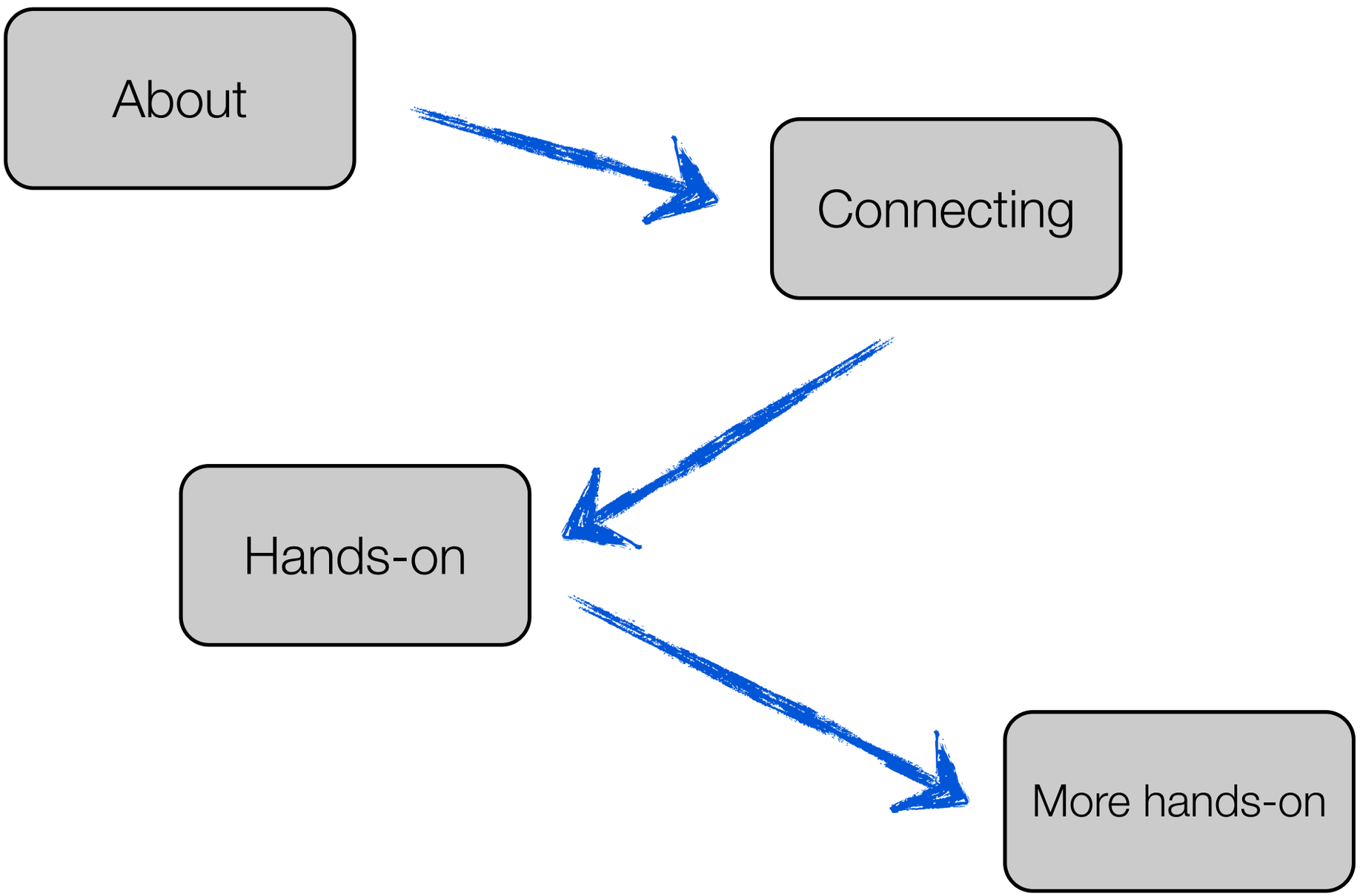
Designing systems in Processing with OOCSI

by Mathias Funk, December 2013

What we do today



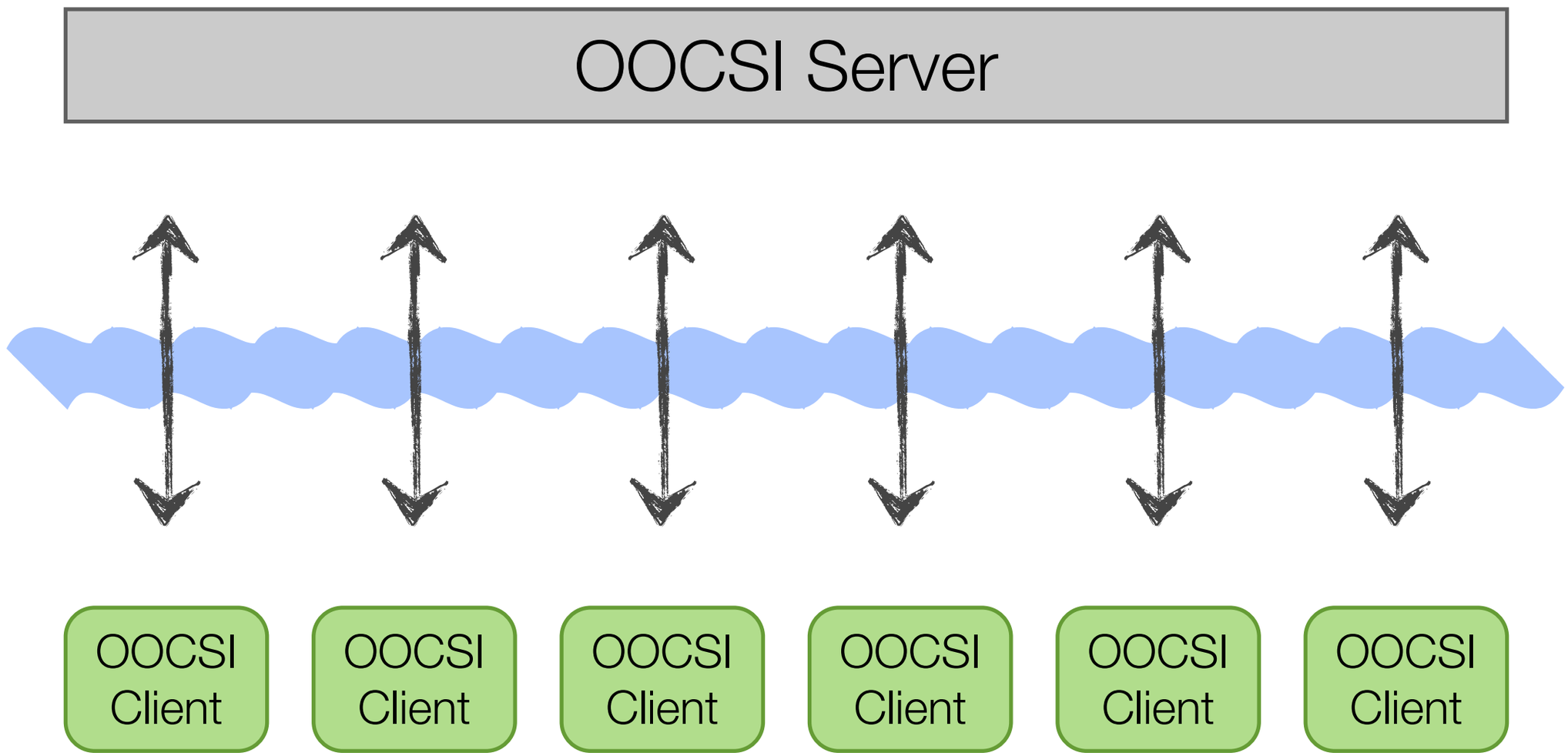
OOCSI

(Out of Control for Semantic Interactivity)

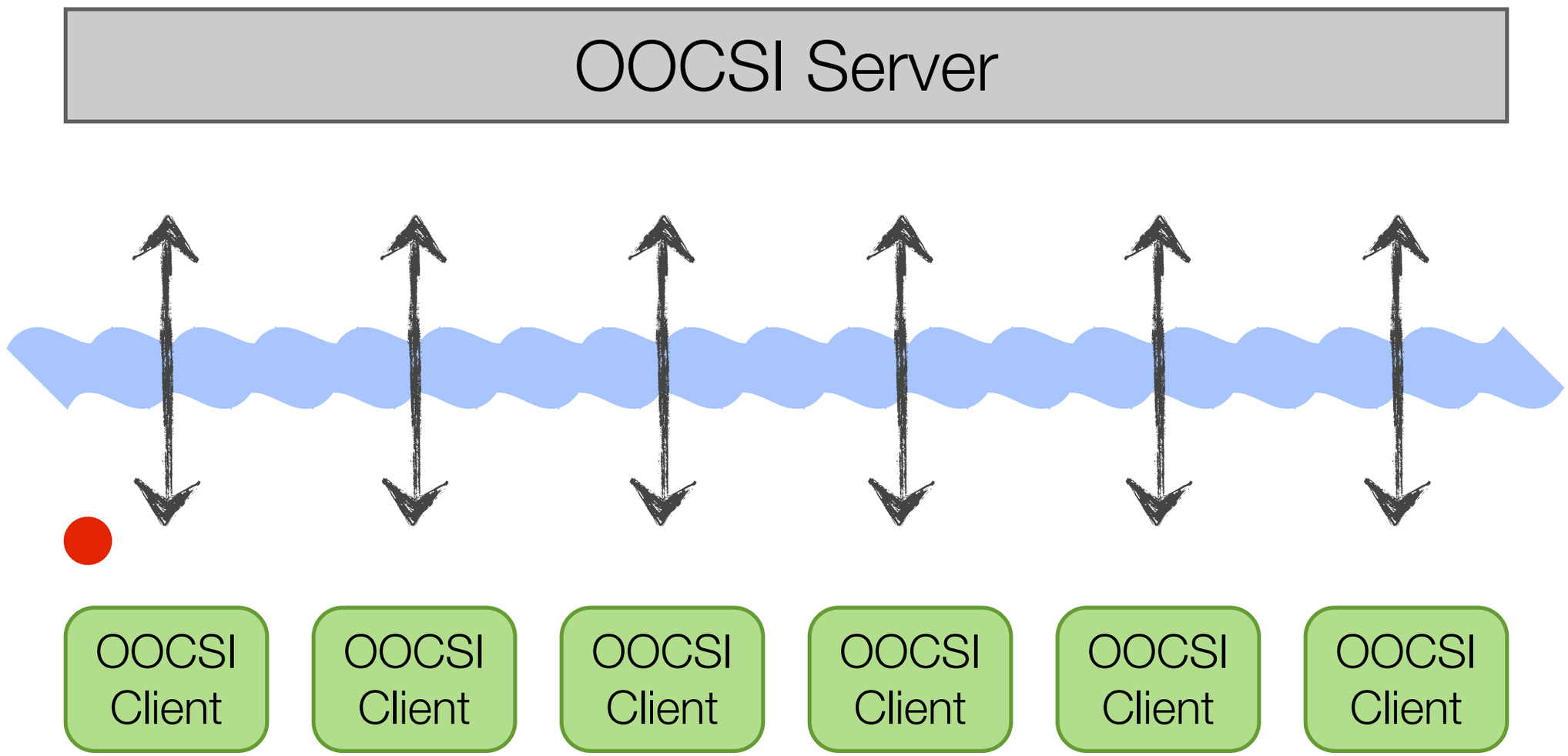
What is it?

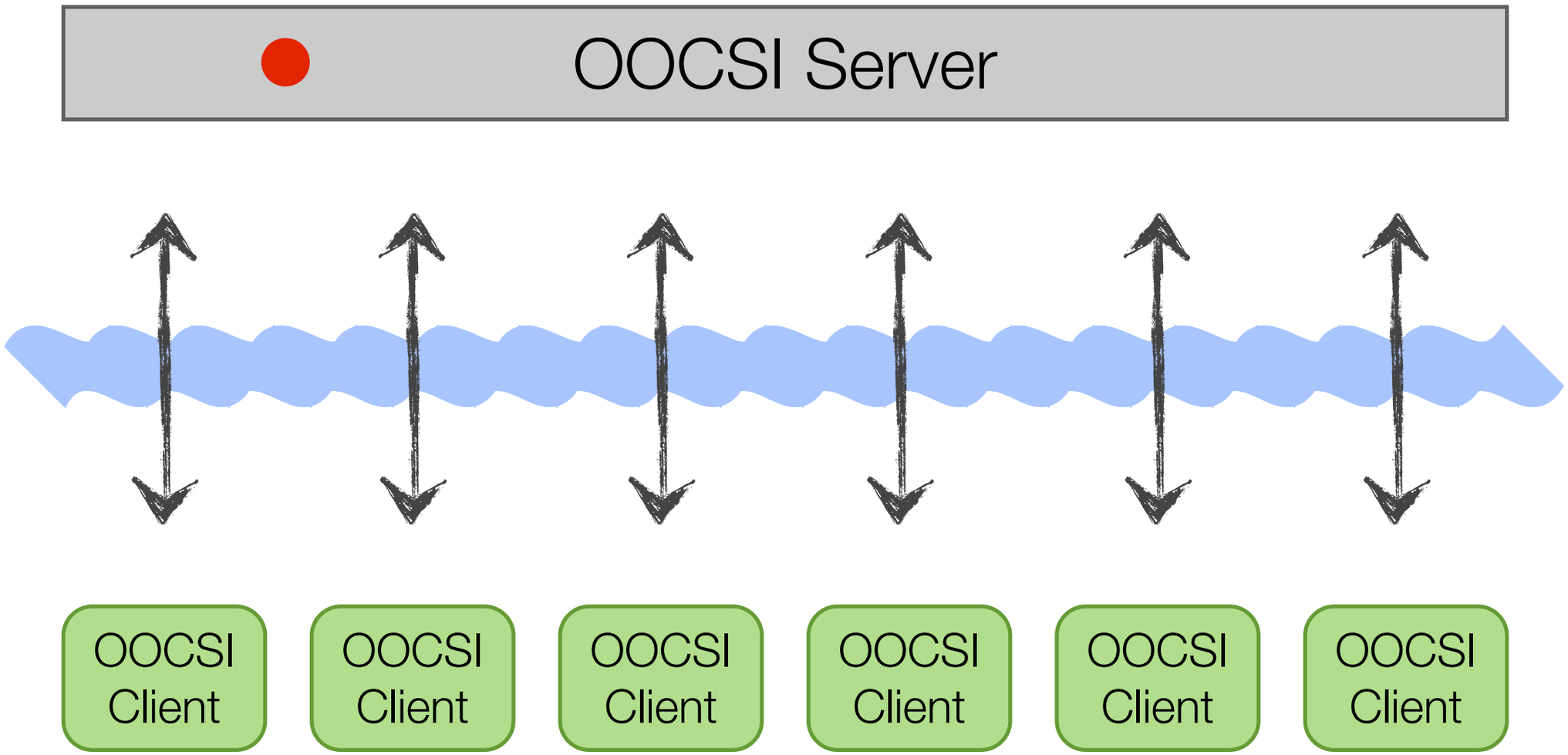
A simple systems-interaction
fabric for use by designers.

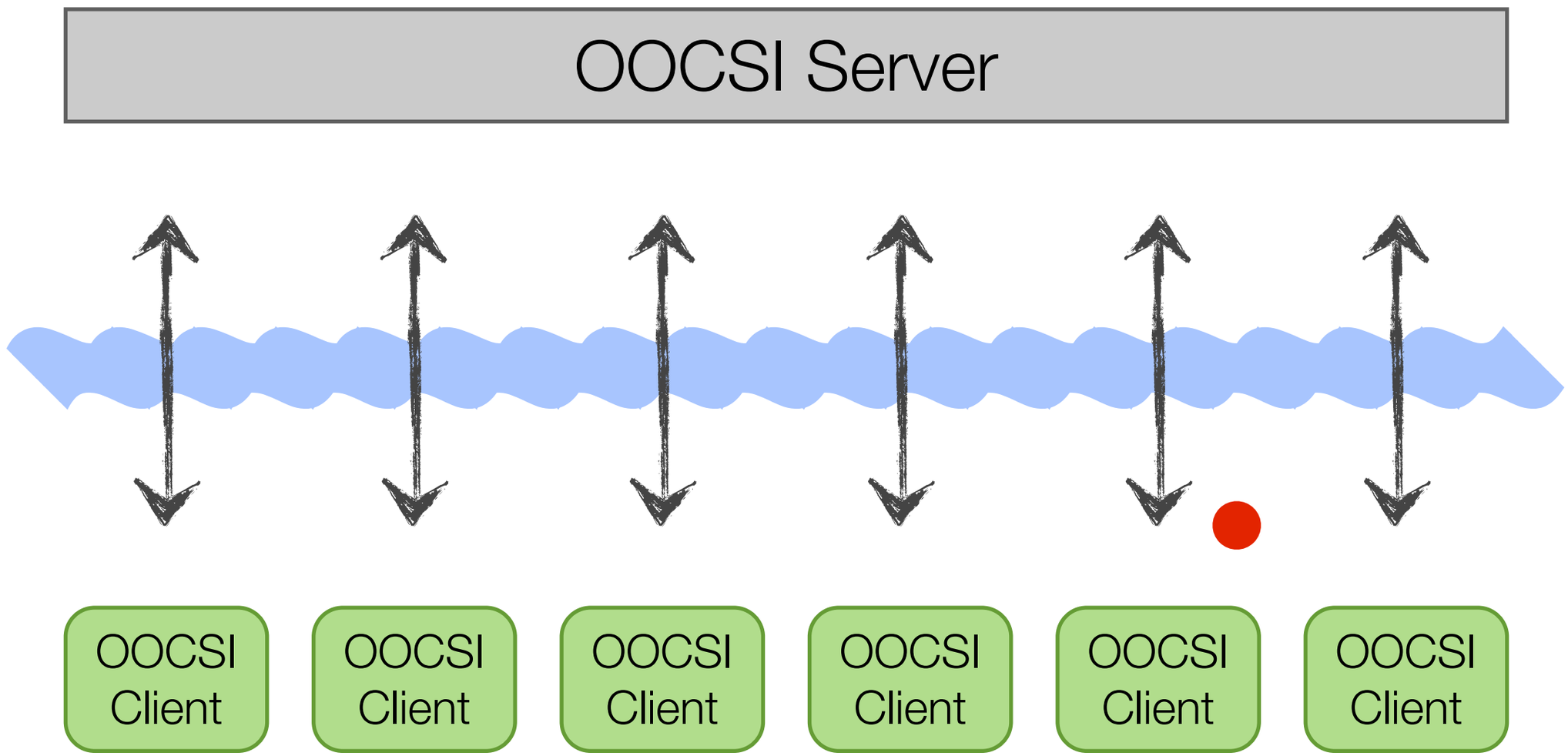
- Easy communication between different computers
- Sending and receiving messages
- Channels (broadcasting to single or multiple computers)
- Data types, semantics, actions


OOCSI Server
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client

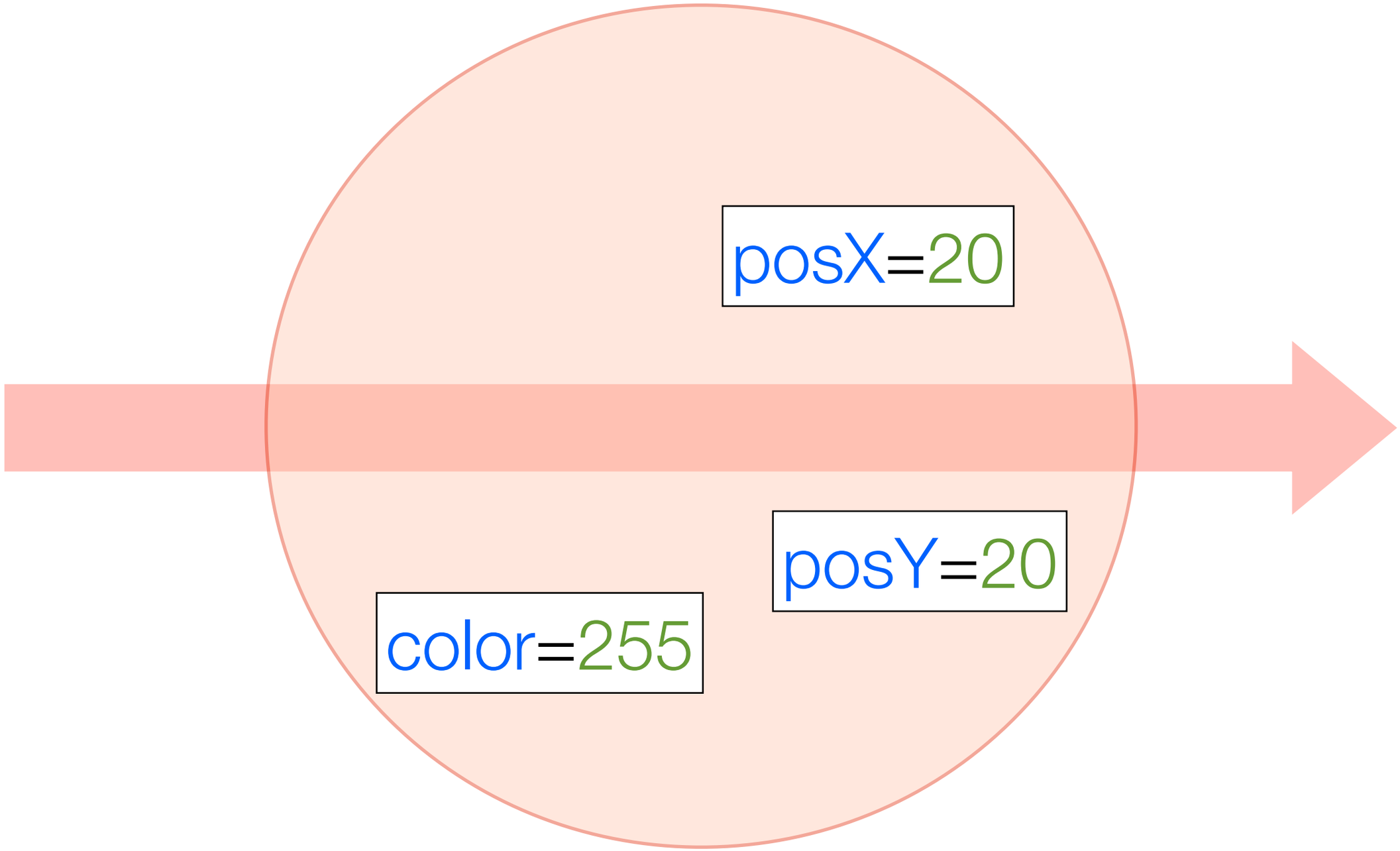
Messages in OOCSI


OOCSI Server
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client


OOCSI Server
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client


OOCSI Server
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client
OOCSI Client

Inside a message

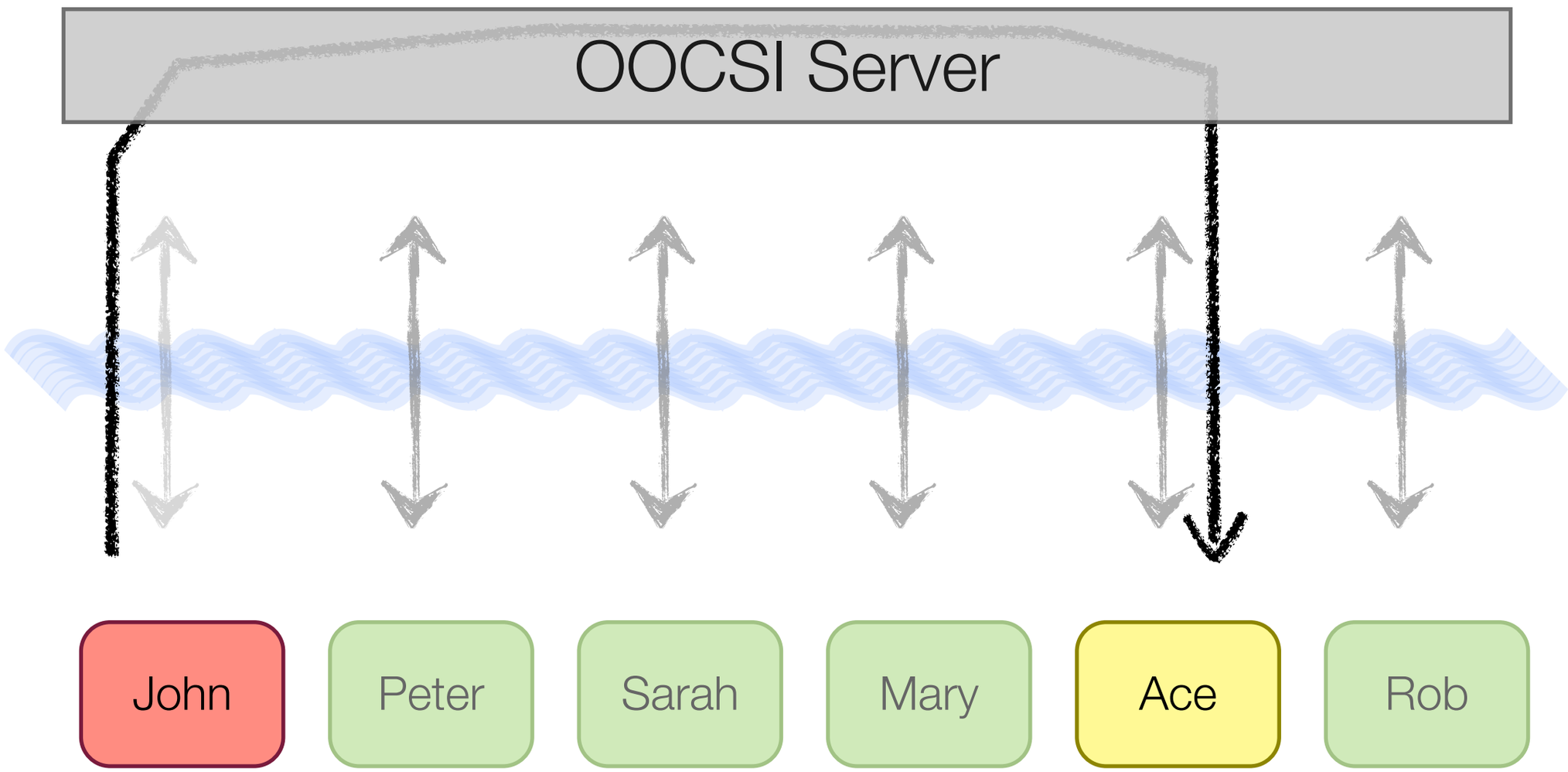


the name of the data

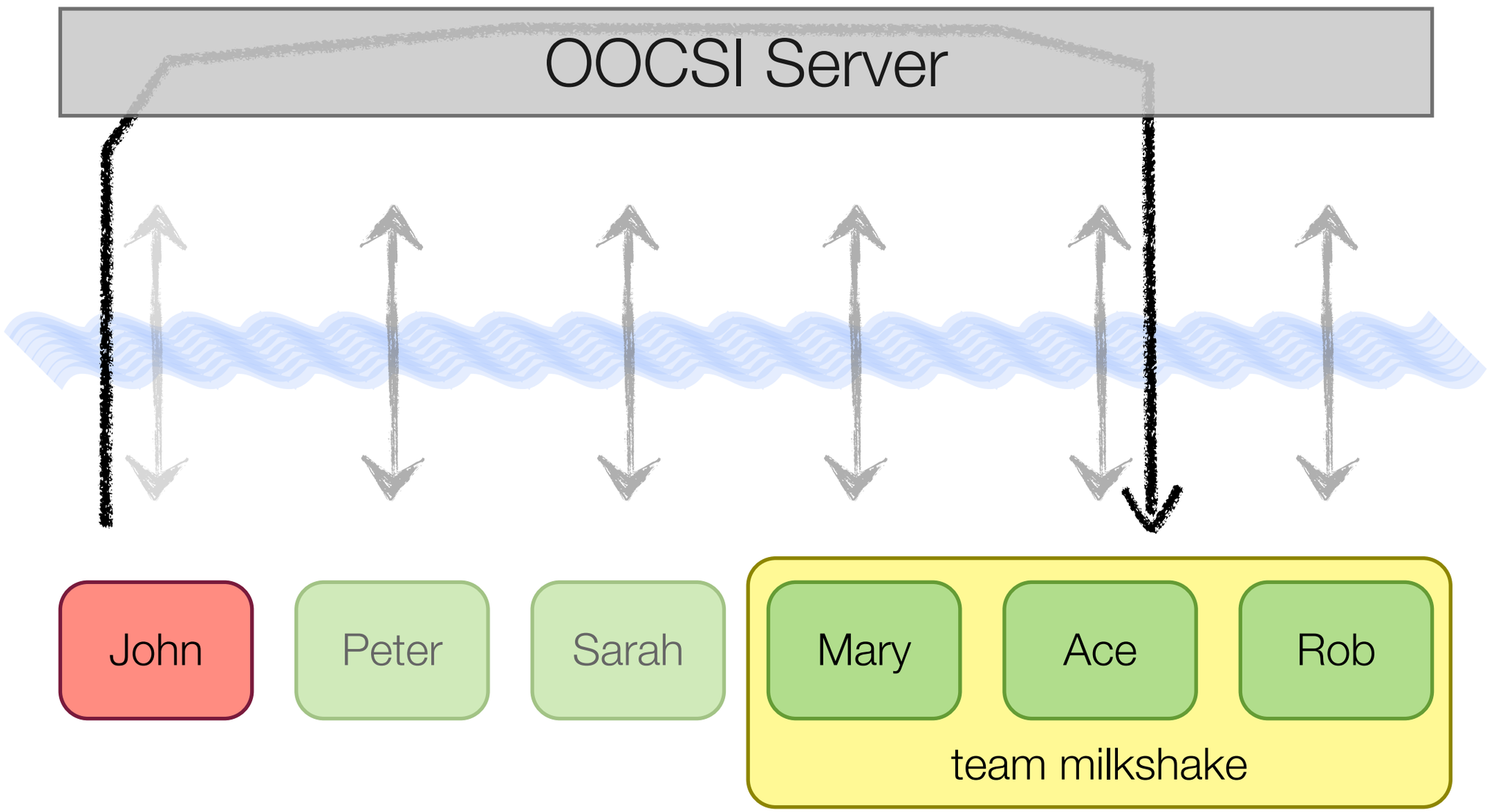
the actual data

key=value

Sending from single client to single client



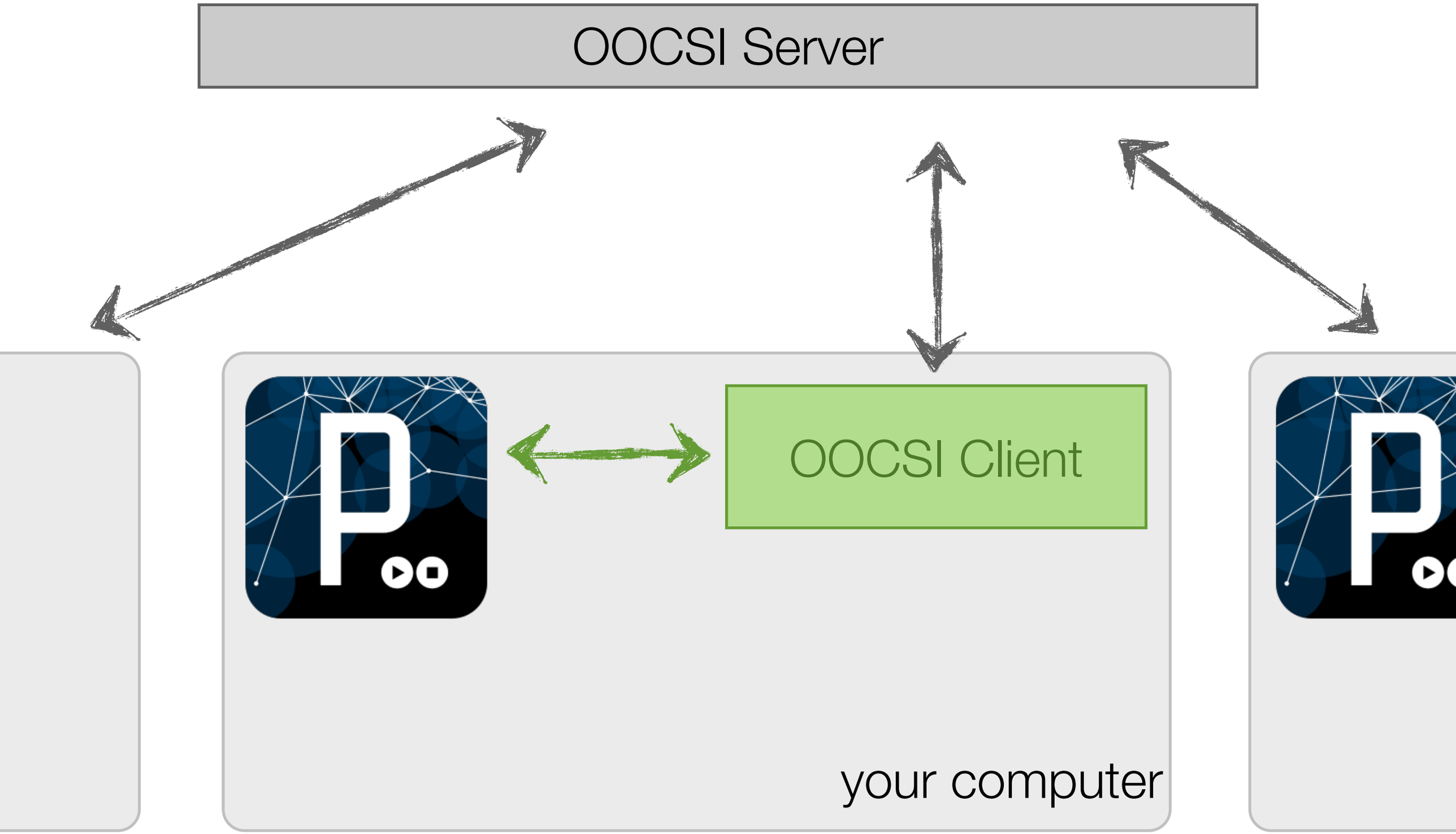
Sending from single client to whole group



Yes, all interesting,
but when can I play
with it myself?!

soon

Setup


OOCSI Server
OOCSI Client
your computer

How to install the OOCSI Processing library

1. Download this: http://bit.ly/1ek8QPA

2. You got Processing installed? (No -> www.processing.org)

3. Close Processing if running

4. Find the directory where it is installed

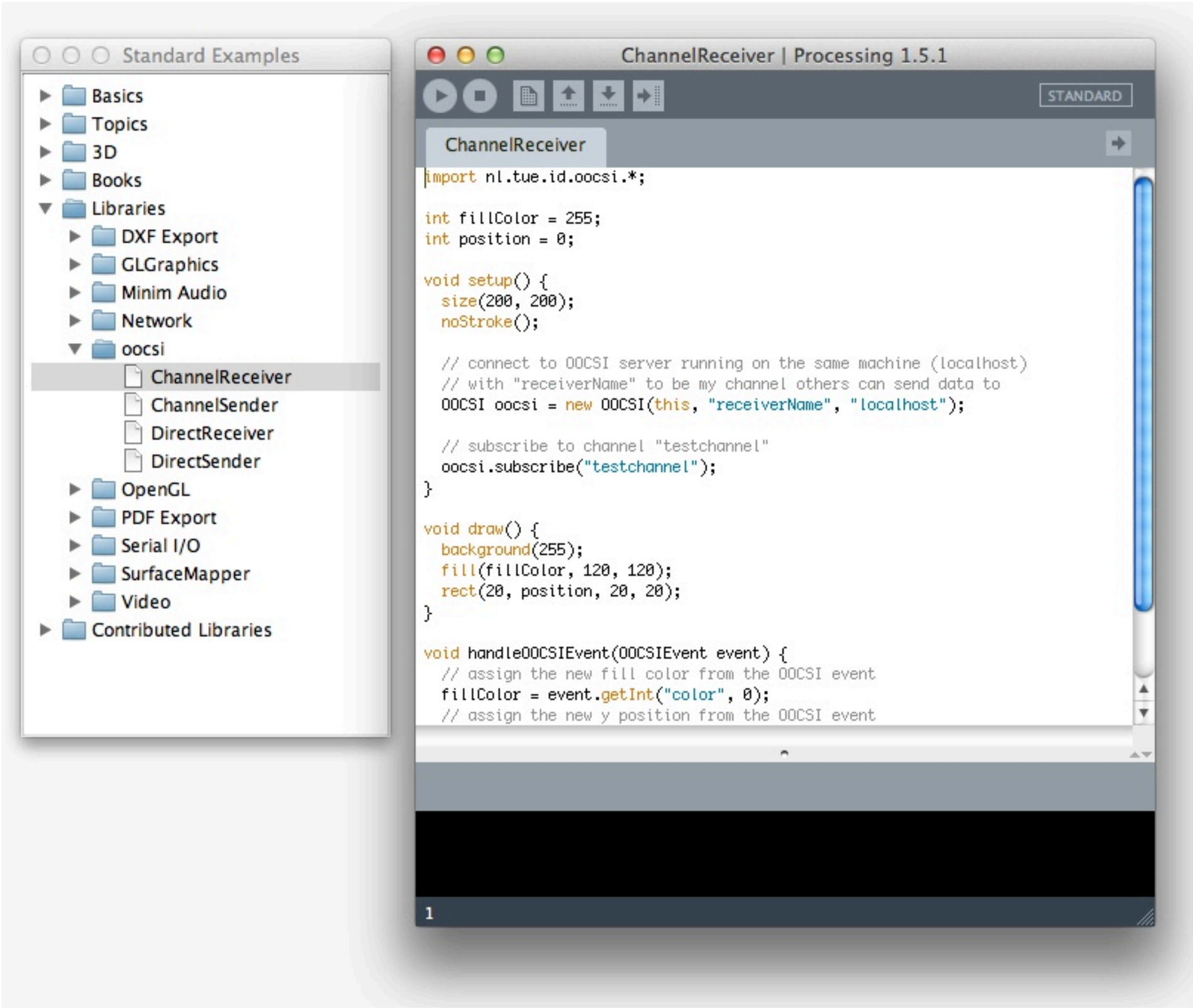
5. Find the subdirectory: modes/java/libraries

6. Extract downloaded zip file to that directory

7. Start Processing

Hands-on

First connection



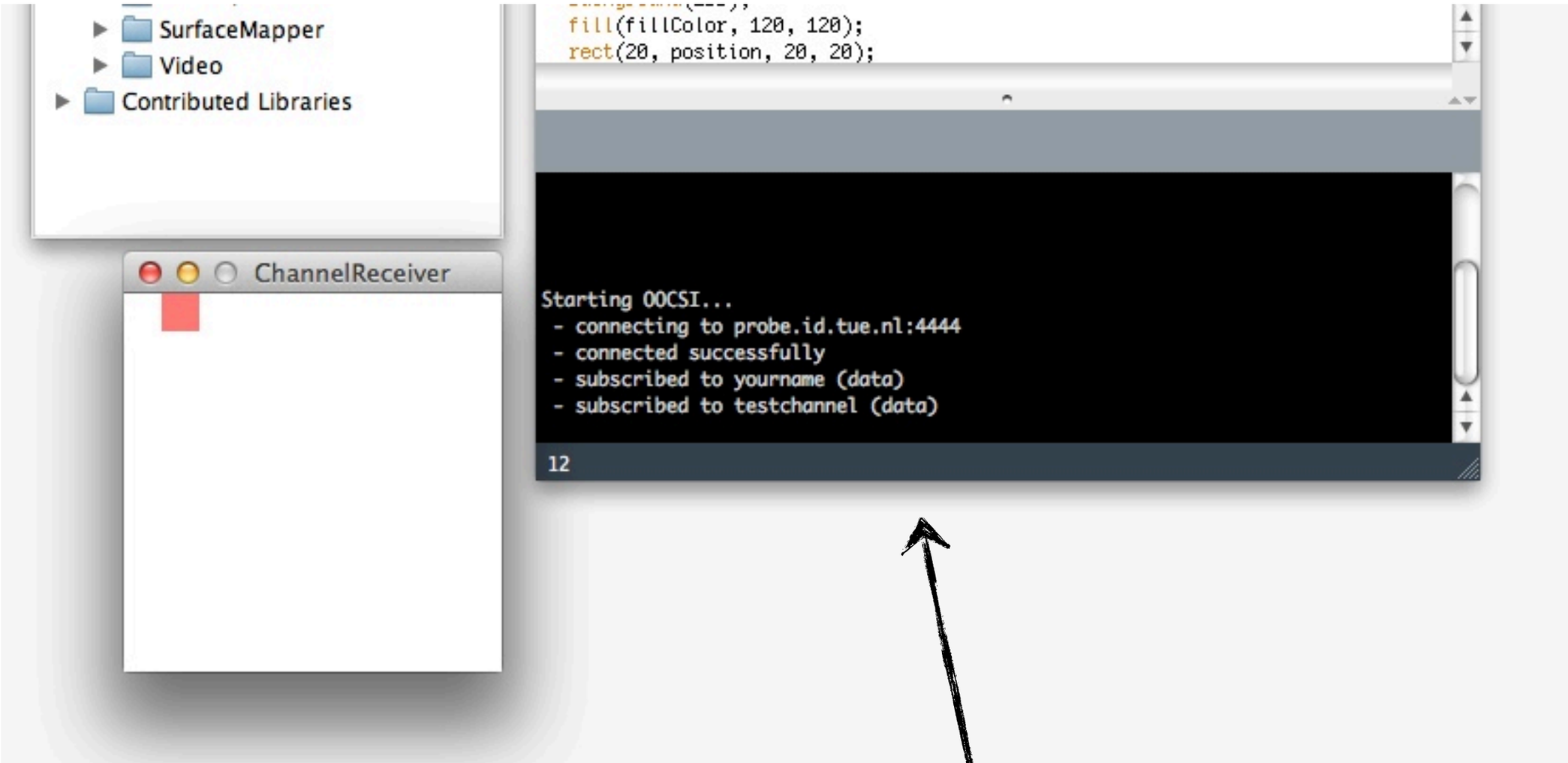
1. Change client name:

```
// connect to OOCSI server running on the same machine (localhost)
// with "receiverName" to be my channel others can send data to
OOCSI oocsi = new OOCSI(this, "yourname", "probe.id.tue.nl");

// subscribe to channel "testchannel"
oocsi.subscribe("testchannel");
```

2. Change server:

```
// connect to OOCSI server running on the same machine (localhost)
// with "receiverName" to be my channel others can send data to
OOCSI oocsi = new OOCSI(this, "yourname", "probe.id.tue.nl");

// subscribe to channel "testchannel"
oocsi.subscribe("testchannel");
```

Running the sketch...



Check out the messages

How does it work?

1 - Connecting to OOCSI

```
import nl.tue.id.oocsi.*;

OOCSI oocsi;

void setup() {
  size(200, 200);
  background(120);
  frameRate(10);

  // connect ot OOCSI server running on the same machine (localhost)
  // with "senderName" to be my channel others can send data to
  oocsi = new OOCSI(this, "John", "localhost");
}
```

- Register with a name, for example: “John” or “group3” or “Dolphin”
- Register at a running OOCSI server, for example: “localhost” or “probe.id.tue.nl”

2 - Subscribing (and receiving)

- Subscribe to messages by adding a method “handleOOCSIEvent”:

```
void handleOOCSIEvent(OOCSIEvent event) {
  // assign the new fill color from the OOCSI event
  fillColor = event.getInt("color", 0);
  // assign the new y position from the OOCSI event
  position = event.getInt("position", 0);
}
```

- Get data out with getInt, getBoolean, getFloat, getString or getObject calls
  - Always give a key to the data, such as “color” or “position”
  - Data types will be int, boolean, float, String or Object respectively
  - Provide default values in case the data does not exist

Standard Examples

- Basics
- Topics
- 3D
- Books
- Libraries
  - DXF Export
  - GLGraphics
  - Minim Audio
  - Network
  - oocsi
    - ChannelReceiver
    - ChannelSender
    - DirectReceiver
    - DirectSender
  - OpenGL
  - PDF Export
  - Serial I/O
  - SurfaceMapper
  - Video
- Contributed Libraries

ChannelSender | Processing 1.5.1

STANDARD

ChannelSender §

```
import nl.tue.id.oocsi.*;

OOCSI oocsi;

void setup() {
  size(200, 200);
  background(120);
  frameRate(10);

  // connect ot OOCSI server running on the same machine (localhost)
  // with "senderName" to be my channel others can send data to
  oocsi = new OOCSI(this, "senderName", "probe.id.tue.nl");
}

void draw() {
  // send to OOCSI ...
  oocsi
    // on channel "testchannel"...
  .channel("testchannel")
    // data labeled "color"...
    .data("color", mouseX)
      // data labeled "position"...
      .data("position", mouseY)
        // send finally
        .send();
}
```

connecting

sending

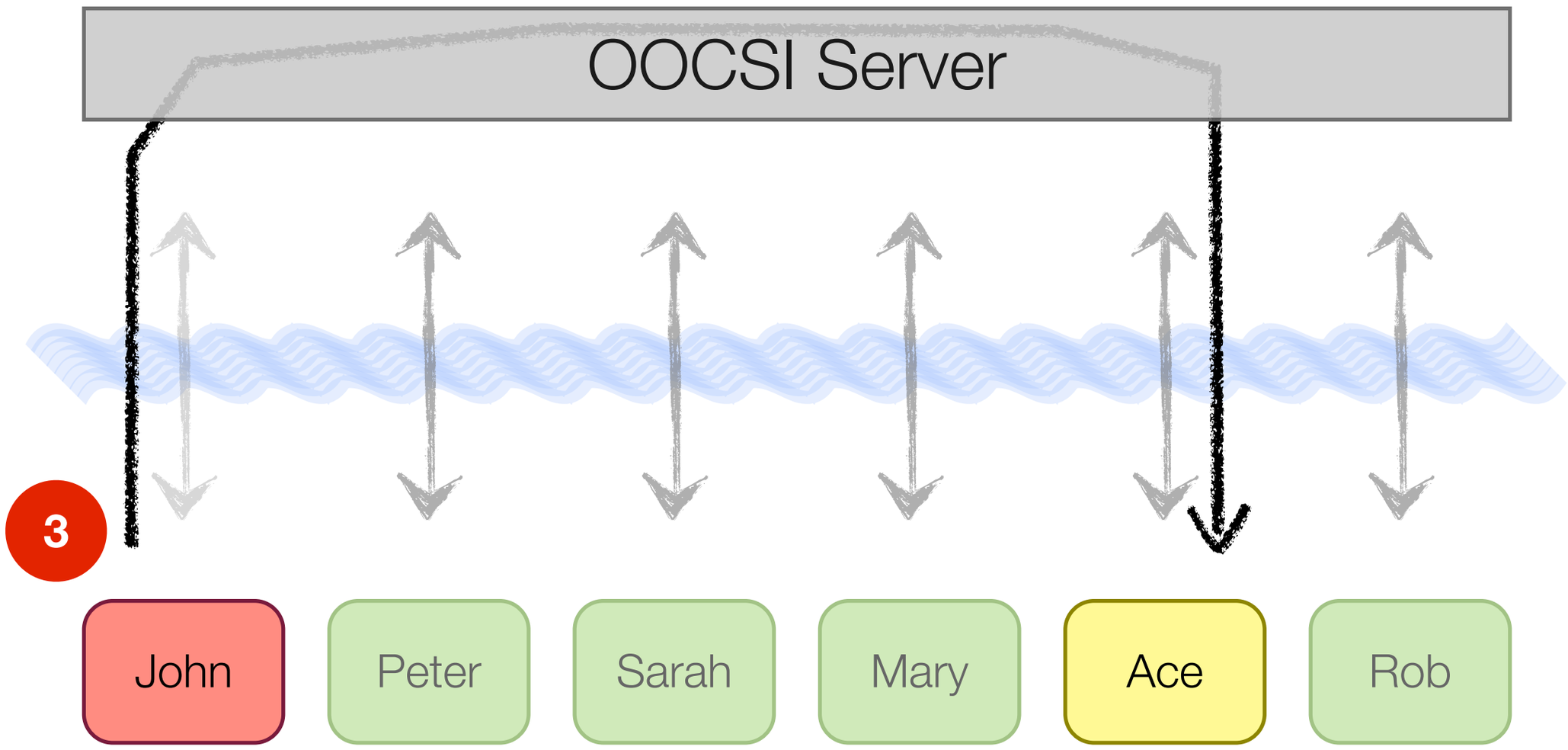
12

3 - Sending data

- Send messages with data by:
  - selecting a channel with `channel("testchannel")` or
  - selecting a single client with `channel("John")`
  - adding data with `data("key", value)`
  - sending it finally with `send()`

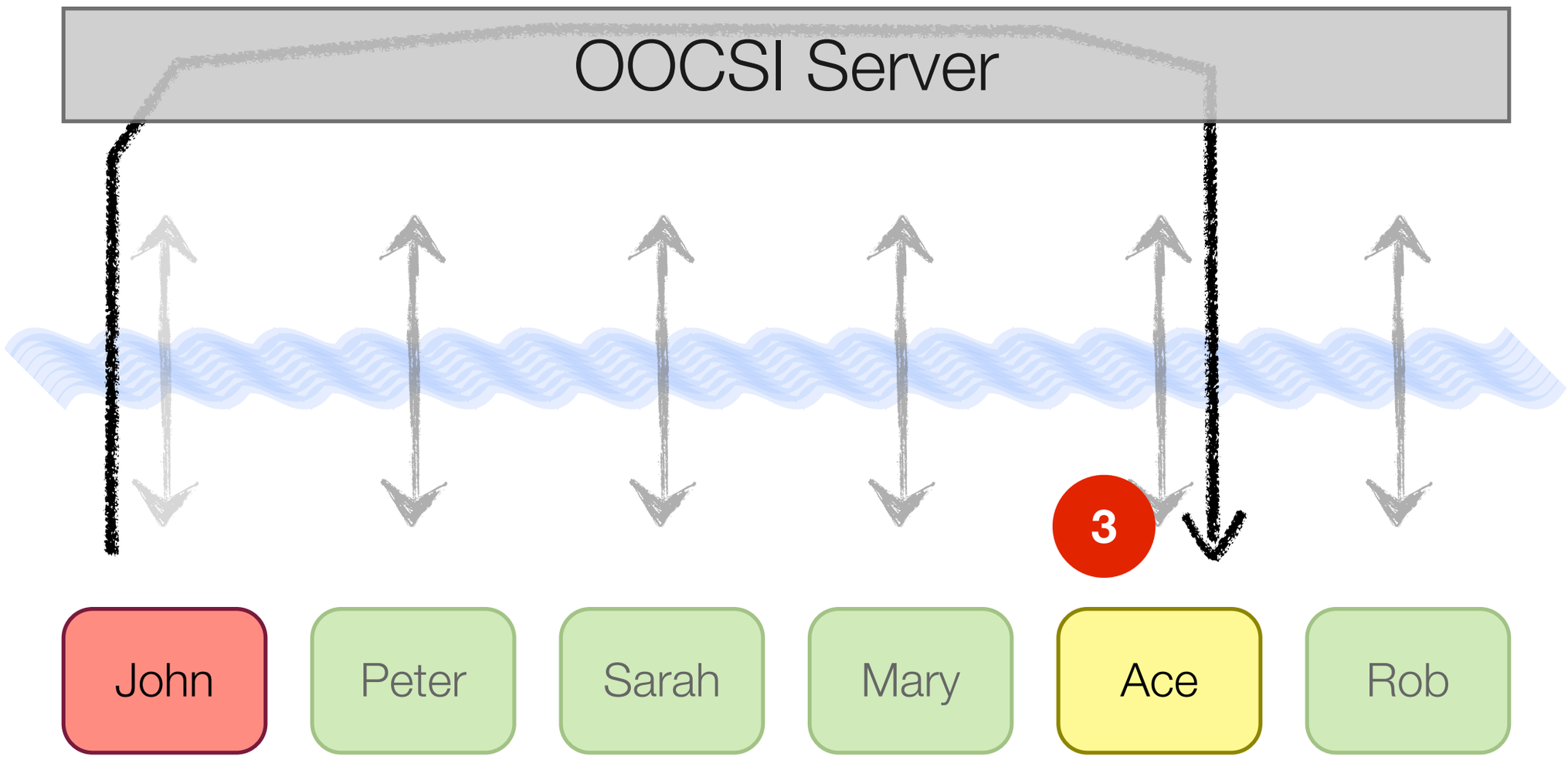
```
// send to OOCSI ...
oocsi
  // on channel "testchannel"...
.channel("testchannel")
  // data labeled "color"...
  .data("color", mouseX)
    // data labeled "position"...
    .data("position", mouseY)
      // send finally
      .send();
```

Sending data: client to client



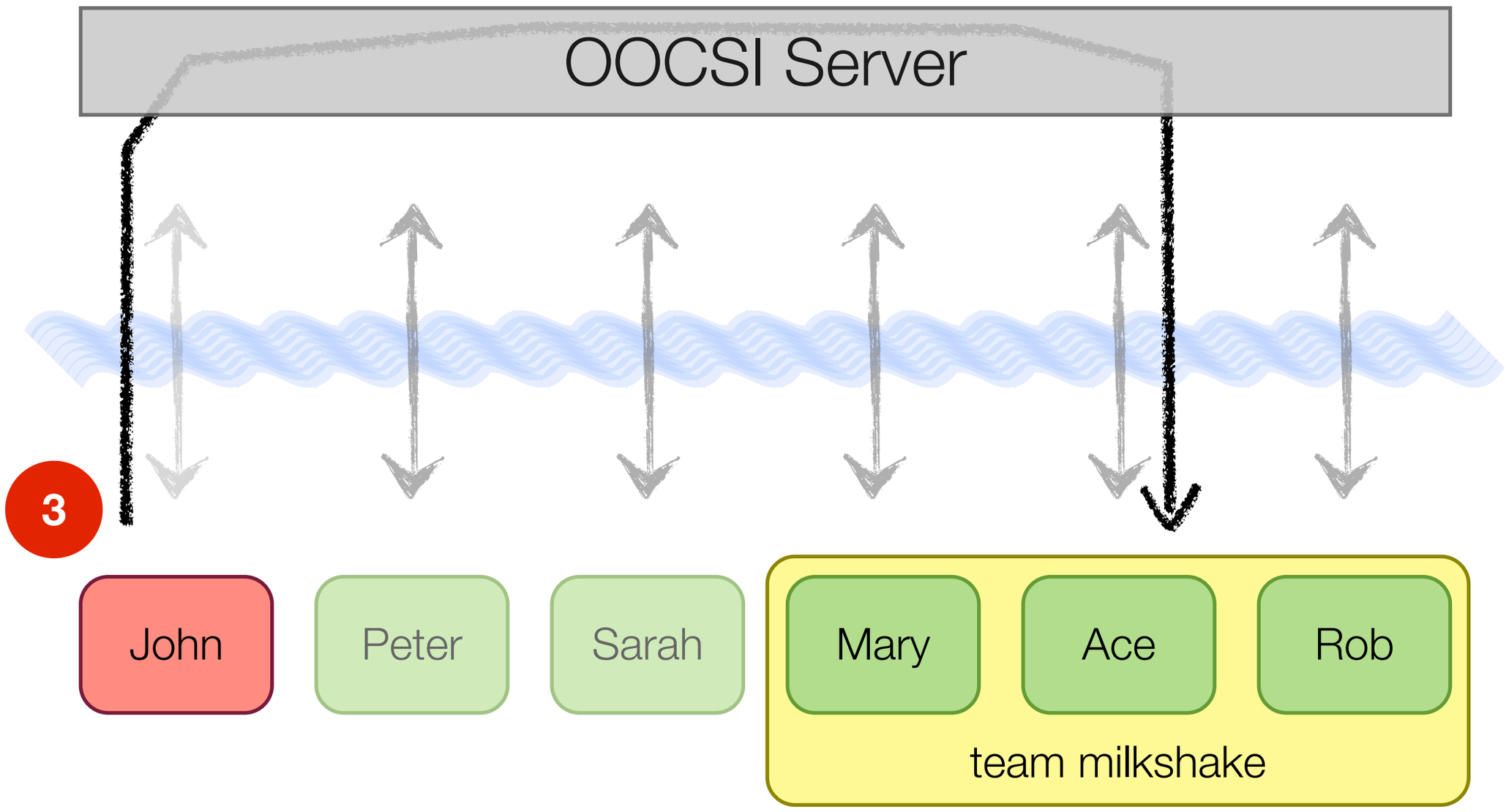

```
oocsi.channel("Ace").data("value", 3).send();
```

Sending data: client to client




```
oocsi.channel("Ace").data("value", 3).send();
```

Sending data: client to group



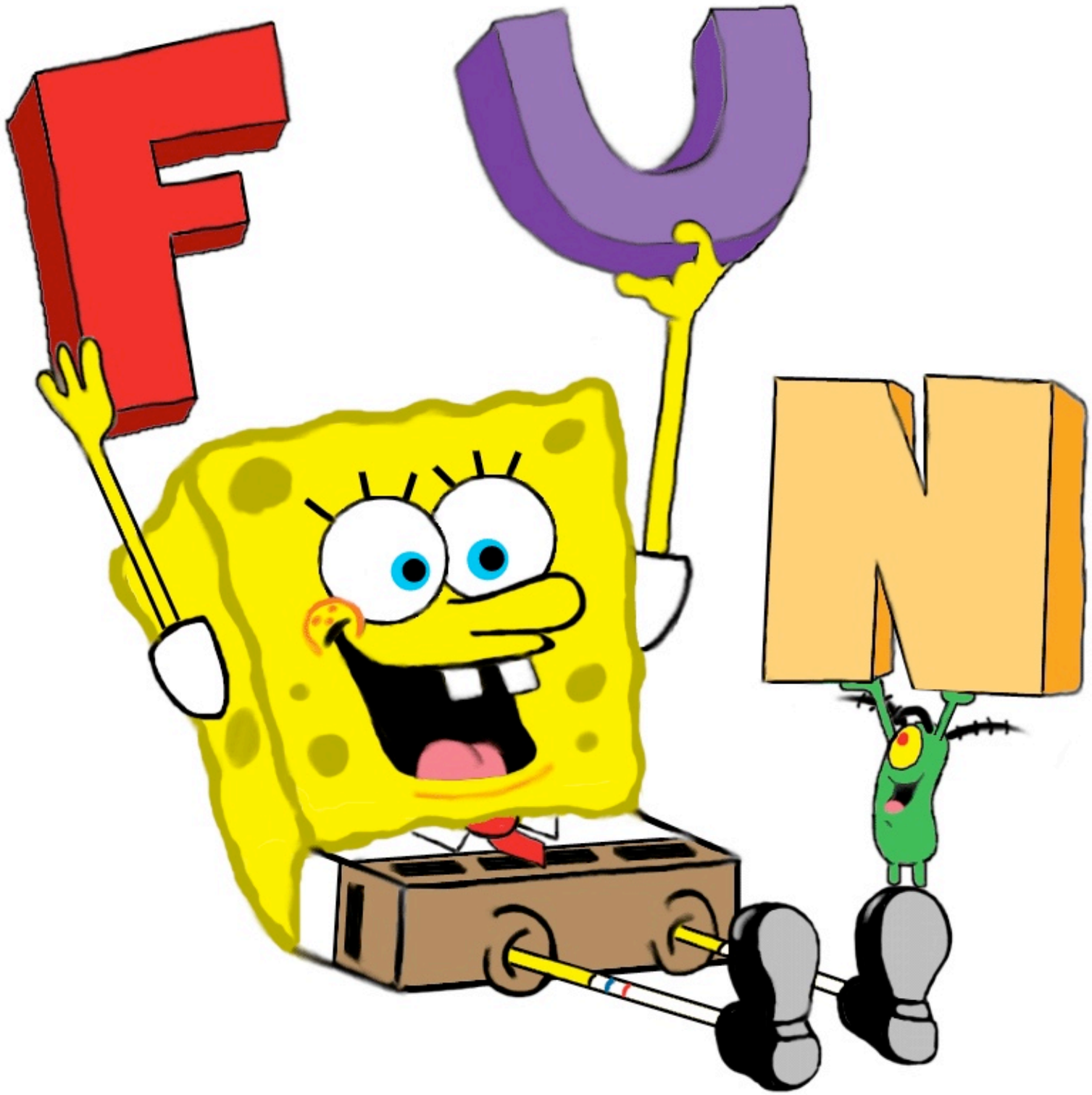

```
oocsi.channel("team milkshake").data("value", 3).send();
```

Remember:

Systems only work with cooperation!

Important hints

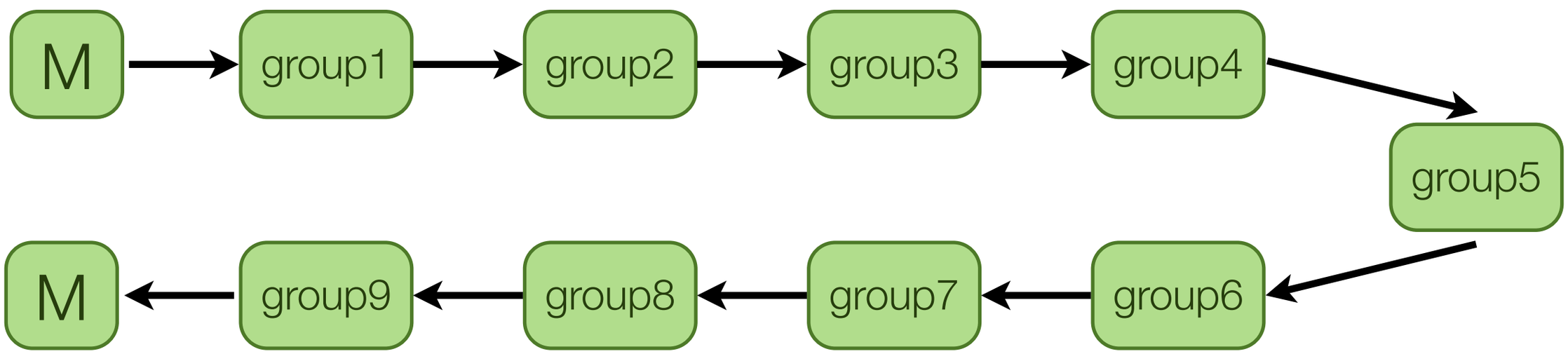
- Connect to the right server (“probe.id.tue.nl”)
- Check output on console
- Sending data
  - Use consistent naming for the data keys
  - Use consistent data types for the data values
  - Agree on what one is sending and how it can be received by others



Now, have some fun with OOCSI!

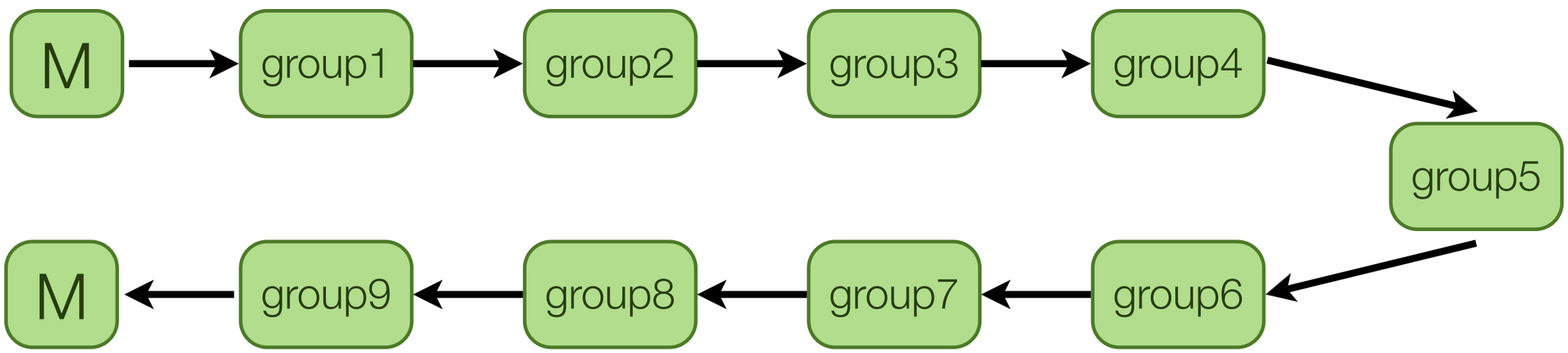
Hands-on: *the chain*

Every group adds 1

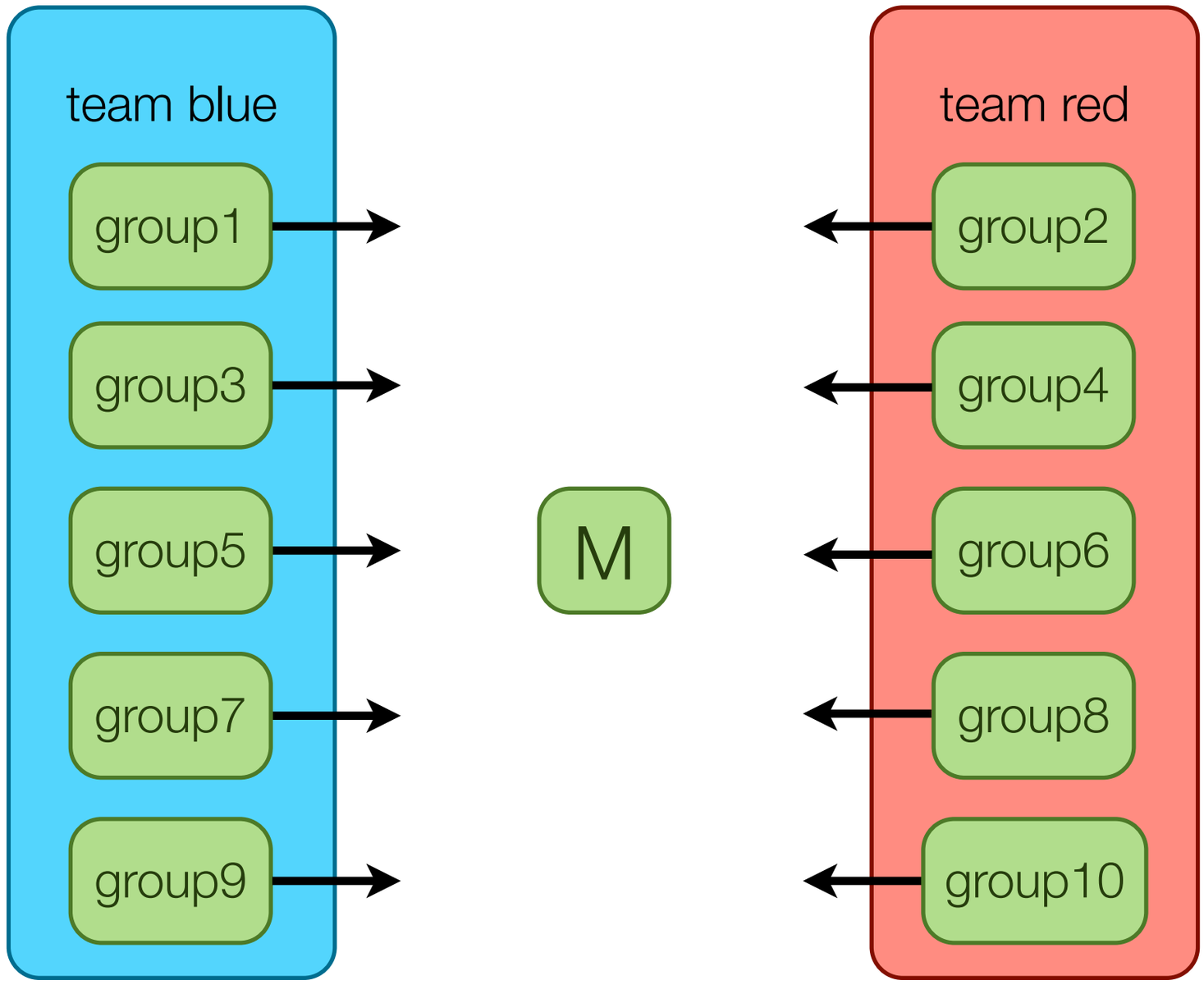


Hands-on: *the chain*

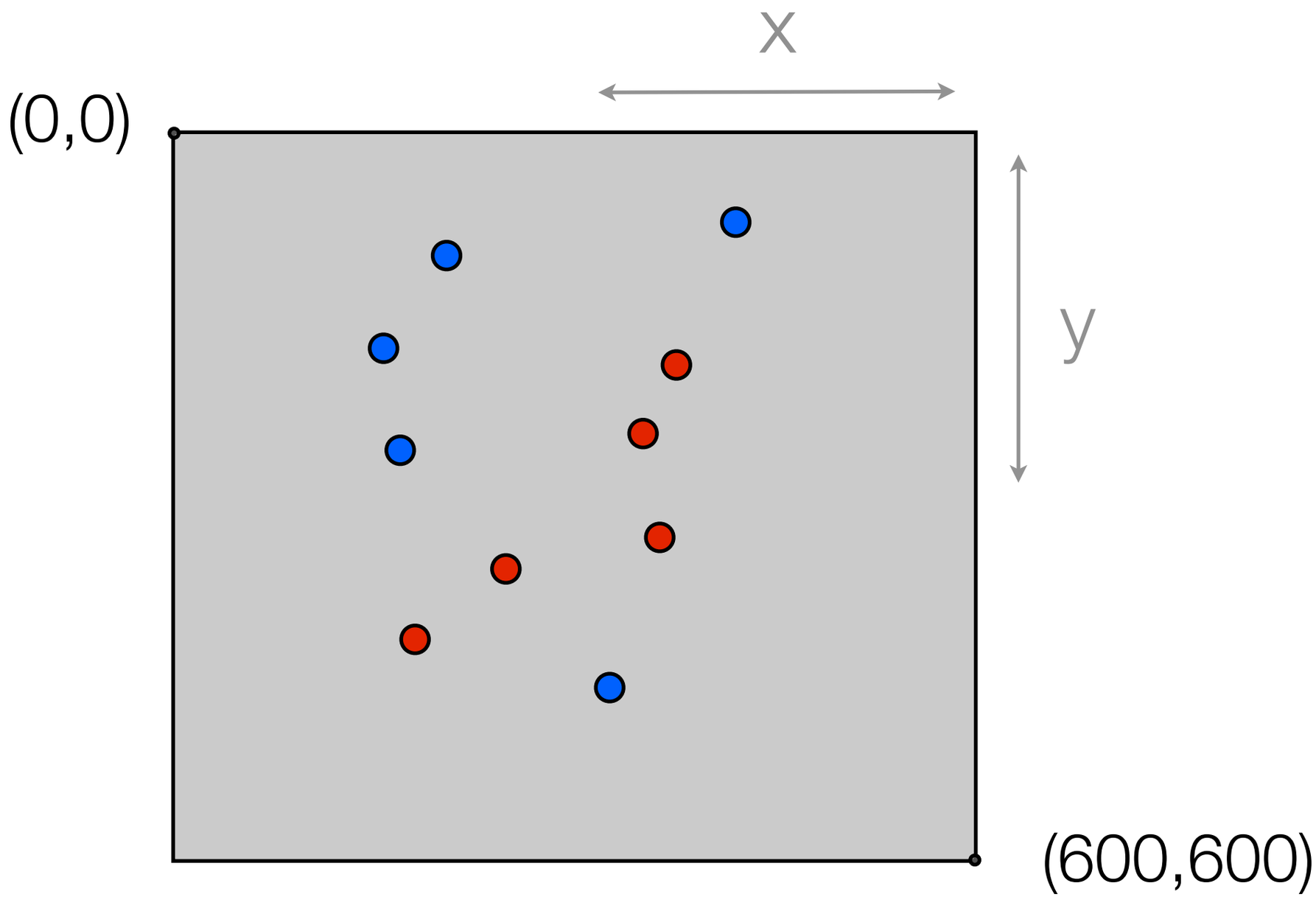
Every group adds group number



Hands-on: *the competition*



Hands-on: *the competition*



Hands-on: *the competition*

Try to stay alive (= don't touch walls)

- Receive: Position

```
void handleOOCSIEvent(OOCSIEvent event) {
  int x = event.getInt("x", 0);
  int y = event.getInt("y", 0);

  ...

}
```

- Send: Acceleration

```
oocsi.channel("M")
  .data("group", 1)
  .data("x", x)
  .data("y", y)
  .data("color", color(<RED>, <GREEN>, <BLUE>))
  .send();
```

Hands-on: *the competition*

Try to stay alive with gravity bursts

- Receive: Position

```
void handleOOCSIEvent(OOCSIEvent event) {
  int x = event.getInt("x", 0);
  int y = event.getInt("y", 0);

  ...

}
```

- Send: Acceleration

```
oocsi.channel("M")
  .data("group", 1)
  .data("x", x)
  .data("y", y)
  .data("color", color(<RED>, <GREEN>, <BLUE>))
  .send();
```

Hands-on: *the competition*

Let your team's balls flock in one place

- Receive: Position

```
void handleOOCSIEvent(OOCSIEvent event) {
  int x = event.getInt("x", 0);
  int y = event.getInt("y", 0);

  ...

}
```

- Send: Acceleration

```
oocsi.channel("M")
  .data("group", 1)
  .data("x", x)
  .data("y", y)
  .data("color", color(<RED>, <GREEN>, <BLUE>))
  .send();
```

Hands-on: *the competition*

Bounce balls into walls.

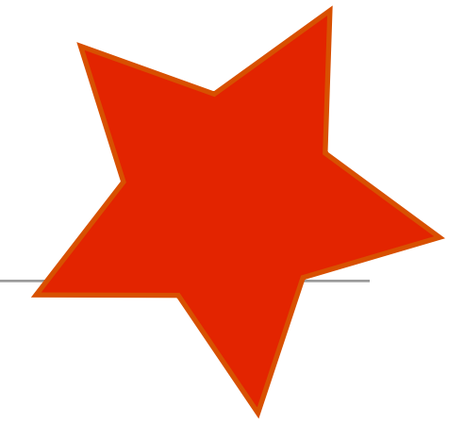
- Receive: Position

```
void handleOOCSIEvent(OOCSIEvent event) {
  int x = event.getInt("x", 0);
  int y = event.getInt("y", 0);

  ...

}
```

- Send: Velocity

```
oocsi.channel("M")
  .data("group", 1)
  .data("x", x)
  .data("y", y)
  .data("color", color(<RED>, <GREEN>, <BLUE>))
  .send();
```

Hands-on: *the collab*



Collaboratively paint the canvas

- Send: Position and color

```
oocsi
  .channel("M")
  .data("x", 100)
    .data("y", 20)
      .data("colorR", 100)
        .data("colorG", 180)
          .data("colorB", 120)
            .send();
```

Finally: *the code*

- If you want to
  - ...know how it works
  - ...extend it
- Check out: GitHub.com
  - Processing client code: https://github.com/iddi/oocsi-processing
  - Server code: https://github.com/iddi/oocsi