
pv dn-host 192.168.0.104

0.1    3.4    9

pack 0. 0. 0.

dn.makenode 999 3 TestData

dn.node 999

999 0.1 3.4 9.
```

Arguments: Node, Number of Slots, and Name of the Node

Send data as a list of floats. Each element corresponds to a slot.

Bang in right inlet re-initializes, as does "init" message sent to "dn.node".

```
init

s dn.node
```

This abstraction learns the host IP address using the "pv dn-host" object - this MUST be included in the parent patcher.

Integration

Integration MiniBee and DataNetwork

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

Integration
» Integration MiniBee and DataNetwork

Examples

Conclusion

- Reading in data from the wireless nodes in the DataNetwork host
- Each MiniBee “node” corresponds to a DataNode
- Each sensor on the MiniBee corresponds to a DataSlot

Integration MiniBee and DataNetwork

- Reading in data from the wireless nodes in the DataNetwork host
- Each MiniBee “node” corresponds to a DataNode
- Each sensor on the MiniBee corresponds to a DataSlot

- Dialog for wireless configuration of nodes
- Storing current configuration of all nodes with DataNetwork setup

Examples

Chronotopia

Chronotopia - a dance piece with the Attakkalari Centre for Movement (India), music from Matthias Duplessy, video from Christian Ziegler, interactive light installation controlled from SuperCollider

Chronotopia

Chronotopia - a dance piece with the Attakkalari Centre for Movement (India), music from Matthias Duplessy, video from Christian Ziegler, interactive light installation controlled from SuperCollider

premiere at the Attakkalari India Dance Biennial in Bangalore (2 shows) and 2 other shows in India (Chennai, Heggodu)
tour in March in Germany (Kassel, Frankfurt) and Sweden (Vara)

(video, performance Feb. 2009, in Bangalore, India)

Hardware: light matrix and handheld lights - camera based videotracking

- 6 control boards for 6 CCFL lights based on XBee-Arduino
- 3 handheld CCFL lights with SenseStage MiniBee

Chronotopia

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

Integration

Examples
» Chronotopia
» JND/Semblance
» SenseStage Workshop

Conclusion

Chronotopia - a dance piece with the Attakkalari Centre for Movement (India), music from Matthias Duplessy, video from Christian Ziegler, interactive light installation controlled from SuperCollider

premiere at the Attakkalari India Dance Biennial in Bangalore (2 shows) and 2 other shows in India (Chennai, Heggodu)
tour in March in Germany (Kassel, Frankfurt) and Sweden (Vara)

(video, performance Feb. 2009, in Bangalore, India)

Shared data: camera motion tracking, beat tracking, pitch tracking, light output data, frame timing

- 2 clients
- 15 datanodes
- with each 1 to 12 slots

JND/Semblance

JND/Semblance - a one-person installation piece by Chris Salter, Marije Baalman and Harry Smoak, interactive sound, light and vibration controlled from SuperCollider

public preview at Empac, Troy, NY, March 3-7, 2010

ISEA, Essen, August 2010; Today's Art, The Hague, September 2010

JND/Semblance

JND/Semblance - a one-person installation piece by Chris Salter, Marije Baalman and Harry Smoak, interactive sound, light and vibration controlled from SuperCollider

public preview at Empac, Troy, NY, March 3-7, 2010
ISEA, Essen, August 2010; Today's Art, The Hague, September 2010

Sensing:

- 24 pressure sensing pads made of paper
- 3 SenseStage MiniBees

JND/Semblance

JND/Semblance - a one-person installation piece by Chris Salter, Marije Baalman and Harry Smoak, interactive sound, light and vibration controlled from SuperCollider

public preview at Empac, Troy, NY, March 3-7, 2010

ISEA, Essen, August 2010; Today's Art, The Hague, September 2010

Sensing:

- 24 pressure sensing pads made of paper
- 3 SenseStage MiniBees

Shared data: light settings, paper sensing, derived data, amplitude tracking

- 2 clients
- 17 datanodes
- with each 1 to 24 slots

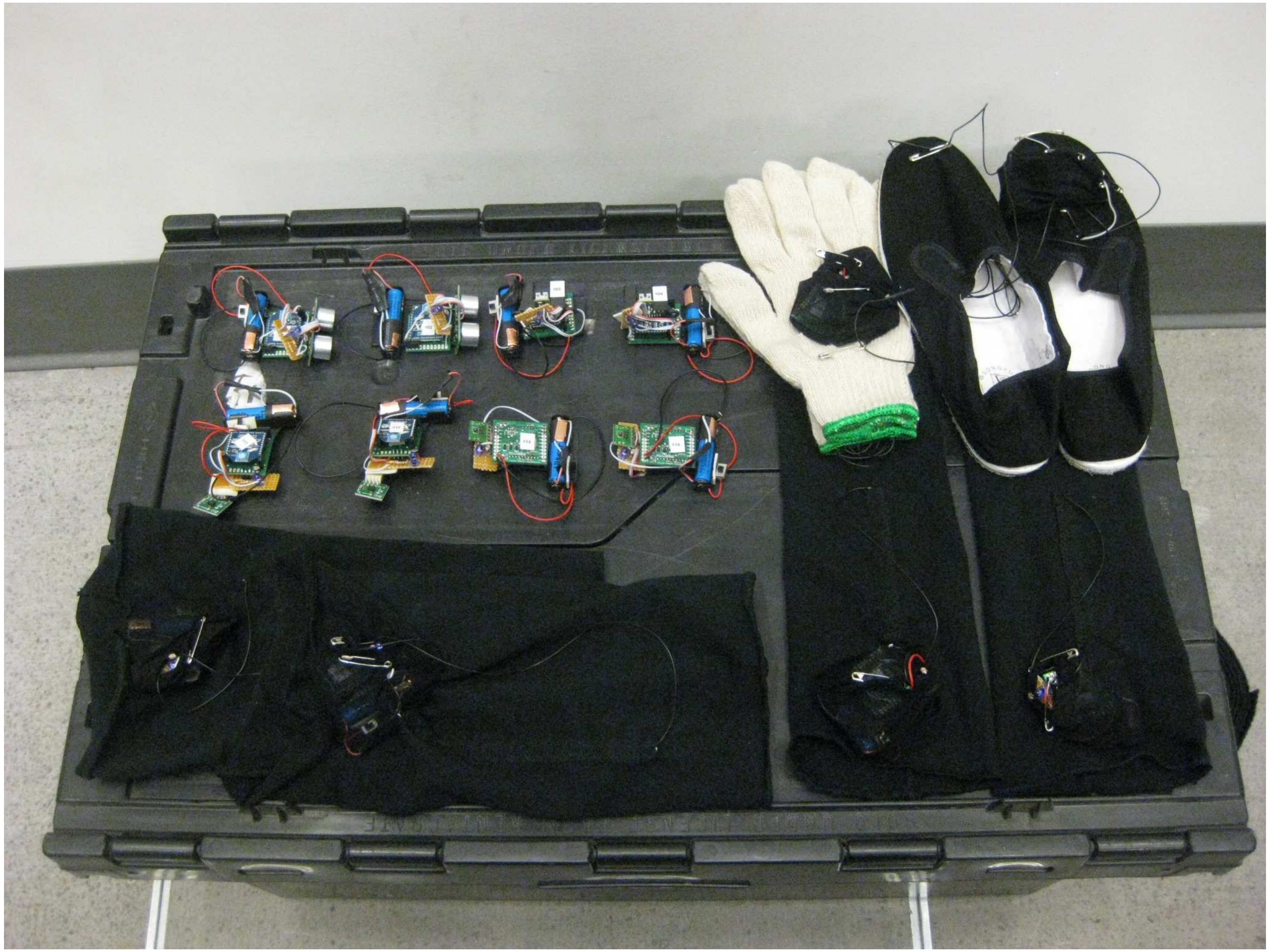
SenseStage Workshop

SenseStage workshop - one-week workshop at Concordia, using 25+ wireless sensor/actuation nodes and camera tracking, in which five projects were created by the participants.

(video, workshop presentations May 2009, Montreal, Canada)

`http://sensestage.hexagram.ca/workshop/`

SenseStage Workshop



SenseStage Workshop

SenseStage workshop - one-week workshop at Concordia, using 25+ wireless sensor/actuation nodes and camera tracking, in which five projects were created by the participants.

(video, workshop presentations May 2009, Montreal, Canada)

`http://sensestage.hexagram.ca/workshop/`

- up to 10 clients
- up to 80 datanodes
- with each 2 to 12 slots

SenseStage Workshop

SenseStage workshop - one-week workshop at Concordia, using 25+ wireless sensor/actuation nodes and camera tracking, in which five projects were created by the participants.

(video, workshop presentations May 2009, Montreal, Canada)

`http://sensestage.hexagram.ca/workshop/`

- up to 10 clients
- up to 80 datanodes
- with each 2 to 12 slots
- server was running for ca. 48 hours nonstop, with clients connecting and disconnecting, hardware nodes turning on and off

Current work

SenseStage MiniBee

- Board revision
- Extensive documentation
- Wireless bootloading
- *Hopefully* MiniBee for sale from second half of 2010

Current work

SenseStage MiniBee

- Board revision
- Extensive documentation
- Wireless bootloading
- *Hopefully* MiniBee for sale from second half of 2010

SenseWorld DataNetwork

- Addition of enabling requests for “derived” data, such as statistical analysis, smoothed data, etc.
- Further integration with the Digital Orchestra Toolset

Future directions

Many data streams require methods and tools to deal with this real time data to make sense of the information and to create meaningful content with it.

- organisation and visualisation of data
- feature extraction (environmental/gestures)
- fusion of data streams
- correlations between data streams
- emergent dynamics

Development of tools integrated with the current infrastructure, as well as a lexicon of techniques.

Website

`http://sensestage.hexagram.ca`

Software available under the GNU/(L)GPL

Acknowledgements

» Overview

Introduction and motivation

SenseStage MiniBee

SenseWorld DataNetwork

Clients

Integration

Examples

Conclusion

» Current work

» Future directions

» Website

» Acknowledgements

Our collaborators: Mark Marshall, Nicholas Munoz, Elio Bidinost and all the “SenseStage Workshop” participants.

Funding by grants from the

Social Sciences and Humanities Research Council of Canada (SSHRC),

Conseil de recherches en sciences humaines du Canada

Social Sciences and Humanities Research Council of Canada

Canada

Fonds de recherche sur la société et la culture, Québec (FQRSC),

and the

Hexagram Institute for Research/Creation in Media Arts and Sciences, Montréal, QC, Canada.

hexagram