Patching Music Together
Collaborative Live Coding in Pure Data

lOhannes m zmölnig
zmoelnig@iem.at
Institute of Electronic Music and Acoustics
University of Music and Dramatic Arts, Graz

- Live Coding
- Motivation/Aims
- existing Pd-based environments
- multiPd
  - architecture
  - implementation
  - problems
- Conclusions

- People writing Software in Real Time
  - not „playing“ (parameterising) ready-made instruments
  - language = instrument
- Software Art / Code Art
  - high abstraction
  - low level interaction
- young performance practice, improvisation

- Quest for Elegance
  - condensed, recursive solutions
  - LISP, Scheme,...
- Environments
  - text-based
    - SuperCollider3/JITlib
    - ChuCK
    - fluxus, impromptu, ...
  - graphical
    - Pd, Max/MSP,... ?

- Code Literacy
  - audience is expected to understand how algorithms are expressed
  - to read algorithms one must read code
  - SC3: what is currently being executed?
  - (((((((LISP)))))))
- ChuCK: meta-visualization
- graphical environments
  - simple metaphor (60's style modular patching)

Collaboration

- environments allow interaction between users on high level („analog“, data-sharing on pre-defined bus,...)
- lower level interaction on the level of code (XP: collaborative development of algorithms)
  - SC3/JITlib: sending code snippets to peers
  - impromptu: ???
  - collaborative editors (Gobby, SubEthaEdit)

blind-data performance series

- several people working together on one patch
  - 2 pairs (audio/video)
- communication via coding (no talking)
- simple collaborative „editor“
  - attach 2 keyboards,... to 1 computer
  - scalability ??
- ideally no hacked environment (Pd)
  - upgrading,...

existing environments in Pd

- serendiPd (h.-c.&j. steiner)
- netpd (r.häfeli)
- DesireData (?)
  - does it work by now?

netPd

- collaboration = tweaking pre-made instruments together
- not really collaborative patching (coding)
- active community (alive!)
- constraints
  - well-defined patch environment needed
    - [_creator], [_chat]
  - shared patches ought to be „netPd-aware“
  - sync between clients not guaranteed
    - (people can have patches not shared)

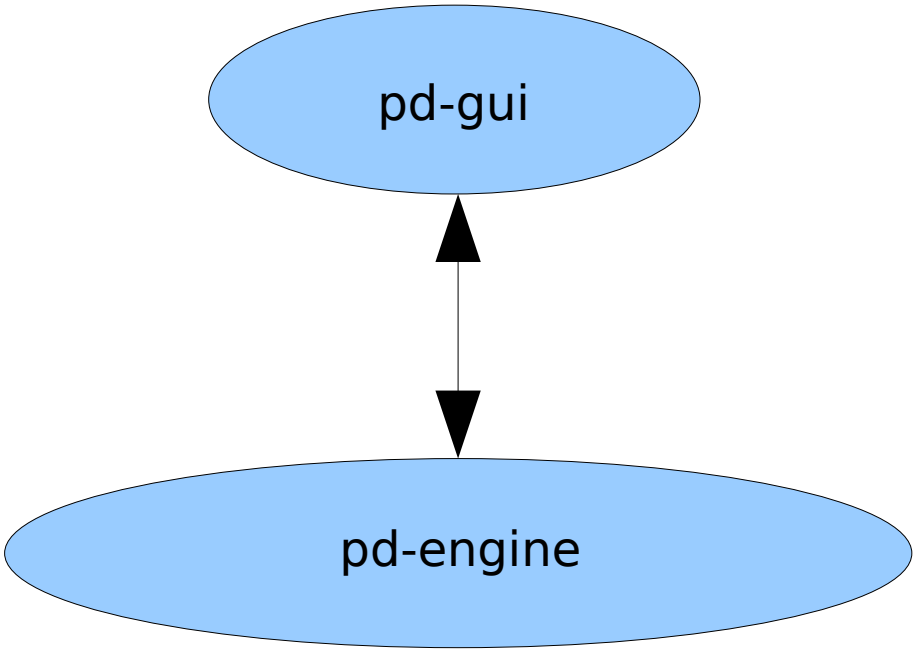
serendiPd

- that's what we want!
- constraints
  - well-defined patch environment needed
    - [serendiPd-gui]
    - client-side dependencies (tot,...)
  - sharing of a single patch
    - sync between clients not guaranteed
- not actively maintained any more

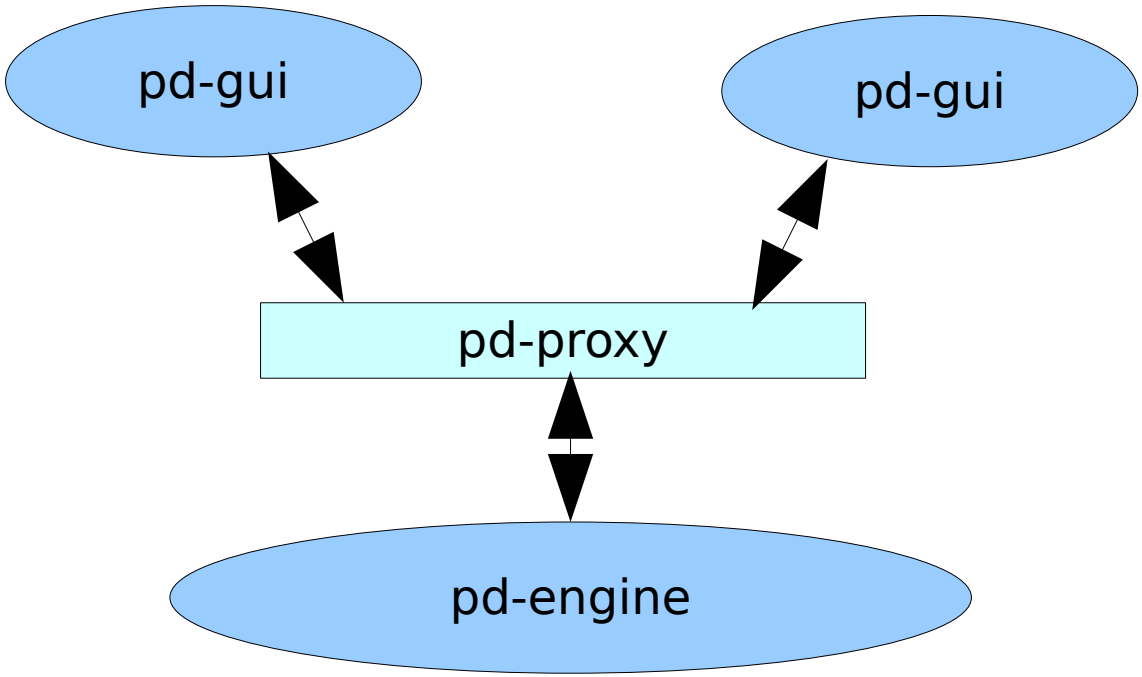
yet another approach: multiPd

- shares idea of collaboration with serendiPd
  - immediate feedback of editing to all participants
- „server“ implementation
  - ALL dependencies on server-side
- NO dependencies on client-side
  - no patches/externals needed for connection
- constraints
  - all clients run compatible („same“) version of Pd (-gui)

use (existing) networking connection between pd-gui and pd-engine

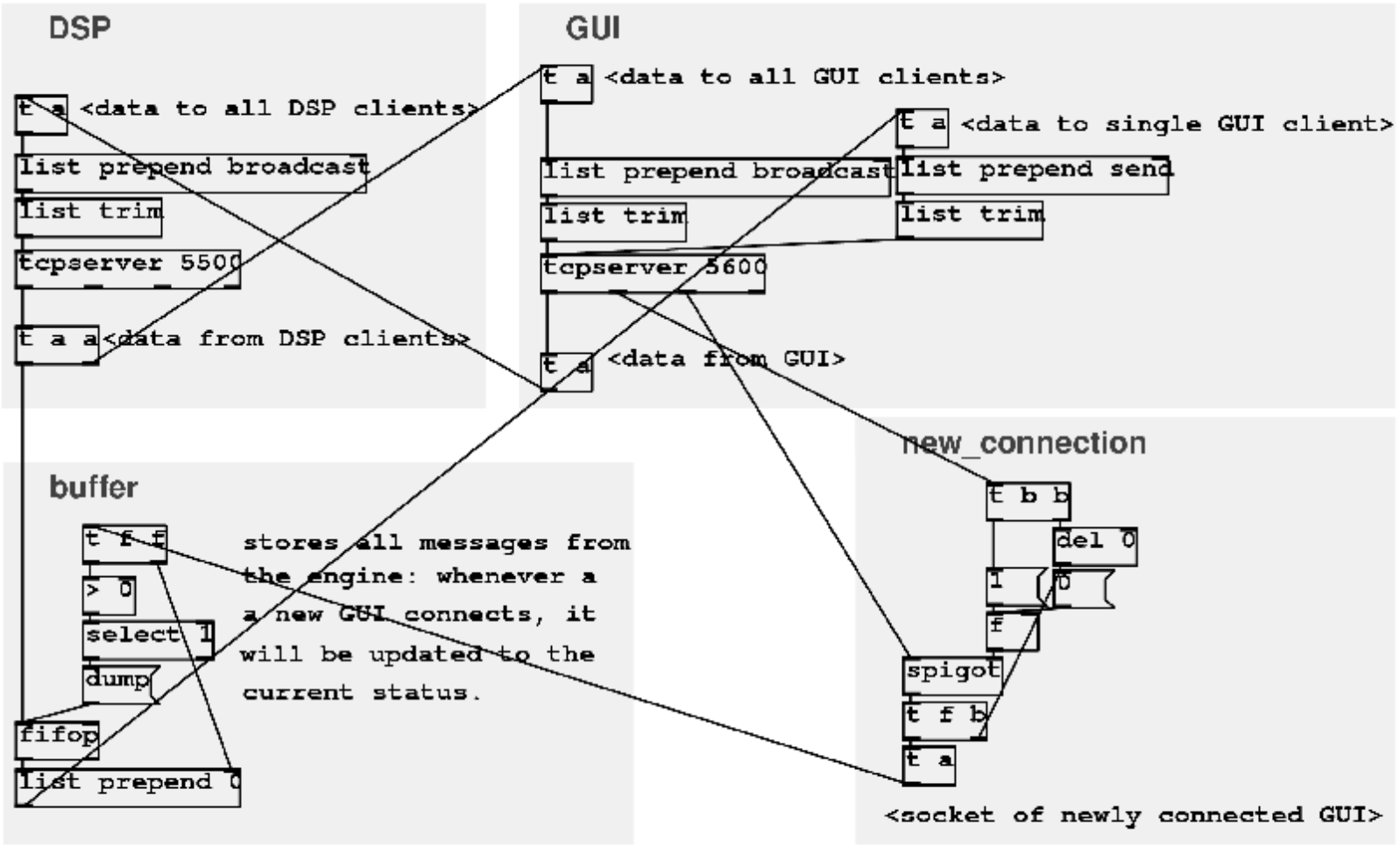


- **pd-gui**s connect to proxy (server)
- **pd-engine** connects to proxy
- **proxy** distributes data between pd-guis and pd-engine(s)



pd-based implementation

- bi-directional TCP/IP connection
  - [netserver] (Martin Peach)
- raw string handling
  - pdstring (Bryan Jurish)
- other stuff
  - zexy


DSP
t a <data to all DSP clients>
list prepend broadcast
list trim
tcpserver 5500
t a a <data from DSP clients>
GUI
t a <data to all GUI clients>
t a <data to single GUI client>
list prepend broadcast
list prepend send
list trim
list trim
tcpserver 5600
t a <data from GUI>
buffer
t f f
> 0
select 1
dump
fifop
list prepend
stores all messages from the engine: whenever a a new GUI connects, it will be updated to the current status.
new_connection
t b b
del 0
1
0
f
spigot
t f b
t a
<socket of newly connected GUI>

- initialization
  - caching messages from DSP
  - cache only „interesting“ messages (memory!)
- synching GUIs
  - keep track of all existing windows
  - tell engine to redraw windows

- concurrent editing
  - single/multiple mousepointers
  - visual feedback
    - pseudo cursors indicating others' actions

- unwanted messages
  - „QUIT“ quits engine!
  - minimizing window (switching desktops) deletes window contents
  - filtering

(time for a demo...)

- multiplication of network traffic
- concurrent editing within 1 window
  - model-view-controller!
  - DesireData (?)
- multiple engines
  - dislocated patching
    - streaming...
  - replicating the engine...
    - no simple way (window-ID generation,...)

- ability to connect multiple pd-gui's to one engine
  - logging of sessions (no replay (yet)...)
- proxy only solution
  - no changes to engine/gui
- benefits from Pd's „stupid gui“ approach
- would benefit from engine/gui separation
- find a better name

thanks for your patience.

get multipd from
https://svn.umlaeute.mur.at/svnroot/zmoelnig/projects/multipd/

- listen to multipd at
- *blind date* by pd-graz tonight!