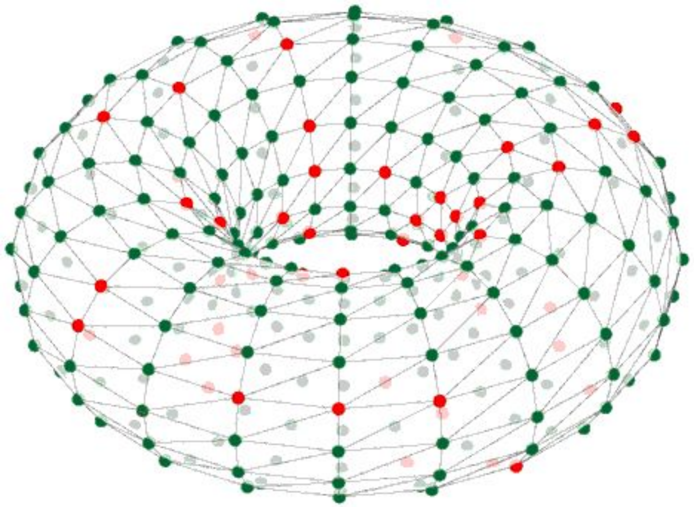
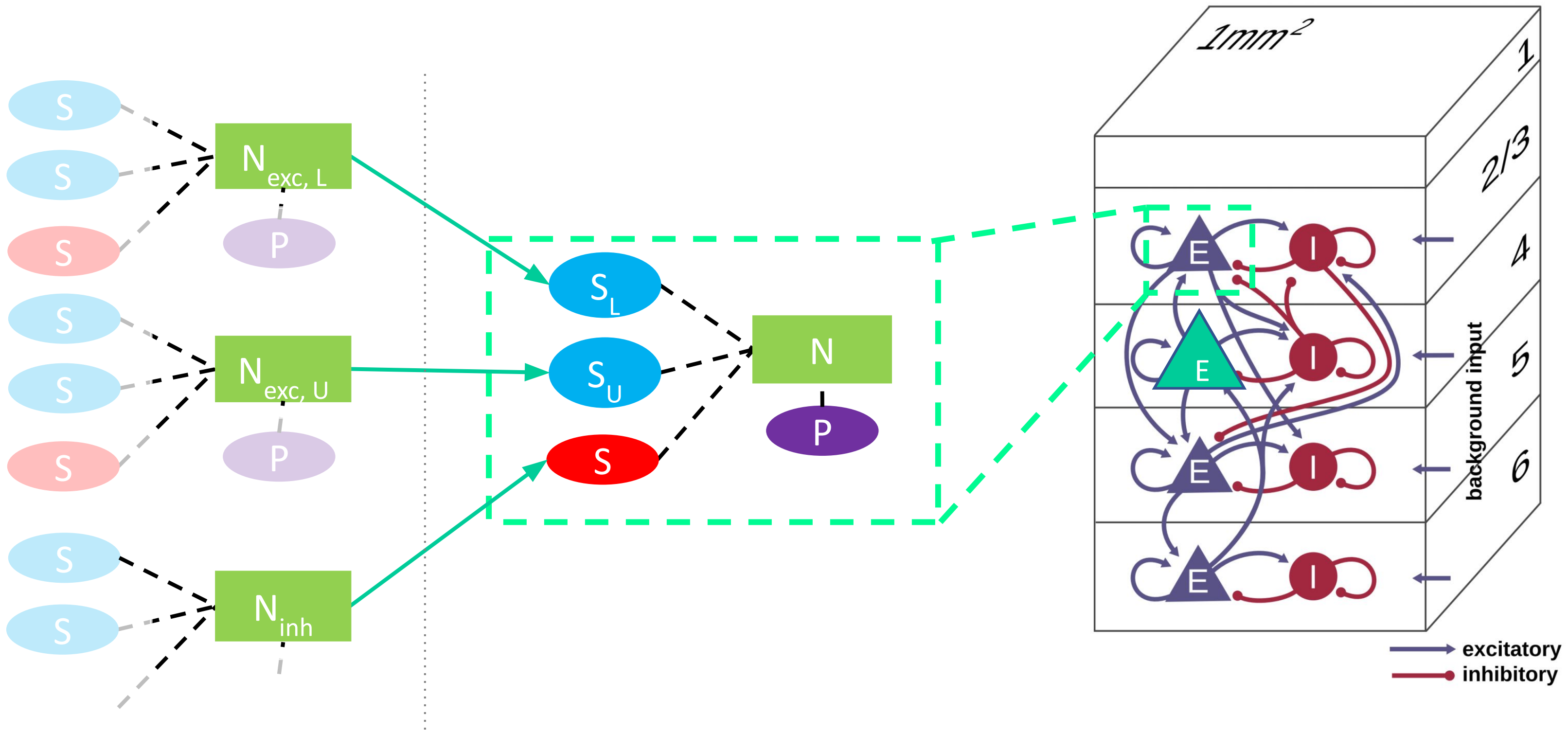
What's New with SpiNNaker

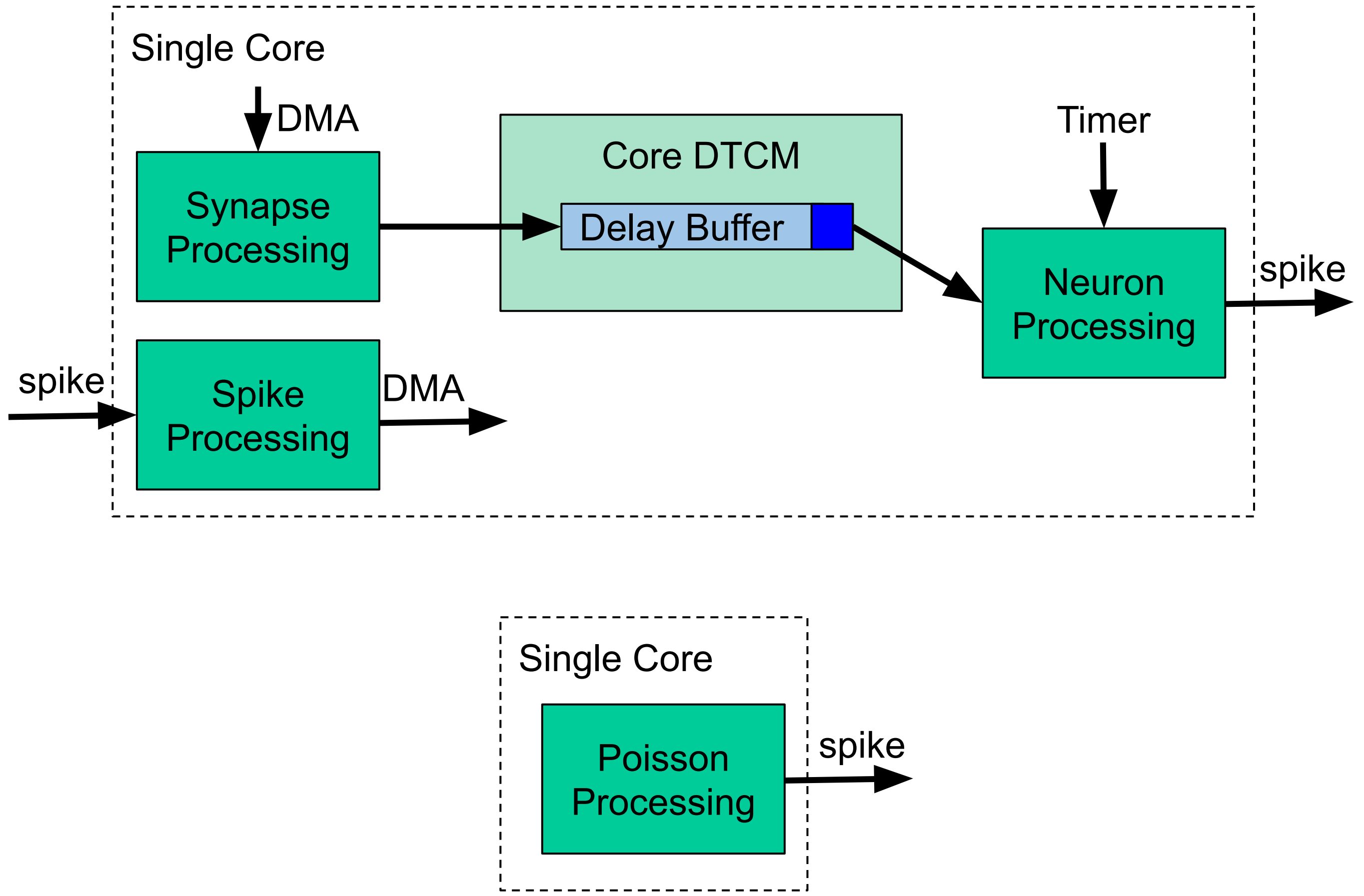


Andrew Rowley, UMAN

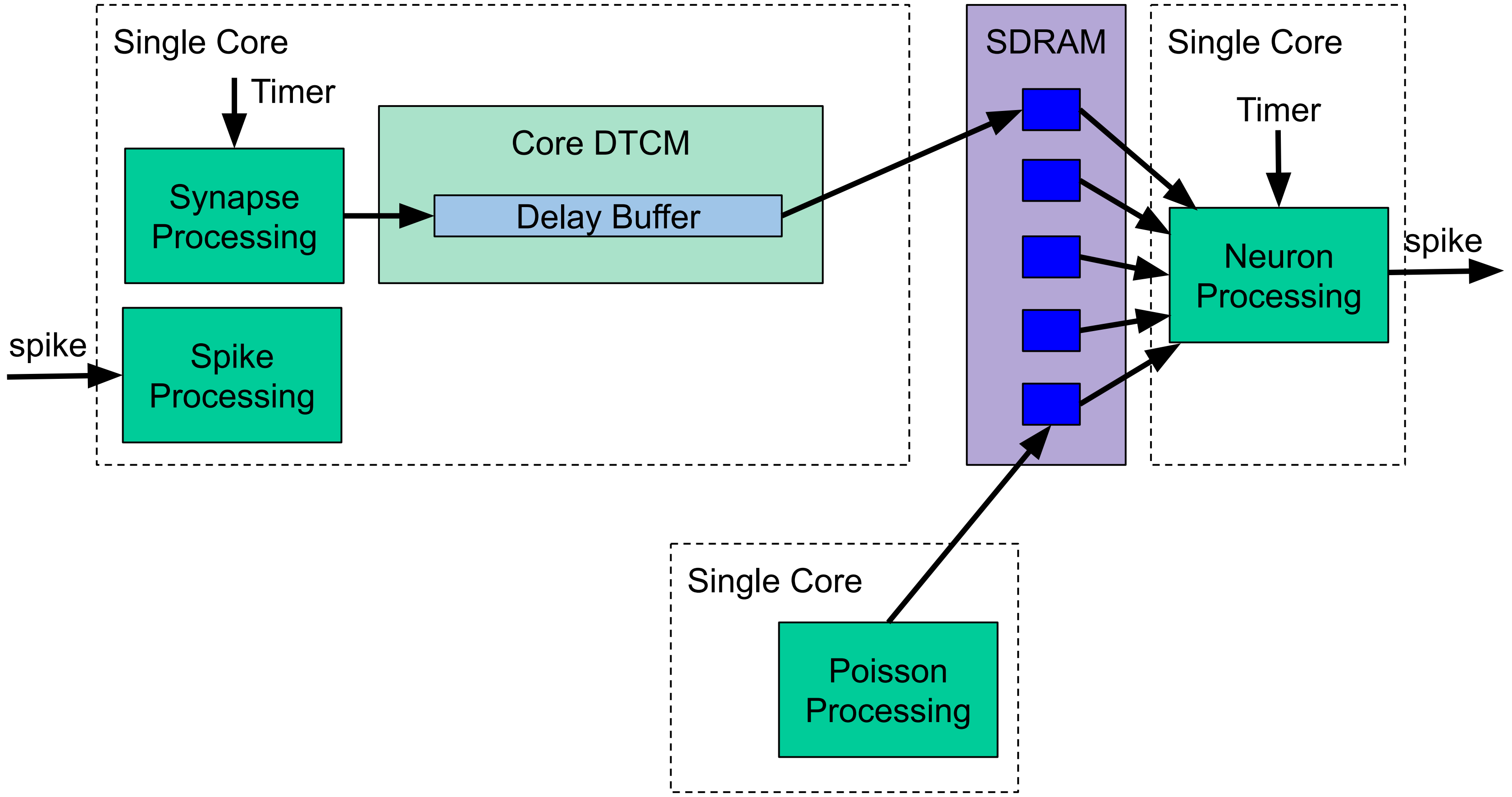
Multi-core Model



Multi-core Model



Multi-core Model



Multi-core Model

```
import pyNN.spiNNaker as p

p.setup()

pop = p.Population(100, p.IF_curr_exp(), label="pop")
source = p.Population(100, p.SpikeSourcePoisson(5), label="source")

p.Projection(source, pop, p.OneToOneConnector(), p.StaticSynapse(weight=5))
```

Multi-core Model

```
import pyNN.spiNNaker as p
from spynnaker.pyNN.extra_algorithms.splitter_components import (
    SplitterAbstractPopulationVertexNeuronsSynapses, SplitterPoissonDelegate)

p.setup()

pop = p.Population(100, p.IF_curr_exp(), label="pop", additional_arguments={
            "splitter": SplitterAbstractPopulationVertexNeuronsSynapses(
                n_synapse_vertices=3, max_delay=128, allow_delay_extension=False)})
```

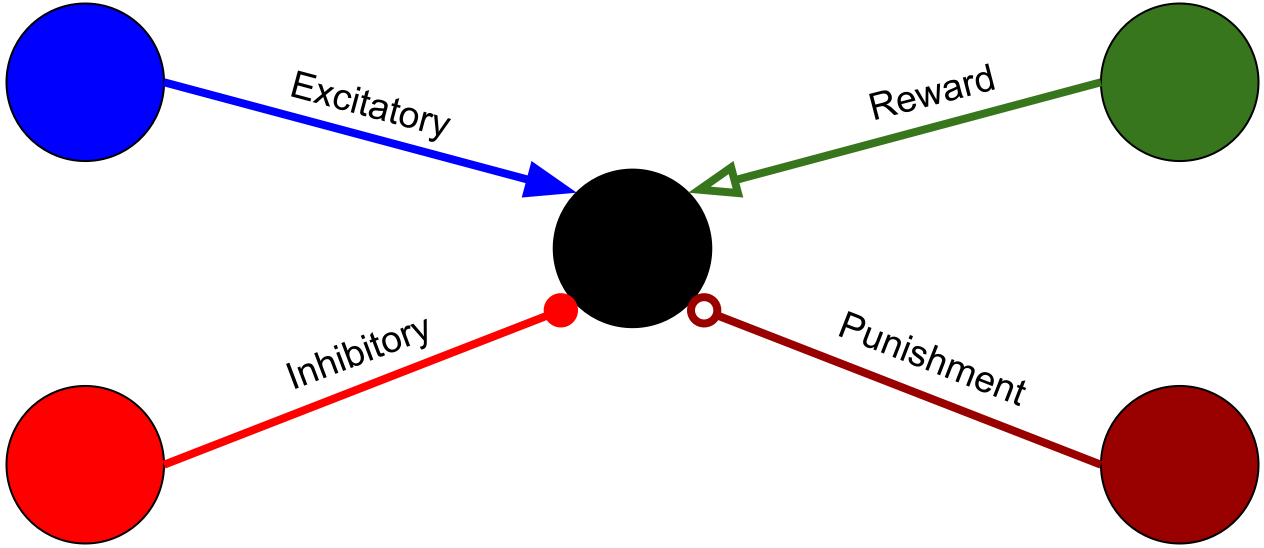
Multi-core Model

```
import pyNN.spiNNaker as p
from spynnaker.pyNN.extra_algorithms.splitter_components import (
    SplitterAbstractPopulationVertexNeuronsSynapses, SplitterPoissonDelegate)

p.setup()

pop = p.Population(100, p.IF_curr_exp(), label="pop", additional_arguments={
            "splitter": SplitterAbstractPopulationVertexNeuronsSynapses(
                n_synapse_vertices=3, max_delay=128, allow_delay_extension=False)})

source = p.Population(100, p.SpikeSourcePoisson(5), label="source", additional_arguments={
            "Splitter": SplitterPoissonDelegate()})
```

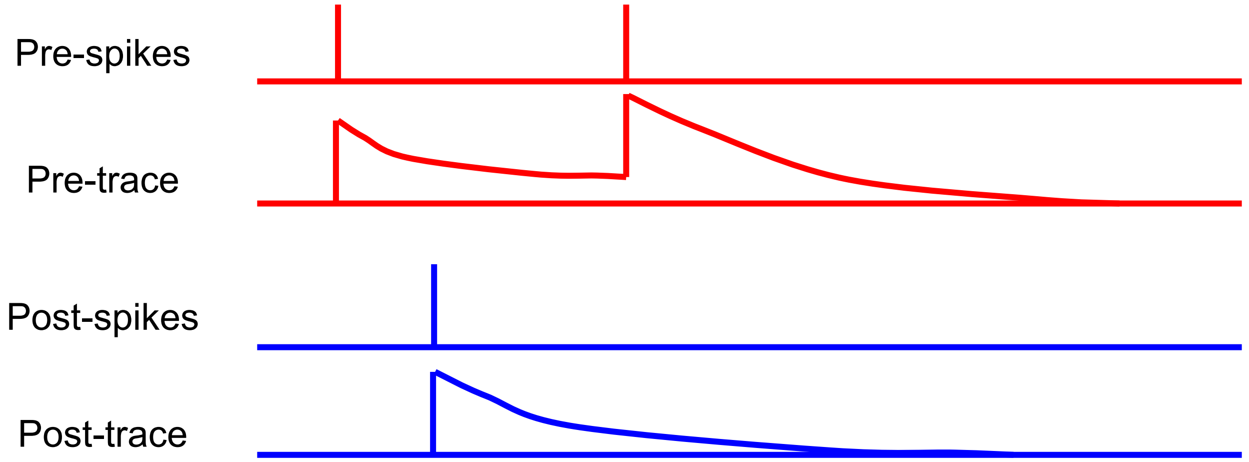
Multi-core Model

```
import pyNN.spiNNaker as p
from spynnaker.pyNN.extra_algorithms.splitter_components import (
    SplitterAbstractPopulationVertexNeuronsSynapses, SplitterPoissonDelegate)

p.setup()

pop = p.Population(100, p.IF_curr_exp(), label="pop", additional_arguments={
            "splitter": SplitterAbstractPopulationVertexNeuronsSynapses(
                n_synapse_vertices=3, max_delay=128, allow_delay_extension=False)})

source = p.Population(100, p.SpikeSourcePoisson(5), label="source", additional_arguments={
            "Splitter": SplitterPoissonDelegate()})

p.Projection(source, pop, p.OneToOneConnector(), p.StaticSynapse(weight=5))
```

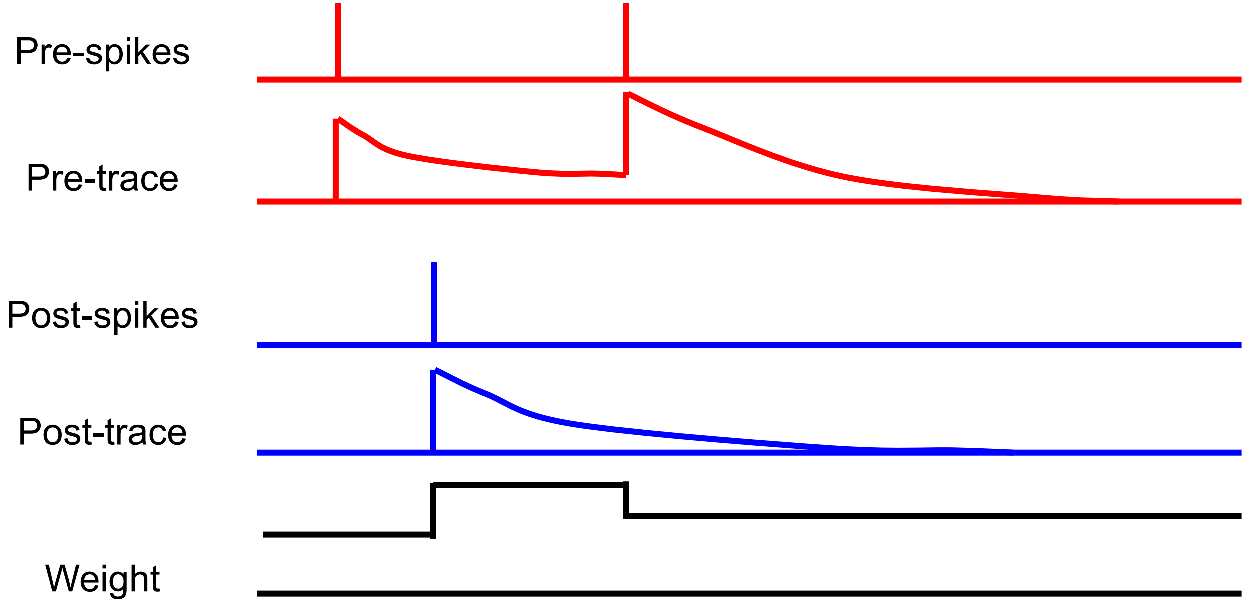
STDP + Neuromodulation



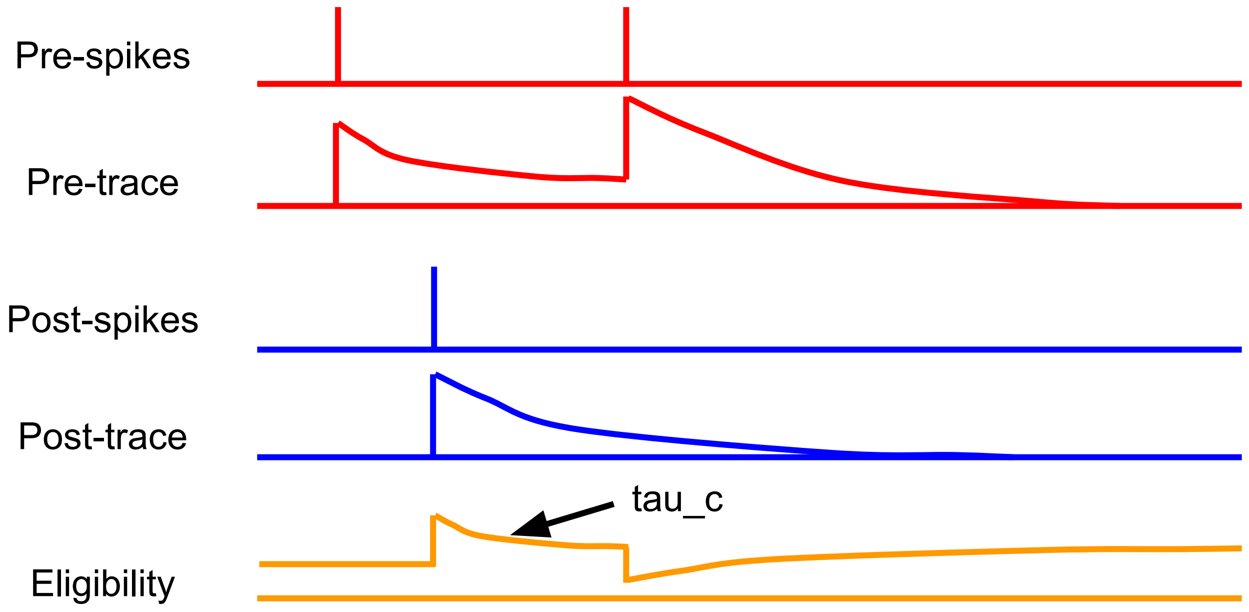
STDP



STDP

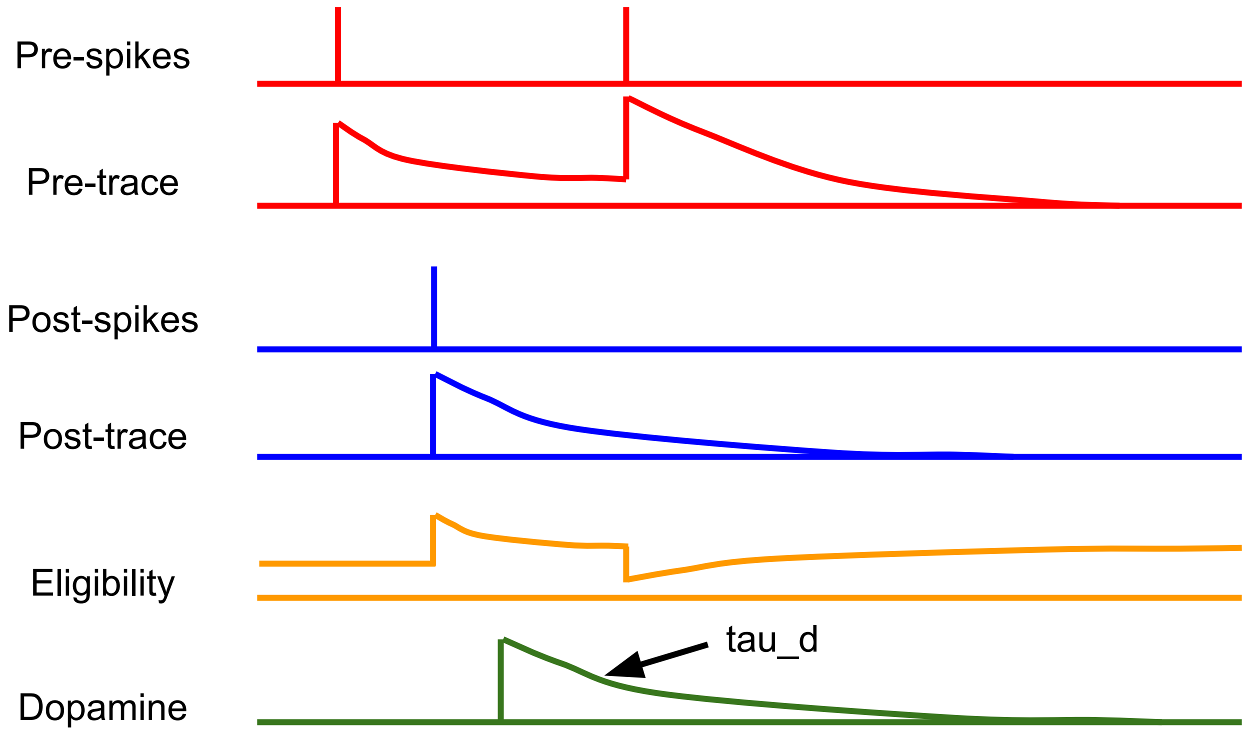


STDP + Neuromodulation

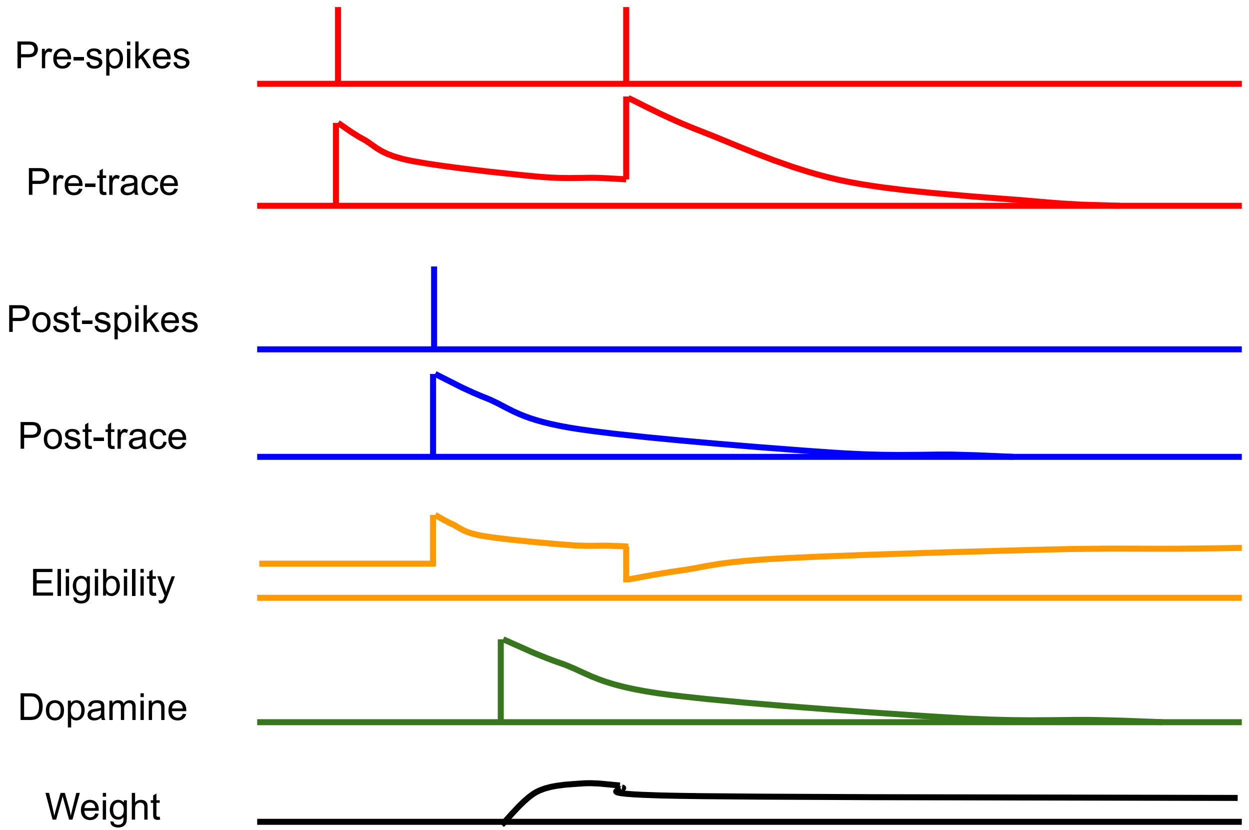


MANCHESTER 1824

STDP + Neuromodulation



STDP + Neuromodulation



STDP + Neuromodulation

```
import pyNN.spiNNaker as p

p.setup()
pop = p.Population(100, p.IF_curr_exp(), label="pop")
source = p.Population(100, p.SpikeSourcePoisson(5), label="source")

p.Projection(source, pop, p.AllToAllConnector(),
             synapse_type=p.STDPMechanism(
                 timing_dependence=p.SpikePairRule(),
                 weight_dependence=p.AdditiveWeightDependence(w_max=20.0),
                 weight=0))
```

STDP + Neuromodulation

```
import pyNN.spiNNaker as p

p.setup()
pop = p.Population(100, p.IF_curr_exp(), label="pop")
source = p.Population(100, p.SpikeSourcePoisson(5), label="source")

p.Projection(source, pop, p.AllToAllConnector(),
             synapse_type=p.STDPMechanism(
                 timing_dependence=p.SpikePairRule(),
                 weight_dependence=p.AdditiveWeightDependence(w_max=20.0),
                 weight=0))

reward_pop = p.Population(1, p.SpikeSourceArray([100]), label="reward")
punish_pop = p.Population(1, p.SpikeSourceArray([200]), label="punish")
```

MANCHESTER 1824

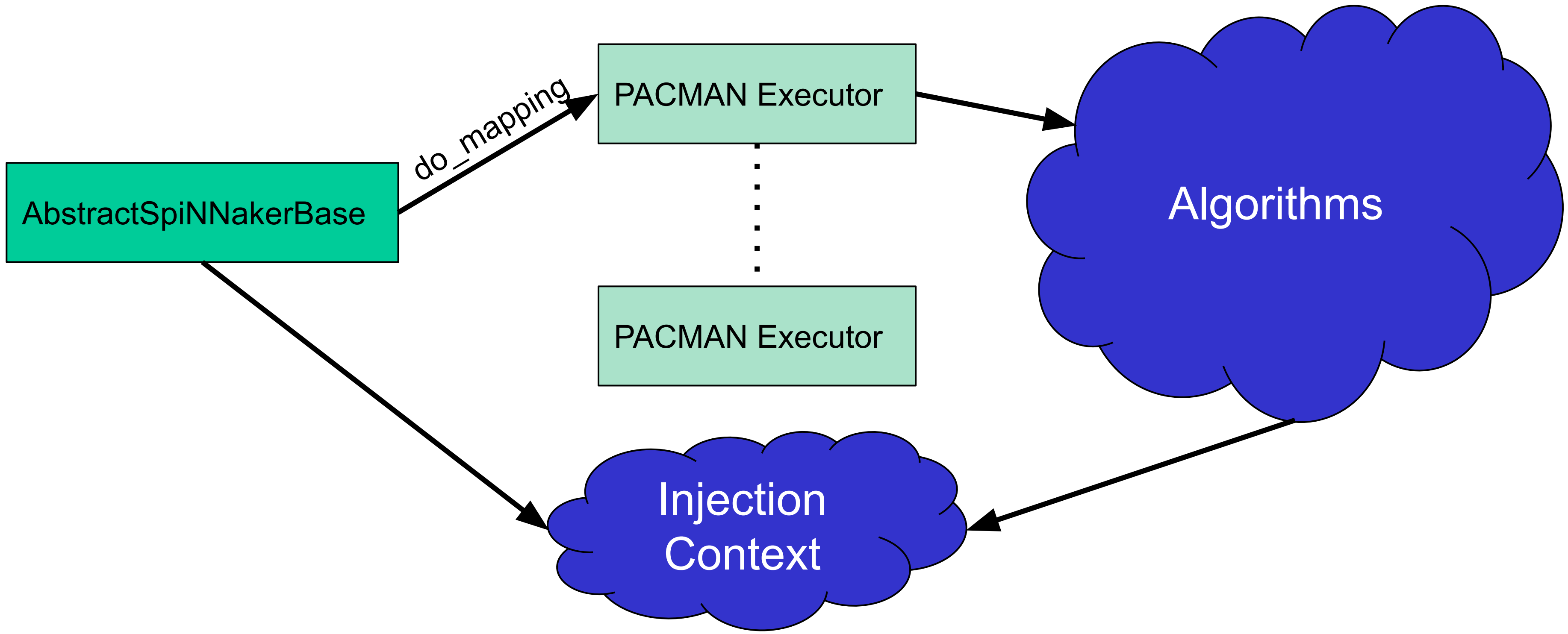
STDP + Neuromodulation

```
import pyNN.spiNNaker as p

p.setup()
pop = p.Population(100, p.IF_curr_exp(), label="pop")
source = p.Population(100, p.SpikeSourcePoisson(5), label="source")

p.Projection(source, pop, p.AllToAllConnector(),
             synapse_type=p.STDPMechanism(
                 timing_dependence=p.SpikePairRule(),
                 weight_dependence=p.AdditiveWeightDependence(w_max=20.0),
                 weight=0))

reward_pop = p.Population(1, p.SpikeSourceArray([100]), label="reward")
punish_pop = p.Population(1, p.SpikeSourceArray([200]), label="punish")

p.Projection(reward_pop, pop, p.AllToAllConnector(),
             synapse_type=p.extra_models.Neuromodulation(
                 weight=0.05, tau_c=100, tau_d=5.0, w_max=20.0), receptor_type="reward")
```

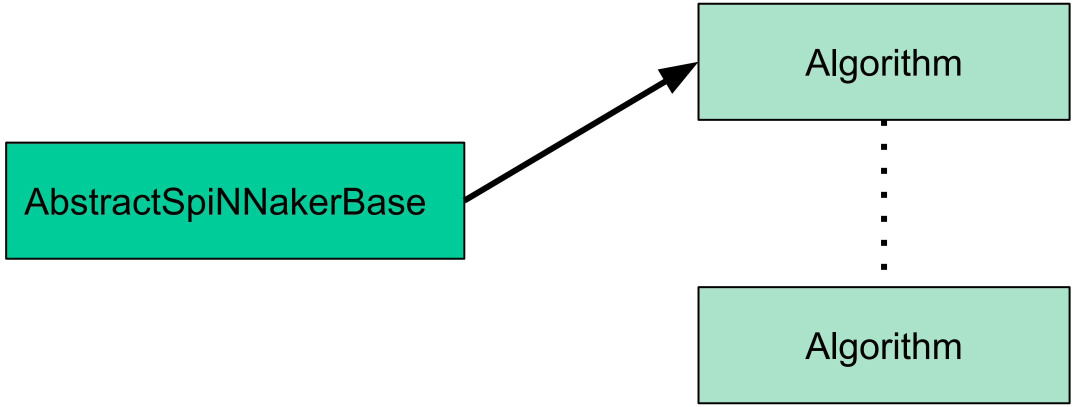
STDP + Neuromodulation

```
import pyNN.spiNNaker as p

p.setup()
pop = p.Population(100, p.IF_curr_exp(), label="pop")
source = p.Population(100, p.SpikeSourcePoisson(5), label="source")

p.Projection(source, pop, p.AllToAllConnector(),
             synapse_type=p.STDPMechanism(
                 timing_dependence=p.SpikePairRule(),
                 weight_dependence=p.AdditiveWeightDependence(w_max=20.0),
                 weight=0))

reward_pop = p.Population(1, p.SpikeSourceArray([100]), label="reward")
punish_pop = p.Population(1, p.SpikeSourceArray([200]), label="punish")

p.Projection(reward_pop, pop, p.AllToAllConnector(),
             synapse_type=p.extra_models.Neuromodulation(
                weight=0.05, tau_c=100, tau_d=5.0, w_max=20.0), receptor_type="reward")
p.Projection(reward_pop, pop, p.AllToAllConnector(),
             synapse_type=p.extra_models.Neuromodulation(
                weight=0.05, tau_c=100, tau_d=5.0, w_max=20.0), receptor_type="punishment")
```

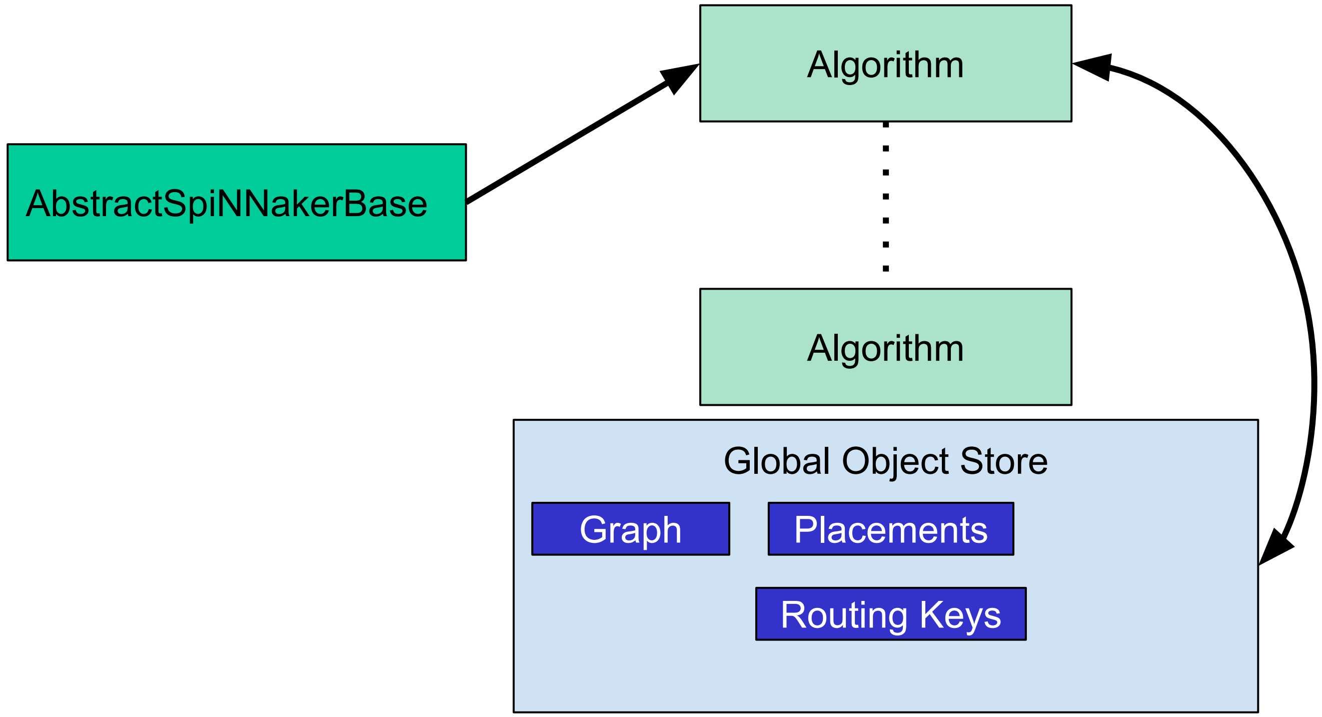
Simplified Common Code



Simplified Common Code



Simplified Common Code - In Progress



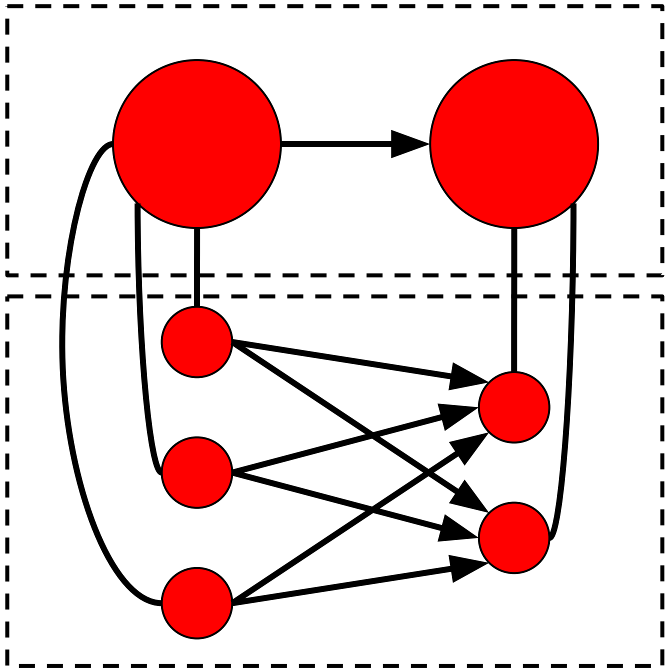
Data Simplification - In Progress

Application Level

Machine Level

Placement

Routing



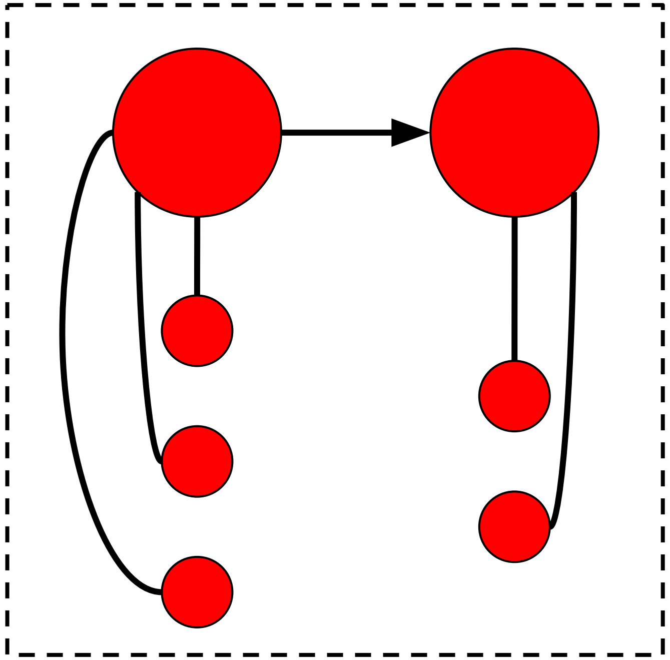
MANCHESTER 1824

Data Simplification - In Progress

Application Level

Placement

Routing



Convolution Networks - In Progress