NEMO

A PARALLEL DISCRETE EVENT NEUROMORPHIC HARDWARE SIMULATION MODEL

OVERVIEW

- A Brief Overview of PDES
- Neuromorphic Hardware
- Demand & Potential
- Contributions of NeMo
- Neuromorphic Software Models
- Features of NeMo
- Experimental Results
- Future Work

A BRIEF OVERVIEW OF PDES

PARALLEL DISCRETE EVENT SIMULATION

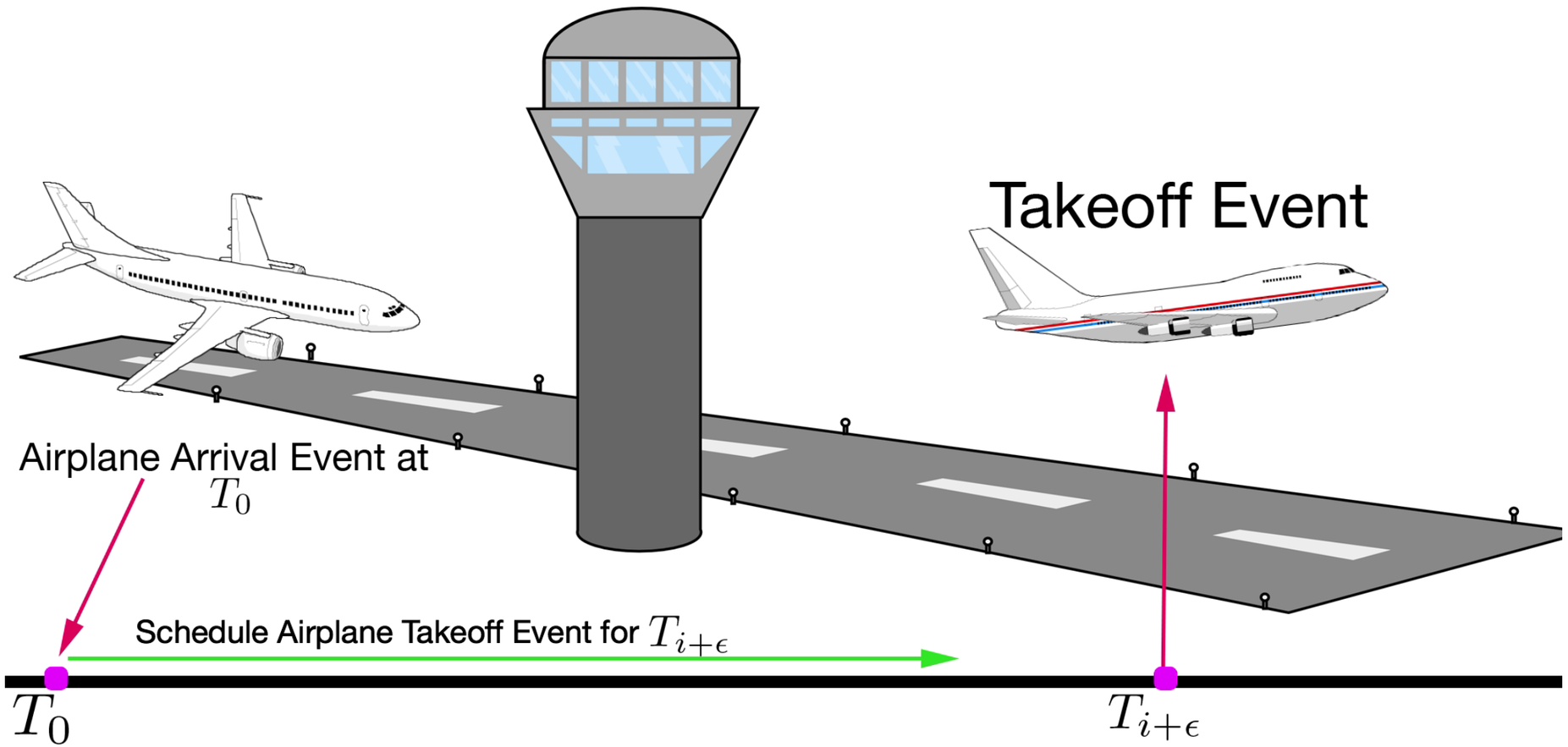
- Provides a fast way to simulate large systems
- Uses event based processing
- Computation only occurs when a process receives an event

KEY TERMS

LP: Logical Process - A simulated item. Can be an airport, a cell phone tower, etc...

PE: Physical Process - A running process in the simulation.
Can contain one or more running **LPs**

Event: A communication between running LPs. Events drive the simulation.


Takeoff Event
Airplane Arrival Event at
T_0
Schedule Airplane Takeoff Event for $T_{i+\epsilon}$
T_0
$T_{i+\epsilon}$

ROSS

ROSS (Rensselear's Optimistic Simulation System) is a PDES simulation tool
ROSS enables use of the TimeWarp algorithm

TIMEWARP ALGORITHM

- With TimeWarp all PEs process events as fast as they can
- If an event is received in the past, events are "rolled back":
  Events are undone until received event is in the proper order.

NEUROMORPHIC HARDWARE

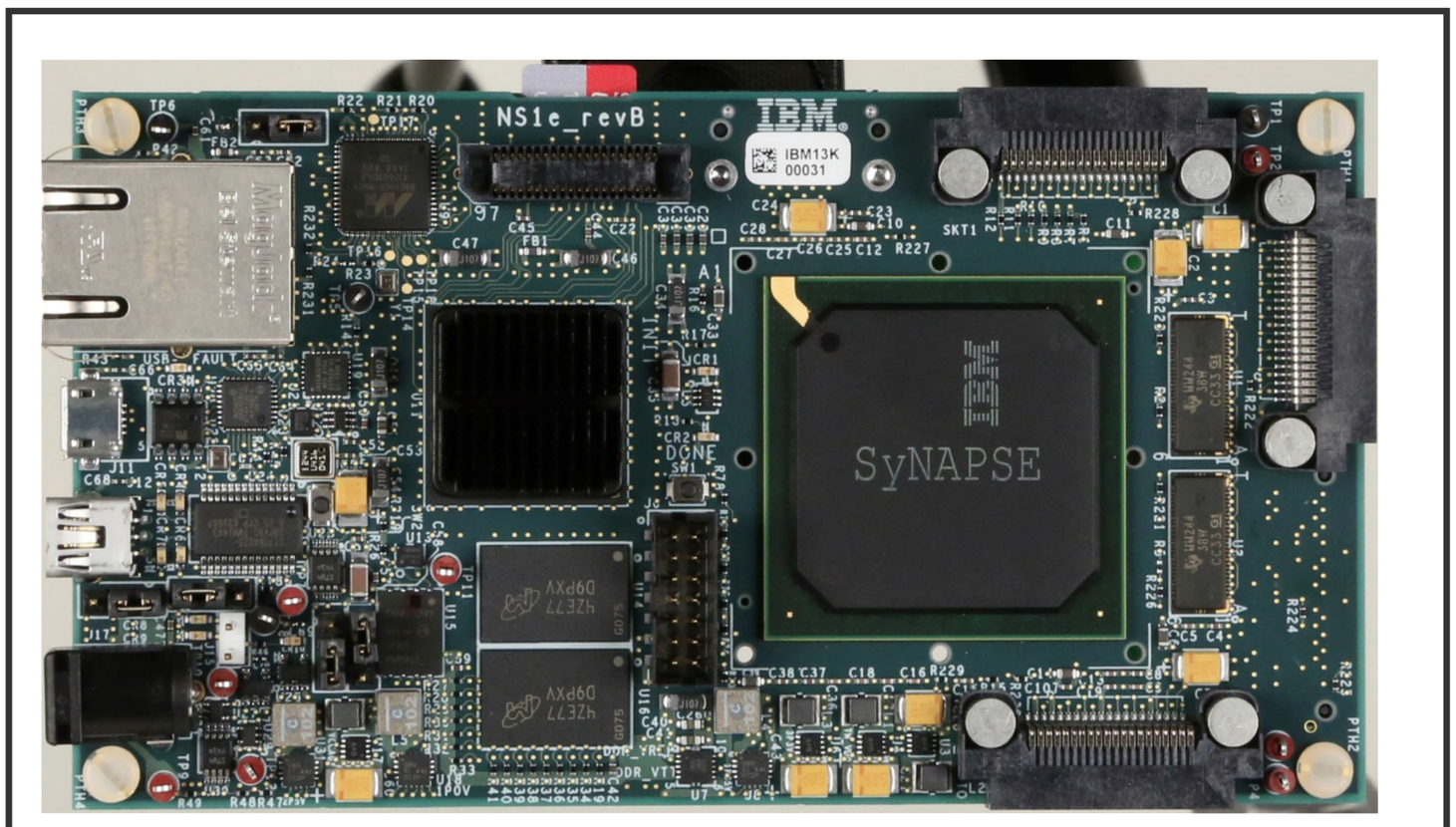
SPECIALIZED PROCESSING

- Uses Artificial Neural Network concepts
- Excellent for finding patters in data
- Great data analysis capability

NEUROMORPHIC HARDWARE

LOW POWER

- IBM TrueNorth processor uses 65 mW *(That's Milliwatts!)*



NEUROMORPHIC HARDWARE

NEW ALGORITHMS

Neuromorphic computing is "Non von Neumann"

- Does not operate using traditional algorithms
- New programming paradigms needed for adoption

DEMAND AND POTENTIAL

EXASCALE COMPUTING

- Transitioning to “Fat Nodes”
- Titan
  - Released in 2013 has 18,688 CPUs and 18,688 GPUs
  - Uses 8.2 MW of Power
  - 10 petaflops
- Tianhe-2 has 32,000 CPUs with accelerators
  - 33.86 petaflops
  - Uses 24 MW of Power!

NASA VISION REPORT SUGGESTS IN 2030:

- New systems will have only 20,000 compute nodes
- New systems will *need* accelerator cards

EXASCALE COMPUTING

- Accelerator cards are becoming increasingly important
- GPU, Intel PHI
- Why not neuromorphic hardware?
  - Low Power
  - Excellent Machine Learning

DESIGNING THE NEXT GENERATION SUPERCOMPUTERS

SIMULATION OF NEW TECHNOLOGIES

- Allows testing of hardware configurations
- Enables rapid prototyping of systems

CODES

- Based on ROSS
- Enables simulation of new supercomputer designs

NEUROMORPHIC HARDWARE SIMULATION

- Should Allow for Chip Simulation

NEEDS TO SIMULATE

- Current hardware design
- Future and theoretical hardware

CONTRIBUTIONS

NEMO – AN OPEN SOURCE NEUROMORPHIC HARDWARE SIMULATION MODEL

NEMO

DESIGN AND IMPLEMENTATION

- Implemented using ROSS
- Event-Driven
- Massively Parallel
- Optimistic Event Scheduling
- Supports Reverse Computation

NEMO

- Open Source
- Flexible Hardware Models
- Supports current neuromorphic design
- Can simulate novel designs
- Can even simulate currently "impossible" designs

NEMO

LARGE SCALE SIMULATION SUPPORT

- Currently tested to simulate 65,536 neurosynaptic cores
- Further scale-ups very feasible

NEMO

POTENTIAL FOR NEW SPIKING NEURON MODELS

- Will support other Spiking Neural Network models in the future
- Validated using IBM’s TrueNorth Model

NEUROMORPHIC COMPUTING MODELS

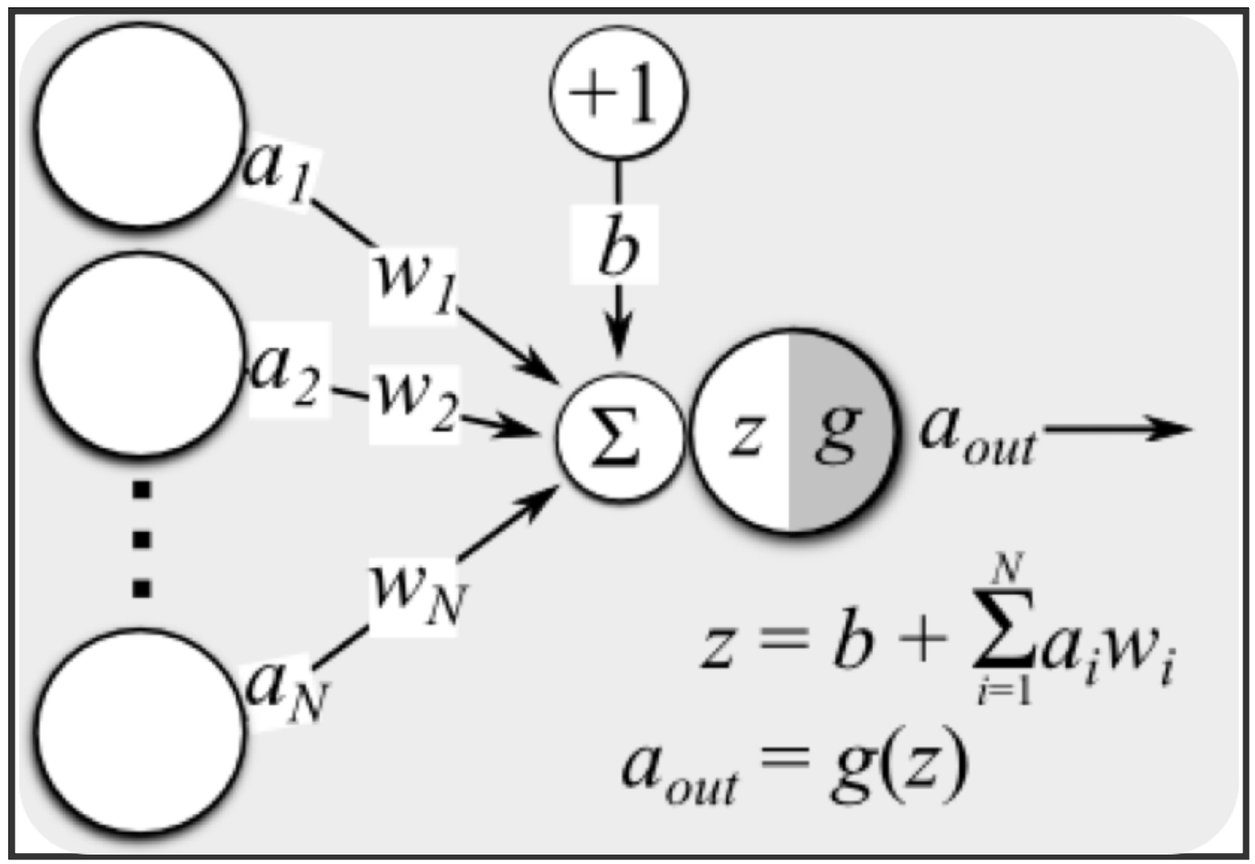
A BRIEF OVERVIEW

BASICS

- Based on Artificial Neural Network (ANN) concept
- Third generation neuron simulation
- Developed to simulate biological functions - not for machine learning
- Neuromorphic computing has been shown to be viable for computing

ANNS AND MLPS

- ANNs (Artificial Neural Network)s are:
  - Machine Learning Concepts
  - Inputs are modified by weights
  - Neurons sum inputs and apply output functions
  - Output is result of this function

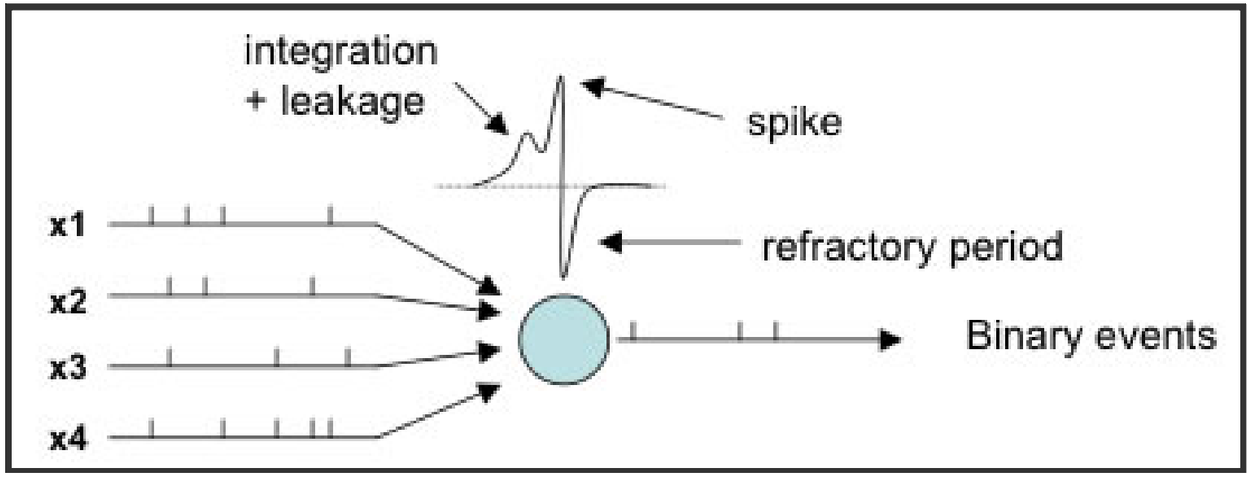


ANNS AND MLPS

- MLPs (Multi Layer Perceptron)s are:
  - Multiple layers of perceptron neurons
  - The basis for many ANN network designs
  - Hugely popular

SPIKING NEURONS

- Include concept of time
- Neurons do not need to fire at every time-step, t
- Activation level is increased with spikes

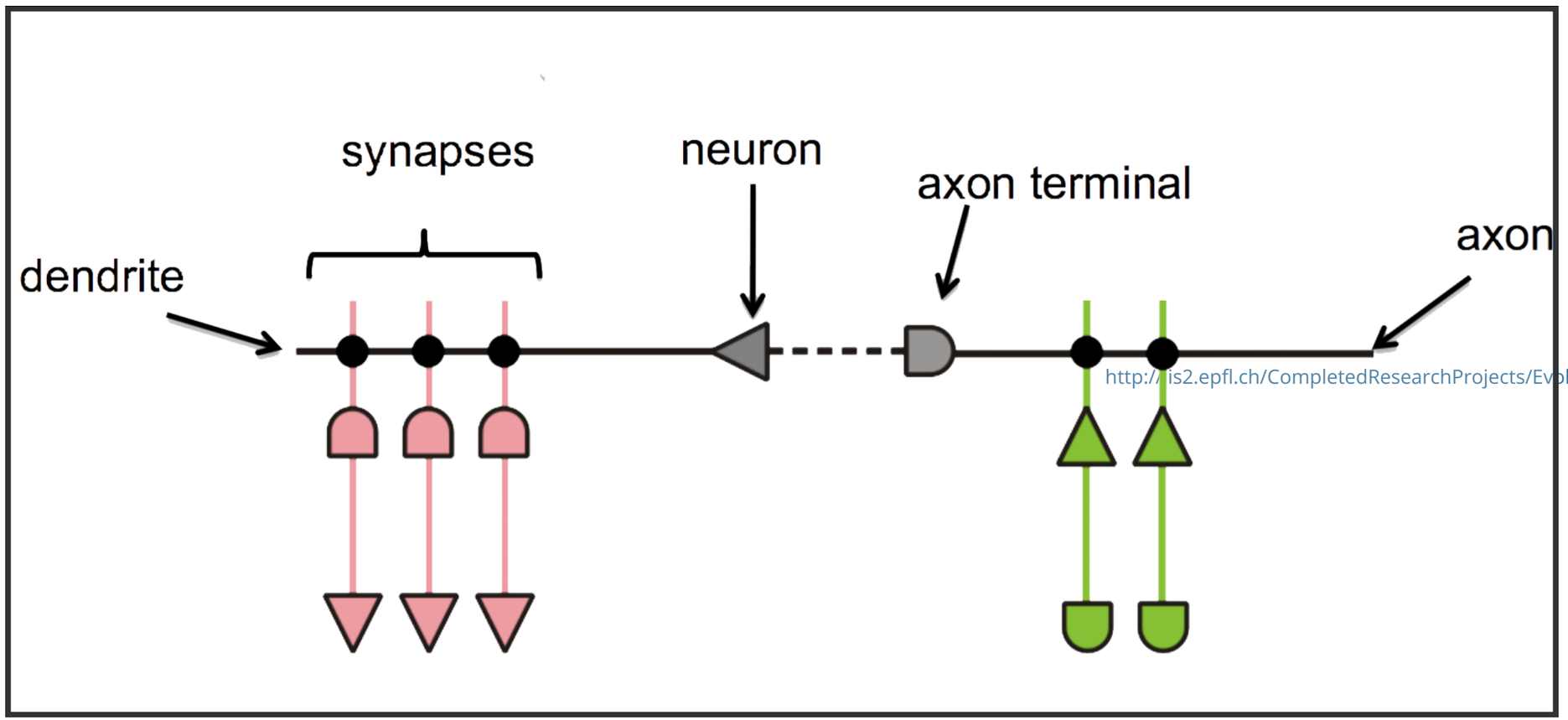


A BRIEF HISTORY

- First scientific model developed in 1952 by Hodgkin and Huxley
- Developed further into commonly used models including:
  - Hodgkin and Huxley model
  - Integrate and fire
  - Leaky integrate and fire
  - Many more...

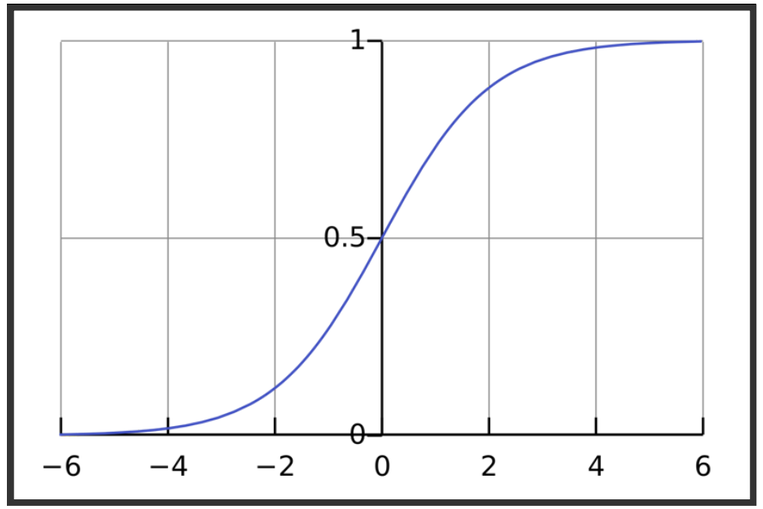
NAMING CONVENTIONS

Spiking neural networks use bio inspired terms



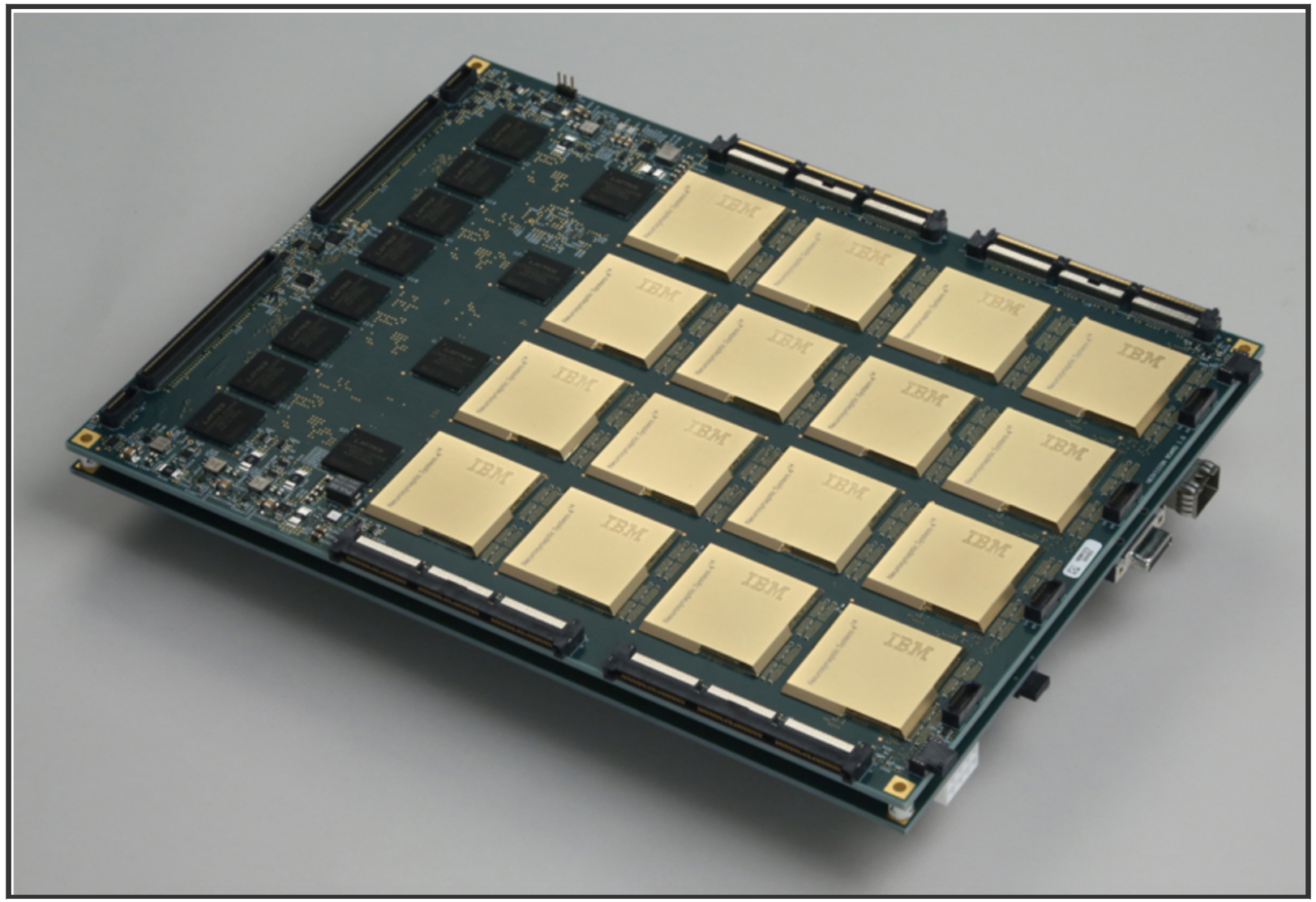
CURRENT HARDWARE

- Current hardware uses spiking neurons
  - Neurons output a 1 or 0
  - Integration is generally simple (no sigmoid functions yet)
  - Significant research is underway



TRUE NORTH CHIP

Implementation of an enhanced Leaky Integrate and Fire neuron



LEAKY INTEGRATE-AND-FIRE

A BASIC SPIKING NEURON MODEL

LEAKY INTEGRATE-AND-FIRE:

$$V_j(t) = V_j(t-1) \sum_{i=0}^{n-1} [x_i(t)\, s_i]$$

- Takes the sum of all of the input synapses.
  - x is the on/off state of the synapse
  - s_i is the synapse weight
  - $V_j(t)$ is the neuron voltage at time t

LEAKY INTEGRATE-AND-FIRE

Leak and reset for neuron j:

- Leak:

$$V_j(t) = V_j(t) - \lambda_j$$

- Spike & Reset:

$$\textbf{if } V_j(t) \geq \alpha_j :$$
$$\text{Spike}$$
$$V_j(t) = R_j$$
$$\textbf{end if}$$

- λ_j is the leak value of the neuron.
- α_j is the threshold of the neuron.
- R_j is the reset voltage.

- R_j is the reset voltage.

LEAKY INTEGRATE-AND-FIRE

The Combined Formula

Integration:

$$V_j(t) = V_j(t-1) \sum_{i=0}^{n-1} [x_i(t)\, s_i]$$

Leak:

$$V_j(t) = V_j(t) - \lambda_j$$

Threshold check & Fire:

$$\textbf{if} \;\; V_j(t) \geq \alpha_j \;:$$

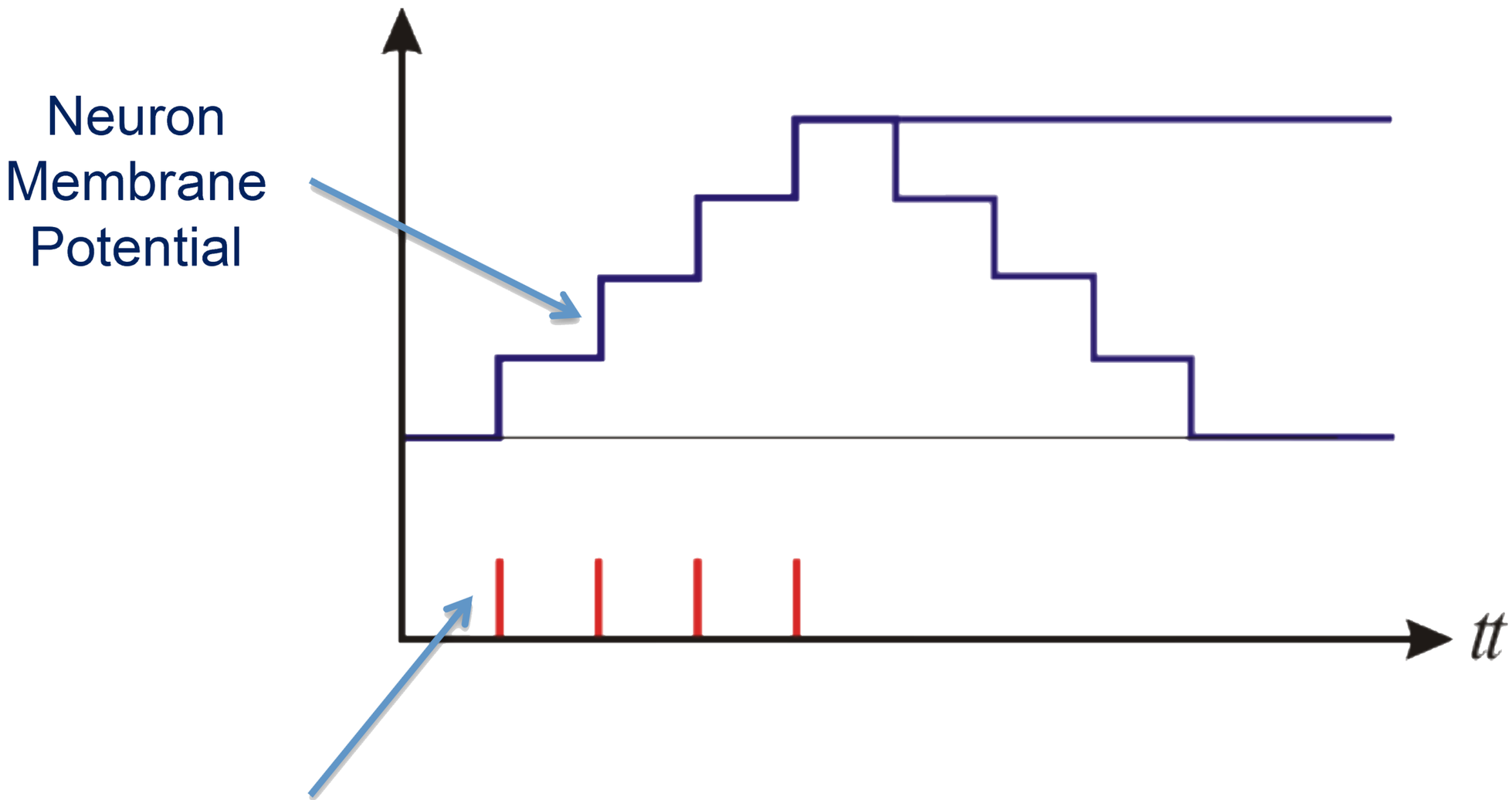
Spike

$$V_j(t) = R_j$$

$v_j(t) = K_j$

IN PRACTICE

end if


Neuron
Membrane
Potential
tt
Input Spikes

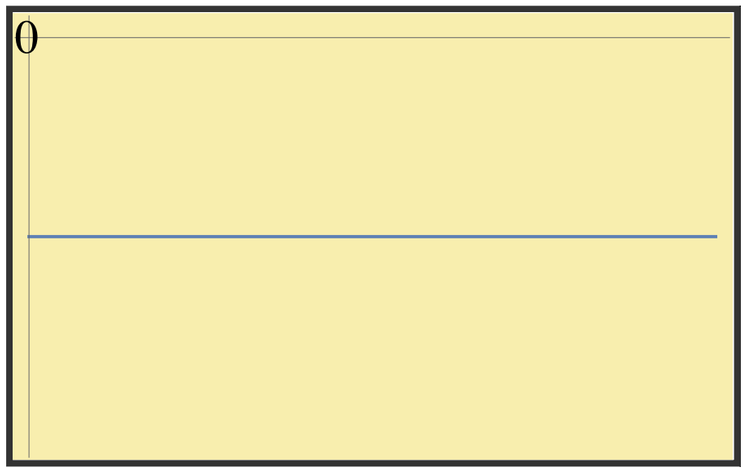
IBM TRUENORTH NEURON

NEURON IMPLEMENTATION

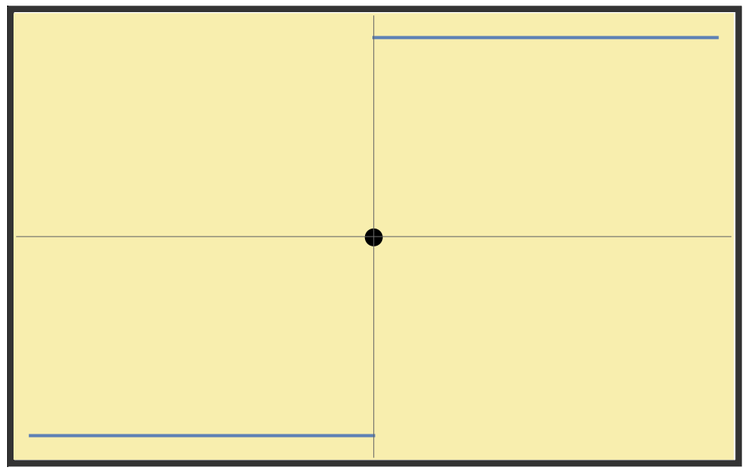
- IBM Enhanced the Leaky Integrate and Fire Neuron
- Added stochastic integration:
  - Enables probabilistic leak and synapse integration
  - Enhanced integration allows for conversion of Caffe models

NEURON IMPLEMENTATION

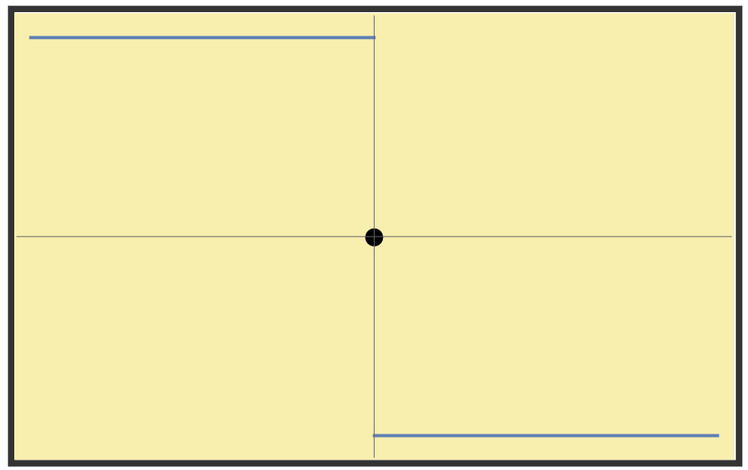
IBM added 3 new leak modes:



Negative



Divergent



Convergent

Stochastic integration allows for probabilistic features

Leak modes allow for greater flexibility over LIF model

Able to reproduce Izhikevich's biologically important neuron models

FOR THOSE INTERESTED

Full IBM TrueNorth neuron formula and description available as handout

PROGRAMMING TRUENORTH CONCEPTS

- TrueNorth is inspired by biology but does not seek to mimic it exactly
- TrueNorth chip is a network of neurosynaptic cores
- A TrueNorth “program” is a complete specification of that network, including inputs and outputs
- That program is specified as a “corelet”
- Tools like Caffe can be used to learn TrueNorth-compatible corelets from data

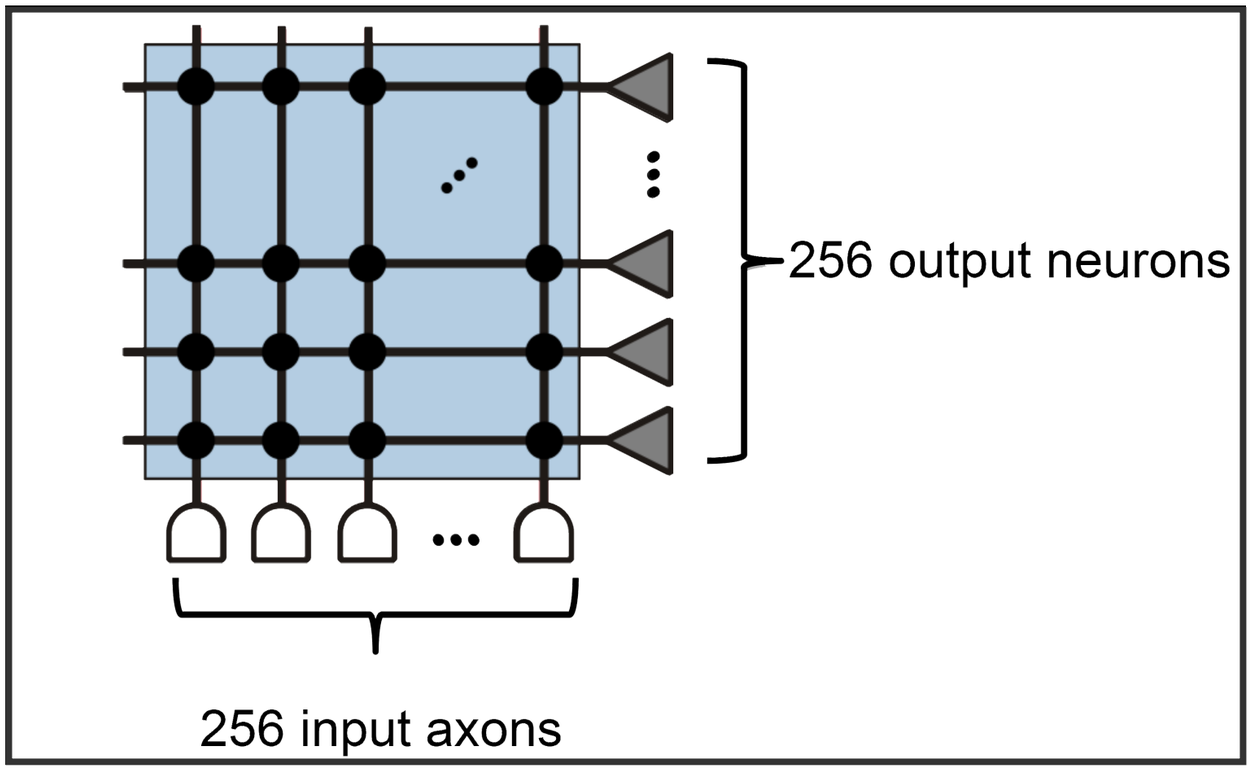
IBM TRUENORTH HARDWARE LAYOUT

- Developed through DARPA project

TrueNorth is a neurosynaptic processor • Designed as a low power dedicated processor

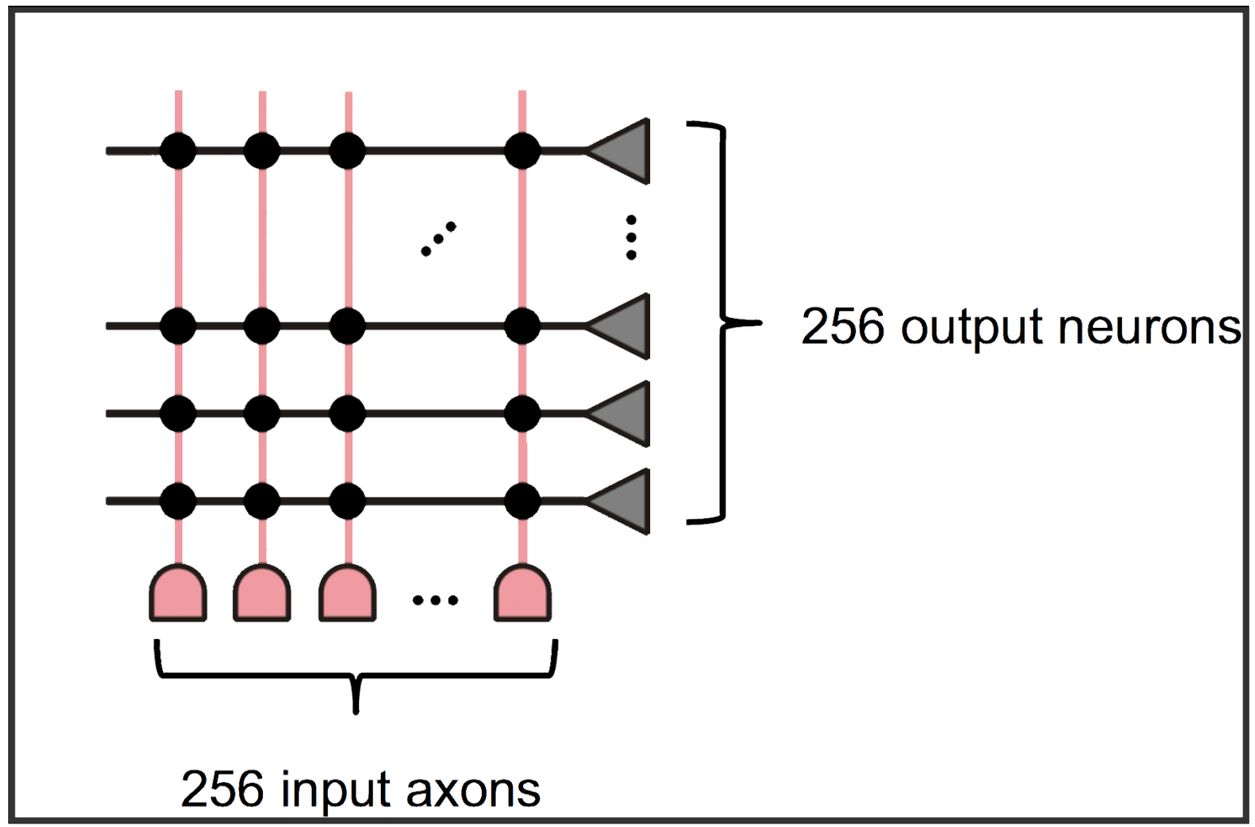
TRUENORTH LAYOUT

A Neurosynaptic Core is 256 neurons, interconnected



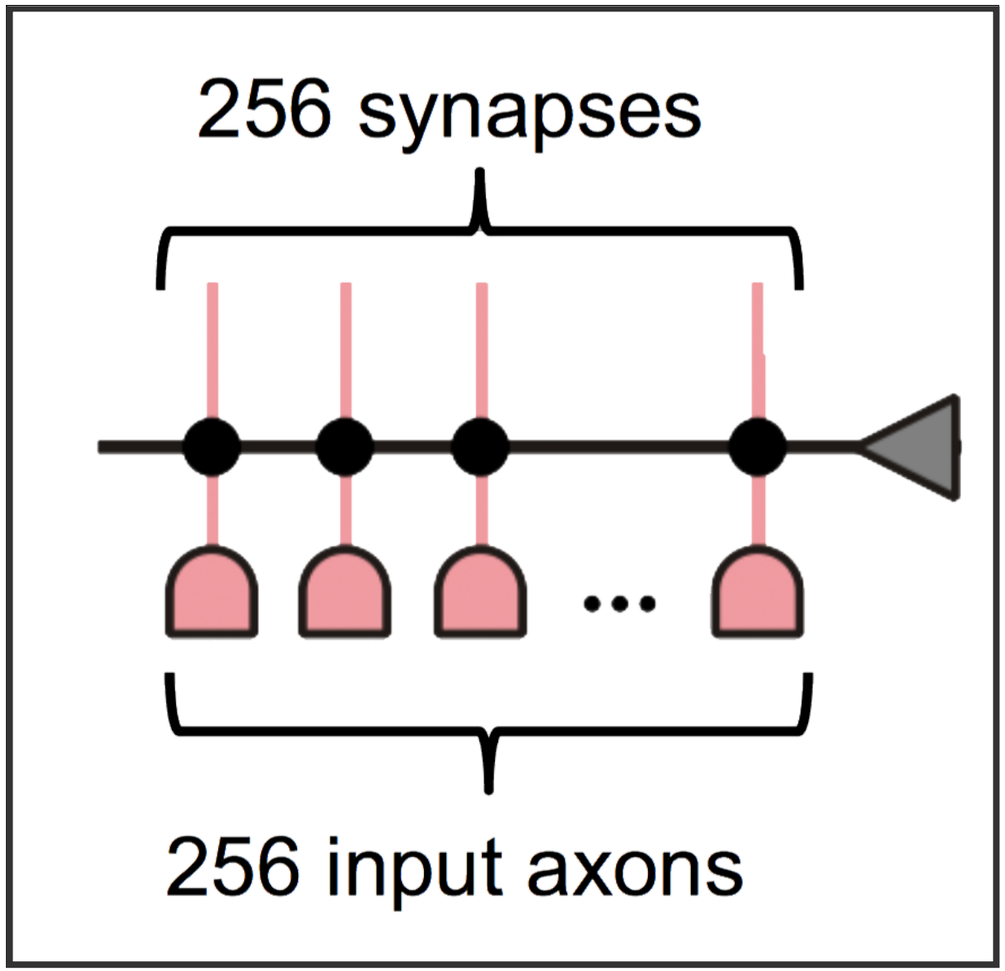
INPUT AXONS

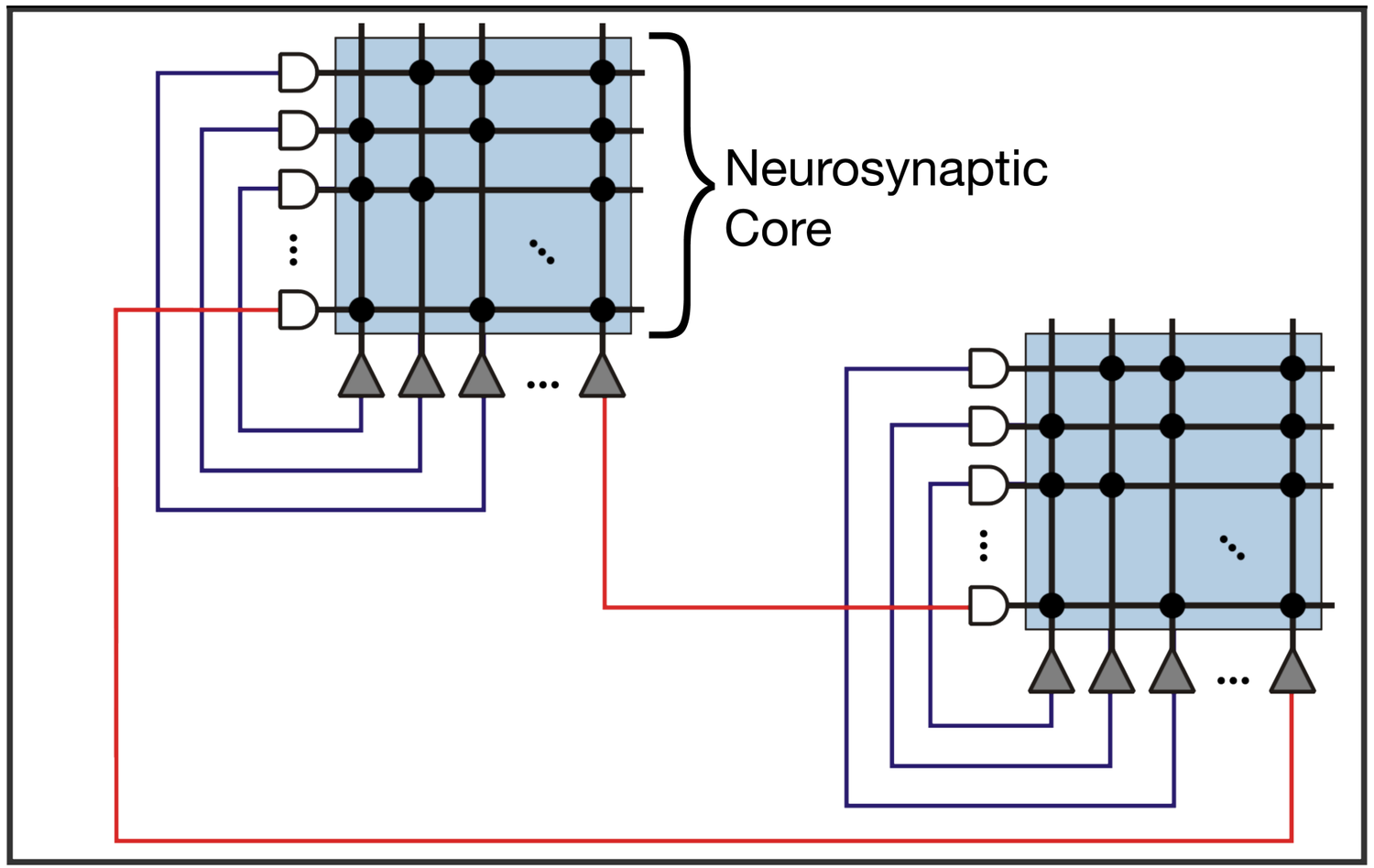
Input axons are arranged in a connected grid



Synapses exist as a logical connection between axons and nurons.

There are 256^2 synapses per core





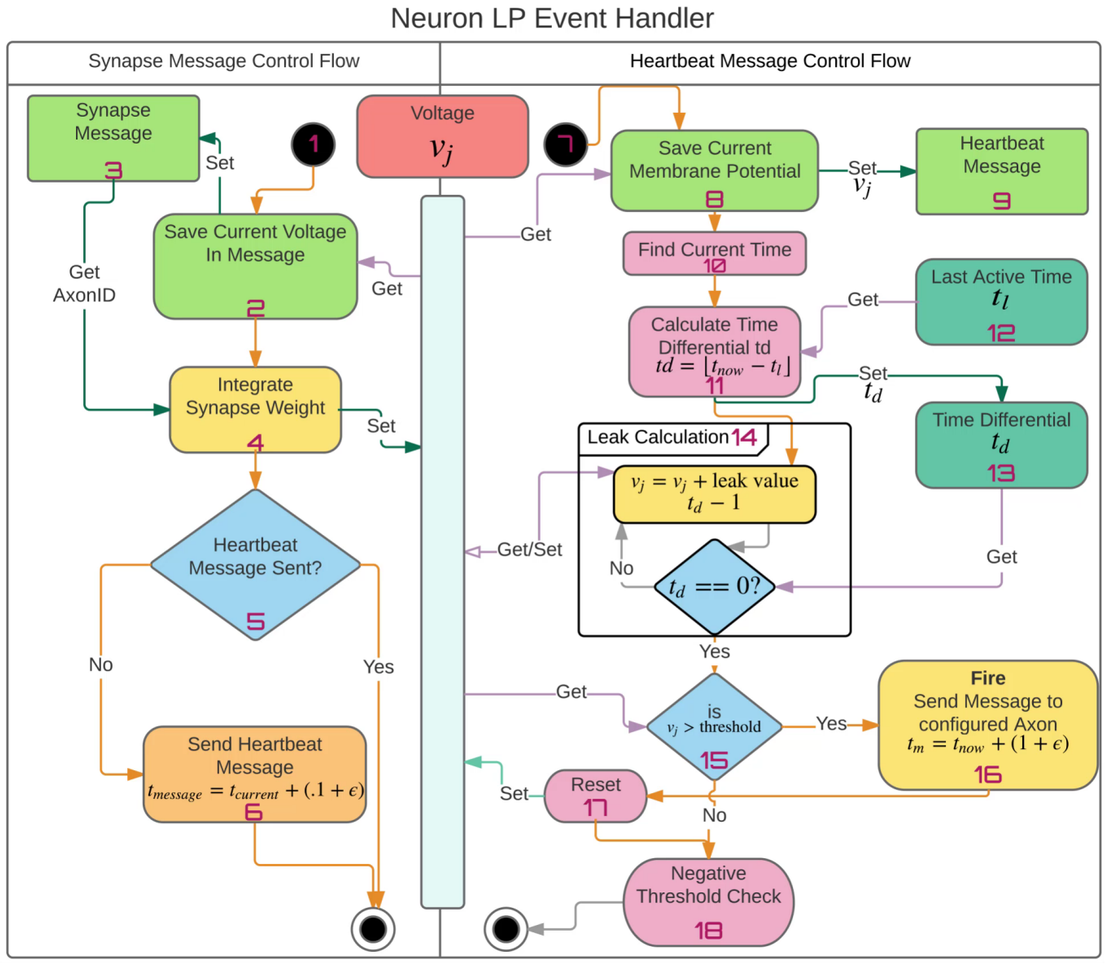
One neuron connects to one axon

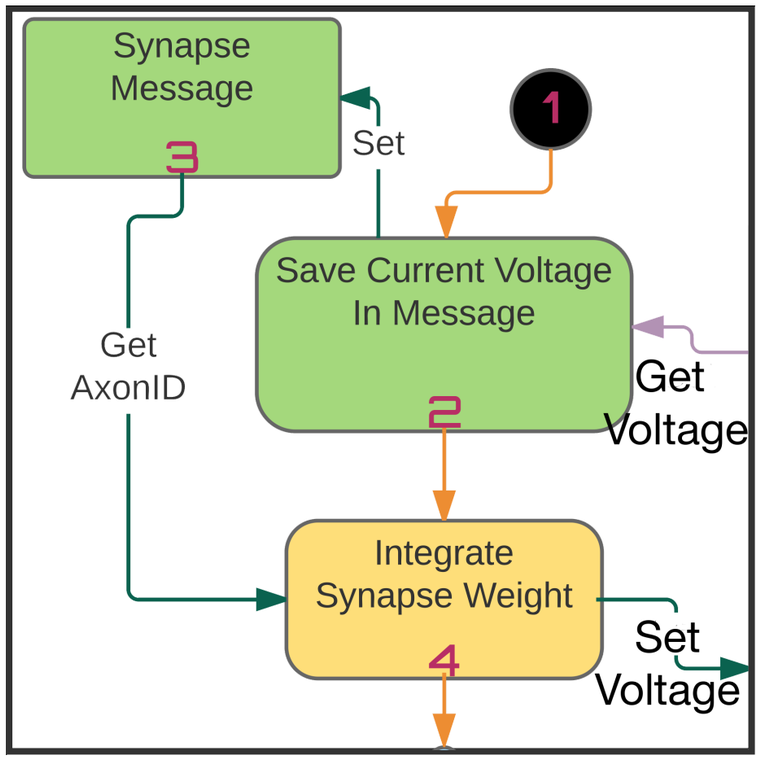
One axon connects to all neurons in a core

NEMO IMPLEMENTATION

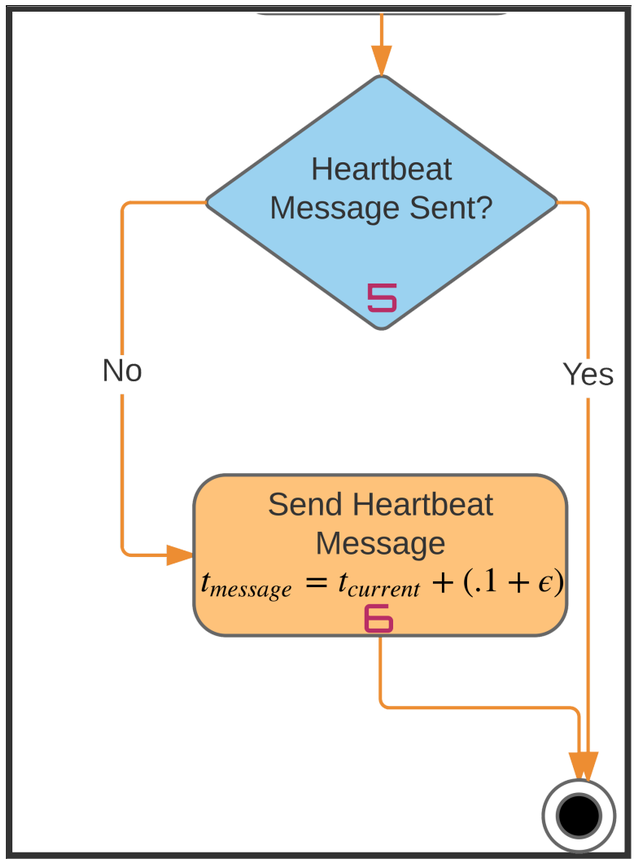
NEMO FEATURES

- "Heartbeat" messages that provide implicit synchronization
- Message fanout reduction through message routing
- Reverse computation for optimistic scheduling


Neuron LP Event Handler
Synapse Message Control Flow
Heartbeat Message Control Flow
Synapse Message
3
Set
1
Voltage
v_j
7
Save Current Membrane Potential
8
Set
v_j
Heartbeat Message
9
Save Current Voltage In Message
2
Get
Get
Get AxonID
Find Current Time
10
Last Active Time
t_l
12
Get
Calculate Time Differential td
$td = \lfloor t_{now} - t_l \rfloor$
11
Set
t_d
Integrate Synapse Weight
4
Set
Time Differential
t_d
13
Leak Calculation 14
$v_j = v_j + \text{leak value}$
$t_d - 1$
Get
Get/Set
No
$t_d == 0?$
Yes
Heartbeat Message Sent?
5
No
Yes
Get
is
$v_j > \text{threshold}$
15
Yes
Fire
Send Message to configured Axon
$t_m = t_{now} + (1 + \epsilon)$
16
Send Heartbeat Message
$t_{message} = t_{current} + (.1 + \epsilon)$
6
Set
Reset
17
No
Negative Threshold Check
18

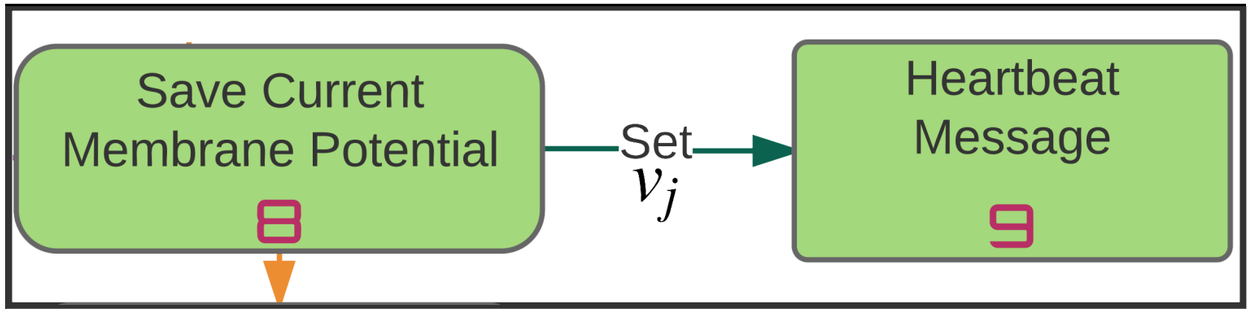


- Synapses send event messages to the neuron
- Neurons integrate the synapse weight at this point
- Previous voltage is saved in message



Synapse Message Part 2

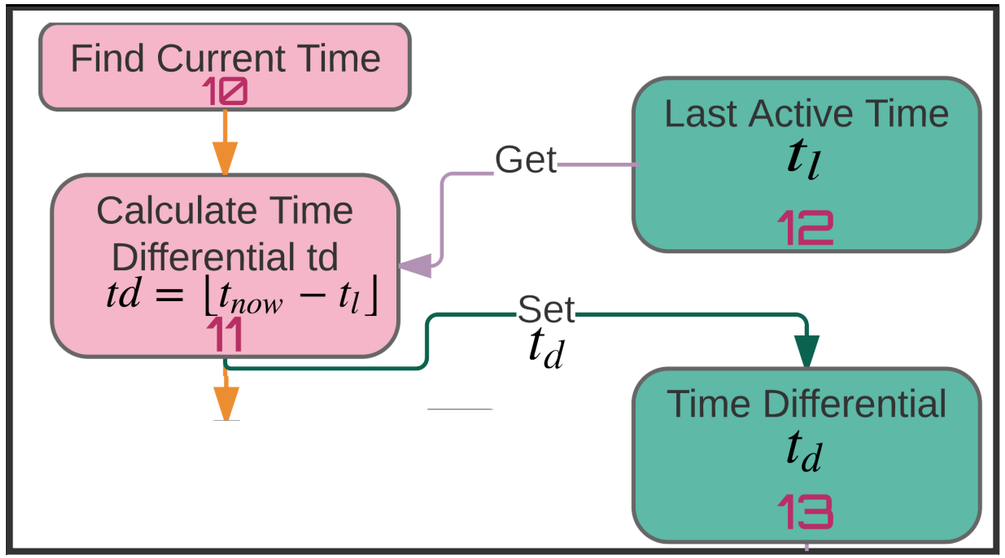
- Neurons check to see if a "heartbeat" message has been sent
- Send heartbeat message if one has not been sent



HEARTBEAT MESSAGE RECEVIED

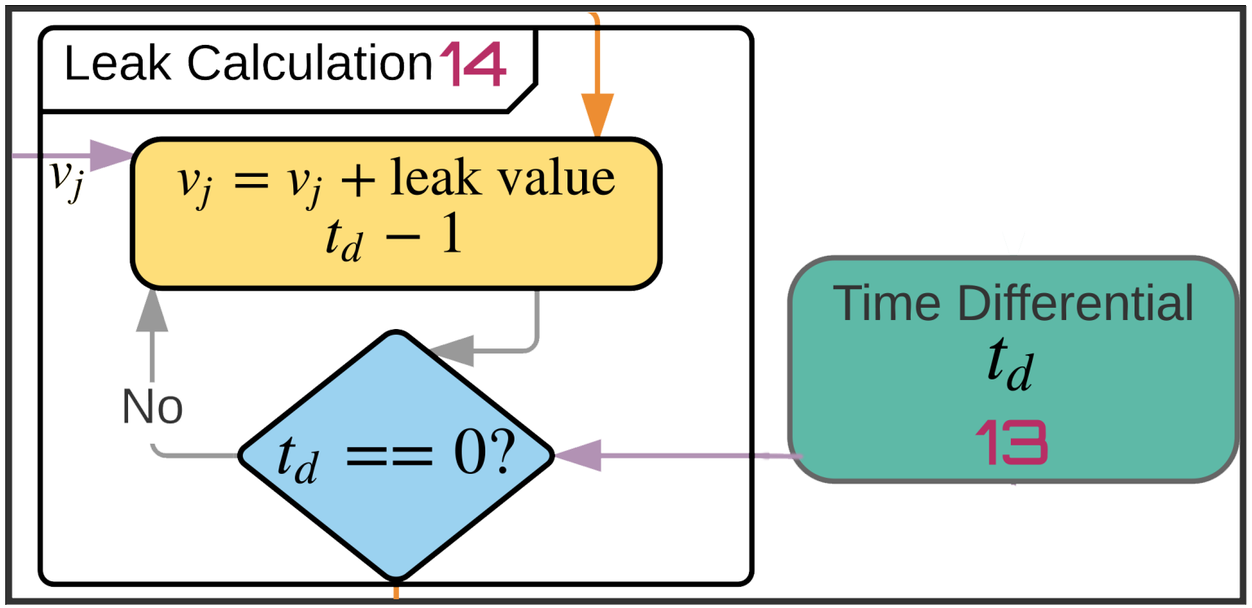
Neurons receiving a heartbeat message have either:

- Had synaptic activity this tick
- Have the potential to be self-firing
  - Positive leak
  - Negative leak with specific reset values
  - And More!
- Neurons save current membrane potential in the message



Neurons then:

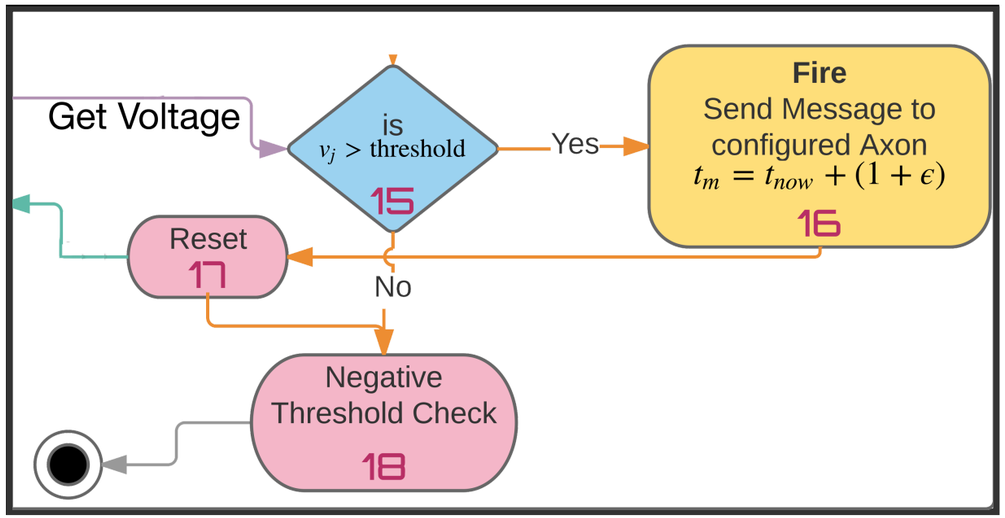
- Retreive current time
- Calculate the number of ticks since last activating
- Store this as t_d



Leak calculation:

```
while (td > 0) {
    voltage = voltage - leakFunction();
    td --;
}
```

THRESHOLD, RESET & FIRE

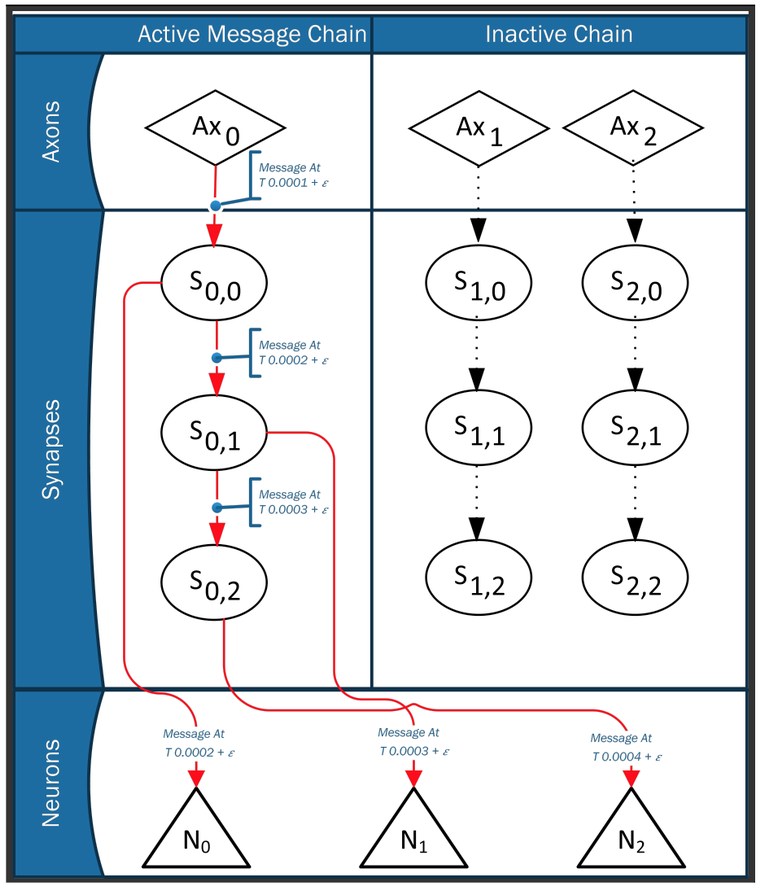


- Check if voltage V_j is $>$ than threshold
- Send fire message
- Reset or check negative threshold

NEMO EVENT FLOW

Given an event at Ax_0(axon 0)

- Synapse $S_{0,0}$ through $S_{0,2}$ activate
- All neurons integrate



MORE NEMO FEATURES

NeMo is built on ROSS!

This allows for large hardware simulations!

Further integration with existing supercomputer simulation models

Nemo will have support for JSON neuron models

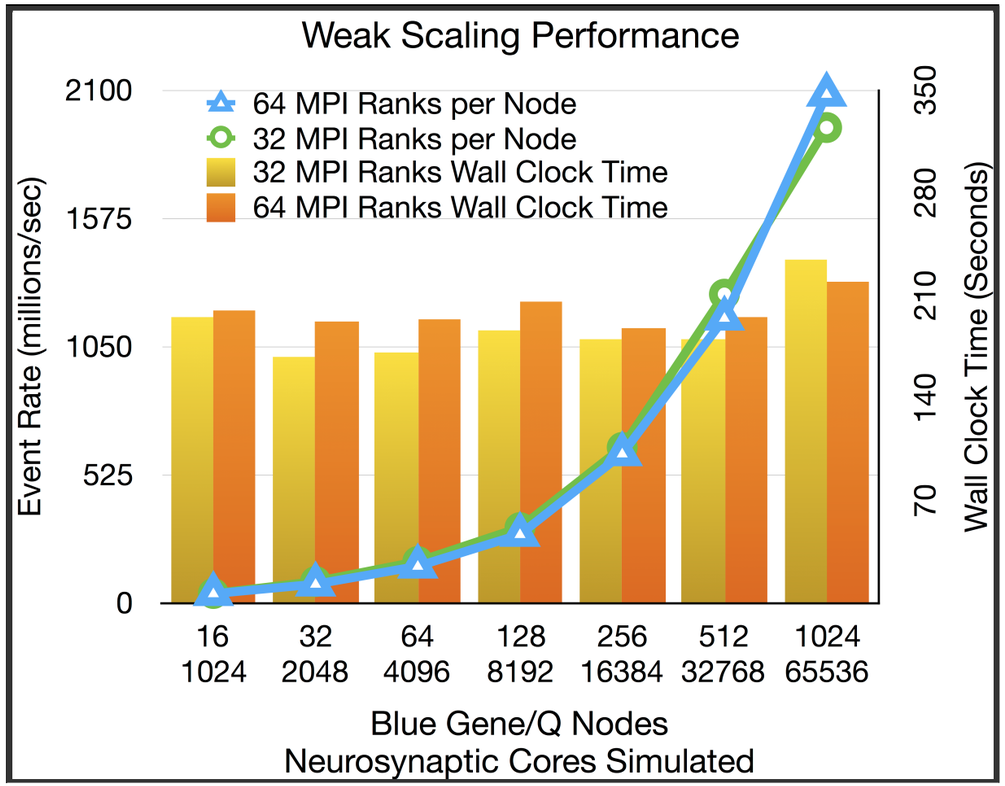
Input support through Python

With Pythonic descriptors of models

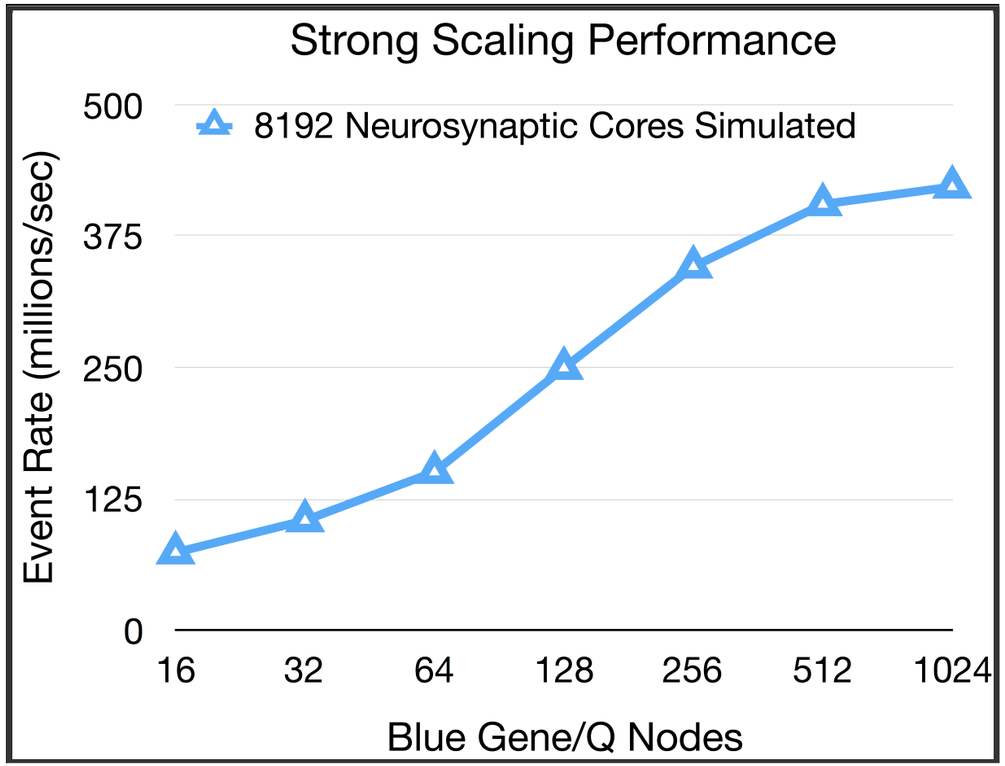
NeMo will gain flexibility

Comming soon to a simulator near you!

NEMO EXPEREMENTAL RESULTS



- NeMo shows excellent weak scaling performance



- Strong scaling shows initially excellent results
- The simulation begins to loose parallelism
- Larger simulation runs should improve strong scaling

FUTURE WORK

The next steps for nemo include:

- Integration with supercomputer simulation software
- JSON neuromorphic core design support
- Pythonic I/O for integration with spiking neural network tools