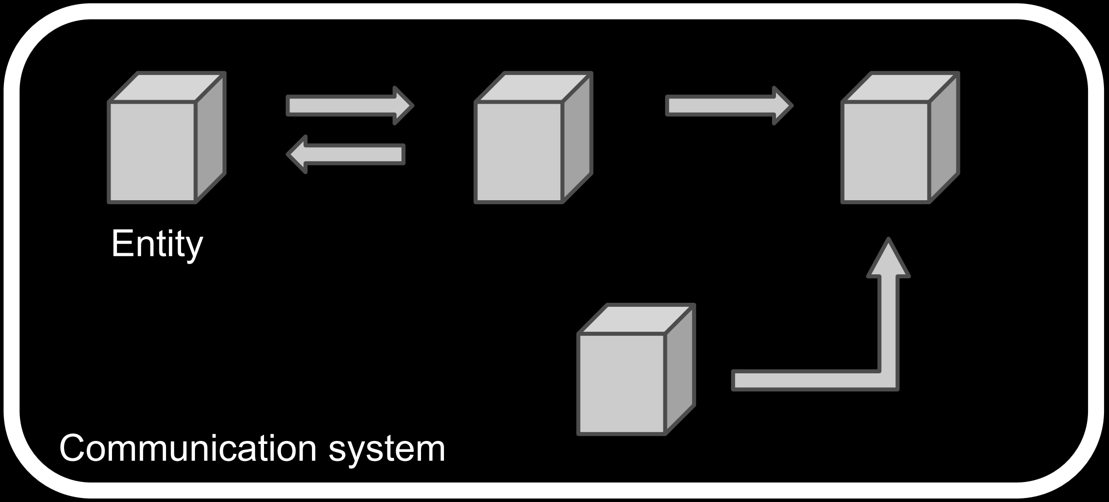
Communication Protocols

Make things communicate

Definition

In telecommunications, a communications protocol is a system of rules that allow two or more entities of a communications system to transmit information.



DMX

DMX 512 aka DMX

Stands for Digital MultipleX and 512 is the number of addresses/channels

Lighting industry standard

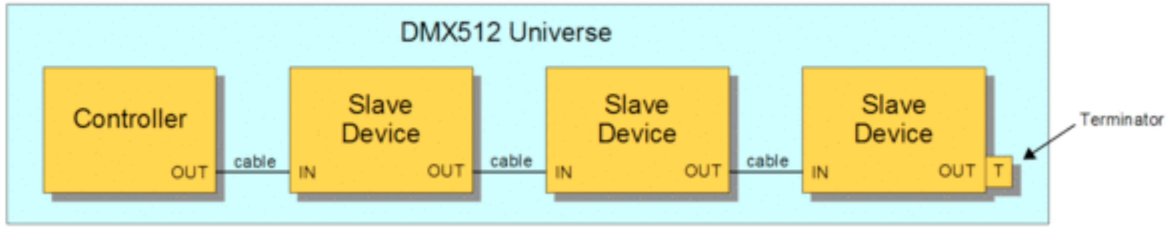
Send commands to light fixtures in real time

Might be used for other purpose than lighting

Network topology

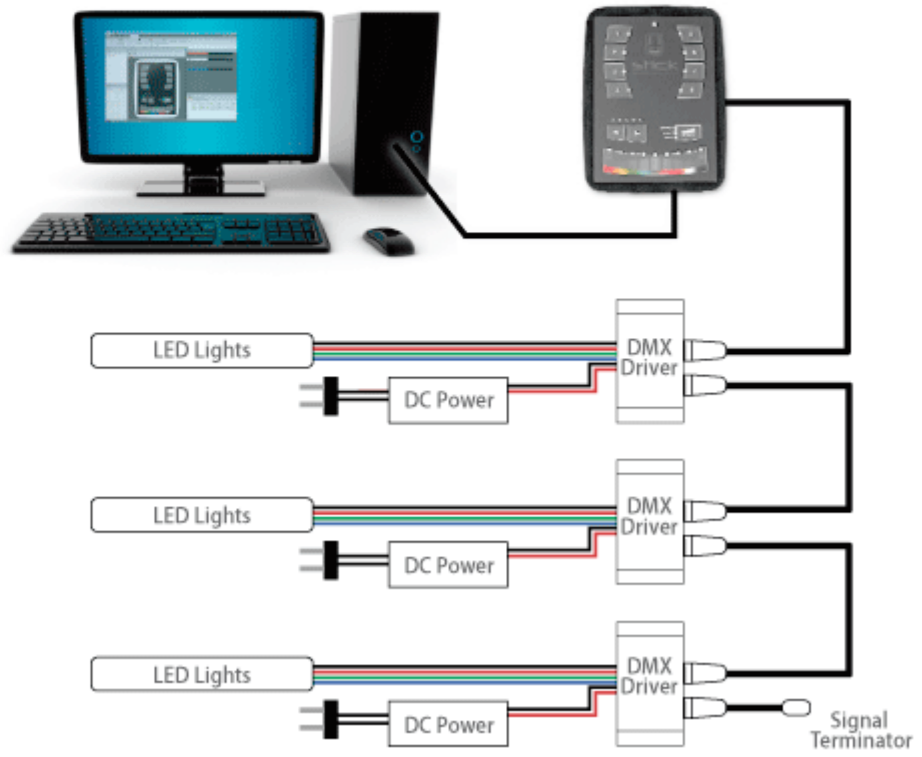
A single controller can manage several universes

Each universe can have to 512 channels splitted across the chained devices
A universe is made of a single cable/line ending with a terminator

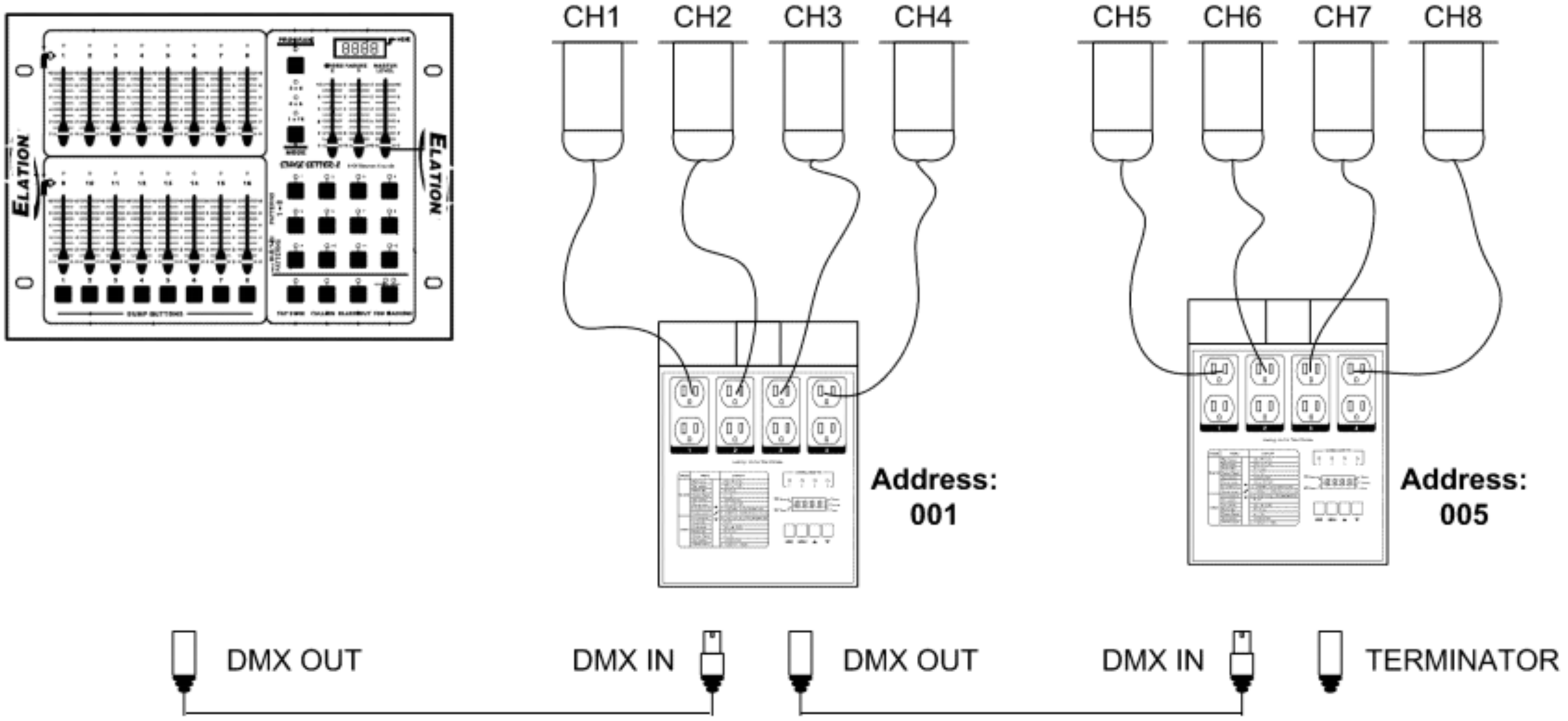


source : Wikipedia

Example


LED Lights
DMX Driver
DC Power
LED Lights
DMX Driver
DC Power
LED Lights
DMX Driver
DC Power
Signal Terminator

Channels / Split over the chain



Practice

Create a virtual DMX chain

Split a DMX message across the chain

Software

Processing library: DMXP512 (for DMXPro and Lanbox)

http://motscousus.com/stuff/2011-01_dmxP512/

openFrameworks addon : ofxDMX (targets the Enttec DMXPro)

https://github.com/kylemcdonald/ofxDmx

OSC

Open Sound Control

OSC - Open Sound Control

OSC is a content format by Adrian Freed and Matt Wright

Developed at the CNMAT (Center for New Music and Audio Technologies)

Originally intended for sharing music performance data between :

- musical instruments
- computers
- other multimedia devices

Alternative to MIDI

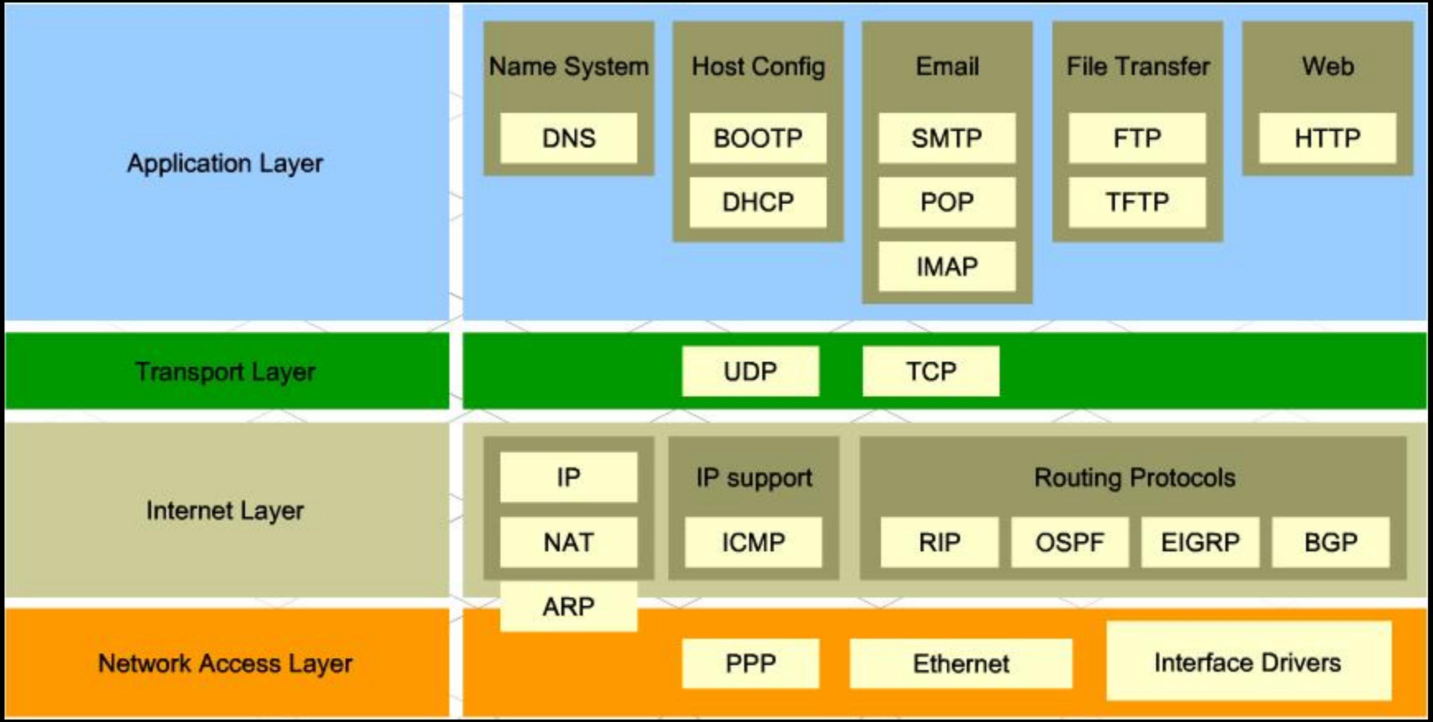
More flexibility in the kinds of data one can send over the wire

Transported across the Internet and within home studio subnets using UDP/IP (or TCP/IP), Ethernet

The SLIP protocol can also be used through USB between gestural controllers (ie: with Arduino)

Enables new applications that can communicate with each other at a high level

Communication Layers



OSC Message content

Address pattern

Hierarchical name space, Unix or URL like (ie: /OSCcontainerA/OSCcontainerB/methodName)

Type tag

Compact string representation of the argument types (ie: iiffsF)

Arguments

Binary form with 4-byte (32 bits) alignment

Contiguous sequence of the binary representations of each argument (each type provided by the tag string)

Optional time tag

Note : when receiving an OSC Messages the OSC Server should invoke the corresponding OSC methods immediately

OSC Bundle content

Contains 1 or more OSC Messages

OR

Encapsulates other OSC Bundles

Note : encapsulated OSC Bundles must have a timestamp equal or greater than their parent

Contains a Timestamp

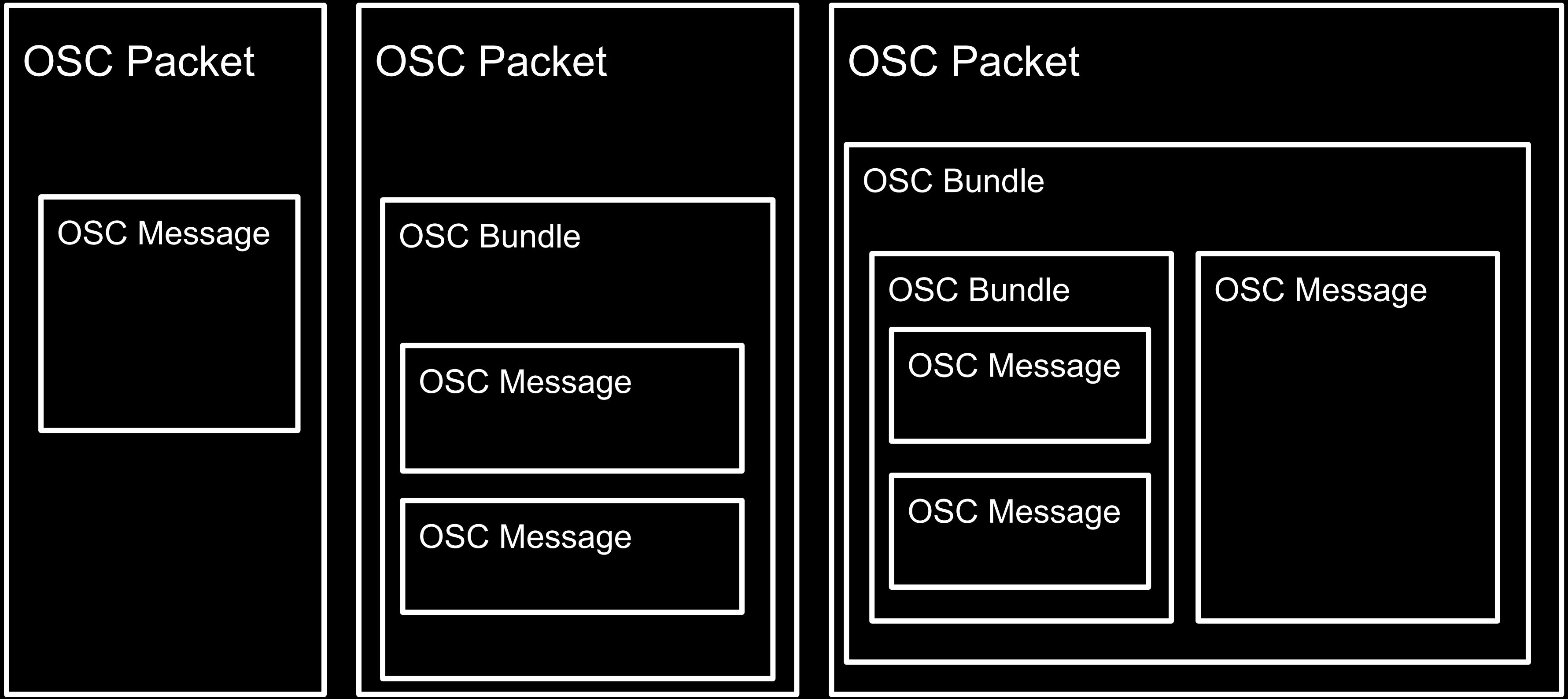
Notes :

When receiving an OSC Bundle, the OSC Server should invoke the corresponding OSC methods based on the timestamp :

If less or equals the current time, then execute immediately (except if the Server is configured to drop belated messages)

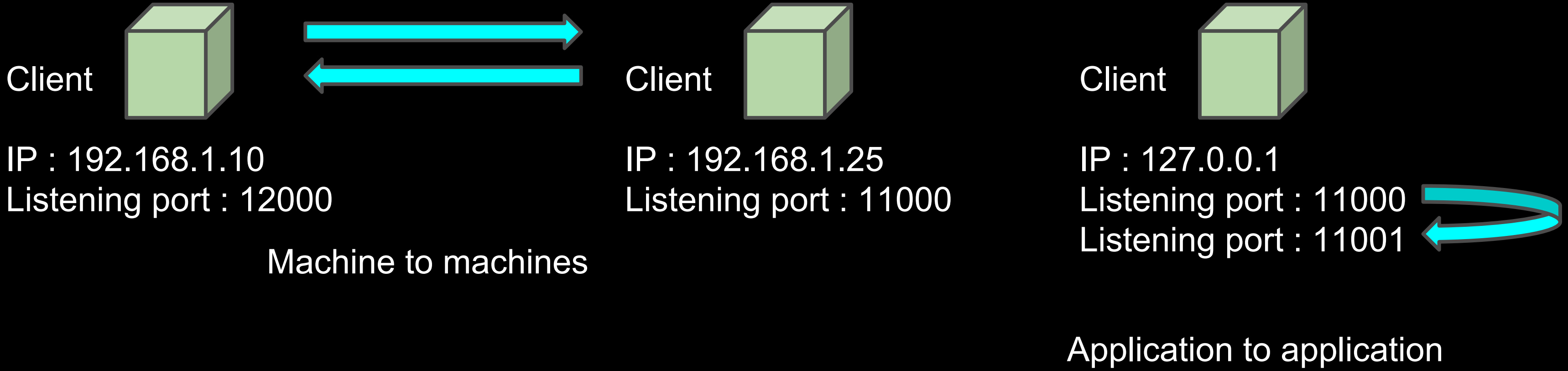
If later, keep the OSC Bundle until the specified time then interpret it

OSC Packet


OSC Packet
OSC Message
OSC Packet
OSC Bundle
OSC Message
OSC Message
OSC Packet
OSC Bundle
OSC Bundle
OSC Message
OSC Message
OSC Message

Exchanging OSC messages

Send message



Define your logic

Define your commands and the type of expected values (ie: integer, float, ...)

Example :

Address patterns :

- /color/setBackground
- /color/setFill

Type tag for a grayscale color is i (1 integer)

- i

or for an RGB color is iii (3 integers)

- i
- i
- i

Sending an OSC Message

Prerequisite : know the destination (server) address and port number

Steps :

- Create a message
- Set its address pattern (ie: /color/setBackground)
- Add data (ie: 255, 0 and 127)
- Send it to the server

Receiving an OSC Message

Check the address pattern

Is it “/color/setBackground” ?

Is it “/color/setFill” ?

If so, check the type tag

Is it “i” ?

Is it “iii” ?

Depending on those 2 values :

- get the information from the message arguments
- invoke the appropriate method

Practice

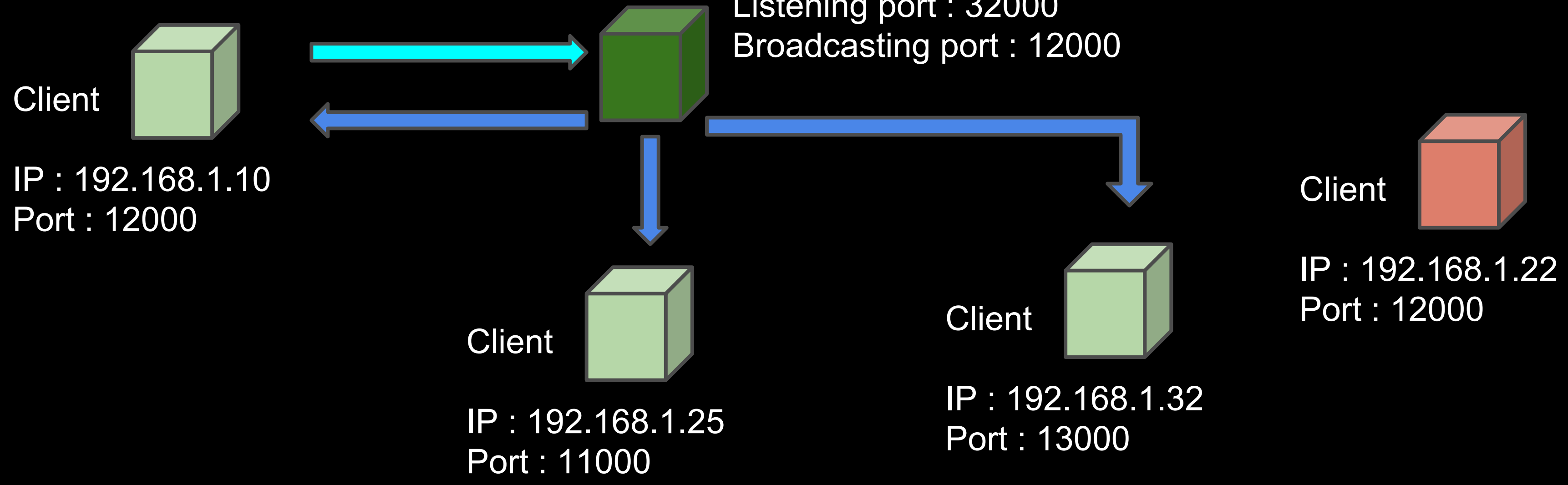
Use sheets of papers and envelopes to mimic the way the OSC protocol works

Send and receive an OSC message

Send and receive an OSC bundle

Broadcasting OSC messages

Step 1: send message
Step 2: broadcast message



Practice

Use sheets of papers and envelopes to mimic the way an OSC broadcasting server works

Connect and disconnect clients

Receive and broadcast OSC packets

Links / Readings

OpenSoundControl

http://opensoundcontrol.org

OSI (Open Systems Interconnection) model

https://en.wikipedia.org/wiki/OSI_model

Open Sound Control: Constraints and Limitations

http://publications.smartcontroller.com.au/Nime2008.pdf

Software

Processing library: oscP5

openFrameworks addon: ofxThreadedOSCReceiver

Arduino : OSCuino

Smartphones : touchOSC (+ touchOSC editor)

also exists for many other languages/platforms (ie: Python, ...)

MIDI

Musical Instrument Digital Interface

Technical standard from 1983

Describes

- a protocol
- digital interface
- connectors

Allows devices to connect and communicate:

- electronic musical instruments
- computers
- other related devices

Connection via USB or MIDI plugs/cables

MIDI Message content

MIDI carries various event messages

- pitch and velocity,
- volume,
- vibrato,
- audio panning,
- cues,
- clock signals (sync tempo between devices)

MIDI message

Up to 16 channels to distinguish devices

Values range from 0 to 127 (1 byte)

Messages are sent on 3 bytes

1 status byte (type of message)

2 data bytes

Channel Voice messages

Usually send by most simple controllers (ie: keyboards, pads)

- Note ON : pitch, velocity
- Note OFF : pitch, velocity = 127
- Control Change :value, velocity
- Program Change : value
- Channel Pressure (After-touch) : value
- Pitch Bend : value (combination of the 2 bytes)

Other type of MIDI messages

Channel messages

ie: all notes OFF

System Common messages

ie: MIDI time code

System Real Time messages :

Used for sync, using MIDI clock and Active Sensing (= ping)

SysEx messages

System Exclusive messages

- Proprietary messages from manufacturer to their devices
- Allows modularity and specific functions
- SysEx ID allows same model devices to be addressed individually

Readings / Links

MIDI Manufacturers Association

http://www.midi.org/

MIDI Specifications

http://www.midi.org/techspecs/midimessages.php

Tools

Audio MIDI setup (Mac OSX)

MIDI Monitor (Mac OSX)

Software

Processing 2.x library: rwmidi

Processing 3.x library: rwmidi-revival

openFrameworks library: ofxMIDI