TVB-multiscale:

TVB and spiking networks (NEST, ANNarchy, NETPYNE(NEURON)) Co-Simulation

D. Perdikis[1], V. Bragin[1], A. Blickensdörfer[1], M. Schirner[1], L. Domide[2], P. Ritter[1]

[1]Brain Simulation Section, Brain Institute of Health, Department of Neurology, Charité—Universitätsmedizin Berlin
[2]S.C. CODEMART S.R.L., Cluj-Napoca, Jud Cluj, Romania

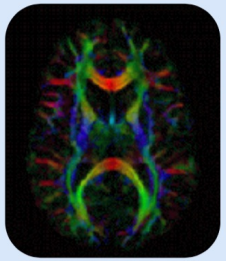
November 2022

Co-funded by the European Union

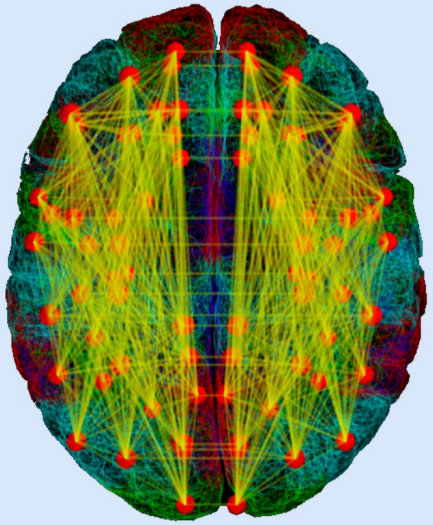
EBRAINS

The Virtual Brain model and simulation: objective

Data

Model

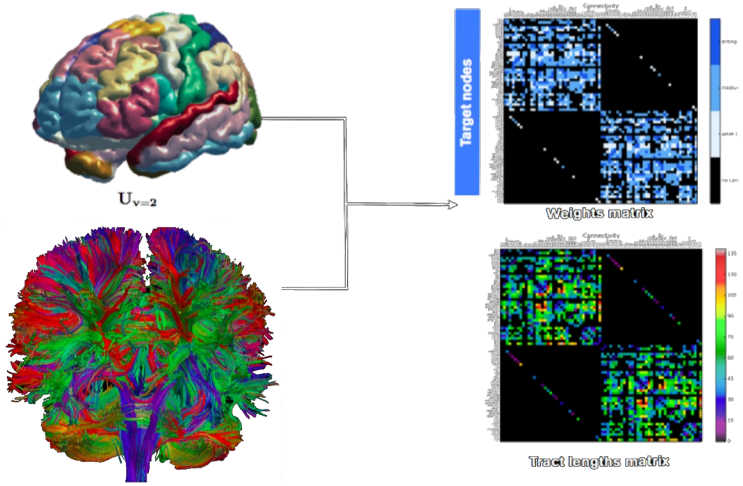


$$\frac{dx}{dt} = f(x, I, \upsilon)$$

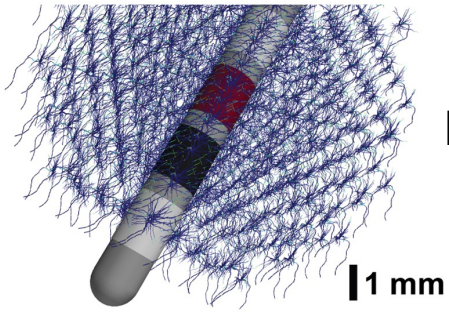
Theory

processes & mechanisms

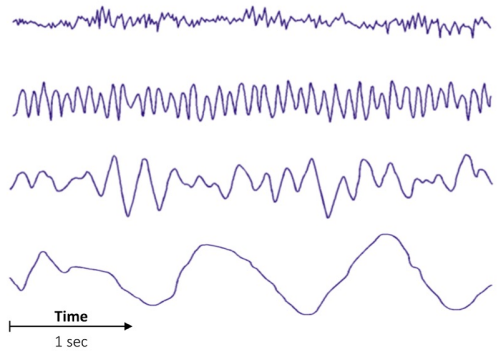
What is the input and output of TVB?



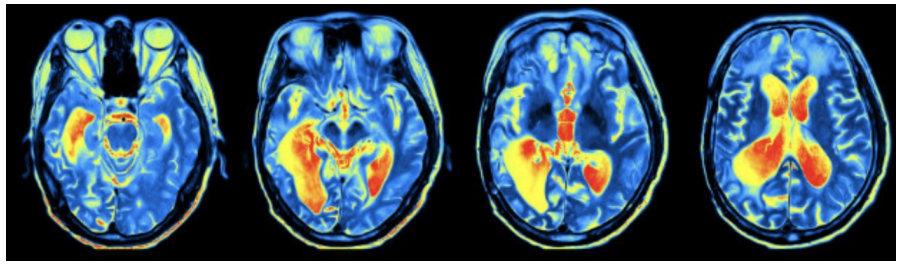
THEVIRTUALBRAIN.



LFP

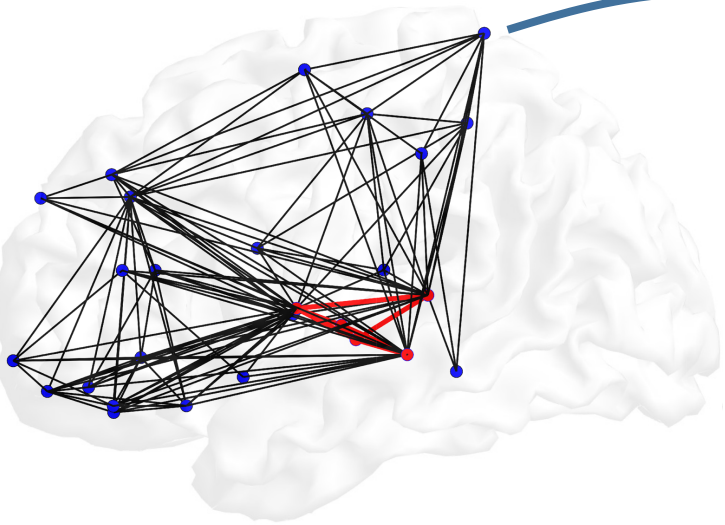


M/EEG

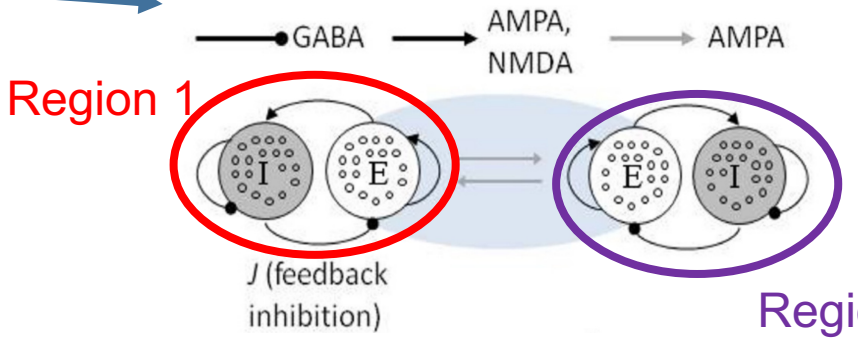


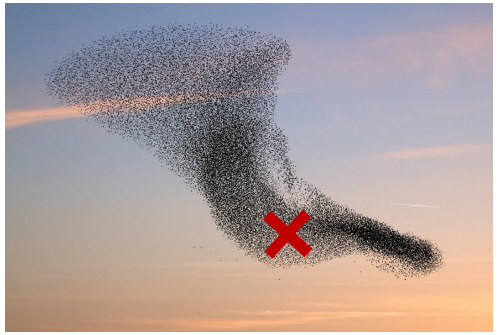
BOLD fMRI

How does TVB work?



Mean-field model





Reduced Wong-Wang model

$$I_i^{(E)} = W_E I_0 + w_+ J_{NMDA} S_i^{(E)} + G J_{NMDA} \sum_j C_{ij} S_j^{(E)} - J_i S_i^{(I)} + I_{external}, \quad (5)$$

$$I_i^{(I)} = W_I I_0 + J_{NMDA} S_i^{(E)} - S_i^{(I)} + \lambda G J_{NMDA} \sum_j C_{ij} S_j^{(E)}, \quad (6)$$

$$r_i^{(E)} = H^{(E)}(I_i^{(E)}) = \frac{a_E I_i^{(E)} - b_E}{1 - \exp(-d_E(a_E I_i^{(E)} - b_E))}, \quad (7)$$

$$r_i^{(I)} = H^{(I)}(I_i^{(I)}) = \frac{a_I I_i^{(I)} - b_I}{1 - \exp(-d_I(a_I I_i^{(I)} - b_I))}, \quad (8)$$

$$\frac{dS_i^{(E)}(t)}{dt} = -\frac{S_i^{(E)}}{\tau_E} + (1 - S_i^{(E)}) \gamma r_i^{(E)} + \sigma \nu_i(t), \quad (9)$$

$$\frac{dS_i^{(I)}(t)}{dt} = -\frac{S_i^{(I)}}{\tau_I} + r_i^{(I)} + \sigma \nu_i(t), \quad (10)$$

(Deco et al., 2014)

A TVB brain model

$$\frac{dx_i(t)}{dt} = N(x_i(t)) + G\sum_j SC_{ij}x_j(t - d_{ij}^{glob}) + L\sum_j LC_{ij}x_j(t - d_{ij}^{loc}) + I_i(t) + v_i(t)$$

Differential operator that maps the system of equations into its derivatives

Neural mass model

State variables (e.g., firing rates, field potentials, ...)

Global coupling scaling factor

Structural connectivity matrix (long-range coupling)

Time delays

Local connectivity

Injected input

Noise

EBRAINS

Motivation

TVB: large-scale simulation linking brain anatomical data ((d)MRI, CT etc) to neuroimaging data, by generating virtual neural source activity.

THEVIRTUALBRAIN.

Brain regions' level

EBRAINS

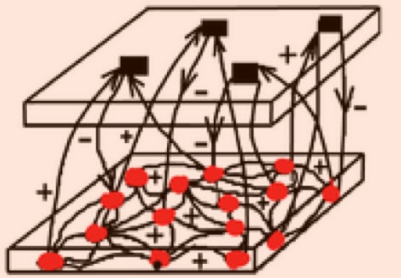
Motivation

TVB: large-scale simulation linking brain anatomical data ((d)MRI, CT etc) to neuroimaging data, by generating virtual neural source activity.

THEVIRTUALBRAIN.

Brain regions' level

Spiking simulators (NEST, ANNarchy, NEURON/NETPYNE etc): fine-scale simulation for investigation of local or systems' neural mechanisms



Neurons' level

nest :: simulated()

ANNarchy

NetPyNE

EBRAINS

Motivation

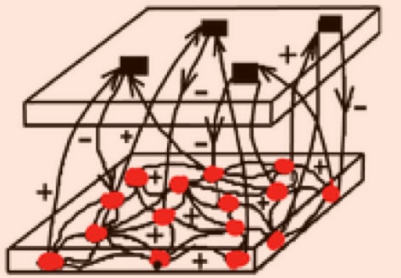
TVB: large-scale simulation linking brain anatomical data ((d)MRI, CT etc) to neuroimaging data, by generating virtual neural source activity.

Spiking simulators (NEST, ANNarchy, NEURON/NETPYNE etc): fine-scale simulation for investigation of local or systems' neural mechanisms

THEVIRTUALBRAIN.

Brain regions' level

THEVIRTUALBRAIN. ⟷ nest :: simulated() / ANNarchy / NetPyNE



Neurons' level

nest :: simulated() / ANNarchy / NetPyNE

EBRAINS

Motivation

TVB-multiscale interfaces for multiscale co-simulation:

- *TVB* ➡ *Spiking simulators*: Generate biologically realistic spatio-temporal context and large-scale connectivity for local neural networks
- *TVB* ⟷ *Spiking simulators*: Interaction of two scales.

THEVIRTUALBRAIN.

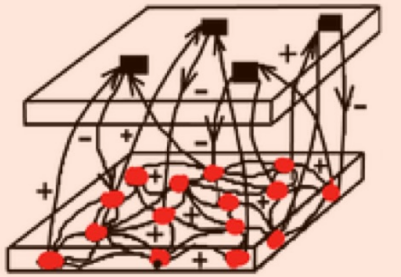
Brain regions' level

THEVIRTUALBRAIN.

nest :: simulated()

ANNarchy

NetPyNE



Neurons' level

nest :: simulated()

ANNarchy

NetPyNE

EBRAINS

Basic concepts

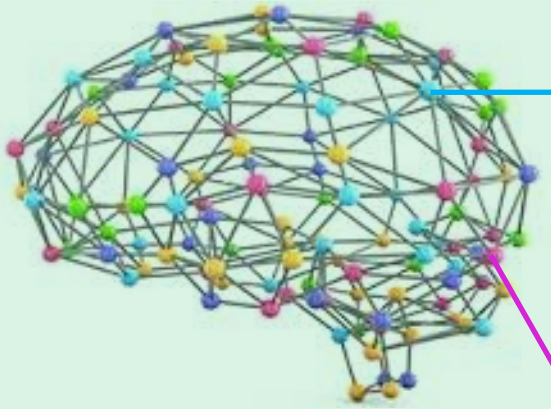
Populations' level

Populations' activity as mean field system state variables

Average activity of populations of spiking neurons

THEVIRTUALBRAIN.

Brain regions' level



THEVIRTUALBRAIN.

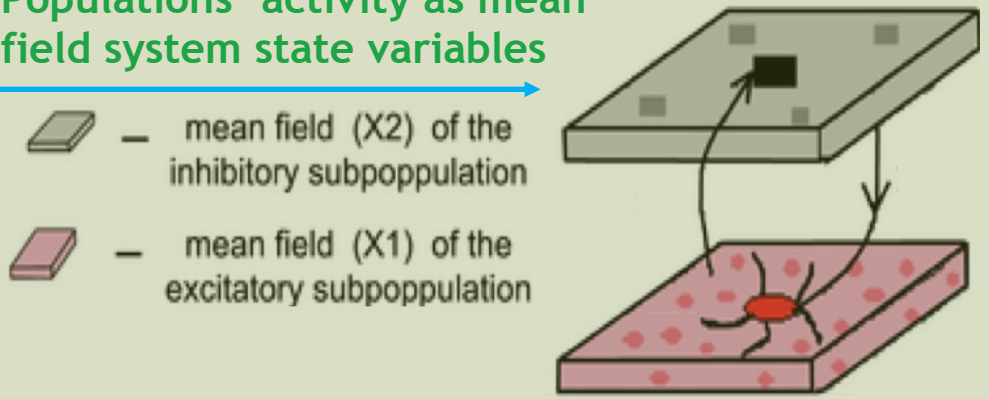
nest :: simulated()

ANNarchy

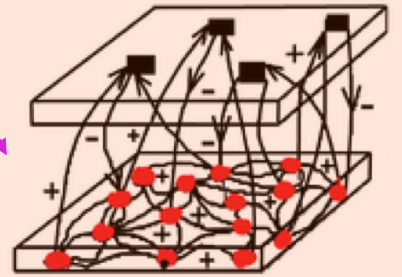
NetPyNE

Populations' level

Populations' activity as mean field system state variables



Average activity of populations of spiking neurons



Neurons' level

nest :: simulated()

ANNarchy

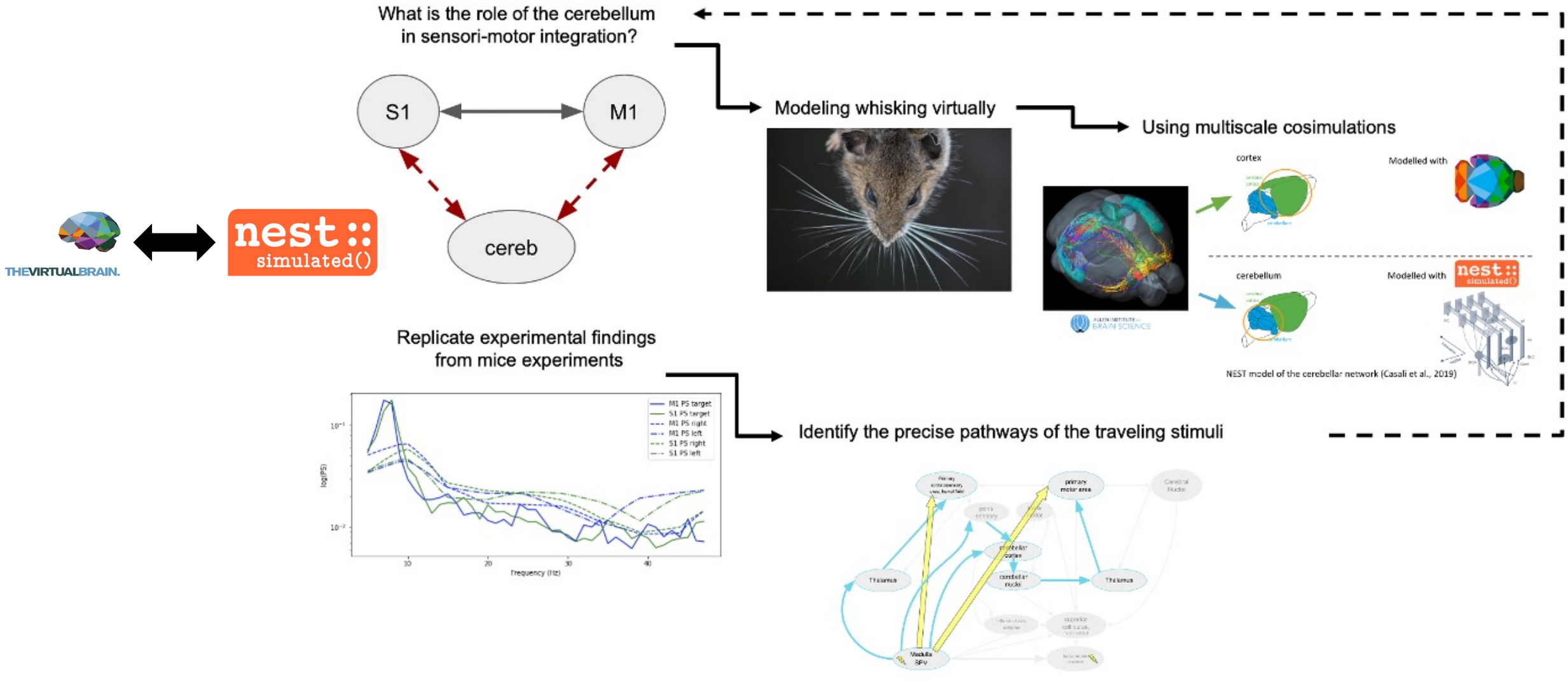
NetPyNE

EBRAINS

First real world scientific applications

Meier, JM*, Geminiani A*, Perdikis, D*, Ouertani S, Casellato C, D'Angelo, E & Ritter, P. in prep. A detailed cerebellar model in whole-brain multiscale cosimulations offers perspective on sensori-motor integration.

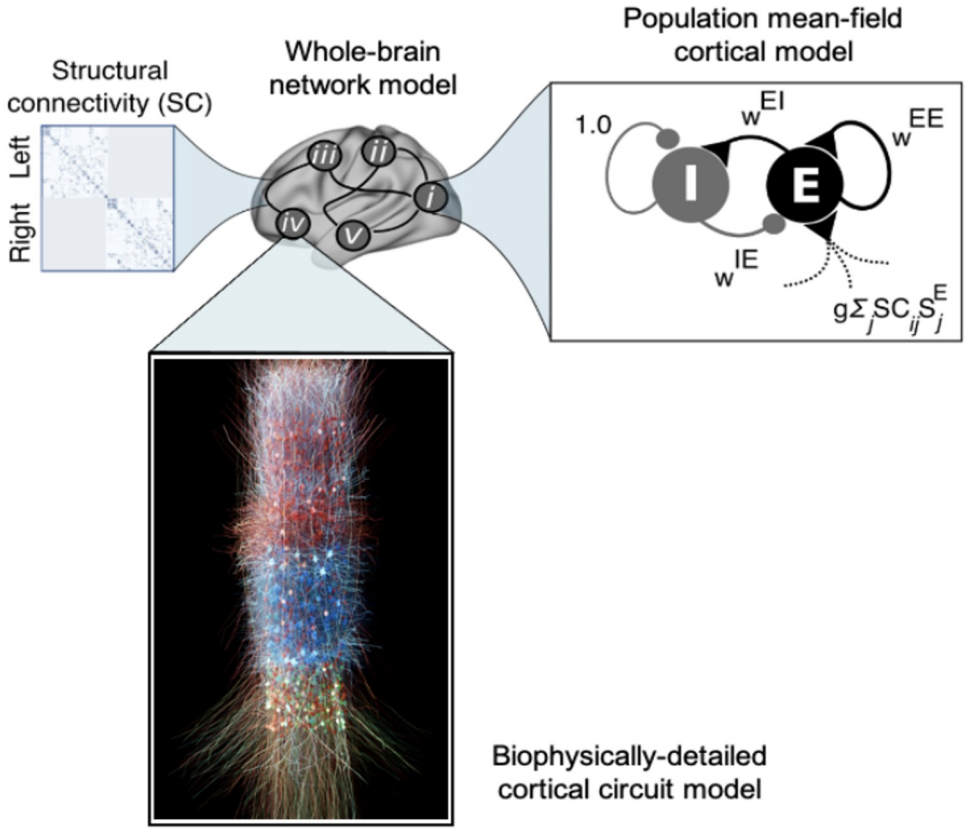
Figure 1: Schematic methods overview.

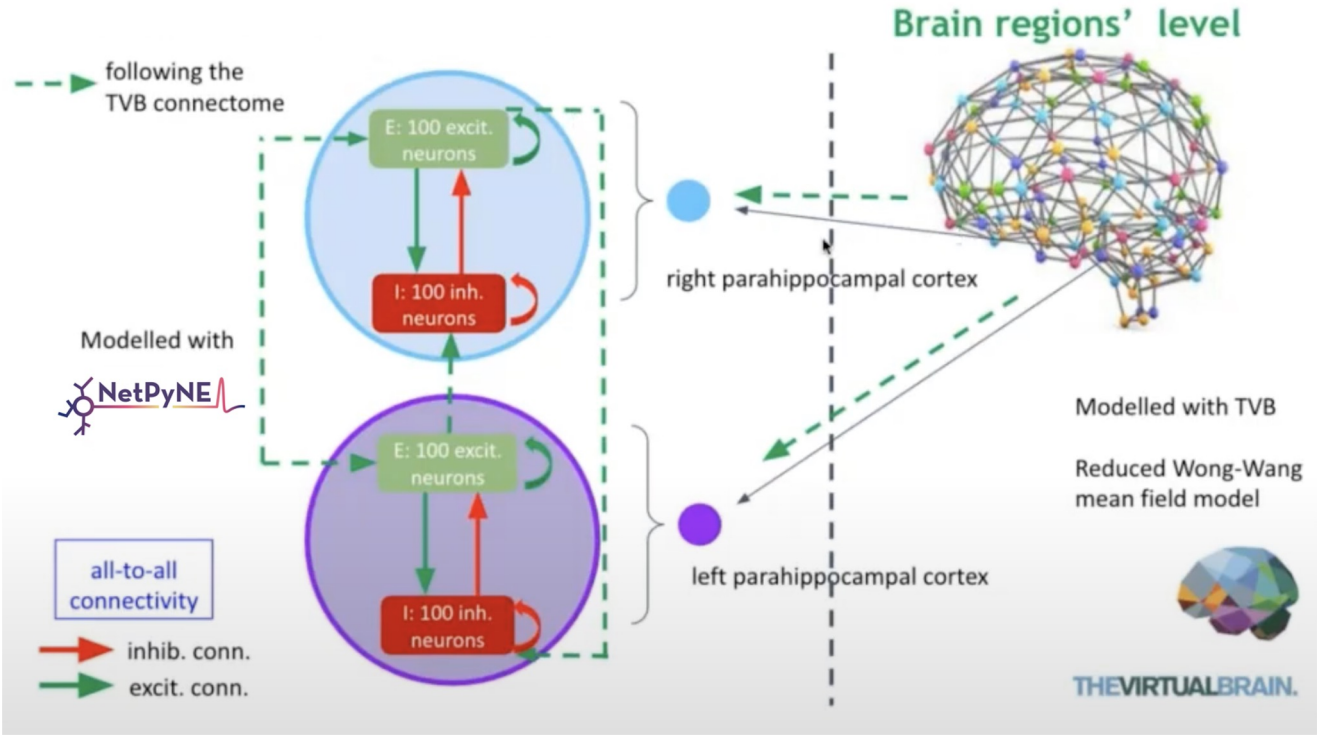


Human Brain Project

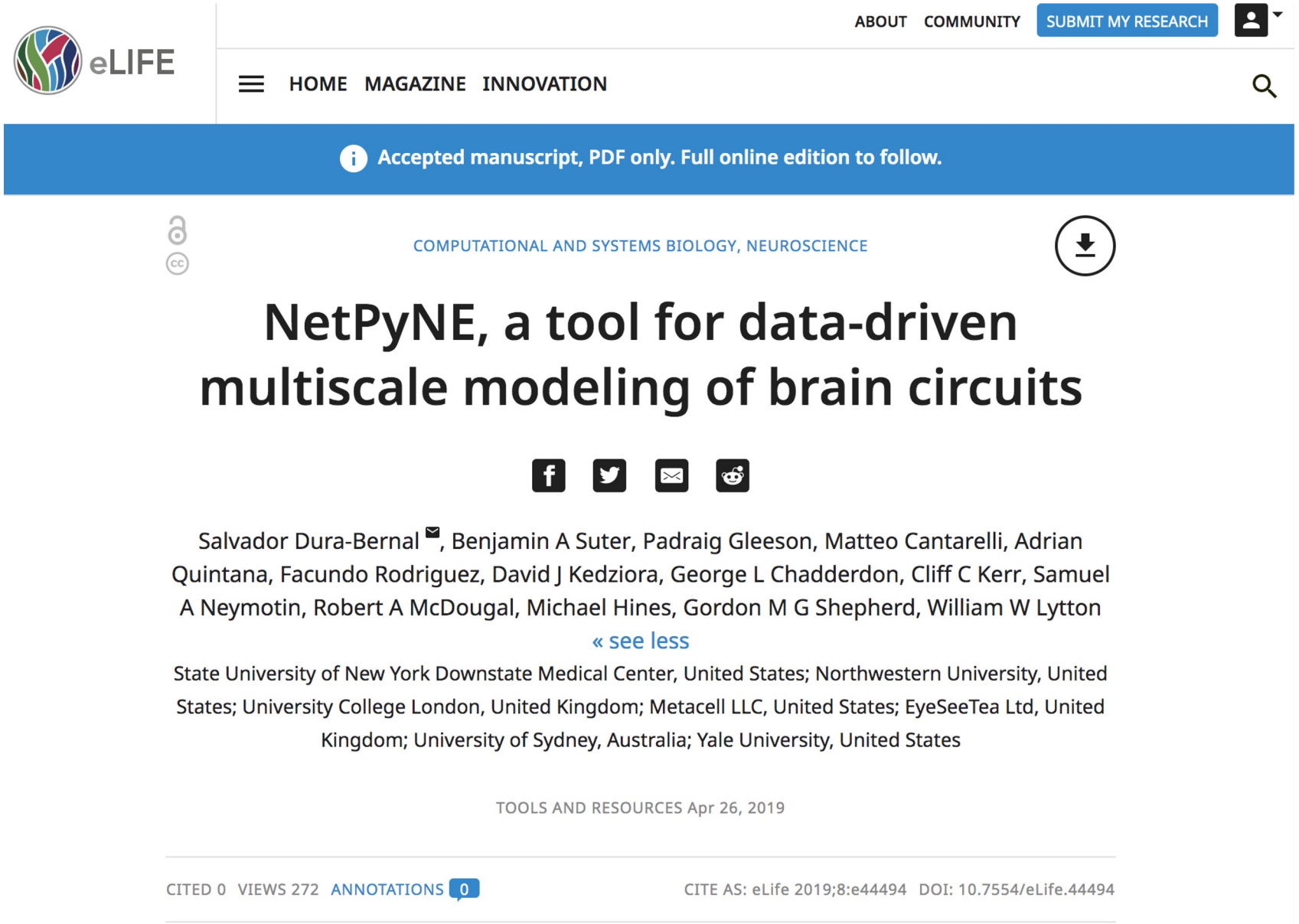
THEVIRTUALBRAIN. ⟷ NetPyNE

TVB-NETPYNE





NetPyNE tool: Publication in eLife



ABOUT COMMUNITY SUBMIT MY RESEARCH

eLIFE

HOME MAGAZINE INNOVATION

Accepted manuscript, PDF only. Full online edition to follow.

COMPUTATIONAL AND SYSTEMS BIOLOGY, NEUROSCIENCE

NetPyNE, a tool for data-driven multiscale modeling of brain circuits

Salvador Dura-Bernal, Benjamin A Suter, Padraig Gleeson, Matteo Cantarelli, Adrian Quintana, Facundo Rodriguez, David J Kedziora, George L Chadderdon, Cliff C Kerr, Samuel A Neymotin, Robert A McDougal, Michael Hines, Gordon M G Shepherd, William W Lytton

« see less

State University of New York Downstate Medical Center, United States; Northwestern University, United States; University College London, United Kingdom; Metacell LLC, United States; EyeSeeTea Ltd, United Kingdom; University of Sydney, Australia; Yale University, United States

TOOLS AND RESOURCES Apr 26, 2019

CITED 0 VIEWS 272 ANNOTATIONS 0

CITE AS: eLife 2019;8:e44494 DOI: 10.7554/eLife.44494

Dura-Bernal, S., Suter, B. A., Gleeson, P., Cantarelli, M., Quintana, A., Rodriguez, F., ... & Lytton, W. W. (2019). NetPyNE, a tool for data-driven multiscale modeling of brain circuits. *Elife*, *8*, e44494. https://elifesciences.org/articles/44494.pdf

EBRAINS

frontiers in Neuroinformatics

ORIGINAL RESEARCH
published: 31 July 2015
doi: 10.3389/fninf.2015.00019

ANNarchy: a code generation approach to neural simulations on parallel hardware

Julien Vitay[1]*, *Helge Ü. Dinkelbach*[1] and *Fred H. Hamker*[1,2]

[1] *Department of Computer Science, Chemnitz University of Technology, Chemnitz, Germany,* [2] *Bernstein Center for Computational Neuroscience, Charité University Medicine, Berlin, Germany*

Many modern neural simulators focus on the simulation of networks of spiking neurons on parallel hardware. Another important framework in computational neuroscience, rate-coded neural networks, is mostly difficult or impossible to implement using these simulators. We present here the ANNarchy (Artificial Neural Networks architect) neural simulator, which allows to easily define and simulate rate-coded and spiking networks, as well as combinations of both. The interface in Python has been designed to be close to the PyNN interface, while the definition of neuron and synapse models can be specified using an equation-oriented mathematical description similar to the Brian neural simulator. This information is used to generate C++ code that will efficiently perform the simulation on the chosen parallel hardware (multi-core system or graphical processing unit). Several numerical methods are available to transform ordinary differential equations into an efficient C++code. We compare the parallel performance of the simulator to existing solutions.

OPEN ACCESS

Edited by:
Andrew P. Davison,
Centre National de la Recherche Scientifique, France

Reviewed by:
Mikael Djurfeldt,
Royal Institute of Technology, Sweden

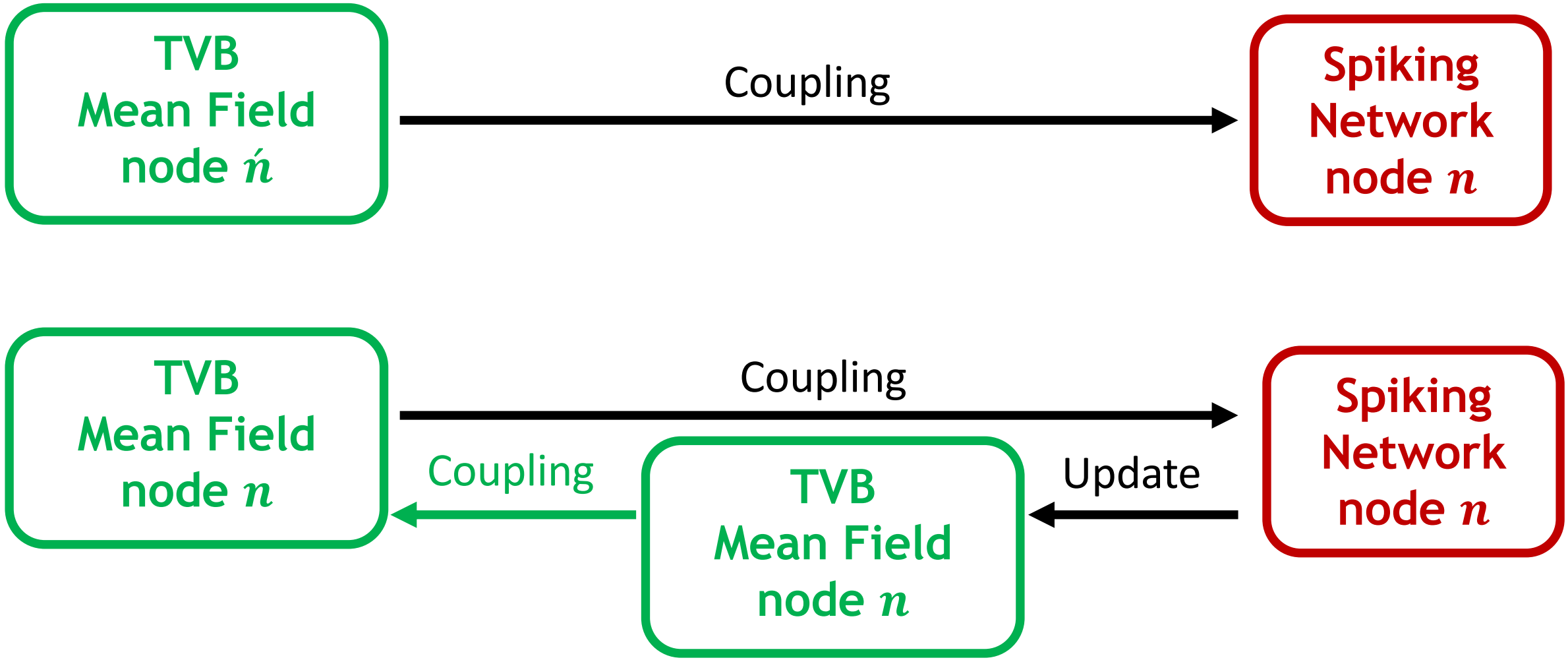
Vitay, J, Dinkelbach, HÜ, & Hamker, FH. 2015. ANNarchy: a code generation approach to neural simulations on parallel hardware. *Frontiers in neuroinformatics*, *9*, 19. https://www.frontiersin.org/articles/10.3389/fninf.2015.00019/full

EBRAINS

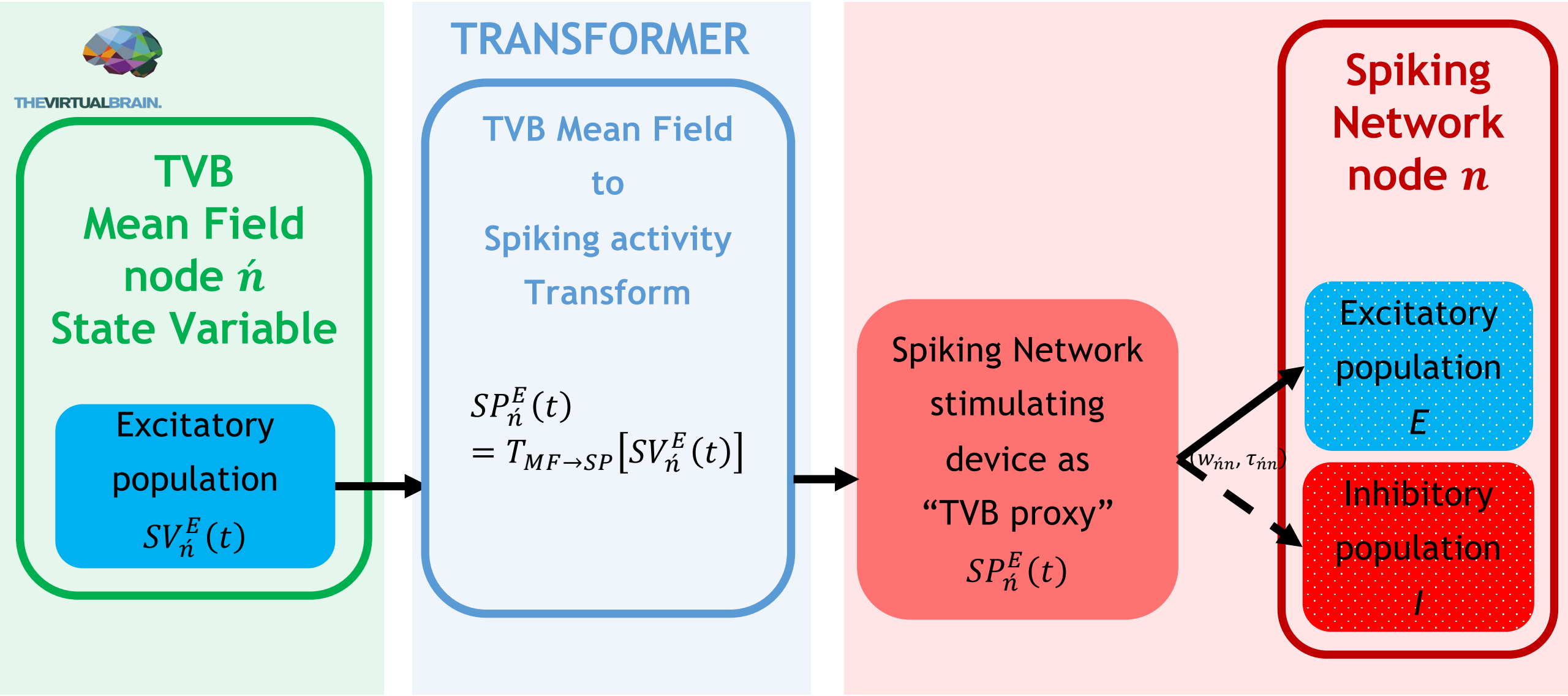
The science behind the implementation
…and some engineering!

EBRAINS

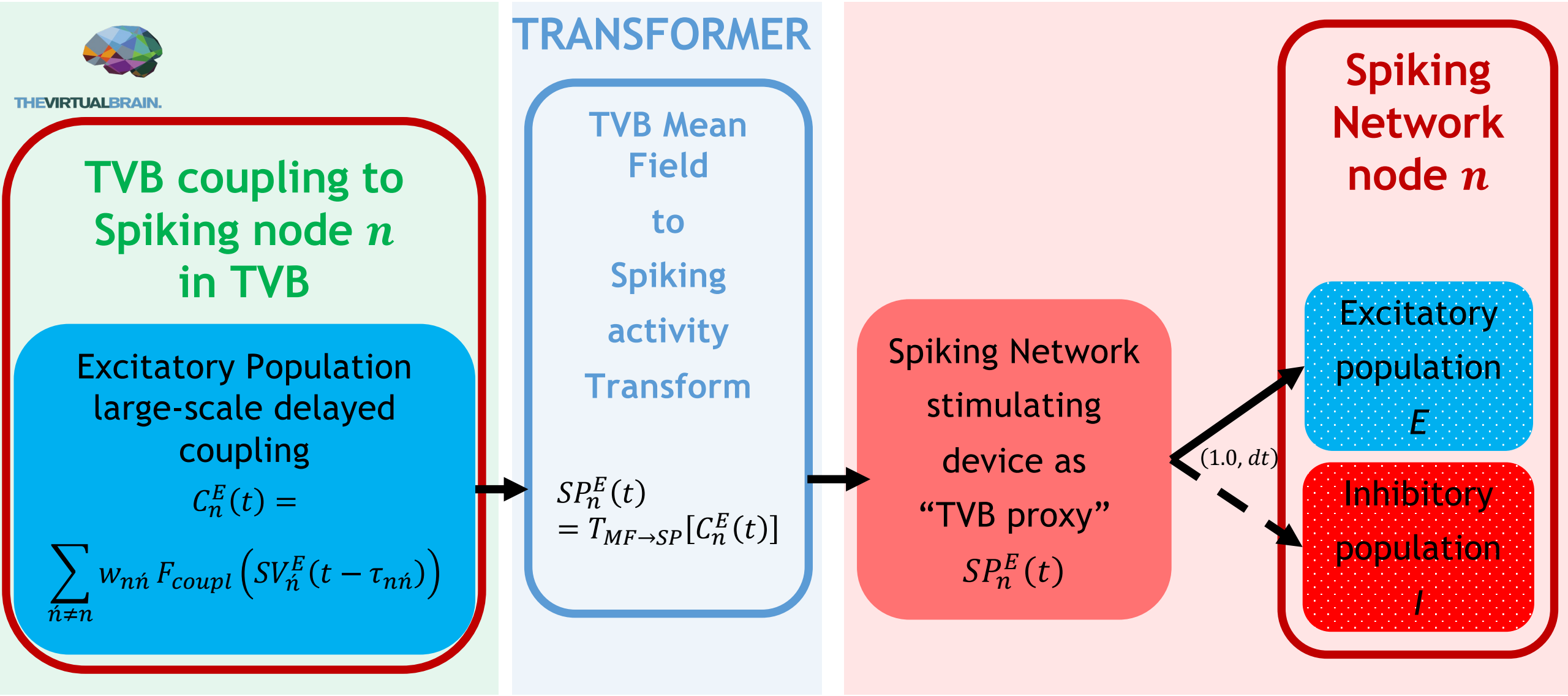
Multiscale Co-Simulation Scenarios



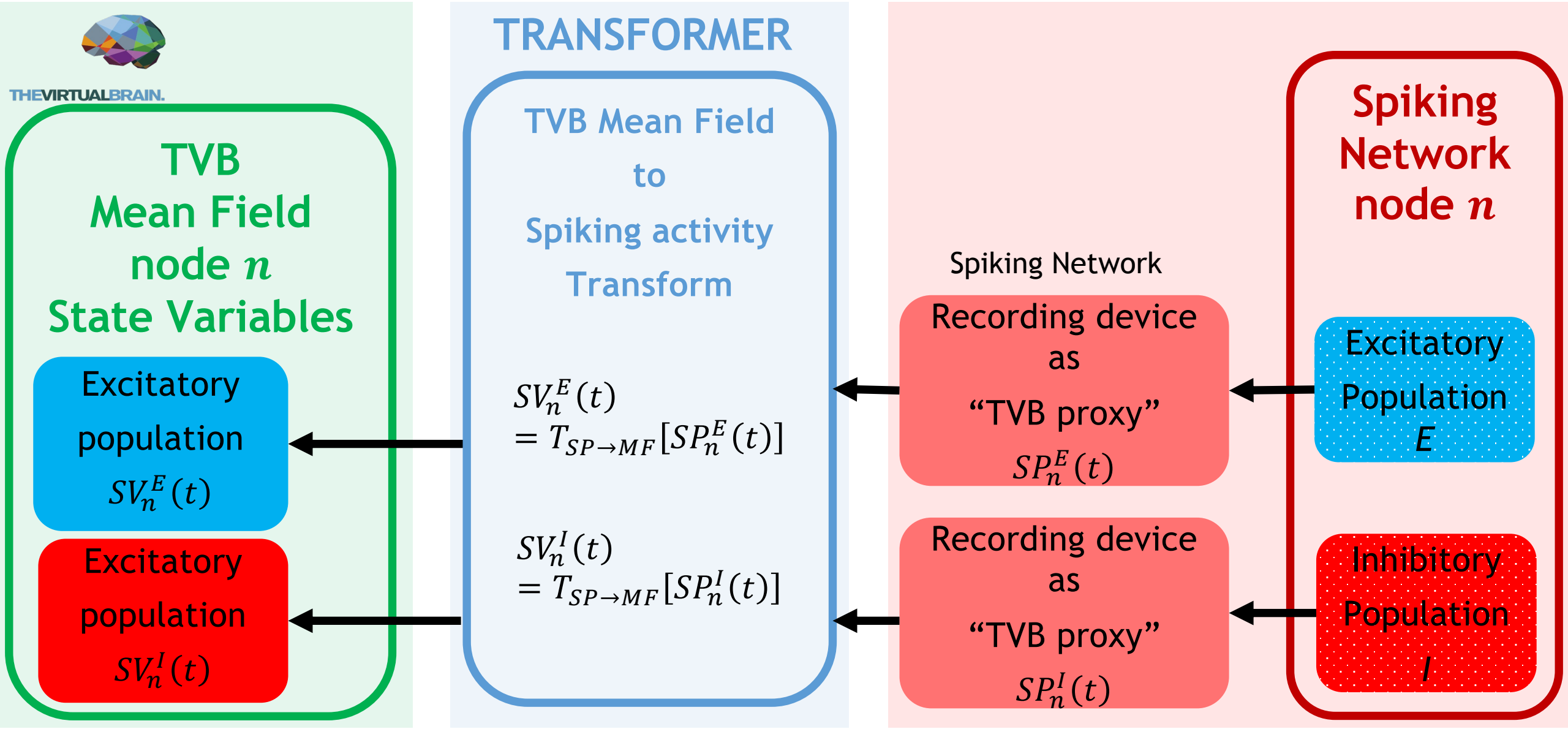
TVB -> Spiking Network Coupling



TVB coupling -> Spiking Network



Spiking Network -> TVB Update



EBRAINS

The tvb-multiscale toolbox

https://github.com/the-virtual-brain/tvb-multiscale

docker hub

https://hub.docker.com/r/thevirtualbrain/tvb-multiscale

Component Ecosystem

1. TVB Model and Simulator
2. Spiking Network Model (consisting of Neurons, Populations, Brain Regions/Nodes, recording and stimulating Devices) and Simulator
3. TVB proxy nodes inside the Spiking Network as Spiking Devices and/or Populations
4. Communicators (Senders, Receivers, via MPI, files, shared memory)
5. Transformers (Rate-to-Spikes, Spikes-to-Rates, Current-to-Current etc)
6. Interfaces (combinations of Communicators and Transformers attached to TVB and/or Spiking Network)

Spiking Network classes:

- addressing spiking network in terms of neuronal populations residing at specific brain region nodes
- Direct access to the spiking simulator module (i.e., imported NEST or ANNarchy)
- Wrapping around NEST/ANNarchy/NETPYNE classes and performing basing functions such as getting or setting properties and parameters recursively, returning the neuronal global/local indices, returning the connections to/from etc

1. **SpikingNodeCollection**
   (e.g., NESTNodeCollection wraping around a NEST NodeCollection and ANNarchyPopulation around an ANNarchyPopulation)
2. **SpikingPopulation(SpikingNodeCollection)**
   (and NESTPopulation(NESTNodeCollection, SpikingPopulation), ANNarchyPopulation(ANNarchyPopulation, SpikingPopulation)): representing a neuronal population of the same model and label, residing at a specific brain region node.
3. **Device(SpikingNodeCollection)**
   (and NESTDevice(Device, NESTNodeCollection), ANNarchyDevice(Device, ANNarchyPopulation)
4. **NodeSet(pandas.Series)**
   (a pandas.Series of SpikingNodeCollection instances)
5. **SpikingRegionNode(NodeSet)**
   (and NESTRegionNode, ANNarchyRegionNode; a NodeSet holding SpikingPopulations)
6. **SpikingBrain(NodeSet)**
   (and NESTBrain, ANNarchyBrain; a NodeSet holding SpikingRegionNodes)
7. **DeviceSet(NodeSet)**
   (i.e., a NodeSet holding **Device instances, each of which couple to specific populations of a SpikingRegionNode**)

Spiking Network class

SpikingNetwork (and NESTNetwork, ANNarchyNetwork):

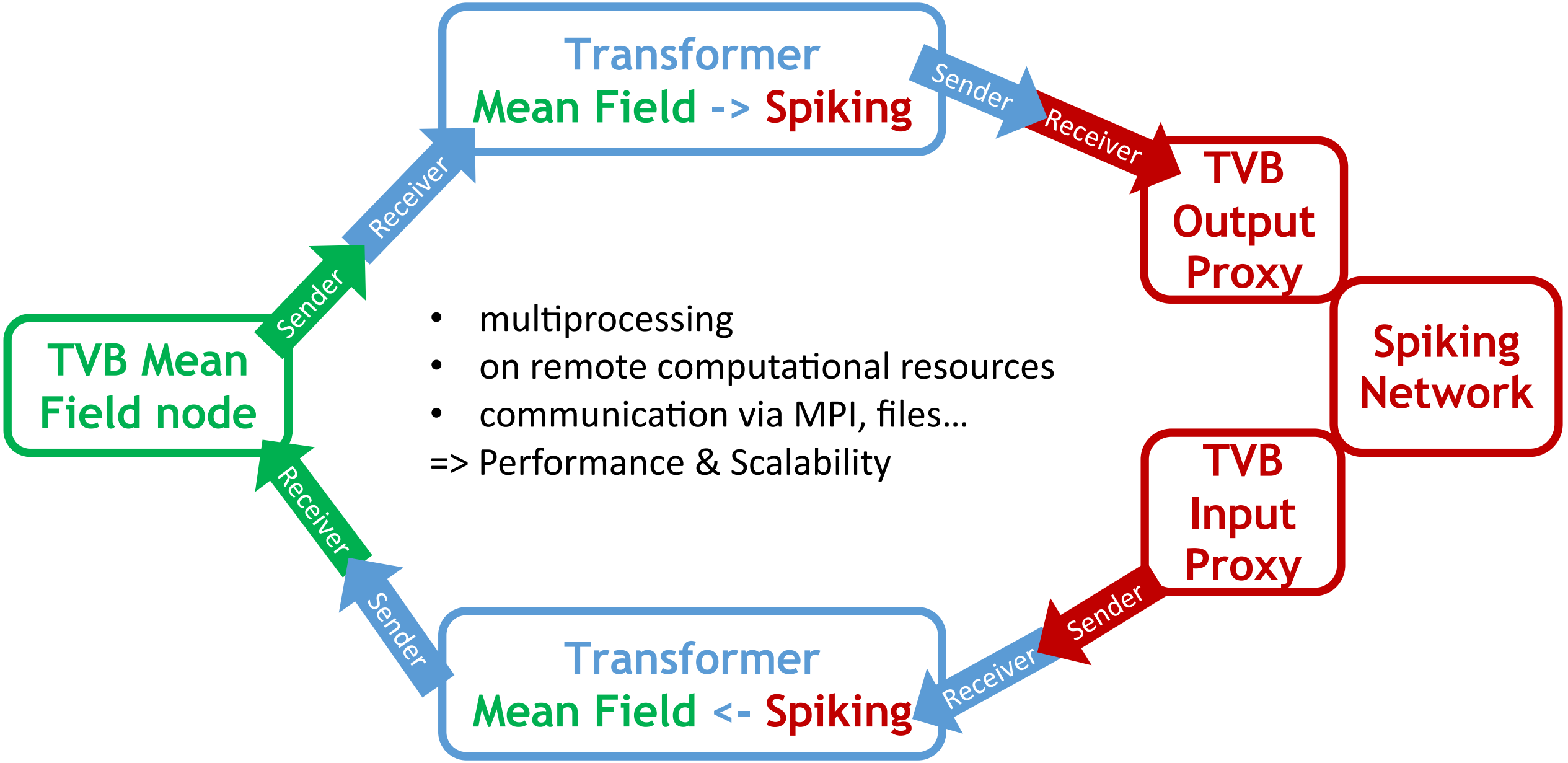
- **brain_regions: SpikingBrain (NESTBrain, ANNarchyBrain)**
- **input_devices: pandas.Series holding stimulating DeviceSet instances**
- **output_devices: pandas.Series holding recording DeviceSet instances**
- **input_proxies: pandas.Series holding stimulating DeviceSet instances used as TVB proxies**
- **output_proxies: pandas.Series holding recording DeviceSet instances used as TVB proxies**

+ Methods to recursively address classes of brain regions, neuronal populations or devices,

down to specific populations or devices that stimulate / record from specific populations and/or regions.

Human Brain Project

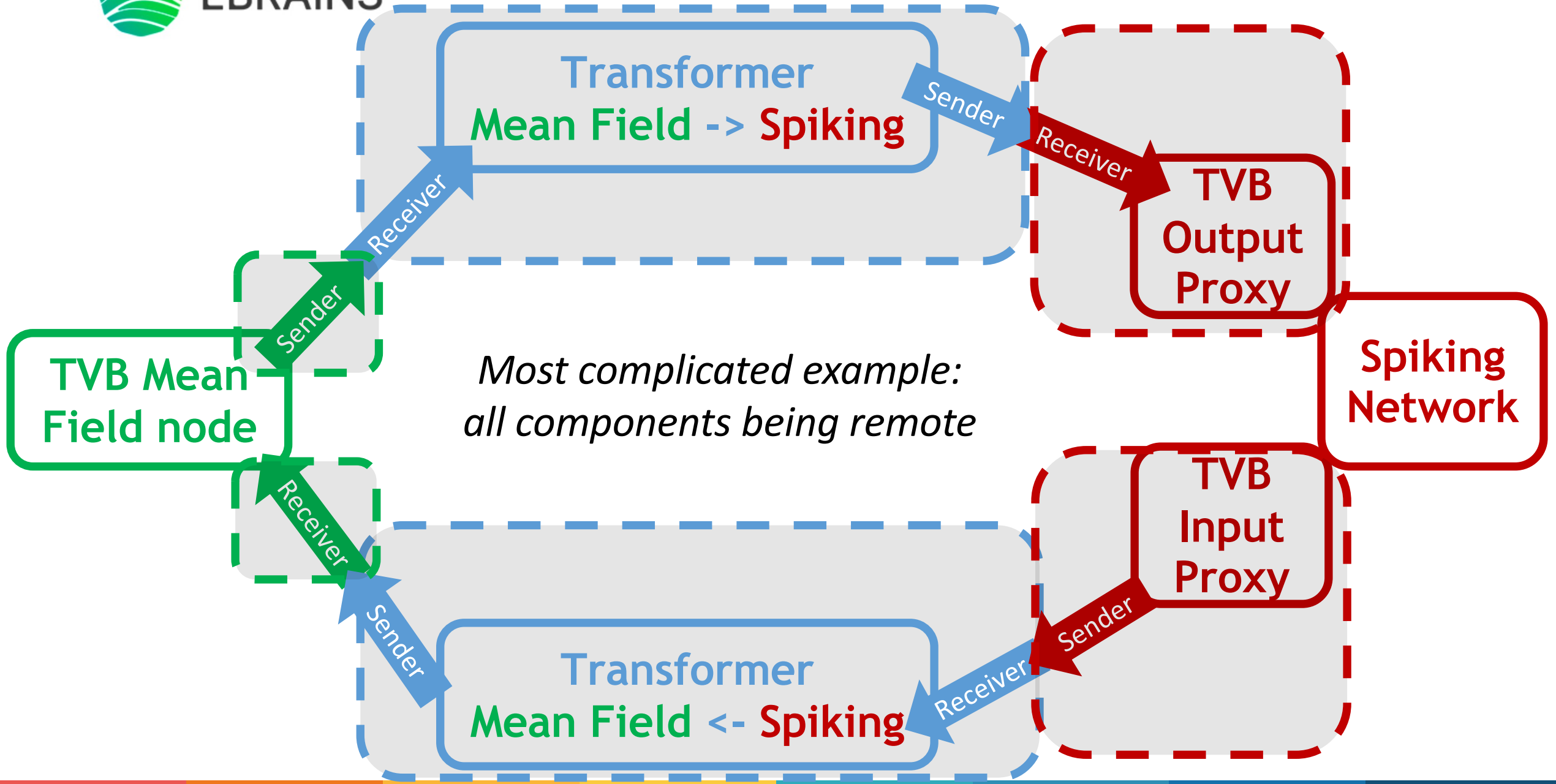
Modular interfaces



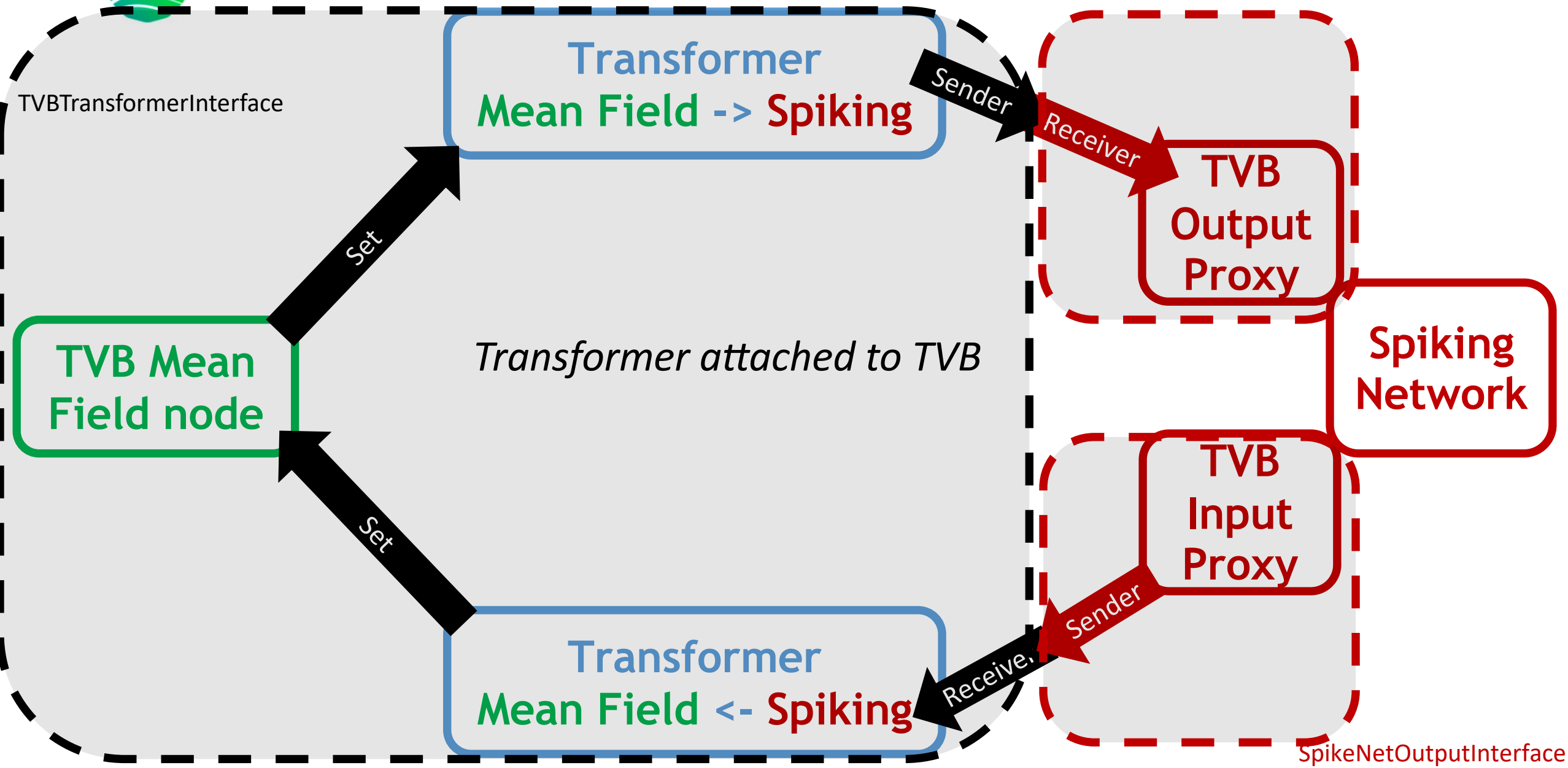
- multiprocessing
- on remote computational resources
- communication via MPI, files…

=> Performance & Scalability

Modular interfaces



Modular interfaces

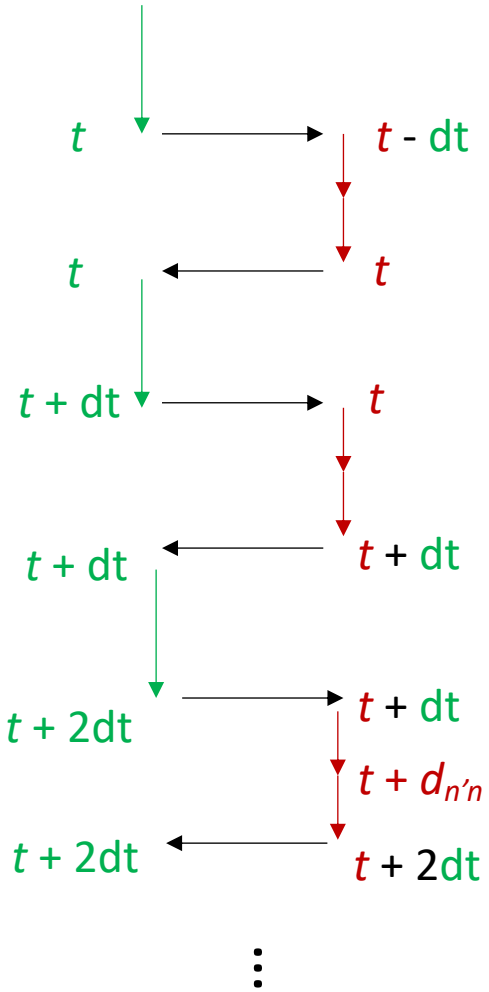


Synchronization

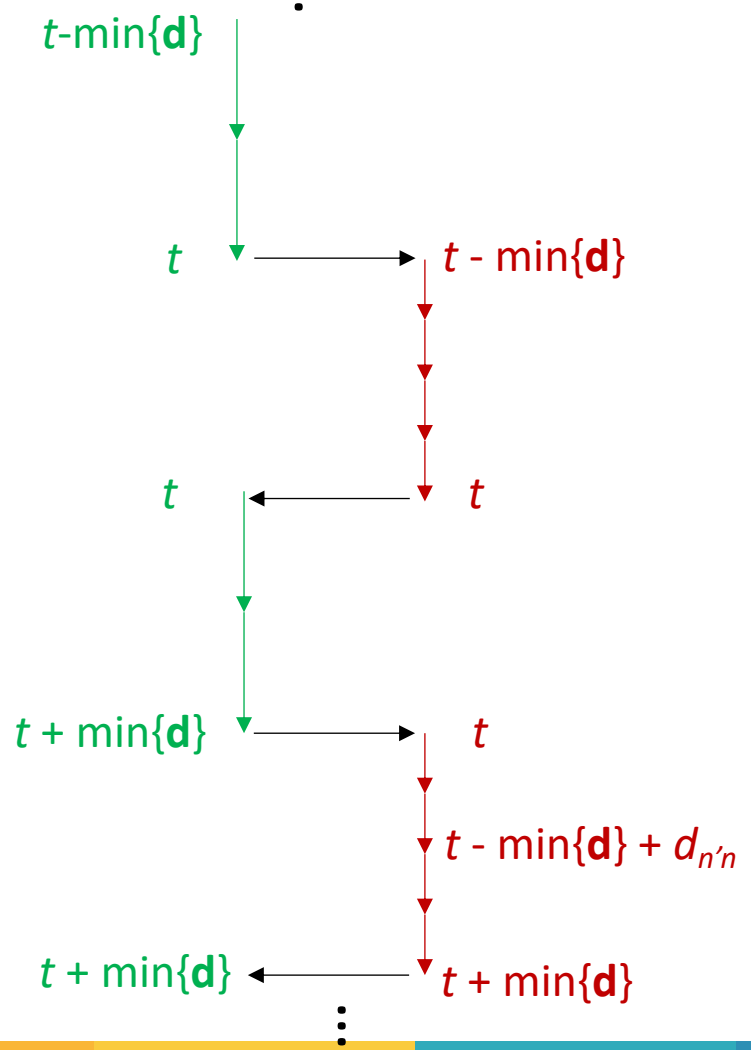
TVB dt

Spiking Network dt

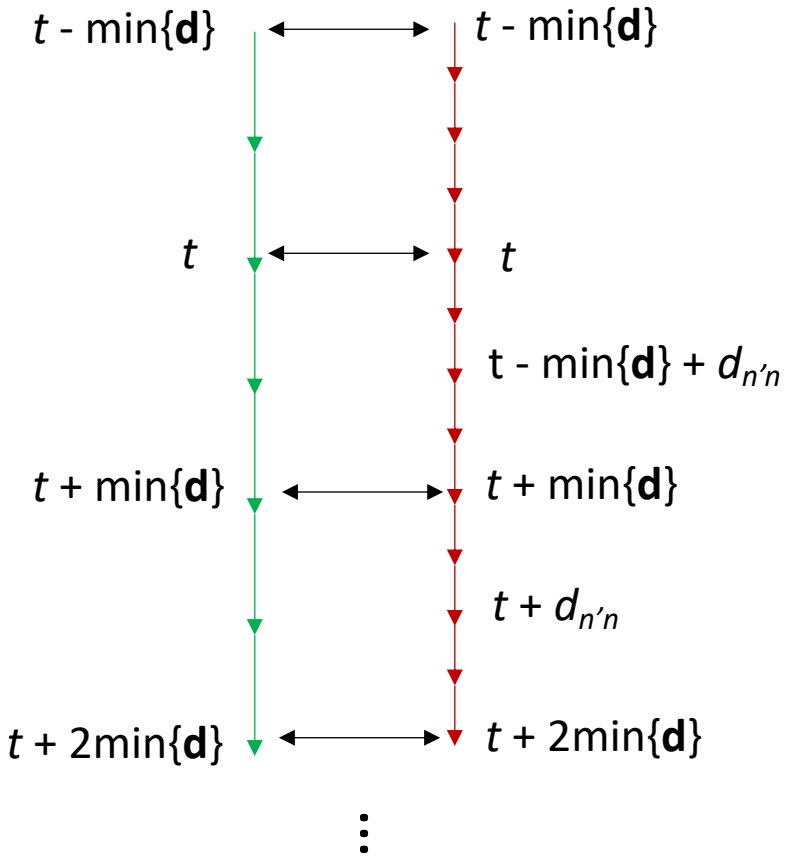
serial cosimulation, every TVB dt



serial cosimulation, every minimum TVB delay



parallel cosimulation, every minimum TVB delay



EBRAINS

User API

1. TVB Model and Simulator **Builder**
2. Spiking Network **Builders**
3. TVB proxy nodes inside the Spiking Network **Builders**
4. Communicators **Builders**
5. Transformers **Builders**
6. Interfaces **Builders** (combinations of Communicators Builders and Transformers Builders attached to TVB and/or Spiking Network)
7. TVBApps to configure, build and run TVB simulation
8. SpikeNetApps to configure, build and run Spiking Network simulation
9. Orchestrator/Simulation manager Apps to configure, build all models and simulate

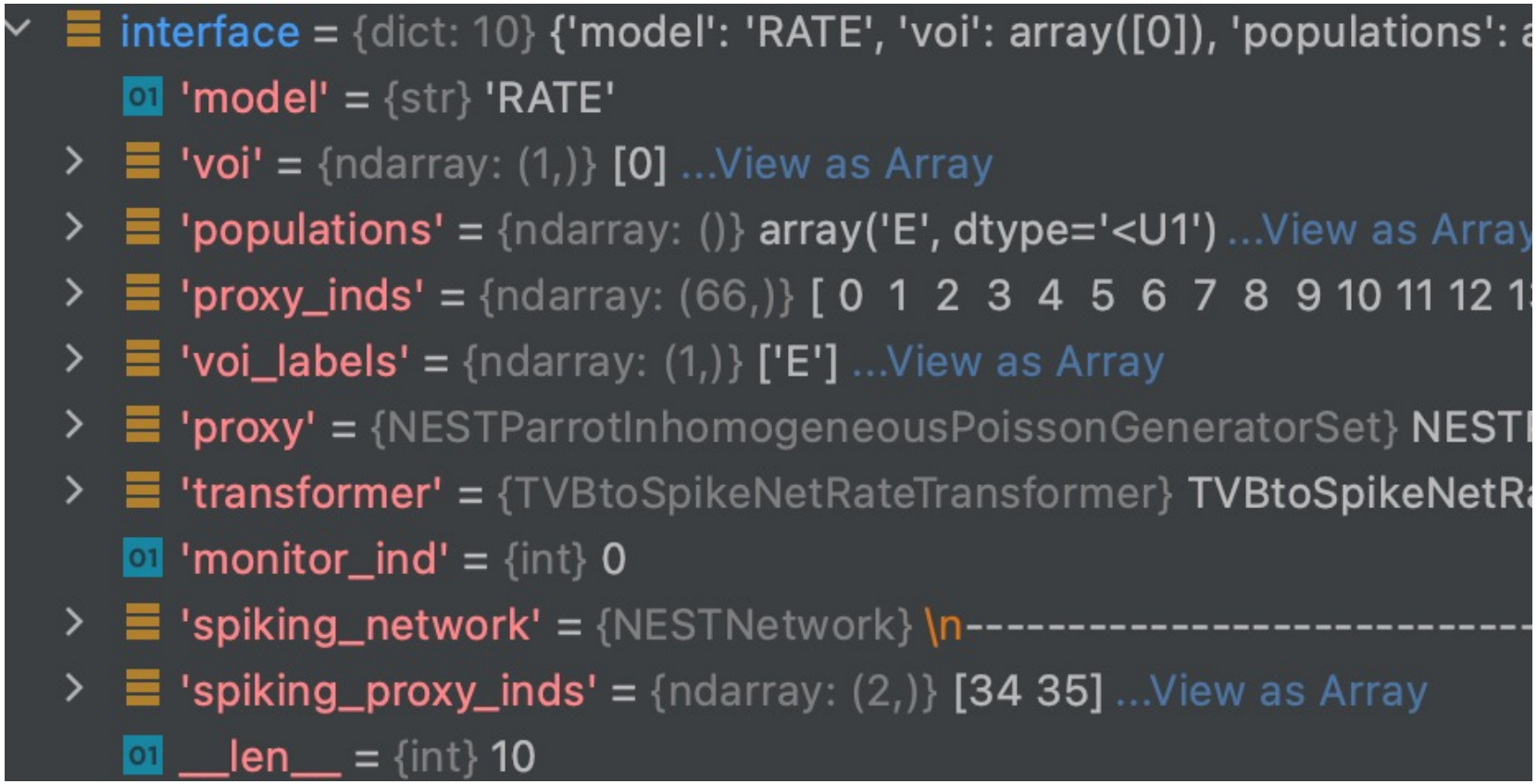
EBRAINS

Example: Code input for configuration of a TVB<->NEST single process interface

```
tvb_to_nest_interfaces = [{"model": "RATE",                          # will determine proxy and transformer defaults
                           "voi": "E",                               # TVB state variable of interest as output to NEST
                            "populations": "E",                      # NEST population label receiving TVB input
                           "coupling_mode " : "spikeNet ",           # “TVB” or something else (e.g., “NEST”, ”spikeNET”)
                            "proxy_params":                          # Parameters for the TVB proxy in NEST
                                           {"number_of_neurons": 100} ,
                          "transformer_params":                      # Parameters for the Rate-to-Spikes transform
                                           {"scale_factor": np.array([1000.0])}}]


nest_to_tvb_interfaces = [{"voi": "E",                                   # TVB state variable of interest to receive NEST input
                           "populations": "E",                           # NEST population label updating TVB state variable
                           "transformer_params":                 # Parameters for the Spikes-to-Rate transform
                                           {"scale_factor": np.array([0.001])/100}}
```

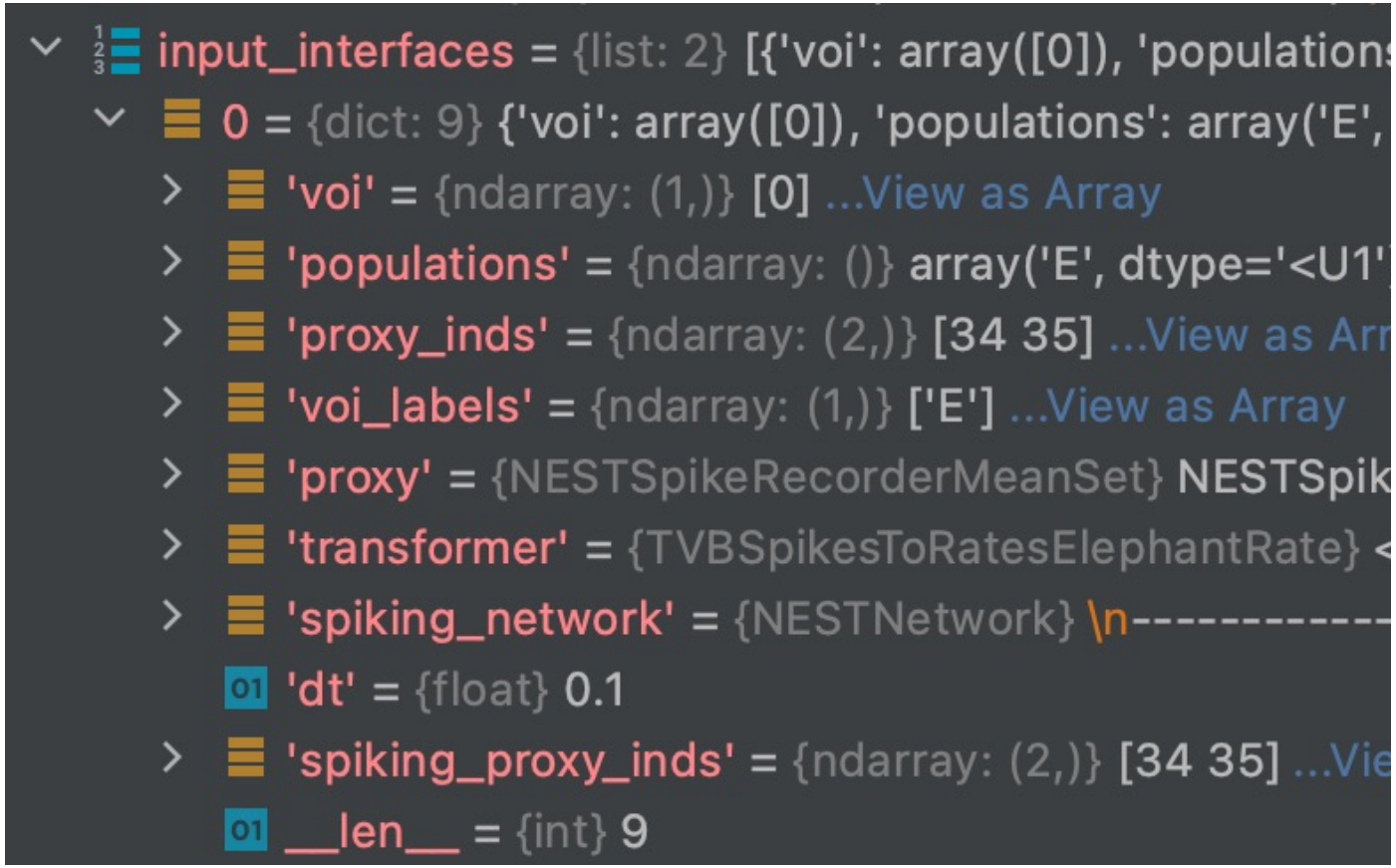
EBRAINS

.he interfaces configured: TVB -> NEST



EBRAINS

The interfaces configured : NEST <- TVB



EBRAINS

Default workflow

1. **Setup configuration**
   (NEST min_delay, TVB and Spiking Simulator integration time resolution etc)
2. **Build TVB model and simulator**
   (connectivity, mean field model, coupling, integrator, monitors etc)
3. **Build Spiking Network model**
   (neural populations, connections, stimulation and measuring devices)
3. **Build interfaces**
   (mappings: state variable <-> neural population,
   transformations: mean-field <-> spike activity)
4. **(Co-)Simulate**
5. Gather results, analyze, plot, write to .h5 files

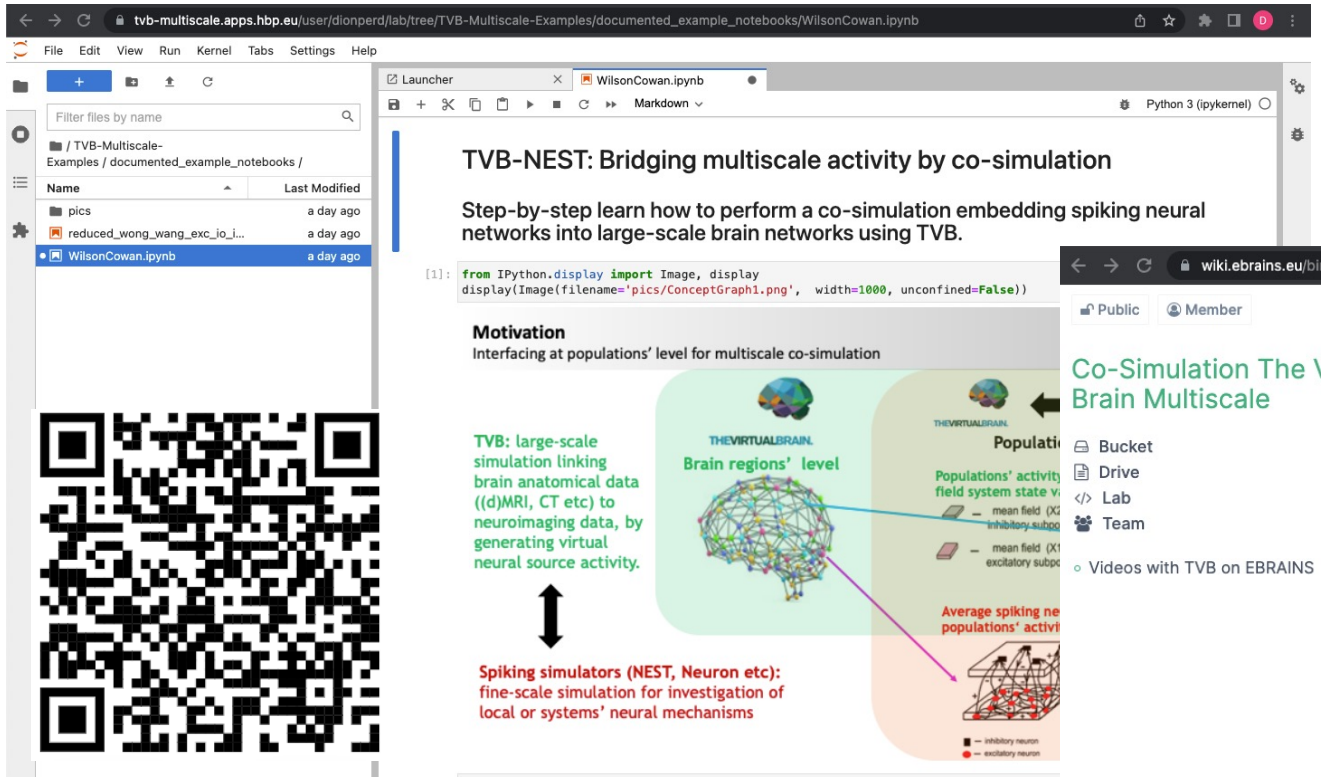
EBRAINS

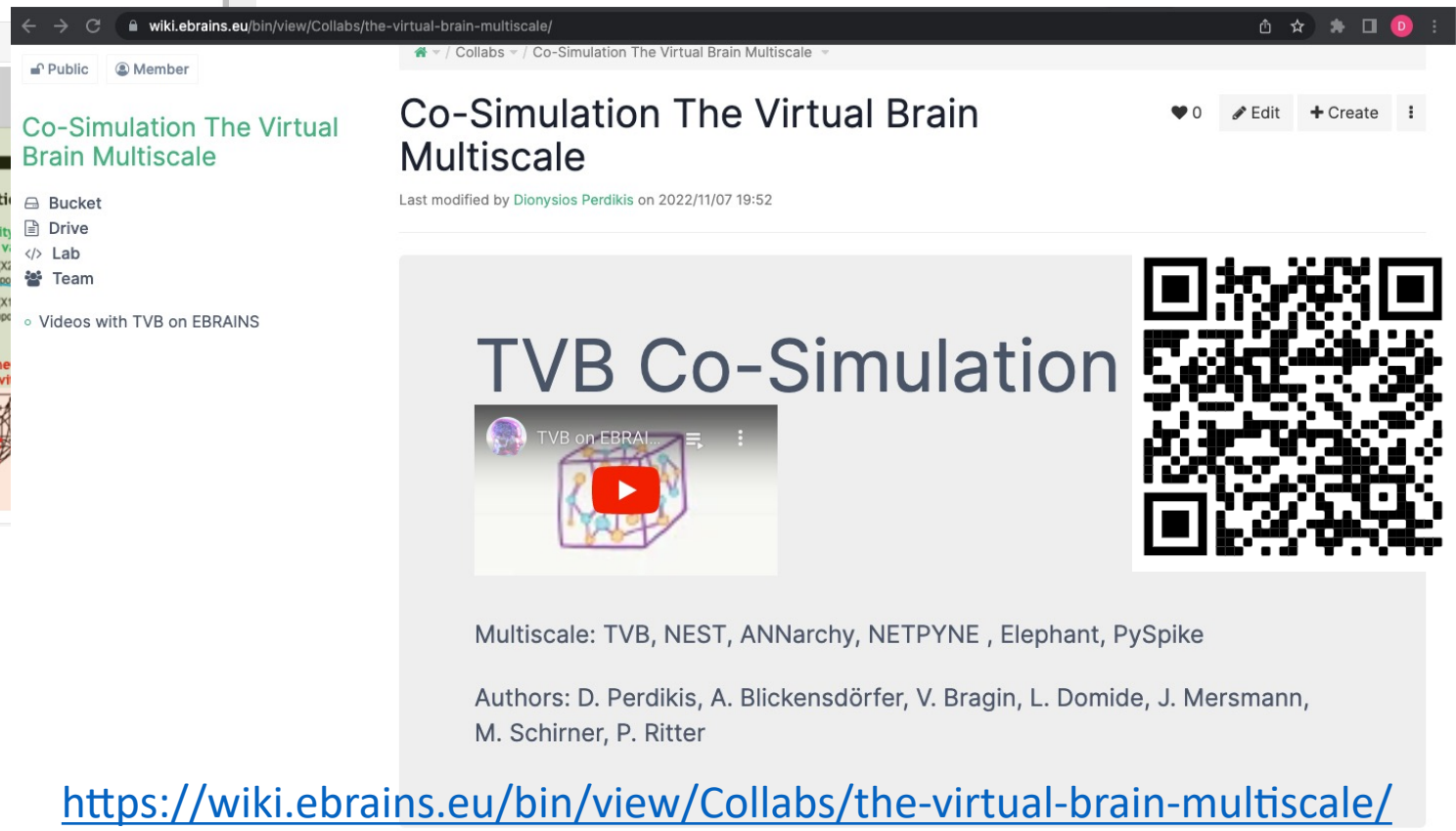
Using Orchestrator and TVB/SpikeNet Apps for a co-simulation workflow

```
orchestrator = TVBNESTSerialOrchestrator(
    config=config,
    logger=logger,
    exclusive_nodes=exclusive_nodes,
    spiking_proxy_inds=np.array(nest_nodes_inds),
    simulation_length=simulation_length
)
orchestrator.start()

# ----------------------------------1. Models' and simulation configuration ----------------------------

# ---------------------------------------a. Configure a TVB simulator builder ----------------------
orchestrator.tvb_app.cosimulator_builder.model = tvb_sim_model()
orchestrator.tvb_app.cosimulator_builder.model_params = model_params
orchestrator.tvb_app.cosimulator_builder.connectivity = connectivity
orchestrator.tvb_app.cosimulator_builder.delays_flag = delays_flag

# ---------------------------------------b. Configure the NEST network model builder-----------------
orchestrator.spikeNet_app.spikeNet_builder = nest_model_builder(config=config)
orchestrator.spikeNet_app.population_order = nest_populations_order

# ---------------------------------------c. Configure the TVB-NEST interface model ------------------
orchestrator.tvb_app.interfaces_builder.output_interfaces = tvb_to_nest_interfaces
orchestrator.tvb_app.interfaces_builder.input_interfaces = nest_to_tvb_interfaces

# ---------------------------------------d. Run the orchestrator configuration ----------------------
orchestrator.configure()

# # ----------------------------------3. Build models, simulators and interfaces----------------------
orchestrator.build()

# ----------------------------------4. Configure, Simulate and gather results----------------------
orchestrator.simulate()
simulator = orchestrator.tvb_cosimulator
results = orchestrator.tvb_app.results
```

Collaboratory and App

https://tvb-multiscale.apps.hbp.eu/





https://wiki.ebrains.eu/bin/view/Collabs/the-virtual-brain-multiscale/

Future work

- Validation and comparative analysis of the different coupling schemes between the coarse (MF)- and fine- scale (SP) models.
- Optimization of TVB-Spiking Network co-simulation to increase speed as well as parallelization and optimal use of EBRAINS hardware resources.
- Improve documentation, test coverage and usability of TVB-multiscale in CSCS cluster and EBRAINS Collaboratory.
- Use the toolbox in real-world scientific problems.

References

1. Ritter P, Schirner M, McIntosh AR, Jirsa VK. 2013. The Virtual Brain integrates computational modeling and multimodal neuroimaging. *Brain Connectivity* 3:121–145.

2. Sanz Leon P, Knock SA, Woodman MM, Domide L, Mersmann J, McIntosh AR, Jirsa V. 2013. The Virtual Brain: a simulator of primate brain network dynamics. *Frontiers in Neuroinformatics* 7:10.

3. Jordan, Jakob, Mørk, Håkon, Vennemo, Stine Brekke, Terhorst, Dennis, Peyser, Alexander, Ippen, Tammo, … Plesser, Hans Ekkehard. (2019, June 27). NEST 2.18.0 (Version 2.18.0). Zenodo.

4. Schirner, M, Domide, L, Perdikis, D, Triebkorn, P, Stefanovski, L, Pai, R, ... & Ritter, P. 2022. Brain simulation as a cloud service: The Virtual Brain on EBRAINS. *NeuroImage*, *251*, 118973.

5. Vitay, J, Dinkelbach, HÜ, & Hamker, FH. 2015. ANNarchy: a code generation approach to neural simulations on parallel hardware. *Frontiers in neuroinformatics*, *9*, 19.

6. Meier, JM, Perdikis, D, Blickensdörfer, A, Stefanovski, L, Liu, Q, Maith, O, ... & Ritter, P. 2022. Virtual deep brain stimulation: Multiscale co-simulation of a spiking basal ganglia model and a whole-brain mean-field model with the virtual brain. *Experimental Neurology*, 114111.

7. Dura-Bernal, S, Suter, BA, Gleeson, P, Cantarelli, M, Quintana, A, Rodriguez, F, ... & Lytton, WW. 2019. NetPyNE, a tool for data-driven multiscale modeling of brain circuits. *Elife*, 8, e44494. https://elifesciences.org/articles/44494.pdf

8. Perdikis D, Schirner M, Domide L, Mersmann J, Ritter P (*in prep*).

TVB -> ANNarchy Coupling

THEVIRTUALBRAIN.

TVB Mean Field node $\acute{n}$ Rate

Population $R_{\acute{n}}(t)$

TRANSFORMER

TVB Mean Field to ANNarchy Spiking activity Transform

$$T^{np}_{MF\to SP}[R_{\acute{n}}(t)] = aG\bar{F}_{np}R_{\acute{n}}(t)$$

$$\bar{F}_{np} = \frac{\boldsymbol{F_{np}}}{\sum_{\acute{n}} w_{n\leftarrow\acute{n}}}$$

ANNarchy

ANNarchy Homogeneous CorrelatedSpikeTrains population

$$\frac{dx}{dt} = \frac{\mu - x}{\tau} + \sigma\frac{\xi}{\sqrt{\tau}}$$

as "TVB proxy"

$$SPIKES_{\acute{n}}(t) = \left\{\delta\left(t - \left(\tau_{n\acute{n}} + t_i^k\right)\right)\right\}_{i\in[1,N_{CTX}]}^{t^k\in(t,t+dt)}$$

$w_{CTX\to n} * w_{n\acute{n}}$, $\tau_{n\acute{n}}$

ANNarchy Spiking Network node $n \in \{STN, SN\}$

Population $p \in \{STN, dSN, iSN\}$

ANNarchy -> TVB Update

EBRAINS

THEVIRTUALBRAIN.

TVB Mean Field node n State Variables

Population

$\left(S_n(t), R_n(t)\right)$

TRANSFORMER

TVB Mean Field to ANNarchy Spiking activity Transform

$$\left(S_n(t), R_n(t)\right) = T_{SP \to MF}\left[SPIKES_n^p(t)\right]:$$

$$\tau_R \dot{R_n} = -R_n + \frac{\sum_{n,p} SPIKES_n^p(t)}{N_p}$$

$$\dot{S_n} = -S_n(t)/\tau_E + (1 - S_n)\gamma R_n$$

ANNarchy

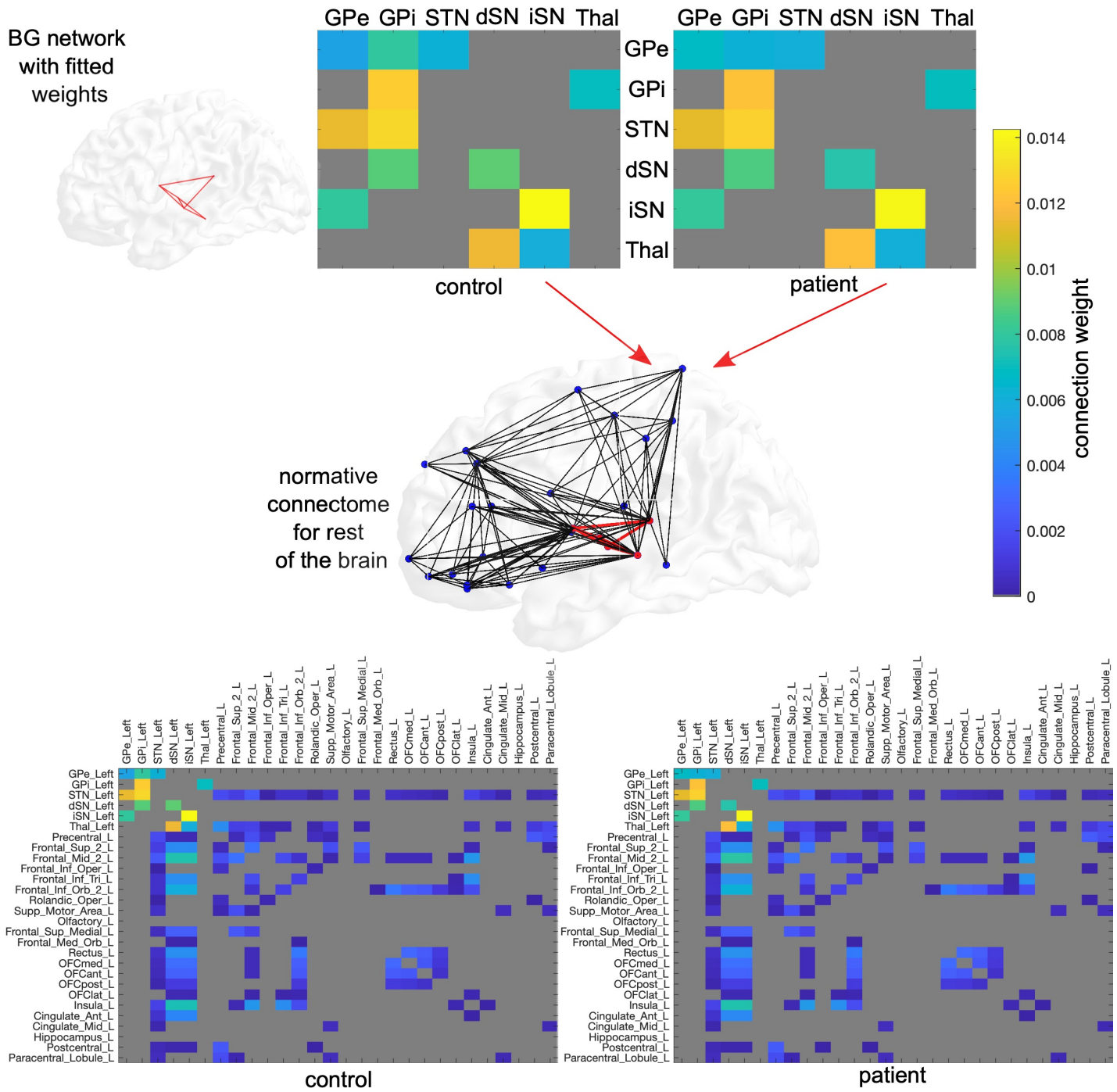
ANNarchy SpikeMonitor as "TVB proxy"

$$SPIKES_n^p(t) = \left\{\delta\left(t - \left(\tau_{\acute{n}n} + t_i^k\right)\right)\right\}_{i \in [1, N_p]}^{t^k \in (t, t+dt)}$$

ANNarchy Spiking Network node $n \in \{STN, SN\}$

Population

$p \in \{STN, dSN, iSN, GPe, GPi, Thal\}$


BG network with fitted weights
GPe GPi STN dSN iSN Thal
GPe
GPi
STN
dSN
iSN
Thal
control
patient
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
connection weight
normative connectome for rest of the brain
GPe_Left
GPi_Left
STN_Left
dSN_Left
iSN_Left
Thal_Left
Precentral_L
Frontal_Sup_2_L
Frontal_Mid_2_L
Frontal_Inf_Oper_L
Frontal_Inf_Tri_L
Frontal_Inf_Orb_2_L
Rolandic_Oper_L
Supp_Motor_Area_L
Olfactory_L
Frontal_Sup_Medial_L
Frontal_Med_Orb_L
Rectus_L
OFCmed_L
OFCant_L
OFCpost_L
OFClat_L
Insula_L
Cingulate_Ant_L
Cingulate_Mid_L
Hippocampus_L
Postcentral_L
Paracentral_Lobule_L
control
patient