MUSIC

Mikael Djurfeldt, PDC/KTH
HBP Neuromorphic SP

Outline

- Interfaces in computational neuroscience software
- What is MUSIC?
- Two problems solved by MUSIC:
  - spatial aliasing problem
  - temporal aliasing problem
- How to use MUSIC from C++, Python and PyNN
- Use cases
- Where to get software and documentation

Interfaces in computational neuroscience

- Simulation environments in computational neuroscience,
  such as NEURON, NEST or Brian, each provide many tools
  needed by the user to carry out high-quality simulation studies.

- Models described differently,
  environments have specific features
  => hard to move models

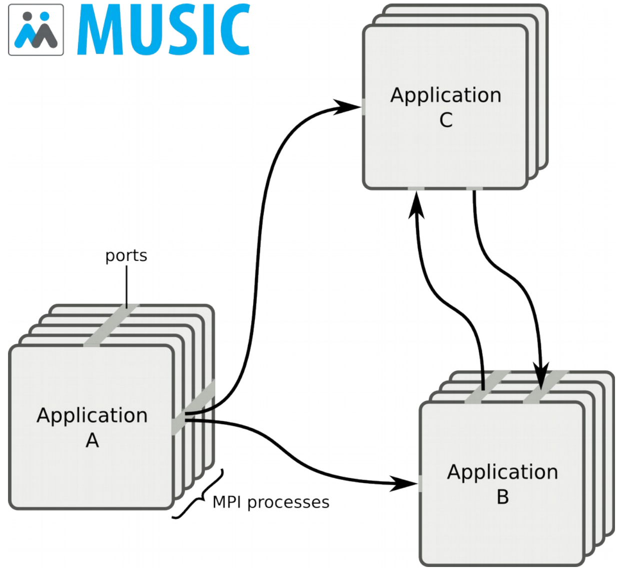
- Difficult to build larger simulations which re-use existing models

Interfaces in computational neuroscience

- As systems grow and encompass more subsystems, they rapidly become unwieldy to develop
- In general, software in computational neuroscience tends to have a monolithic structure
- *Software interfaces* (APIs) allow for use of different implementations of software components
- MUSIC is an API, and implementation in the form of a C++ library, supporting flowing of data between tools during simulation

  (INCF initiative, originally developed by Ö. Ekeberg and M. Djurfeldt)

MUSIC co-simulations

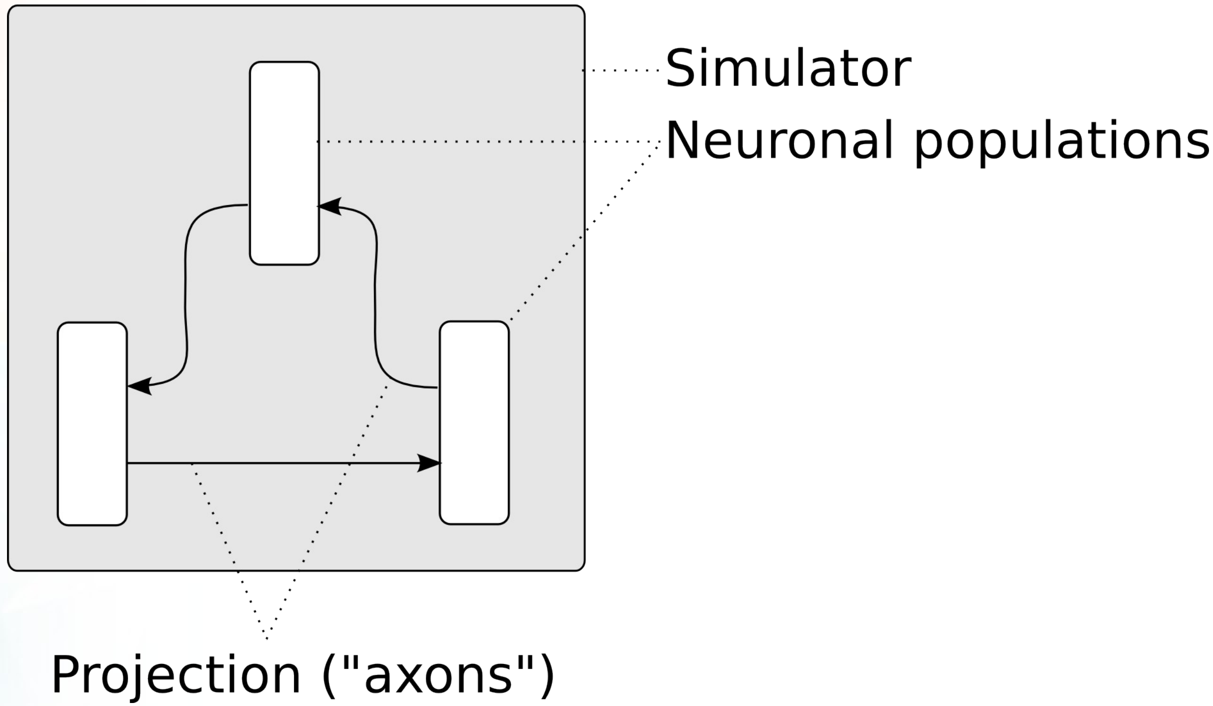


A co-simulation with multiple parallel applications (A, B, C) exchanging runtime data (such as neuronal events)

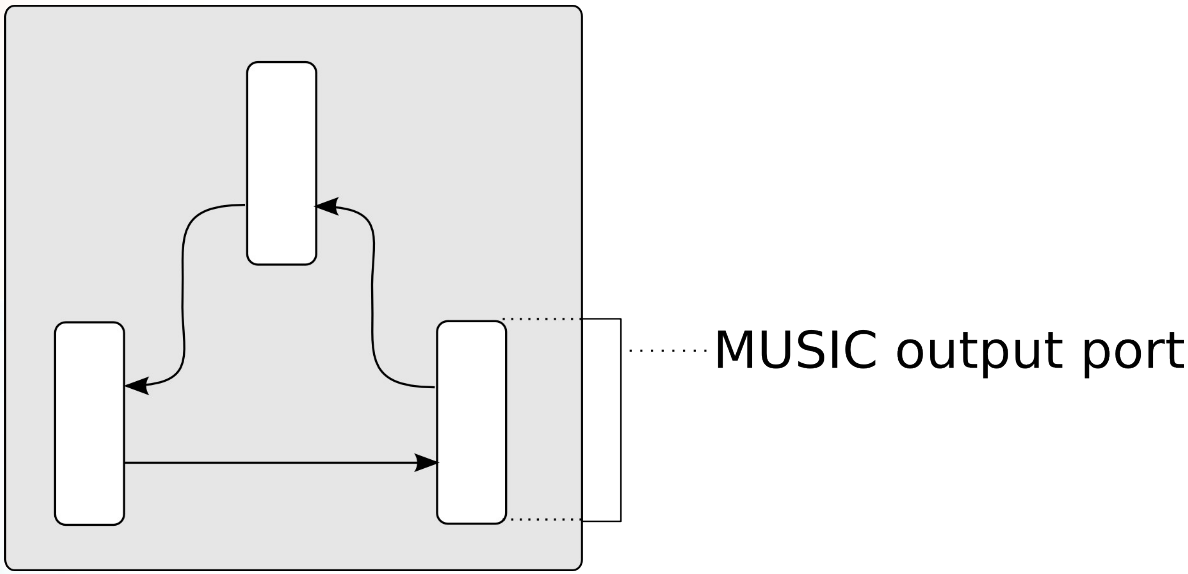
Shipping data around between applications during simulation useful e.g. for:

- Building larger models by combining models as components
- Modeling multiple scales and/or combining different formalisms simultaneously
- Pre/postprocessing and visualization
- Interfacing to external hardware

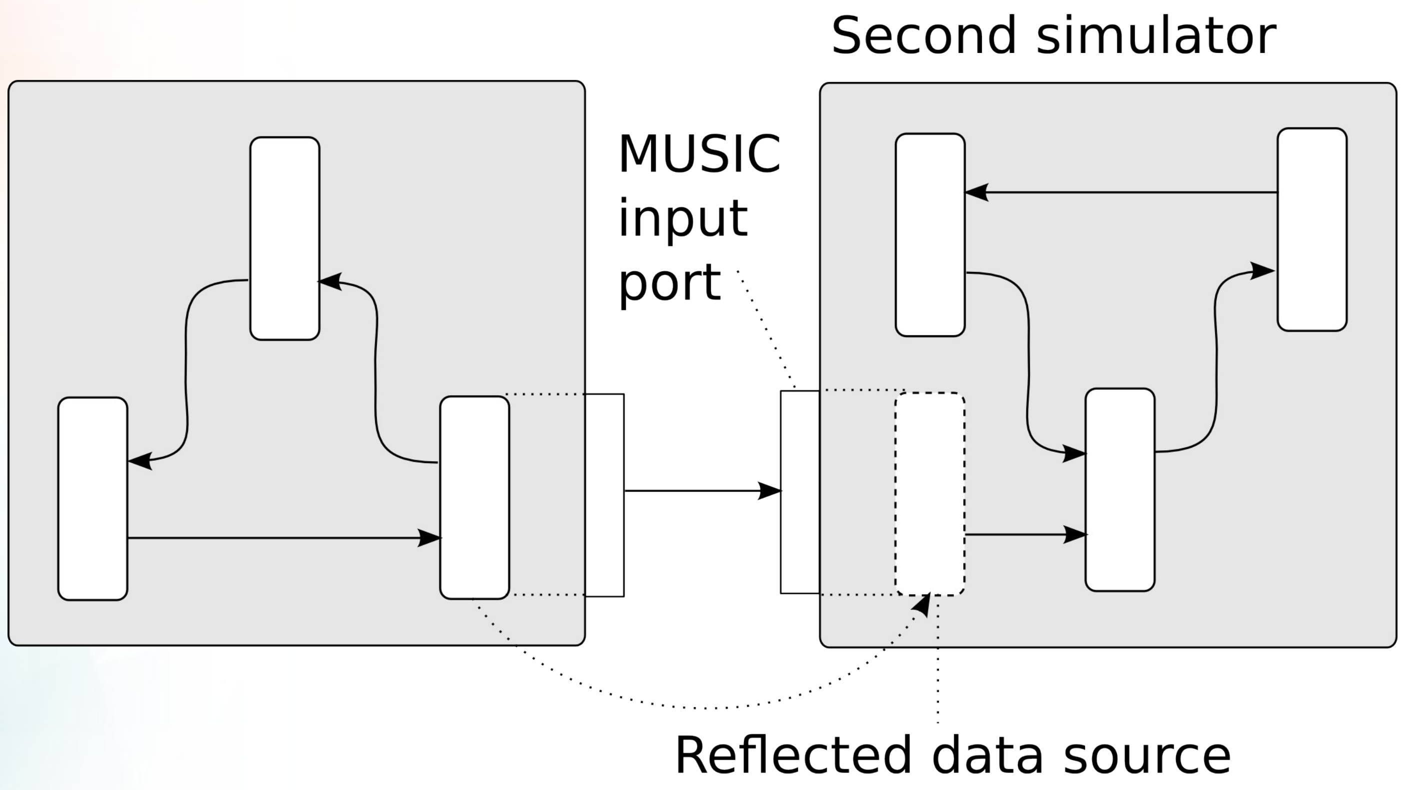
Network simulation



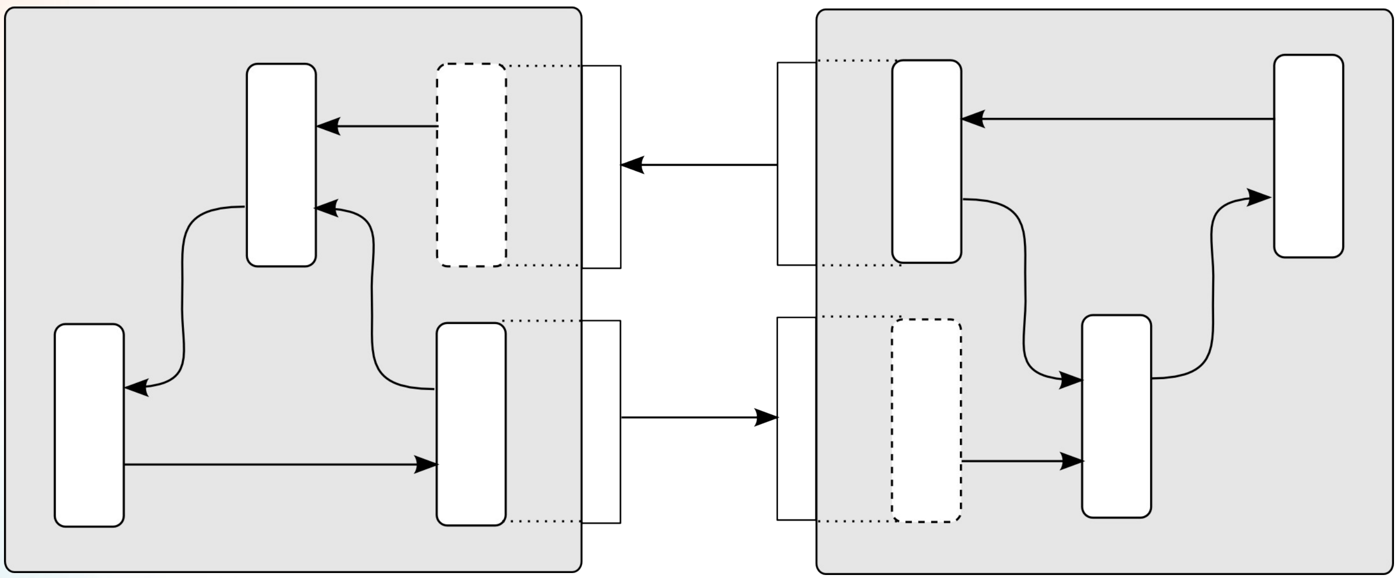
Using MUSIC to expose data



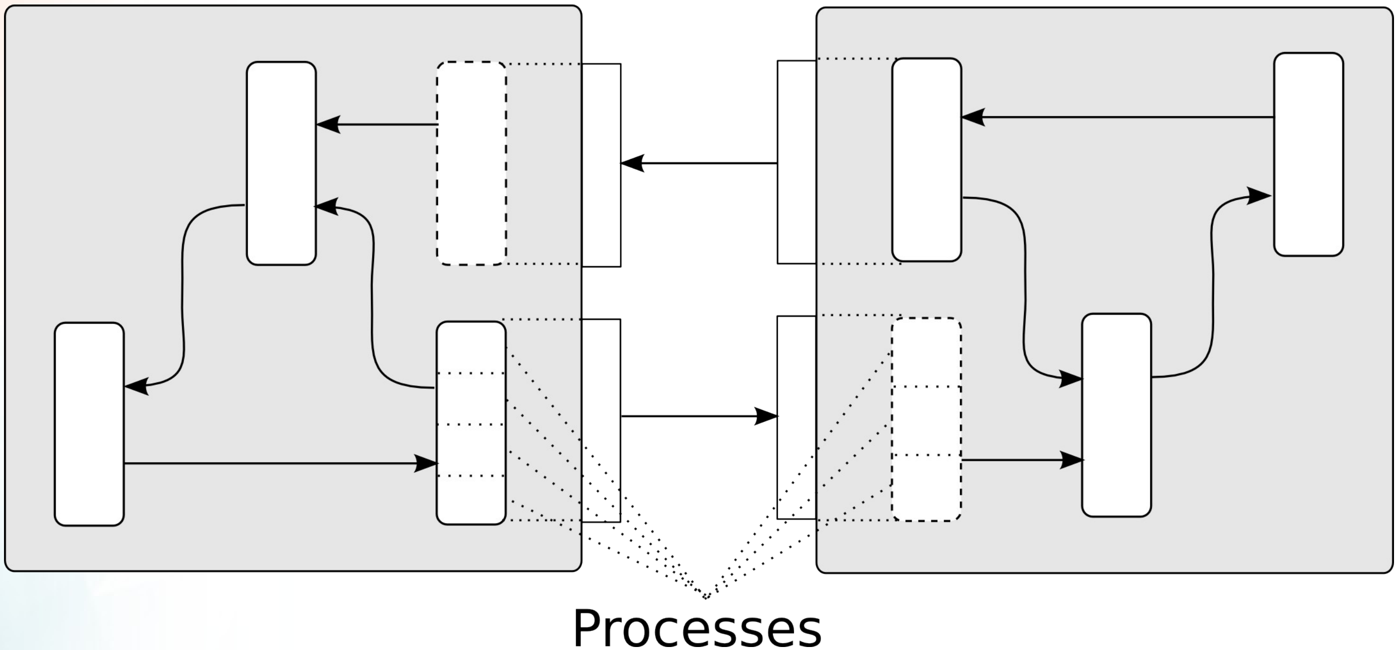
Co-simulation



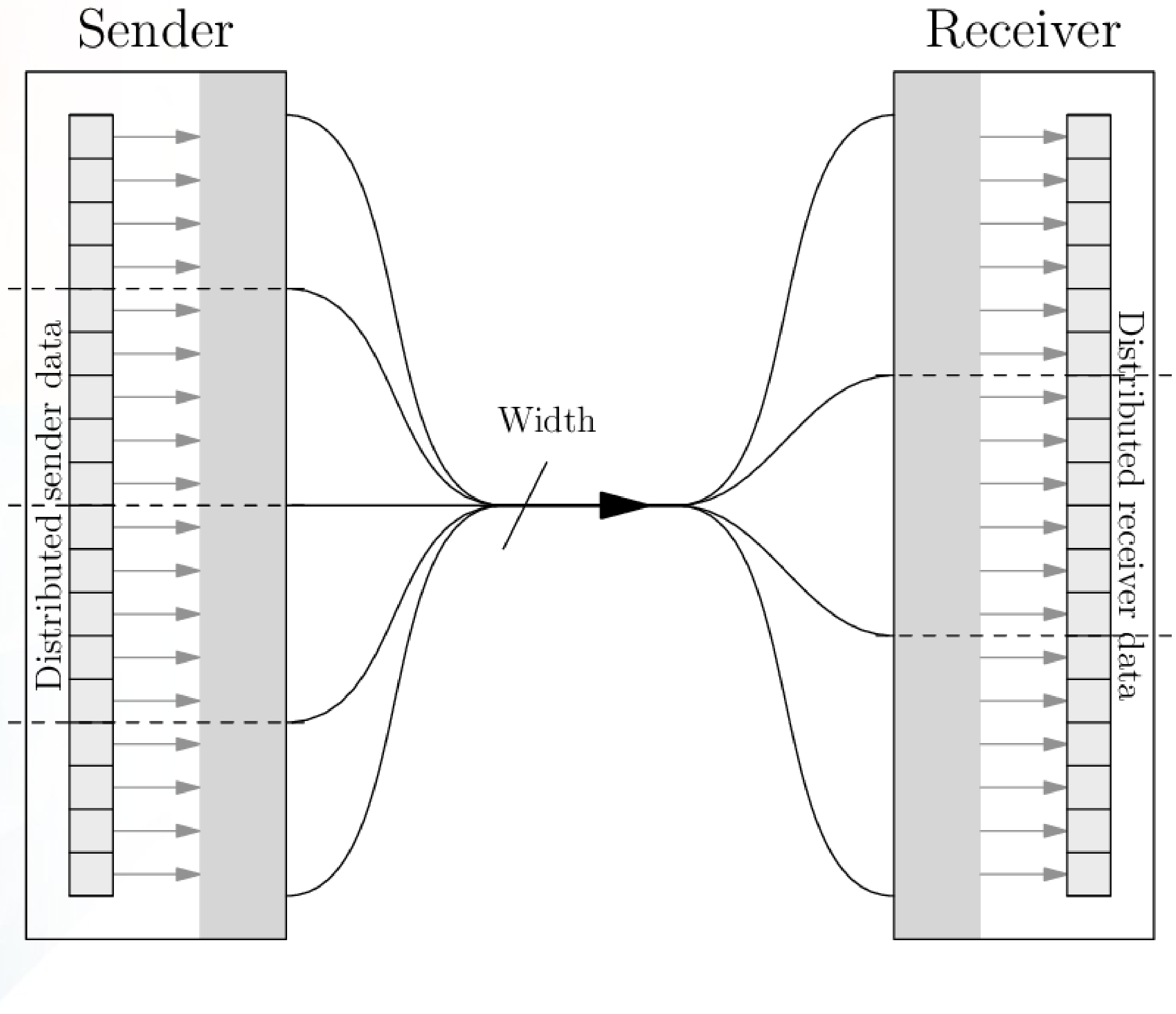
Loop



Spatial aliasing problem



Spatial aliasing



Scheduling of communication

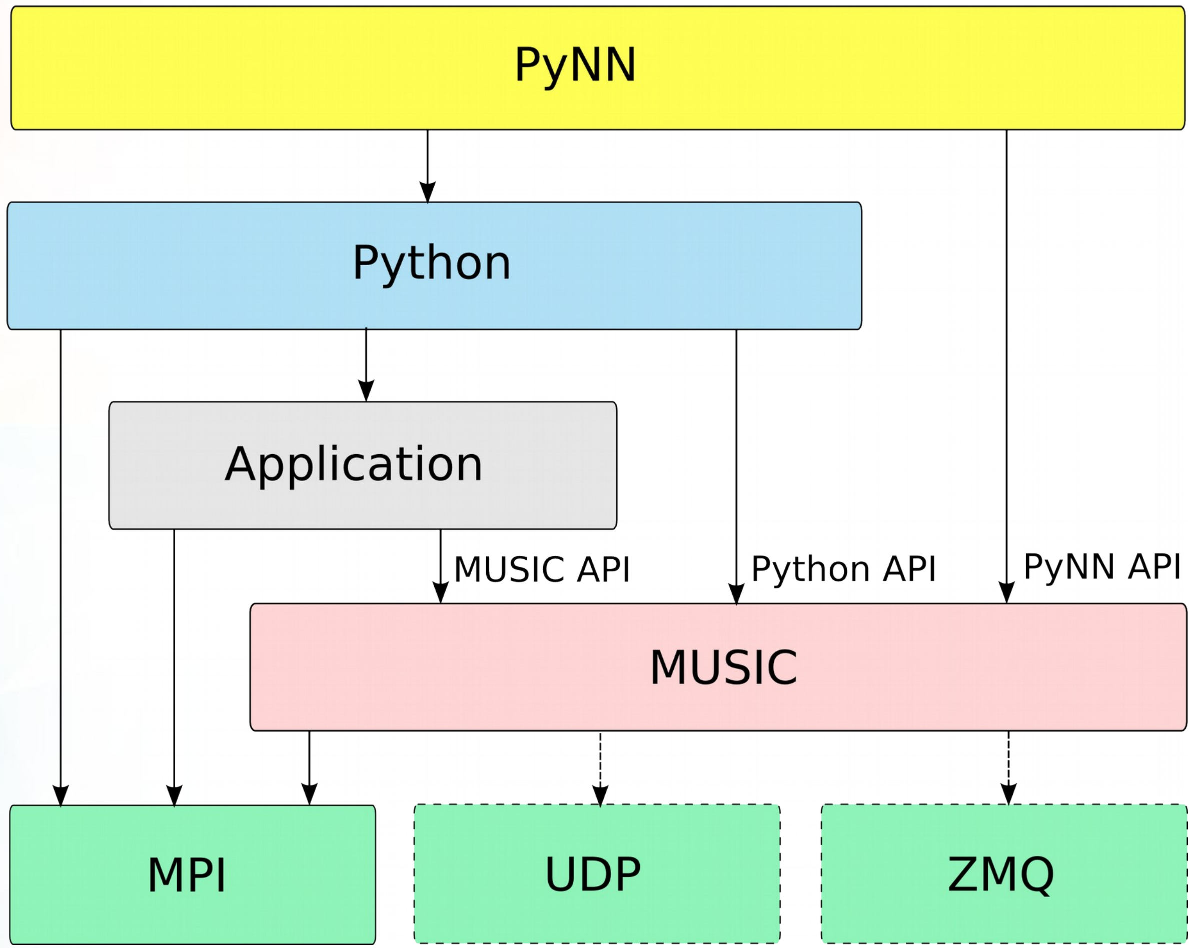
Handling of time in MUSIC

- An application calls MUSIC **`tick()`** at points regularly spaced in simulated time
- This is where data may be sent and/or received
- Different applications are allowed to call **`tick()`** at different rates
- MUSIC may allow applications to run out-of-sync (each with its own offset between simulation time and wallclock time)
- MUSIC allows complex topology of port connectivity

Scheduling problem

- How to deliver data in time while avoiding deadlocks
- How to interpolate continuous data given different tick rates

Interfaces to MUSIC



C++ app: eventsource

```cpp
#include <music.hh>

int
main (int argc, char *argv[])
{
  // Get real argc and argv
  MUSIC::Setup* setup = new MUSIC::Setup (argc, argv);
  ...
  // Publish an output port
  MUSIC::EventOutputPort* out = setup->publishEventOutput (portName);

  // Associate the port with neurons ("map" the port)
  MUSIC::LinearIndex indices (firstIndex, nLocalUnits);
  out->map (&indices, MUSIC::Index::GLOBAL);
  ...
  // Prepare for simulation
  MUSIC::Runtime* runtime = new MUSIC::Runtime (setup, timestep);

  // Simulation loop
  ...

  // End simulation
  runtime->finalize ();
  ...
}
```

C++ app: eventsource

```
#include <music.hh>

int
main (int argc, char *argv[])
{
  ...
  // Simulation loop
  spikeFile >> t >> id;
  double time = runtime->time ();
  while (time < stoptime)
    {
      double nextTime = time + timestep;
      while (!spikeFile.eof () && t < nextTime)
        {
          out->insertEvent (t, MUSIC::GlobalIndex (id));
          spikeFile >> t >> id;
        }
      // Make data available for other programs
      runtime->tick ();

      time = runtime->time ();
    }
    ...
}
```

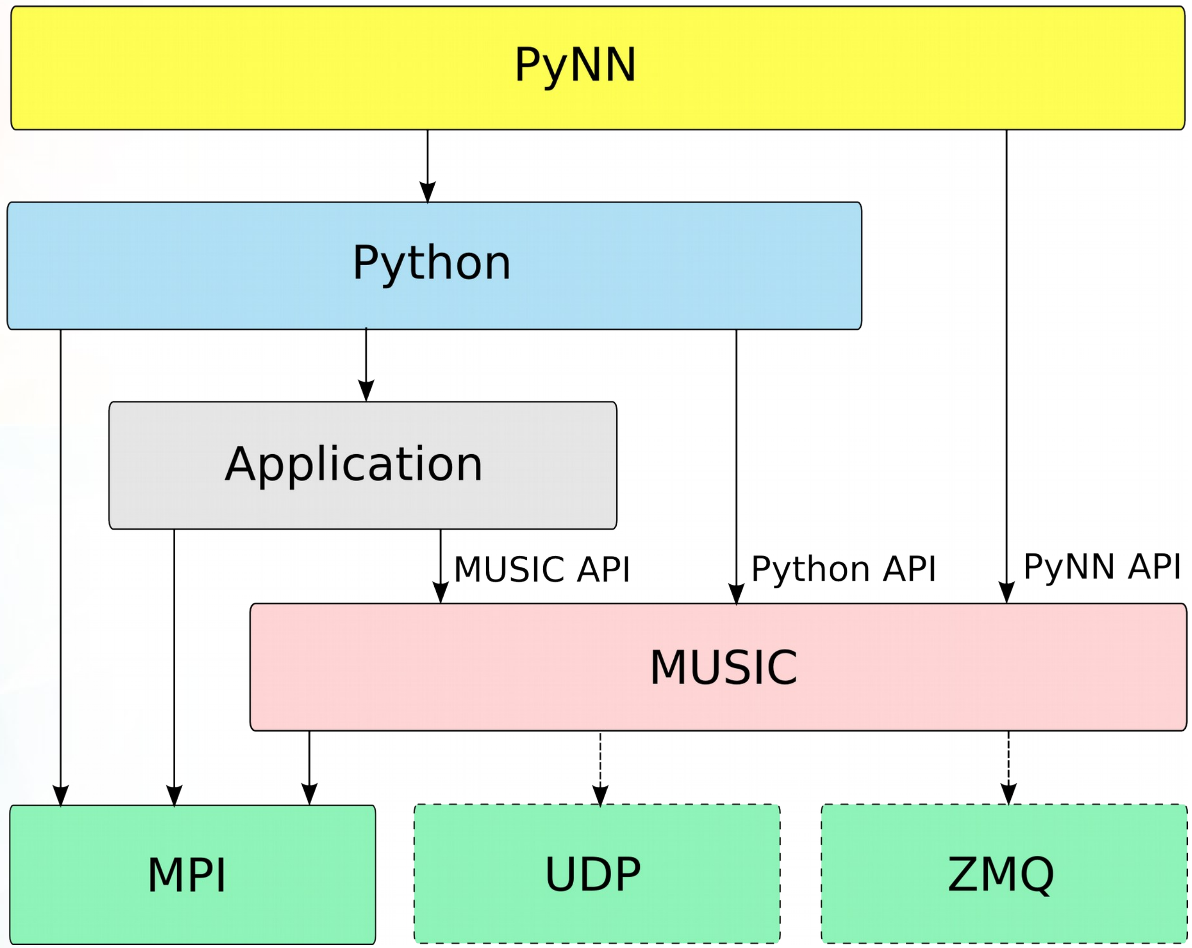
C++ app: eventlogger

```
#include <music.hh>

class MyEventHandlerGlobal : public MUSIC::EventHandlerGlobalIndex {
public:
  void operator () (double t, MUSIC::GlobalIndex id)
  {
    // Print incoming event
    std::cout << t << "\t" << id << std::endl;
  }
};

int
main (int argc, char *argv[])
{
  ...
  double time = runtime->time ();
  while (time < stoptime)
    {
      // Retrieve data from other program
      runtime->tick ();

      time = runtime->time ();
    }
  ...
}
```

MUSIC configuration file

```
np = 1
stoptime = 1.0

[A]
  binary = ./eventsource
  args = 10 spikes

[B]
  binary = ./eventlogger
  args = 10

A.out->B.in [10]
```

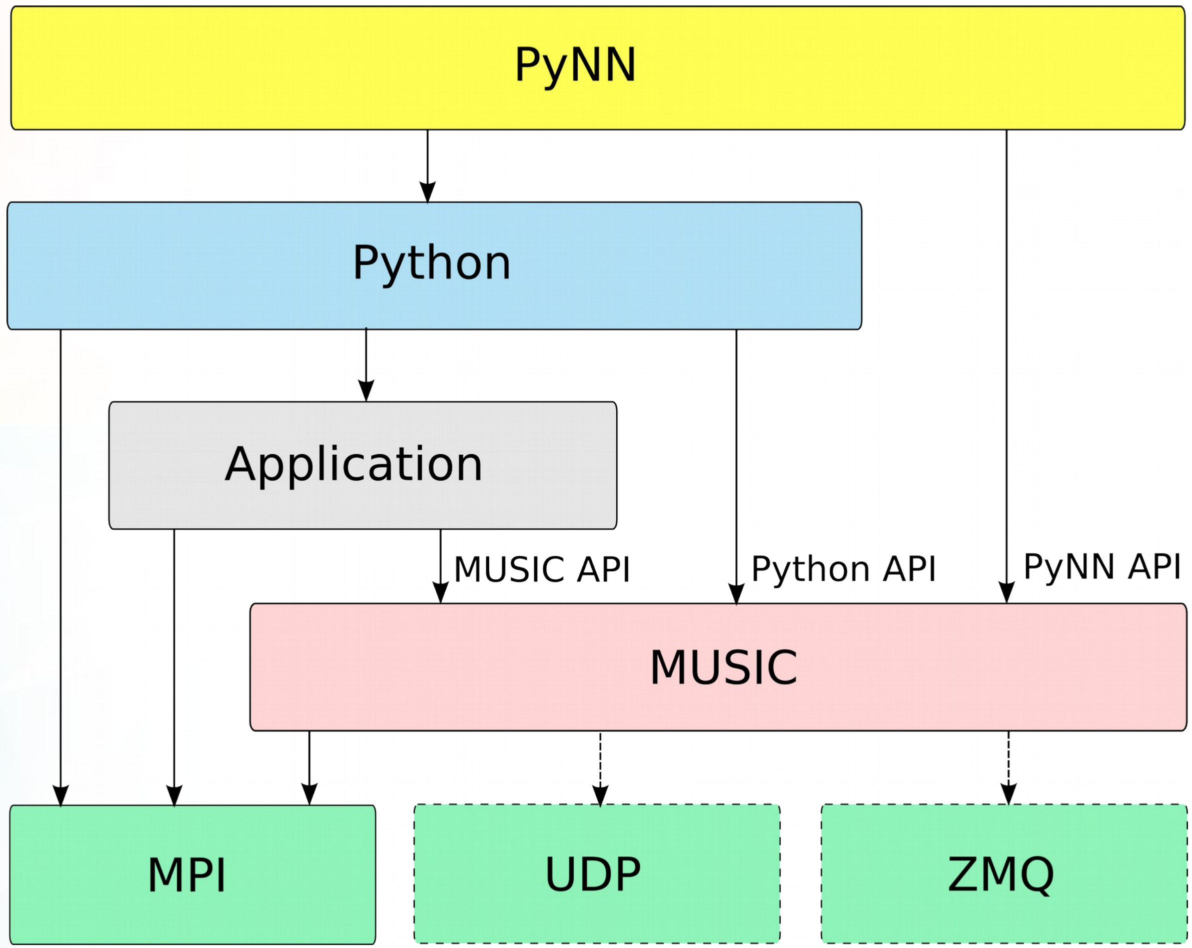
Interfaces to MUSIC



Python app: eventsource

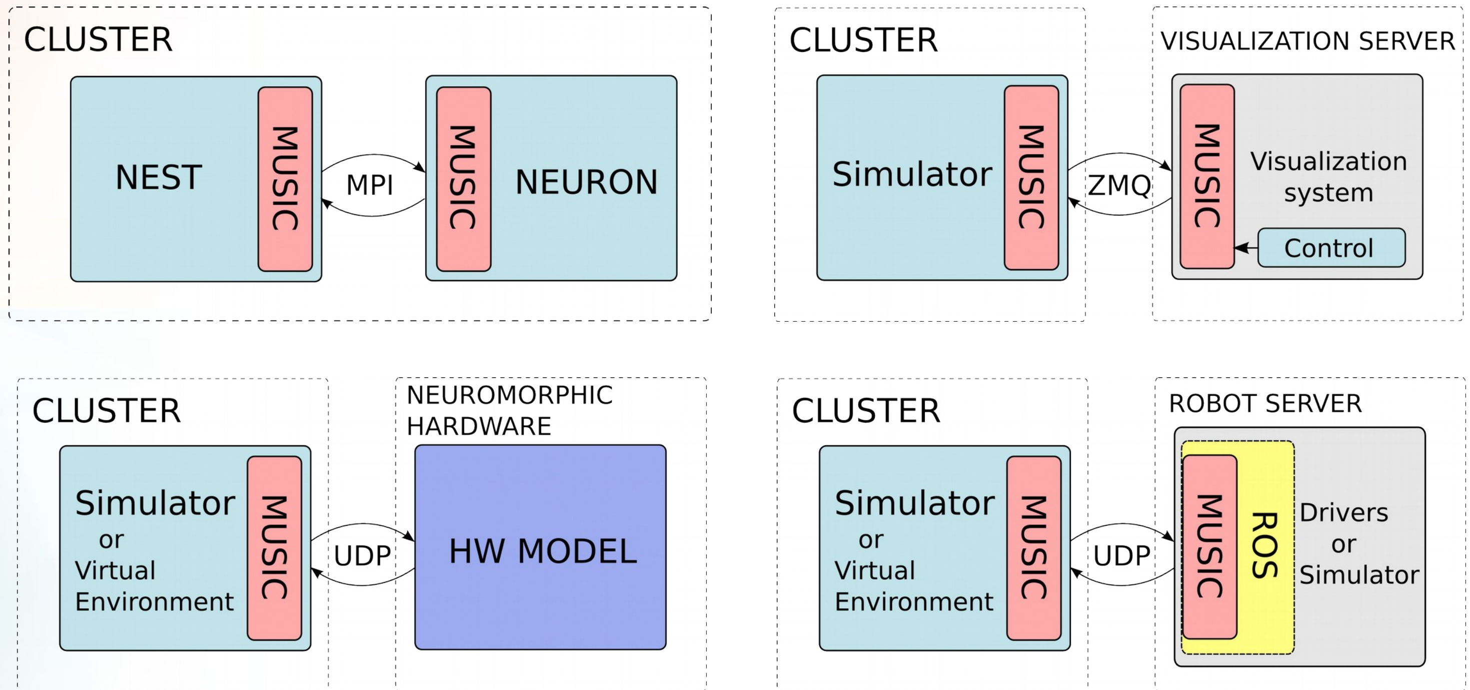
```
import music

# Get setup handle
setup = music.Setup ()

# Publish an output port
out = setup.publishEventOutput ("out")

# Associate the port with neurons ("map" the port)
out.map (music.Index.GLOBAL, base=firstId, size=local)

# Prepare for simulation
runtime = setup.runtime (timestep)
...
# Simulation loop
try:
    t, id = next (spikes)
    while time < stoptime:
        nextTime = time + timestep
        while t < nextTime:
            out.insertEvent (t, id, music.Index.GLOBAL)
        runtime.tick ()
        time = runtime.time ()
except StopIteration:
    pass
```

Interfaces to MUSIC



```python
from pyNN import music

sim1, sim2 = music.setup(music.Config("nest", 1), music.Config("neuron", 1))

[...]

sim1.setup(timestep=0.1, min_delay=0.2, max_delay=1.0)
sim2.setup(timestep=0.1, min_delay=0.2, max_delay=1.0)

input_population  = sim1.Population(1,
                                    sim1.SpikeSourceArray,
                                    {'spike_times': spike_times},
                                    label="input")

output_population = sim2.Population(2,
                                    sim2.IF_curr_alpha,
                                    cell_params,
                                    label="output")

# The connector is used on the receiving side (sim2)
projection = music.Projection(input_population, output_population,
                              sim2.AllToAllConnector())

music.run(tstop)

output_population.printSpikes("Results/simpleNetwork_output.ras")

music.end()
```

Usage scenarios



Integrated simulation of the whole-body musculo- skeletal-nervous system for clarification of motor dysfunctions due to neurological diseases

Jun Igarashi[1], Jan Moren[1], Osamu Shouno[3], Kazuya Shimizu[2], Naoto Yamamura[2], Junichiro Yoshimoto[1]

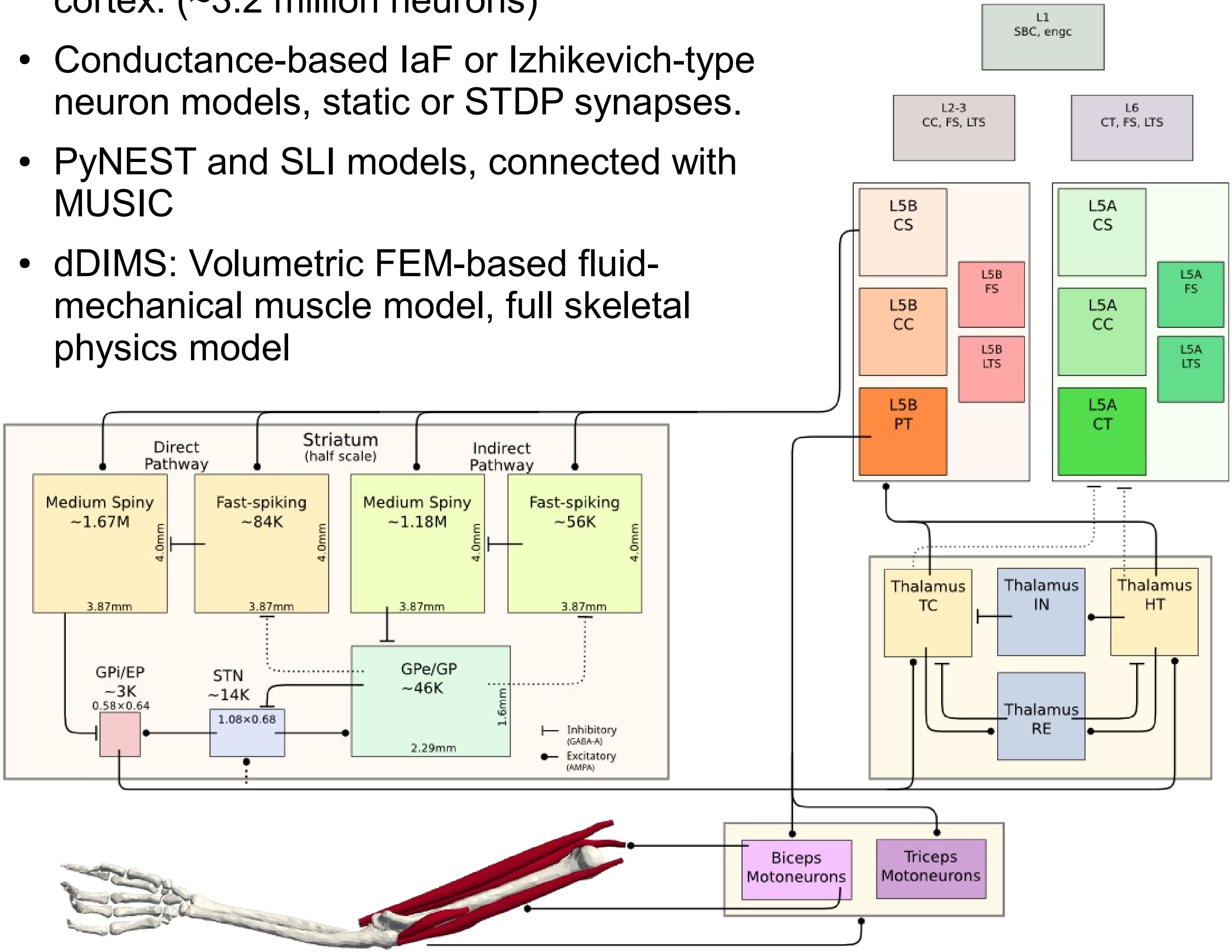
Shu Takagi[2] & Kenji Doya[1]

1: Okiniwa Insitutiute of Science and Technology (OIST)

2: Tokyo University

3: Honda research Institute Japan

- Full size, neuron count of rat BG and motor cortex. (~3.2 million neurons)
- Conductance-based IaF or Izhikevich-type neuron models, static or STDP synapses.
- PyNEST and SLI models, connected with MUSIC
- dDIMS: Volumetric FEM-based fluid-mechanical muscle model, full skeletal physics model

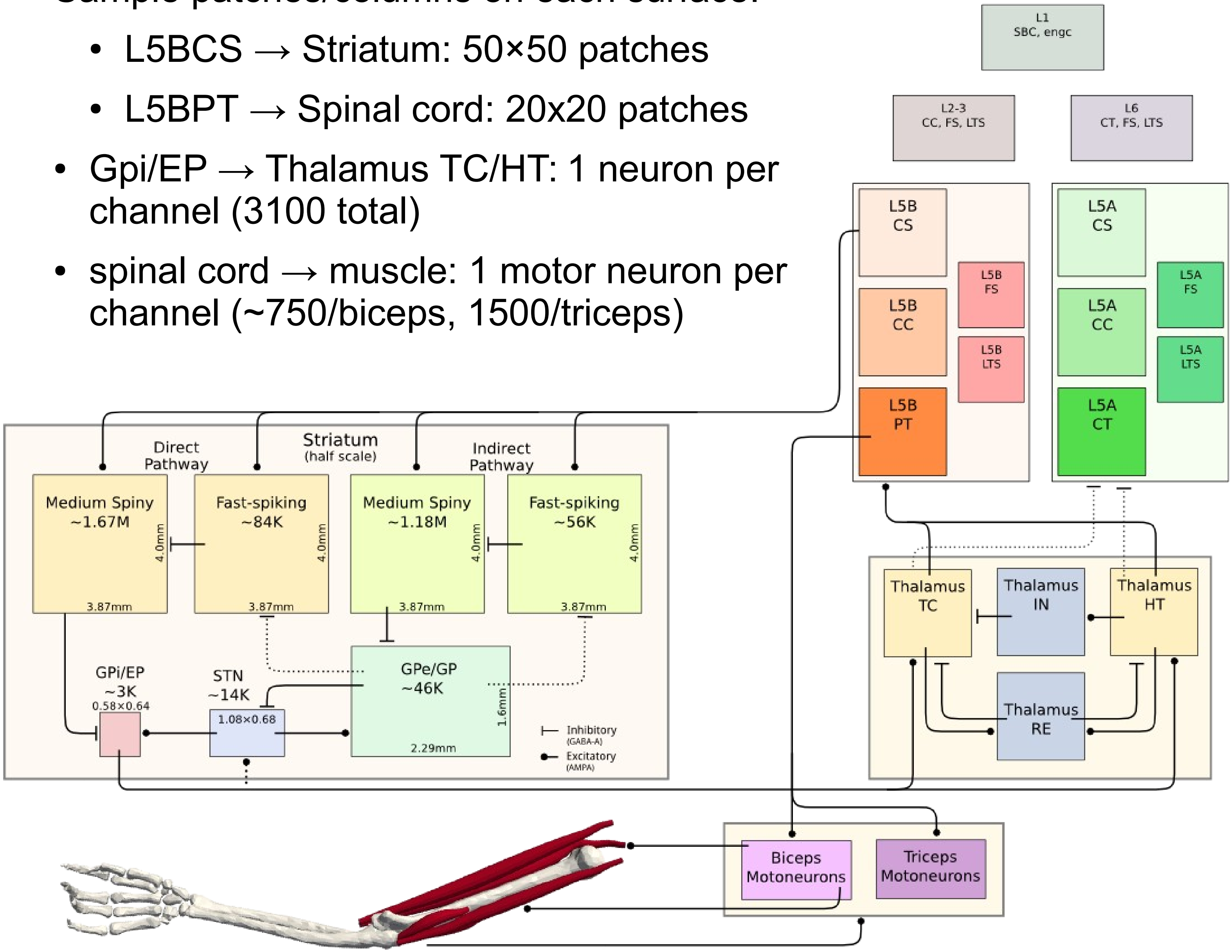


MUSIC organization

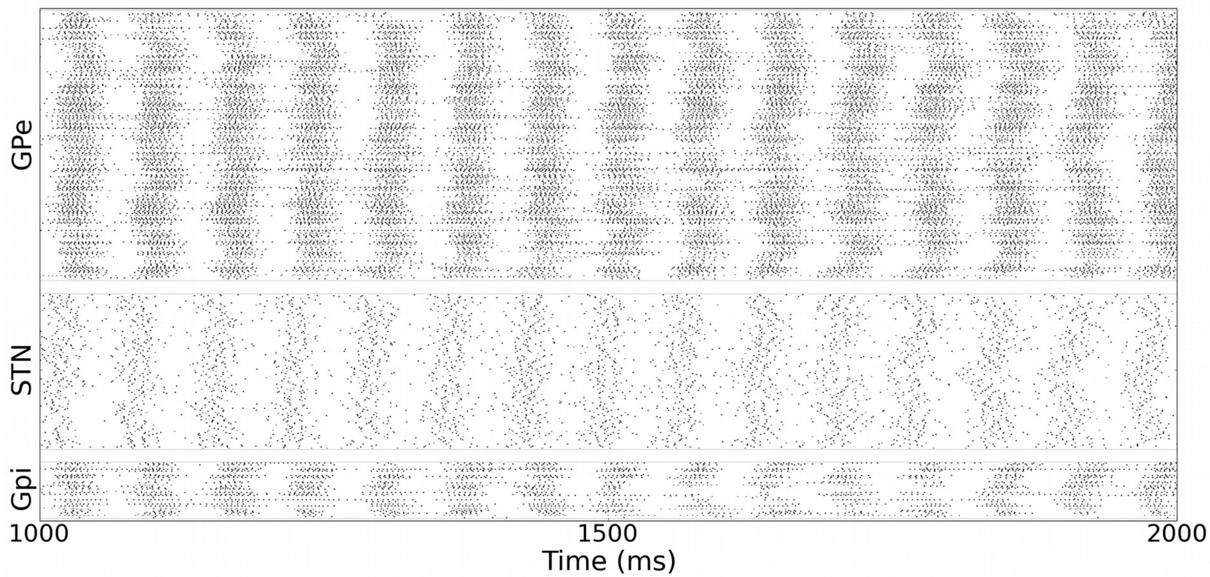
Sample patches/columns on each surface:

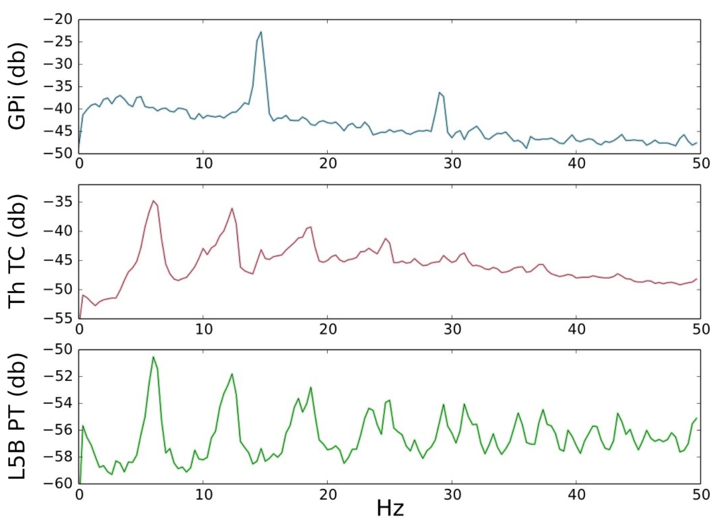
- L5BCS → Striatum: 50×50 patches
- L5BPT → Spinal cord: 20x20 patches

- Gpi/EP → Thalamus TC/HT: 1 neuron per channel (3100 total)
- spinal cord → muscle: 1 motor neuron per channel (~750/biceps, 1500/triceps)



preliminary results





- ***Up:*** *GPe, STN and GPi neurons oscillating at about 14.7Hz.*
- ***Left:*** *power spectrum of GPi, Thalamic CT neurons and L5B PT neurons.*

Other highlighted use cases

- Bluebrain Monsteer
  Library for interactive visualization
- MUSIC-ROS toolchain
  Philipp Weidel Thursday 10:10

Where to get MUSIC

- Github INCF/MUSIC
- MUSIC manual in the distribution
- Djurfeldt et al. (2010) “Run-time interoperability between neuronal network simulators based on the MUSIC framework” Neuroinform.

Contents

Neuroinform
DOI 10.1007/s12021-010-9064-z

Run-Time Interoperability Between Neuronal Network Simulators Based on the MUSIC Framework

Mikael Djurfeldt · Johannes Hjorth · Jochen M. Eppler · Niraj Dudani · Moritz Helias · Tobias C. Potjans · Upinder S. Bhalla · Markus Diesmann · Jeanette Hellgren Kotaleski · Örjan Ekeberg

Abstract MUSIC is a standard API allowing large scale neuron simulators to exchange data within a parallel computer during runtime. A pilot implementation of this API has been released as open source. We provide experiences from the implementation of MUSIC interfaces for two neuronal network simulators of different kinds, NEST and MOOSE. A multi-simulation of a cortico-striatal network model involving both simulators is performed, demonstrating how MUSIC can promote inter-operability between models written for different simulators and how these can be re-used to build a larger model system. Benchmarks show that the MUSIC pilot implementation provides efficient data transfer in a cluster computer with good scaling. We conclude that MUSIC fulfills the design goal that it should be simple to adapt existing simulators to use MUSIC. In addition, since the MUSIC API enforces independence of the applications, the multi-simulation could be built from pluggable component modules without adaptation of the components to each other in terms of simulation time-step or topology of connections between the modules.

Keywords MUSIC · Large-scale simulation · Computer simulation · Computational neuroscience · Neuronal network models · Inter-operability · MPI · Parallel processing

Introduction

Large scale neuronal network models and simulations have become important tools in the study of the brain and the mind (Albus et al. 2007; Djurfeldt et al. 2008a). Such models work as platforms for integrating knowledge from many sources of data. They help to elucidate how information processing occurs in the healthy

Electronic Supplementary Material The online version of this article (doi:10.1007/s12021-010-9064-z) contains supplementary material, which is available to authorized users.

Johannes Hjorth and Jochen M. Eppler have contributed equally to the contents of the article.

M. Djurfeldt (✉) · J. Hjorth · J. Hellgren Kotaleski · Ö. Ekeberg
School of Computer Science and Communication, Royal Institute of Technology, 100 44 Stockholm, Sweden
e-mail: mikael@djurfeldt.com

M. Djurfeldt · M. Diesmann
RIKEN Brain Science Institute, Wako-shi, 351-0198 Saitama, Japan

J. M. Eppler
Honda Research Institute Europe GmbH, Carl-Legien-Straße 30, 63073 Offenbach, Germany

J. M. Eppler · M. Helias · M. Diesmann
Bernstein Center for Computational Neuroscience, Albert-Ludwigs-Universität Freiburg, Hansastraße 9A, 79104 Freiburg, Germany

N. Dudani · U. S. Bhalla
National Centre for Biological Sciences, Bangalore, India

T. C. Potjans
Institute of Neurosciences and Medicine, Research Center Jülich, 52425 Jülich, Germany

T. C. Potjans · M. Diesmann
RIKEN Computational Science Research Program, Wako-shi, 351-0198 Saitama, Japan

Published online: 02 March 2010

Humana Press

Thanks

- Ekaterina Brocke, SciLife lab, KI – communication algorithms
- Alexander Peyser, Simlab neurosci, FZJ – Python interface
- Andrew Davison, Jochen Eppler and Eilif Muller – PyNN interface
- Rajalekshmi Deepu, Simlab neurosci, FZJ – Travis integration
- Jan Morén, OIST – MUSIC application example
- INCF
- HBP
- Simlab neuroscience
- INM6, FZ Juelich