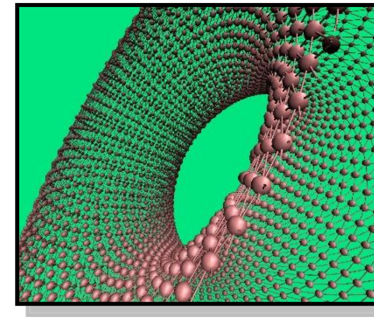
SpiNNaker
Biologically Inspired Massively Parallel Architectures



Discrete event-based neural simulation using the SpiNNaker system

Andrew Brown
University of Southampton
adb@ecs.soton.ac.uk

Jeff Reeve
University of Southampton
jsr@ecs.soton.ac.uk

Kier Dugan
University of Southampton
kjd1v07@ecs.soton.ac.uk

Steve Furber
University of Manchester
steve.furber@manchester.ac.uk

What *is* SpiNNaker?

- It is *not*:

  Just another massive parallel machine

- It *is*:

  1,000,000

  64k:32k D:I memory,
  ARM 9, no floating point

  A *large number* of *relatively small* cores embedded in a *powerful* bespoke *hardware* communication fabric

  Bisection bandwidth
  250 Gb/s

Bio-inspiration: BIMPA

- *How can massively parallel computing resources accelerate our understanding of brain function?*

- *How can our growing understanding of brain function point the way to more efficient parallel, fault-tolerant computation?*

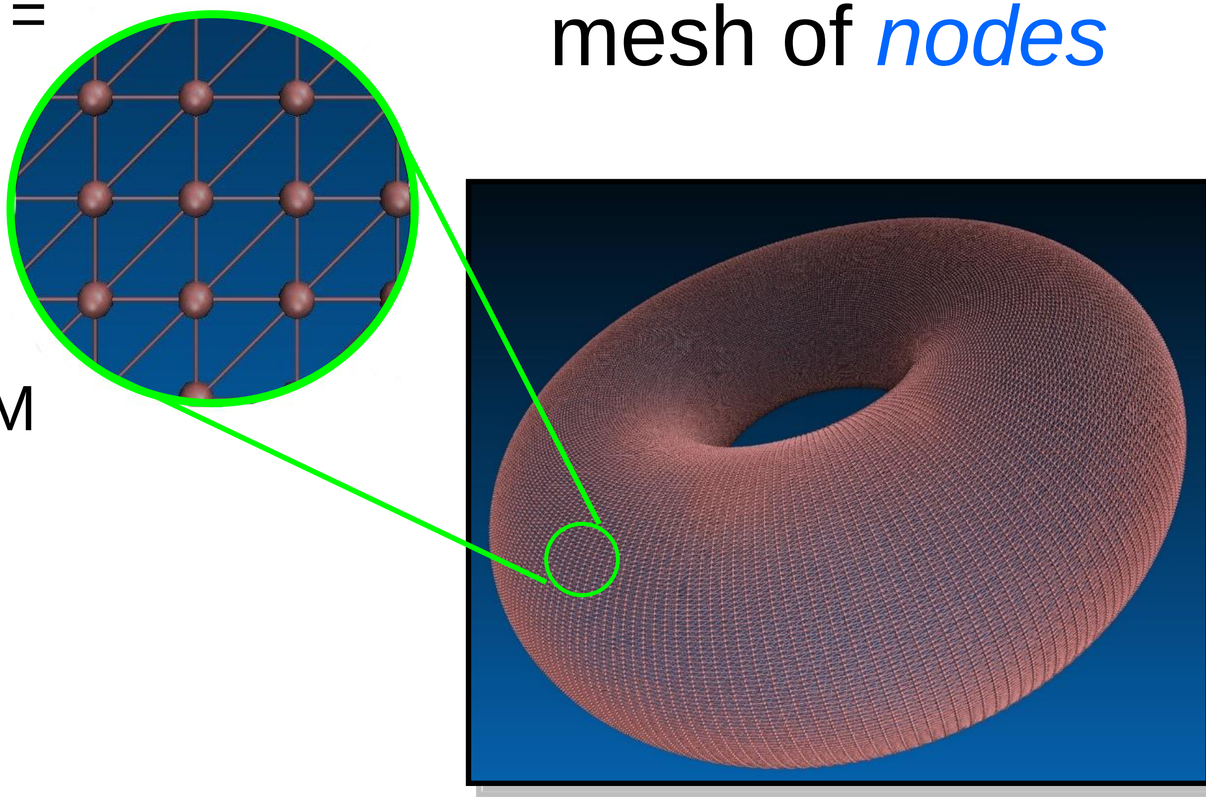
Outline

- The ***SpiNNaker*** system
- Configuration
- Time models itself
- Neural simulation

Machine architecture

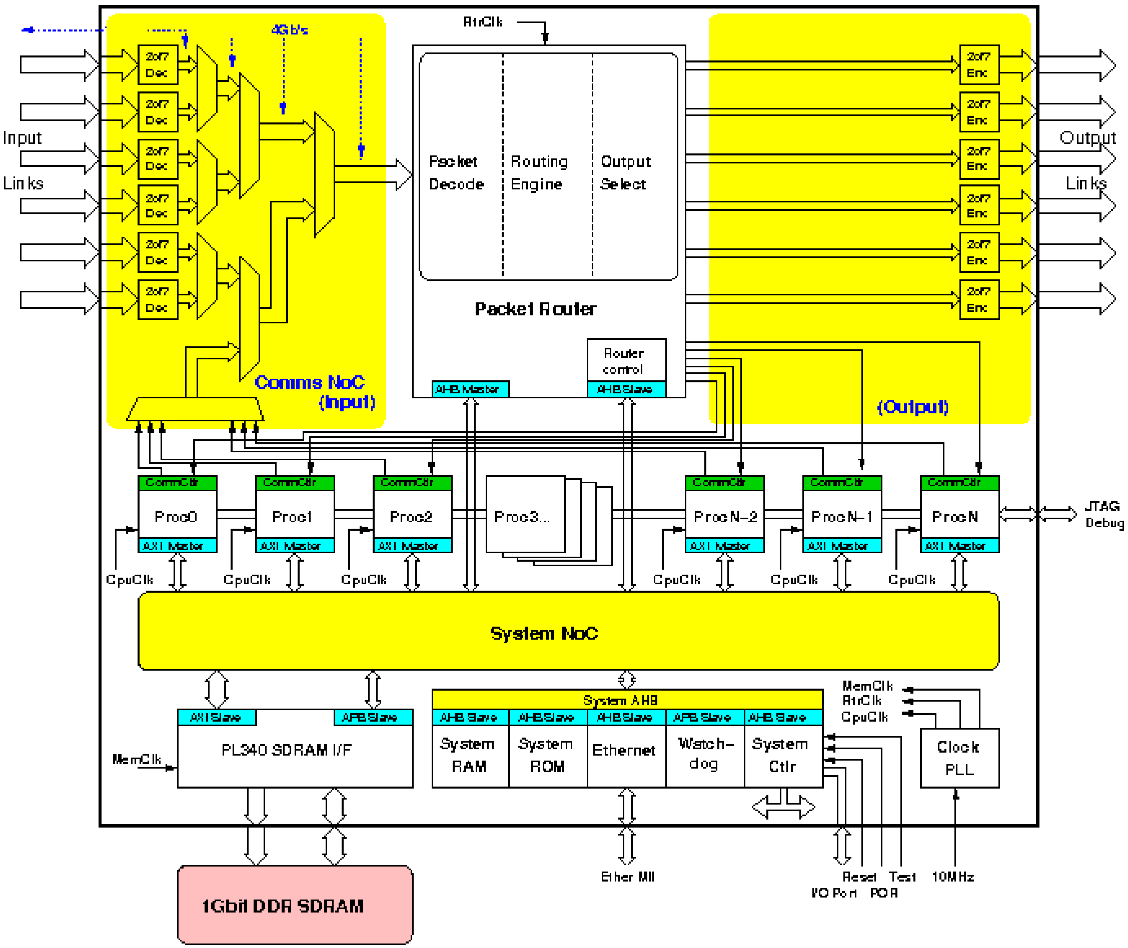
- 1 engine =
  256x256 toroid =
  65536 nodes
- 1 node =
  18 cores
  + comms
  + 128M SDRAM
- 1 core =
  ARM9
  + 64k DTCM
  + 32k ITCM

- Triangular mesh of *nodes*



A Spinnaker node

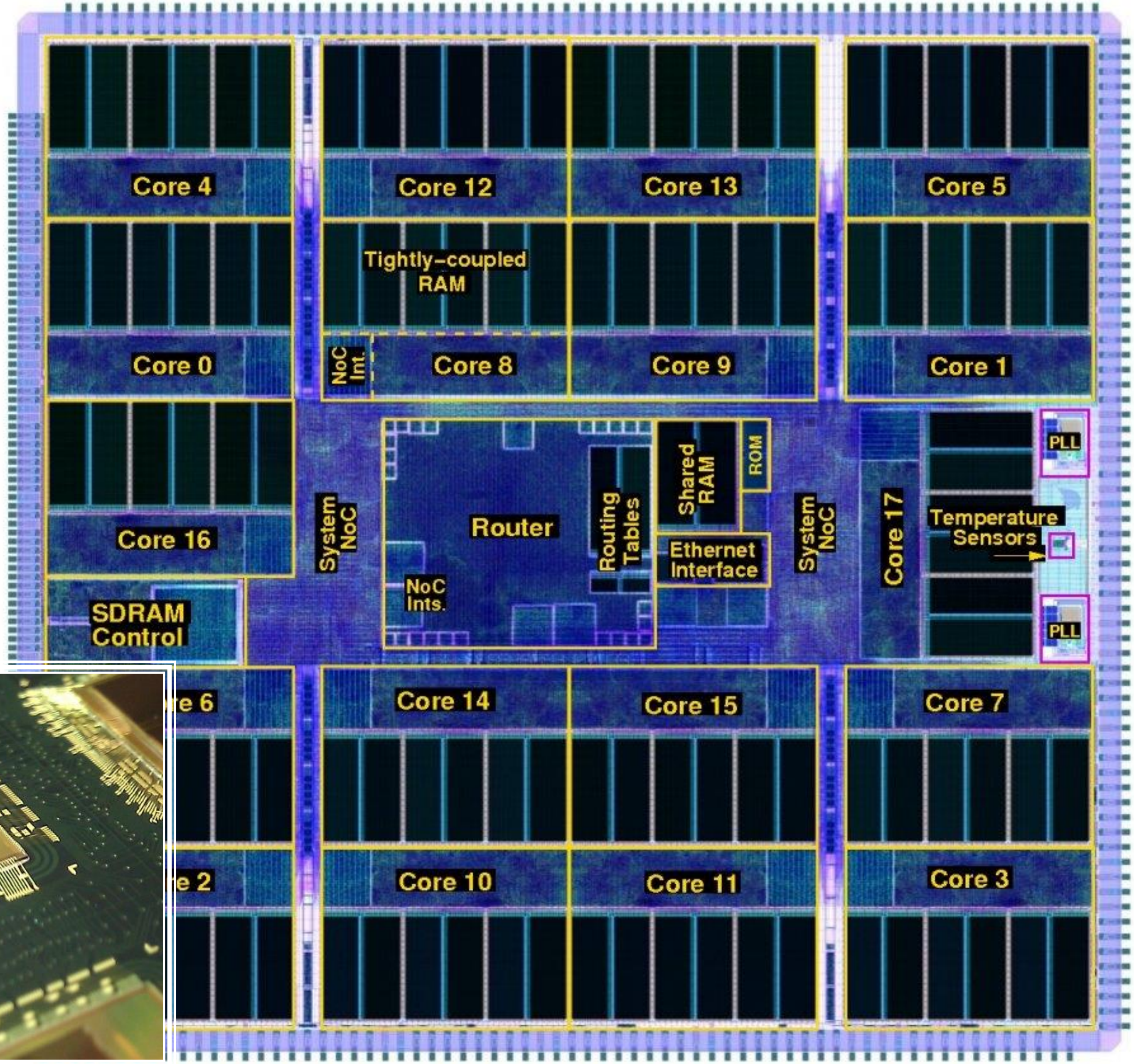
- 6 bi-directional comms links
- Core farm
    - *(1 monitor)*
- System...
    - NoC
    - RAM
    - Watchdogs
- Off-die SDRAM

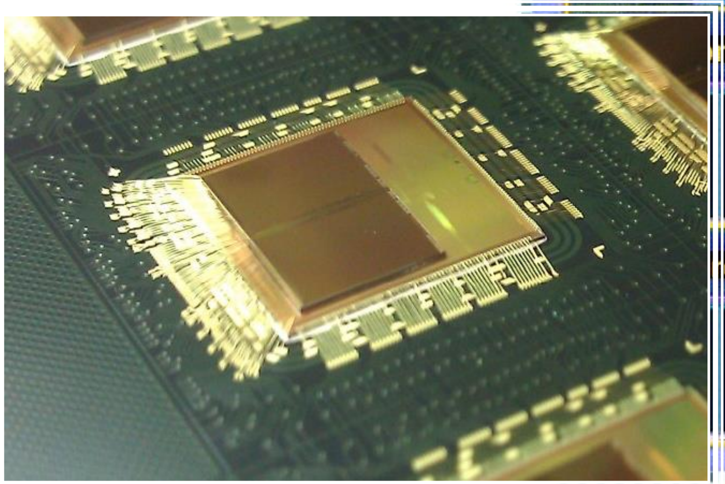


SpiNNaker
Biologically Inspired Massively Parallel Architectures

102 machine

18 cores





Physical construction

103 machine

<u>48 nodes:</u>

48 nodes x

18 cores

= *864* cores

Physical construction

104 machine

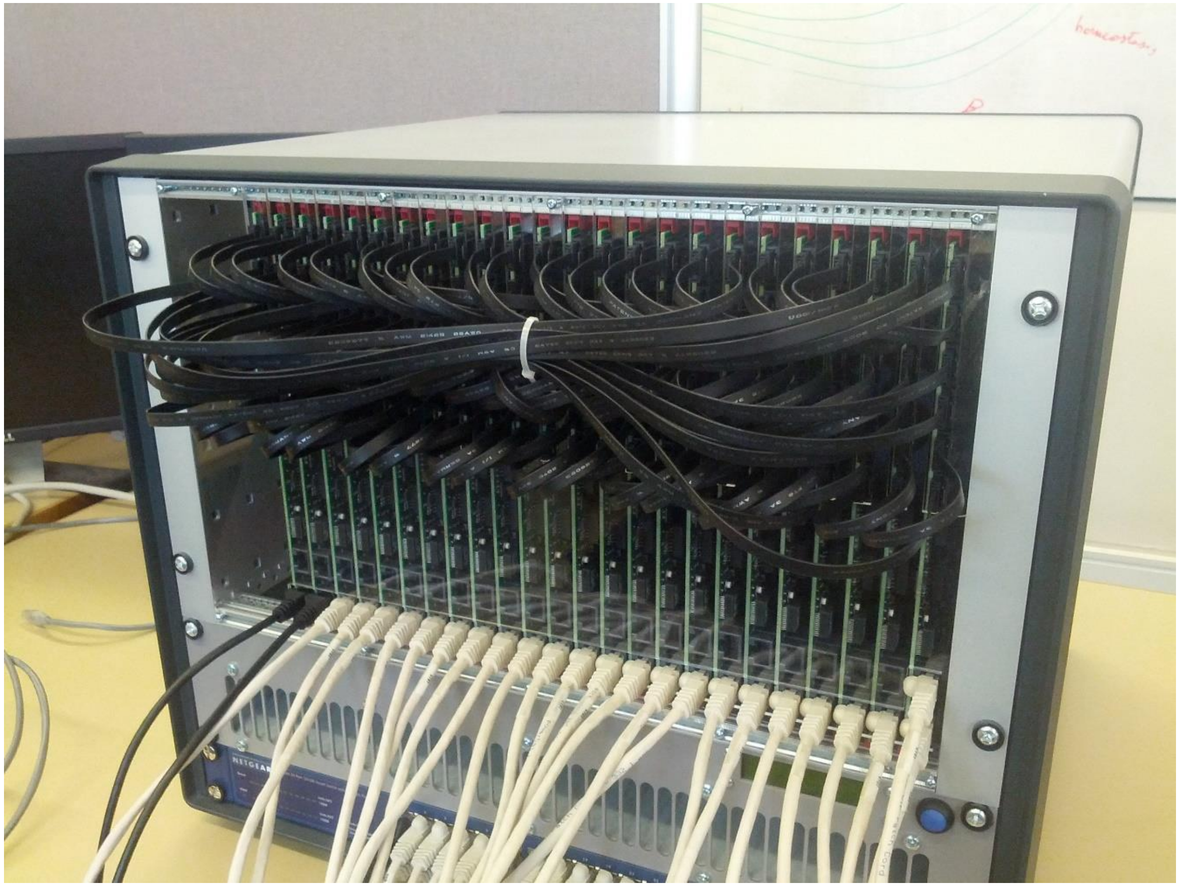
<u>24 boards:</u>

24 boards x

48 nodes x

18 cores

= *20736* cores



Physical construction

105 machine

<u>5 racks:</u>

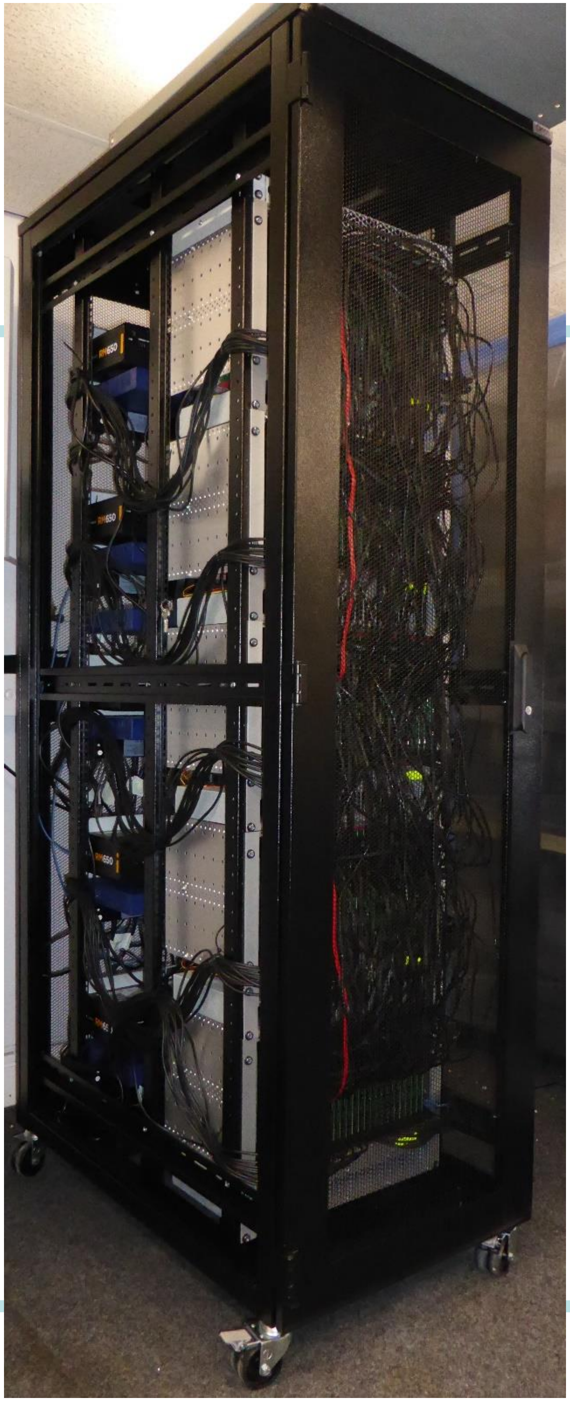
5 racks x

24 boards x

48 nodes x

18 cores

= *103680* cores

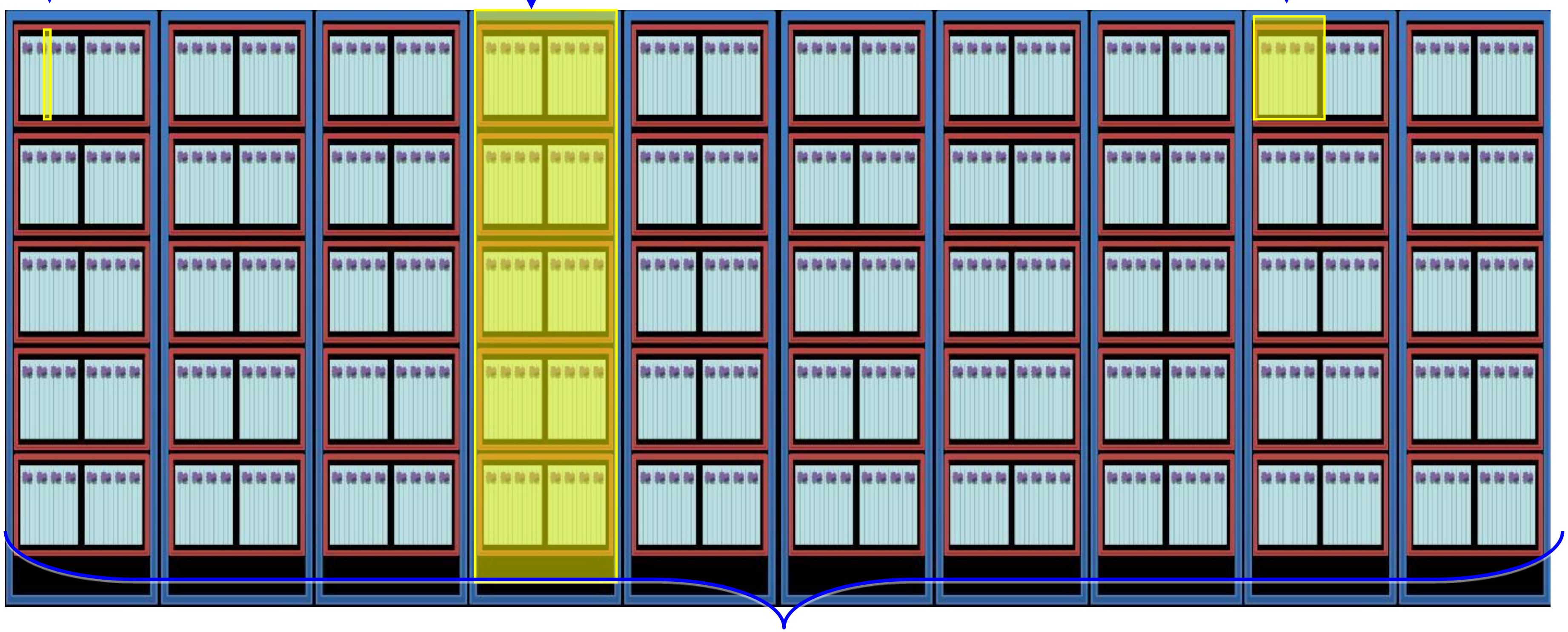


...and the machine yet to be assembled:

103 machine: 864 cores, 1 PCB, ~75W

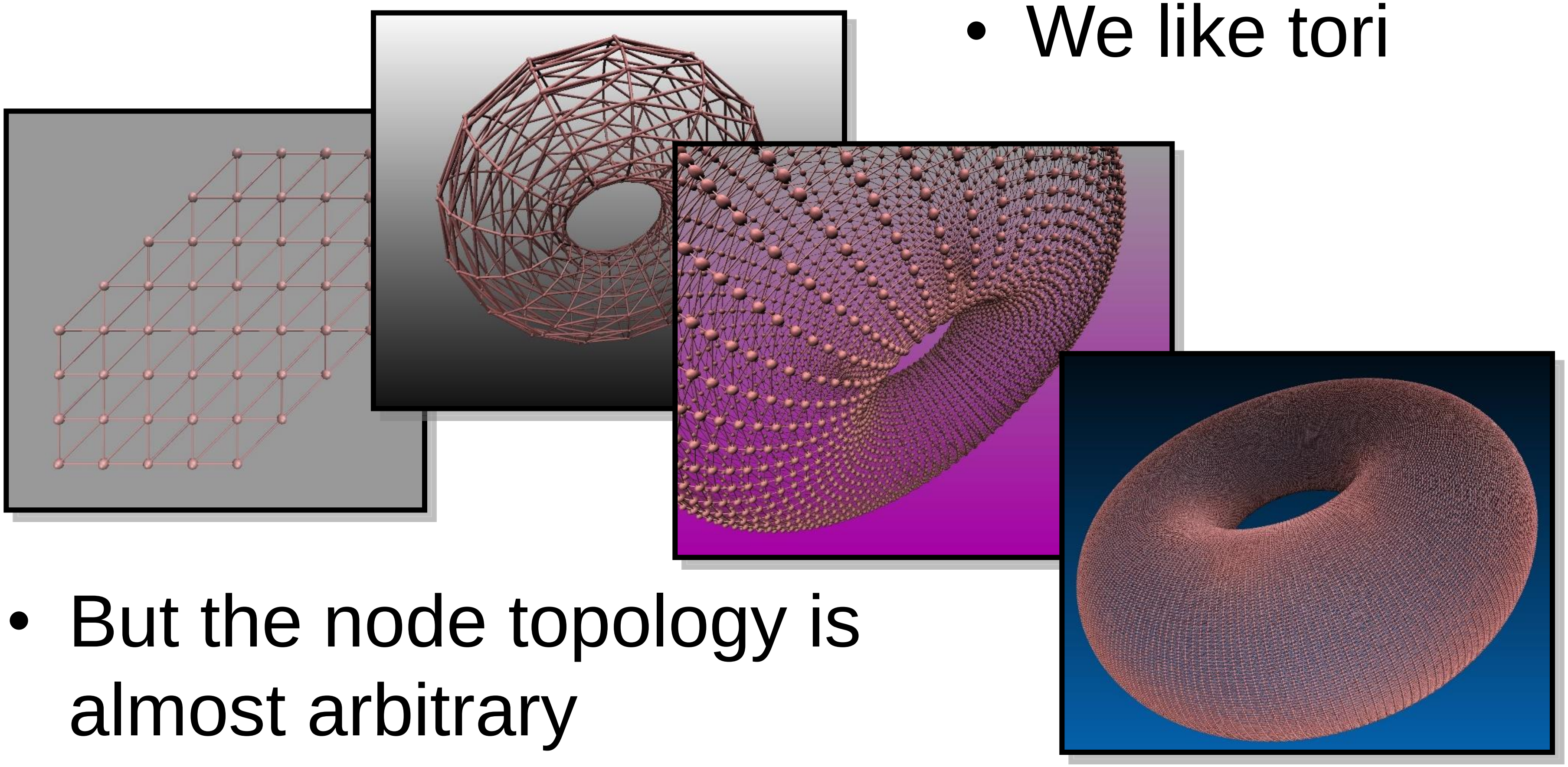
104 machine:20,736 cores, 1 rack, ~1900W
(24 PCBs, operation *without* aircon)

105 machine: 103,680 cores, 1 cabinet, ~9kW



106 machine: 1M cores, 10 cabinets, ~90kW

Scalable system …
… arbitrary topology

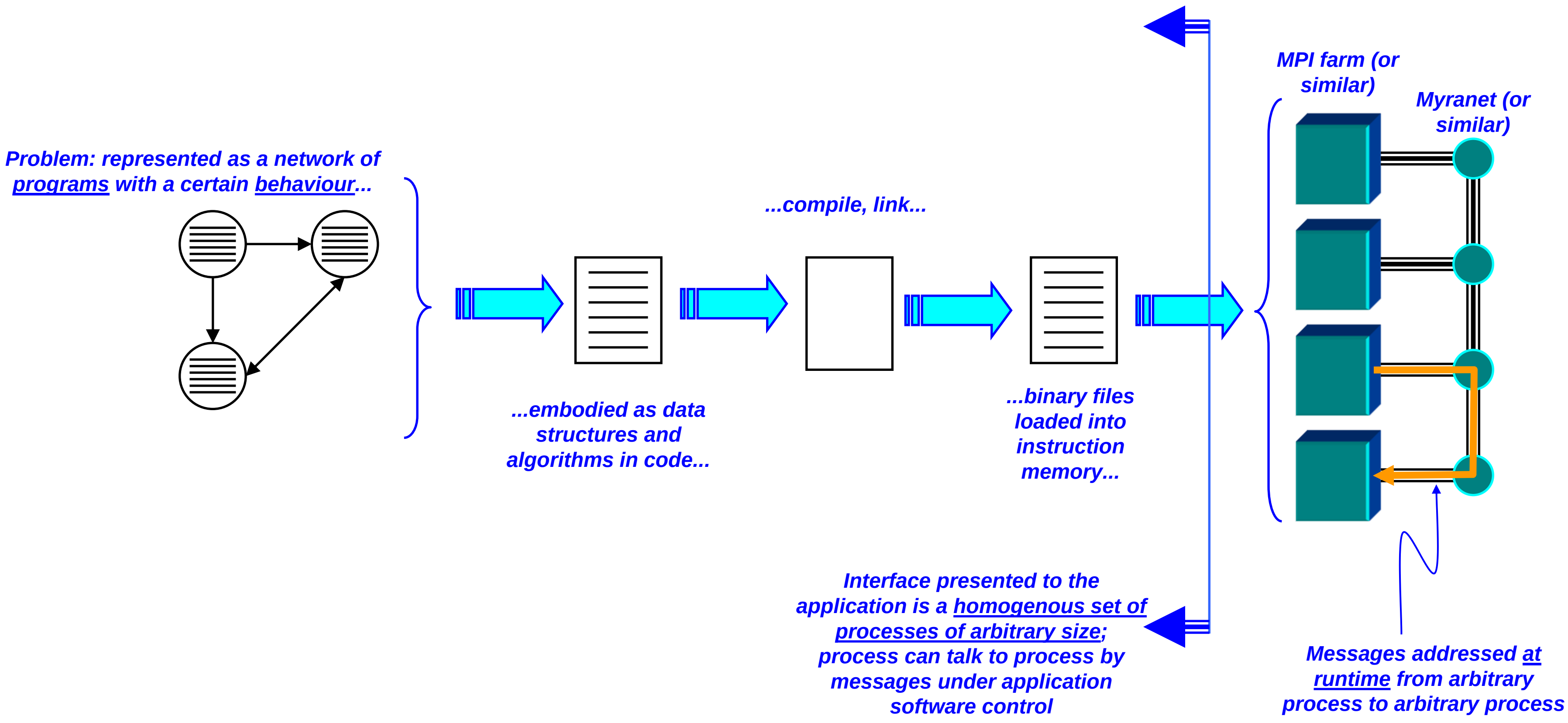


- We like tori
- But the node topology is almost arbitrary

Outline

- The ***SpiNNaker*** system
- Configuration
- Time models itself
- Neural simulation

A conventional multi-processor program:



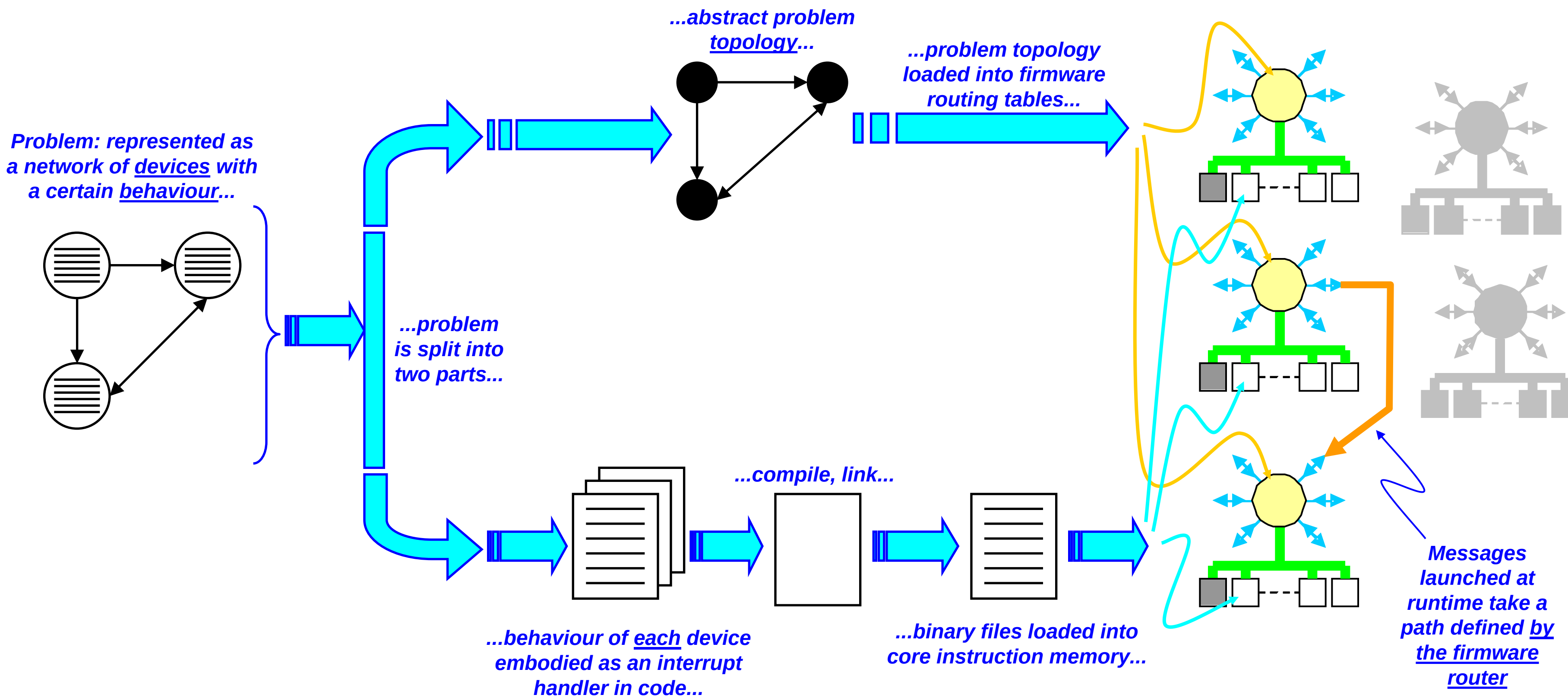
...and you might reasonably expect:

- Blocking and non-blocking send/receive
- Probing the queues
- Broadcasting
- Scatter-gather
- Parallel I/O
- Remote memory access
- Dynamic process management

On SpiNNaker...

- The problem (***C***ircuit ***u***nder ***S***imulation) is defined as a *graph*
- Torn into two components:
  - – CuS *topology*
    - Embodied as hardware route tables in the *nodes*
  - – Circuit device *behaviour*
    - Embodied as software event handlers running on *cores*

On SpiNNaker:



The code says "send message" but has no control where the output message goes - the route tables in each node decide

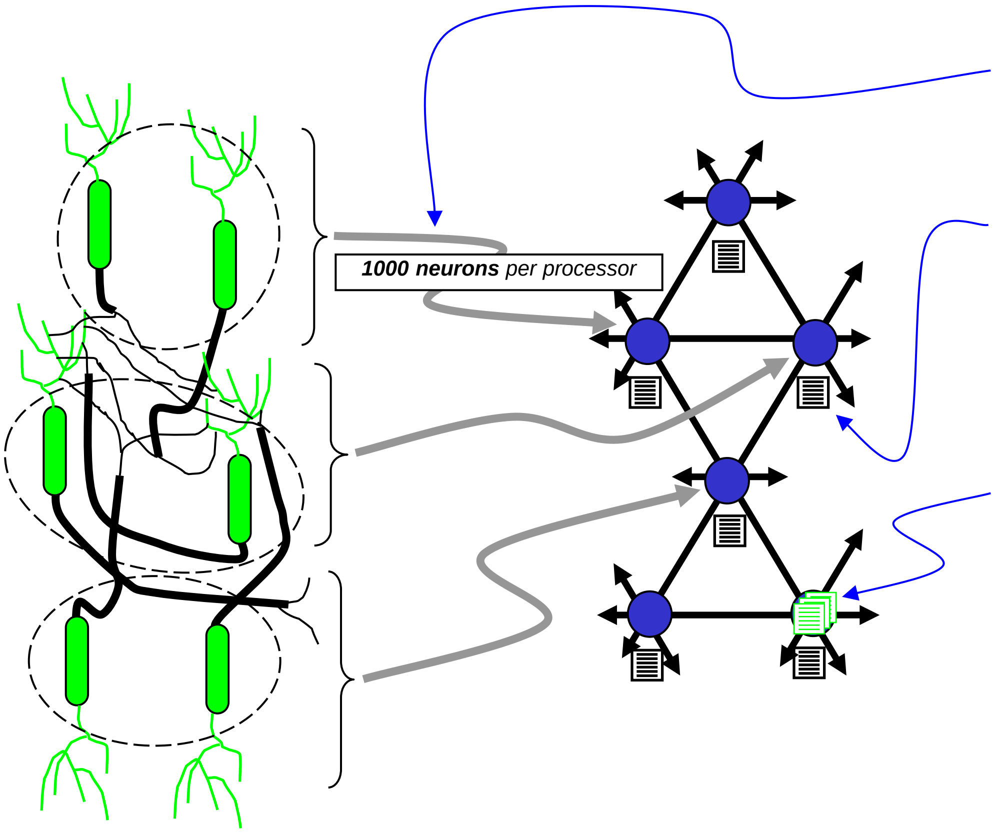
OS, S/W environment

- What you expect:
  - ~~– File I/O~~
  - ~~– Console output~~
  - ~~– Memory management~~
  - ~~– Interactive debug~~
  - ~~– Libraries~~
  - ~~– The time~~
- What each handler gets:
  - Read access to 72 bits of the packet that woke it
  - Knowledge of incoming port (0..5) - not very useful
  - I/O to its own memory map
  - Ability to send packets
  - Knowledge of local node and core identifier
  - Coarse interval signal

And that's all, folks

SpiNNaker configuration

Offline configuration software maps neurons:cores (~1000:1)



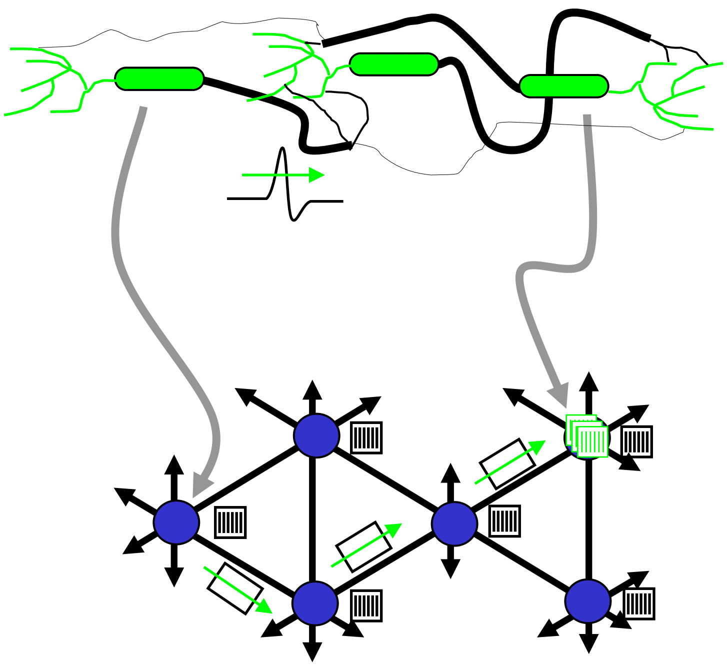
Maps each individual *neuron* to a SpiNNaker *core*

Defines the router tables for each *node*
Connectivity of neural topology is *distributed* throughout the system in the routing tables

Defines the index structures necessary in each core to allow fast retrieval of neuron and synapse state
Defines the packet handling *code (interrupt handlers)*

SpiNNaker configuration

Biology



Neurons communicate via *spikes* traveling along *axons/dendrites*

Cores (and hence the neuron models resident within them) communicate via 72-bit *hardware packets* traveling through the routing structure, hopping from node to node as directed by the *routing tables* in each node

SpiNNaker

Event handlers? Interrupts?

- Packet arrives at a core:
  - Hardware invokes an *interrupt handler*
    - Tied to a neuron
  - Handler modifies neuron state
    - May/may not launch packets as a consequence
- Handlers are *tiny*; they *execute*; they *stop*

And that's all you have to play with

What exactly is a packet?

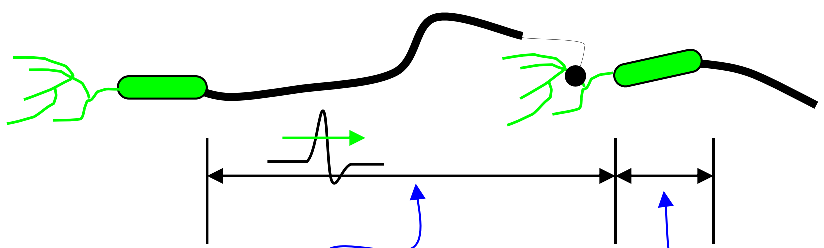
- Hardware
  - Fixed bit length
- Address event representation (AER)
- Packets delivered from source neuron to target neuron
  - Source node address|source core address|source neuron address
- Physical route embodied in route tables
  - Distributed

Outline

- The ***SpiNNaker*** system
- Configuration
- Time models itself
- Neural simulation

Time

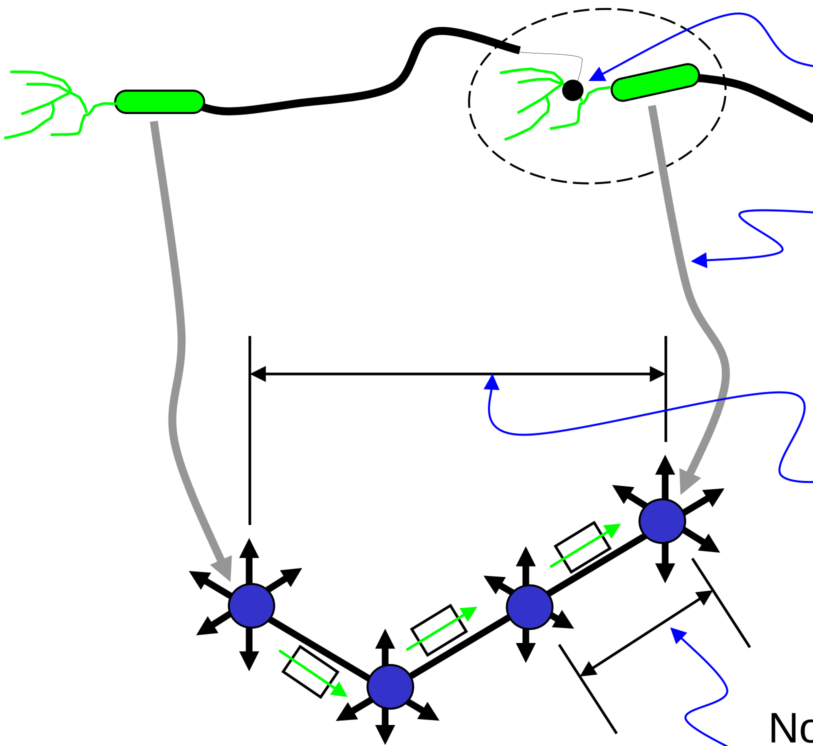
Biology:



Axonal delay ***O***(ms) – fn(biological geometry)

Neuron processing time ***O***(ms) – fn(biology & state(history))

Time



Axonal delay stored as parameter in *synapse* state local to neuron model

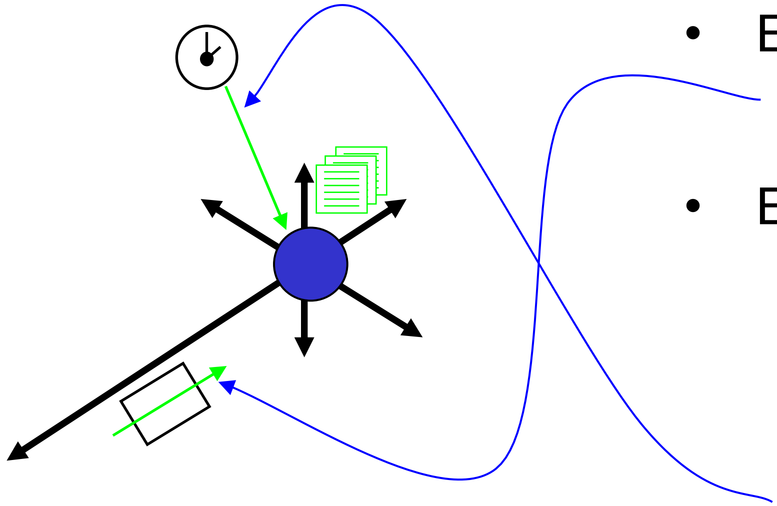
Neuron-core mapping – fn(graph mapping software)

Neuron-neuron wallclock delay *maximum* ***O***(10us) – fn(graph mapping, traffic density & engine size)

Node-node wallclock hop delay ***O***(100ns) – fn(graph mapping & traffic density)

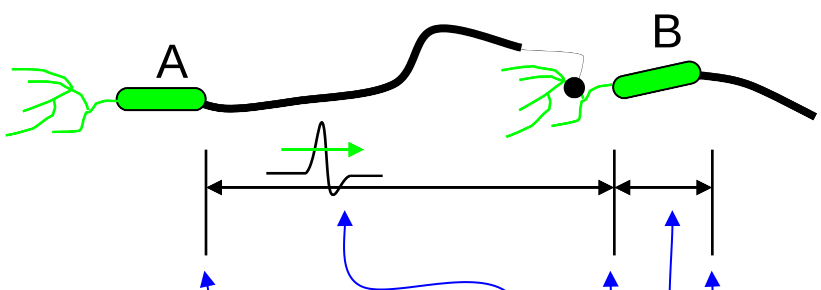
SpiNNaker

There are different <u>sorts</u> of interrupts



- Each *core*
  - Packet handling interrupt
    - Invoked by incoming packet
- Each *node*
  - Biological clock tick handling interrupt
  - Clocks are not phase locked
  - Slow ***O***(kHz)
  - '(Biological) time is passing' signal
  - Asserted on every core

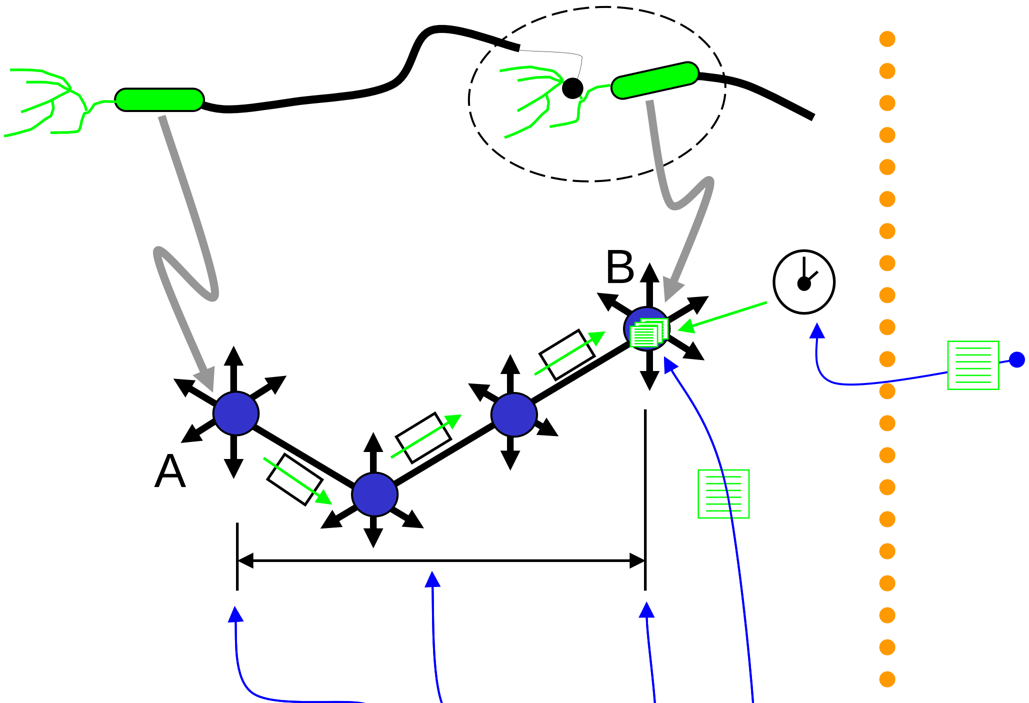
Back to biology



- A fires when it fires
- Pulse propagates to B
- Arrives when it arrives
- B integrates incoming pulse(s)
- Fires when it fires

No synchronising clock
Event driven
Data push

Back to SpiNNaker



In parallel with (and not synchronised to) this:
Biological clock ticks
Triggers an interrupt with each tick

A fires when it fires
Launches a packet to B
Arrives ***O***(us) later
Triggers 'packet arrived' interrupt

A closer look at the interrupt handlers

Packet arrival handler

```
Remove packet from
router;
Store in buffer in
synapse (age = 0)
```

Clock tick handler

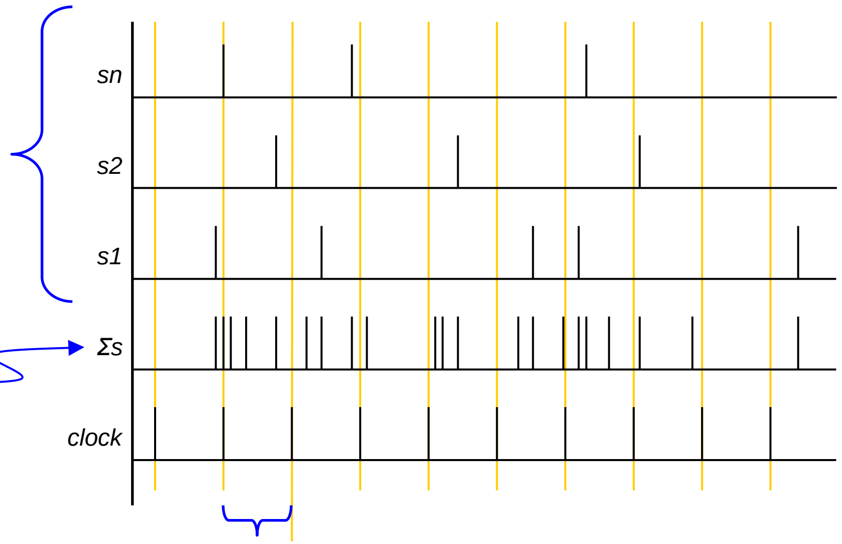
```
Increment age of buffered
packets;
If any 'arrived' (age ==
synapse delay), assert
onto neuron state
equations;
Integrate (one timestep)
neuron state equations
```

SpiNNaker
Biologically Inspired Massively Parallel Architectures

Neural simulation

***Individual** message frequencies < real-time clock*

***Superposition** of all inputs: exact timing = fn(neuron:core) i.e. independent of CuS (bad) BUT message latency << CuS time constants (so it doesn't matter)*



Change of neuron state derived locally, stored until next (biological) timestep

Change of neuron state broadcast (or not) at next (biological) timestep

And this works because:

- Biological wallclock time modelled locally at each node
  - (and thus each neuron modelled within it)
- At each time tick
  - Inputs added if age suitable
  - Equations integrated
  - States updated
- Wallclock packet transit delay is *negligible* and *ignored*
- *Biological* delay captured in target synaptic model state
- Differential equations controlling neuron model behaviour are not stiff
  - All time constants >> biological clock tick
  - Forward Euler / Runge/Kutta stable

Limitations

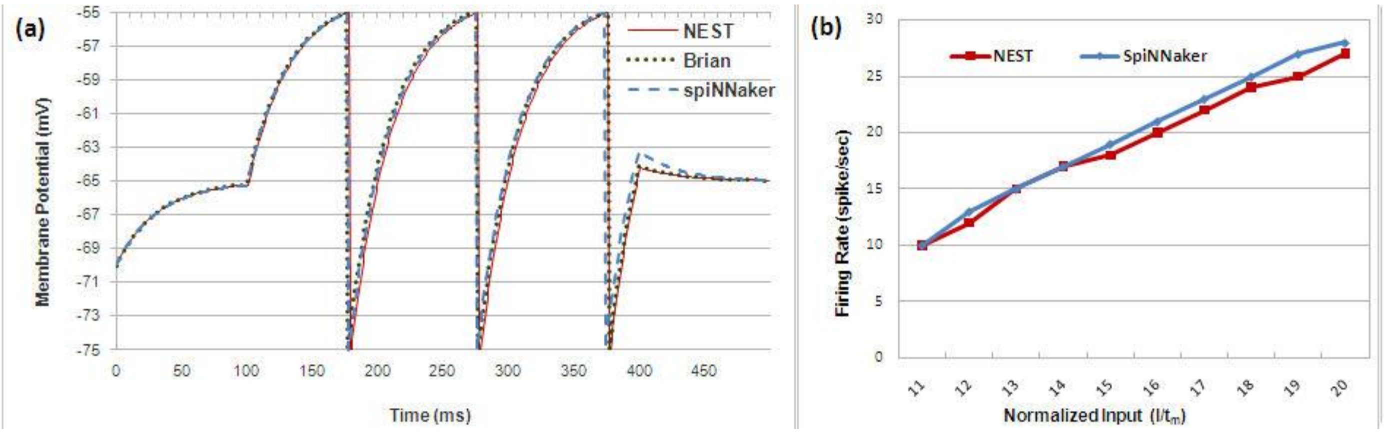
- SpiNNaker designed to operate in *real time*
  - Simulation 'speed' a hard metric to interpret
- Communication via hardware packets
  - 16 bits/node => 65536 nodes/machine
  - 4 bits/core => 16 cores/node
  - 10 bits/neuron => 1024 neurons/core
- Hard limit of 1,073,741,825 neurons

Outline

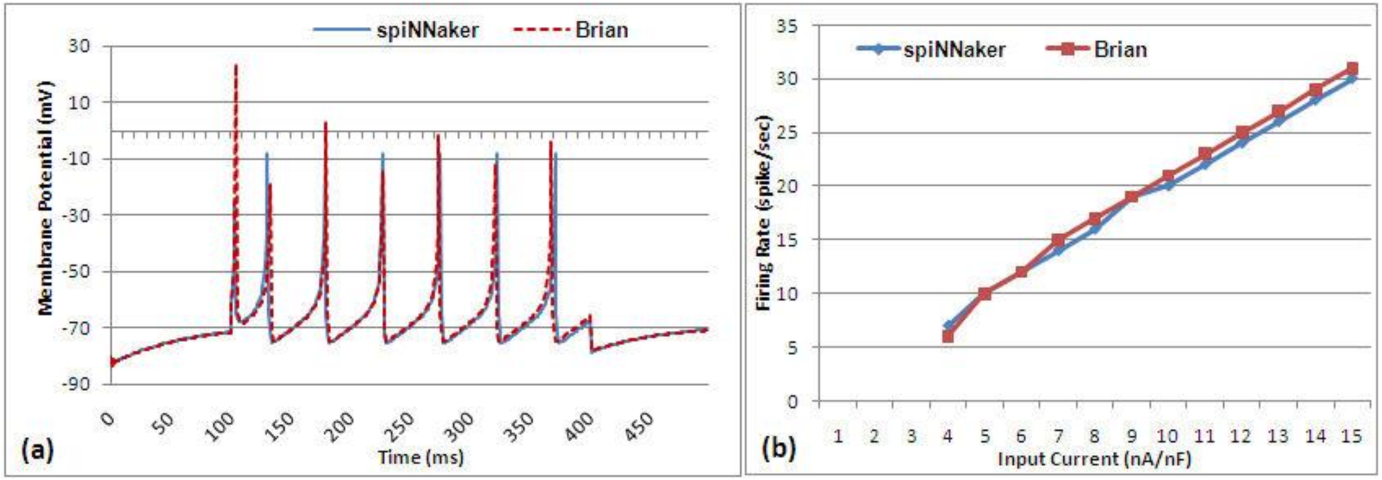
- The ***SpiNNaker*** system
- Configuration
- Time models itself
- Neural simulation

Comparisons

- LIF

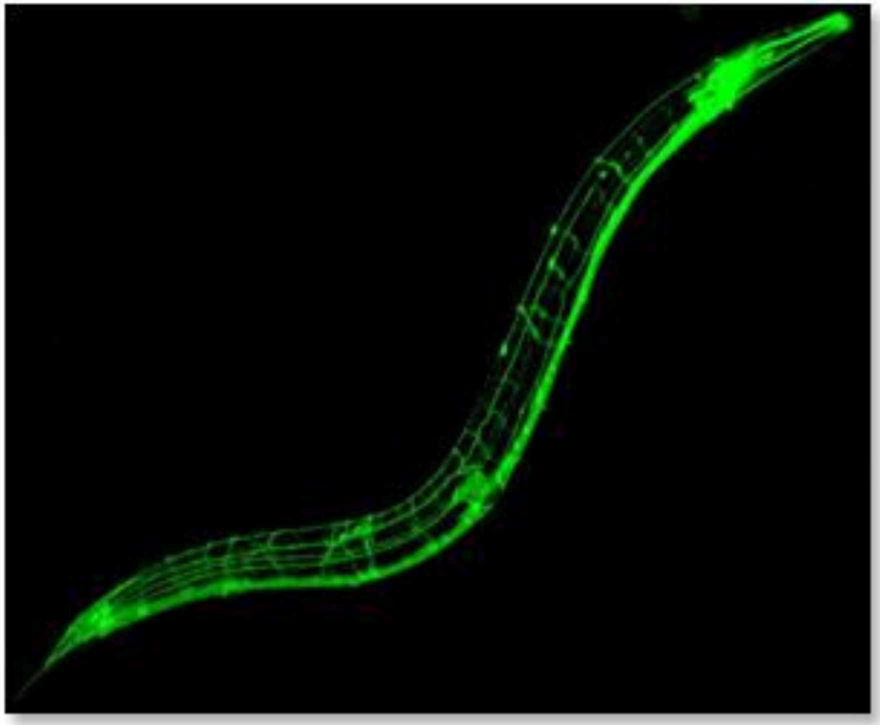


- Izhikevich



Norman the nematode

- C. elegans
  - ~300 neurons
  - Chemotaxic



Bessereau Laboratories

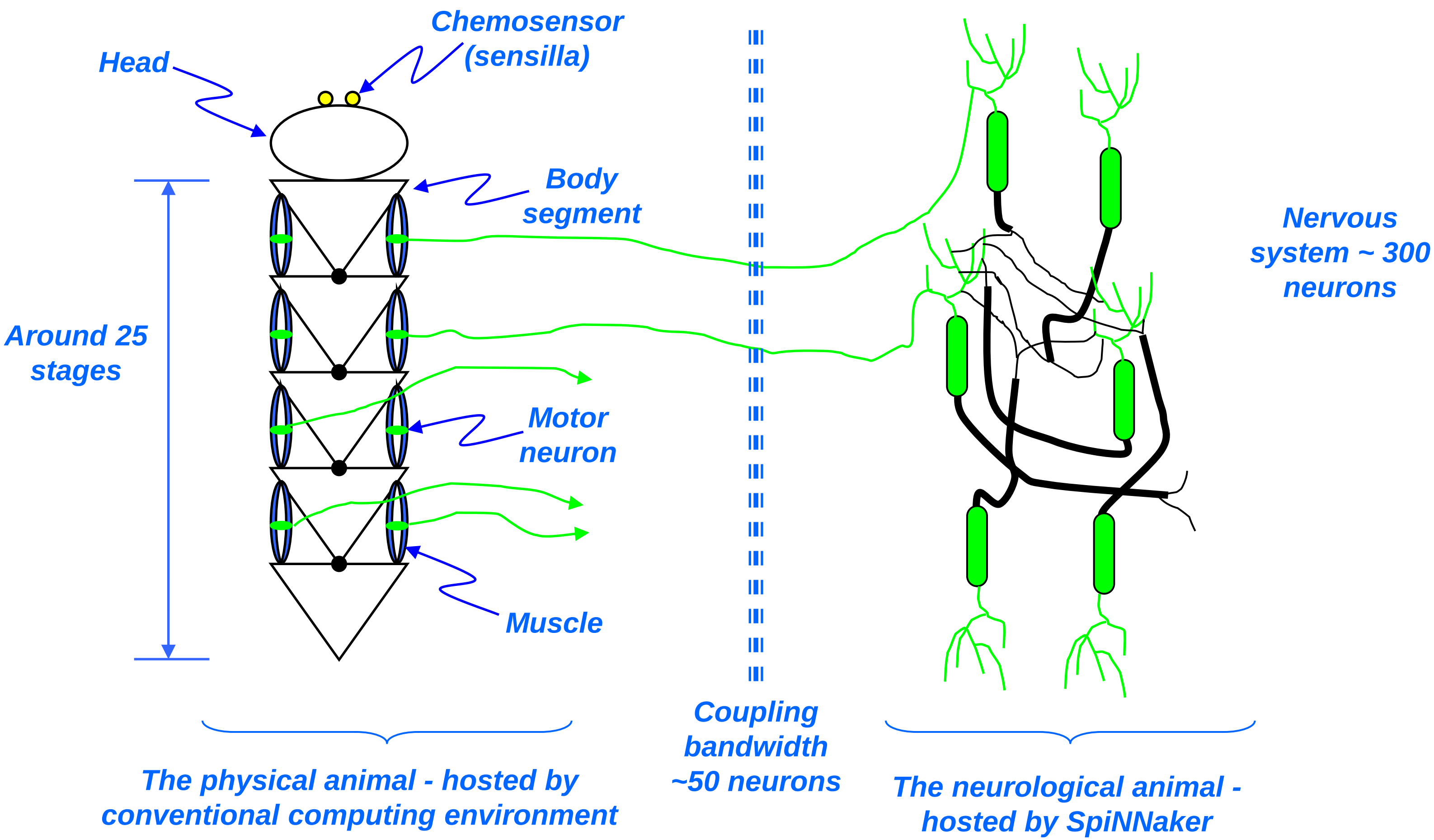
Of worms and environments

- Worm locomotion defined by interaction with the environment
- Motor neuron is *proprioceptive* (bidirectional)
- To move, Norman interacts with ambient on a distance scale *comparable to stride length*
  [viscosity/locomotion studies]

To do useful science.....

- If Norman is in a virtual environment
- Coupling at granularity level requiring ~1 connection/*motor neuron*
- NOT a few connections/*animal*

Norman abstracted



Neuronscape

- A neurophysiological workbench:
  - Can provide this level of interaction
  - Move the focus to a finer level of granularity in the local environment
  - Requires ~ 50 links/animal
- SpiNNaker can do this
  - Group dynamics ~5000 animals
  - Replace mechanical linkage in the virtual environment
    - Non-neural physical interactions
    - Brokered by SpiNNaker packets

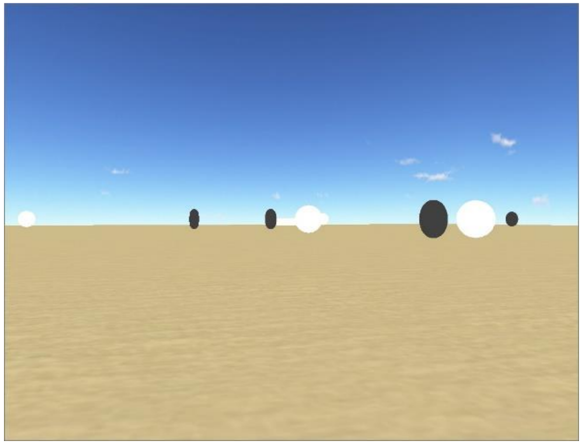
Neuronscape - concept

Artificial environments

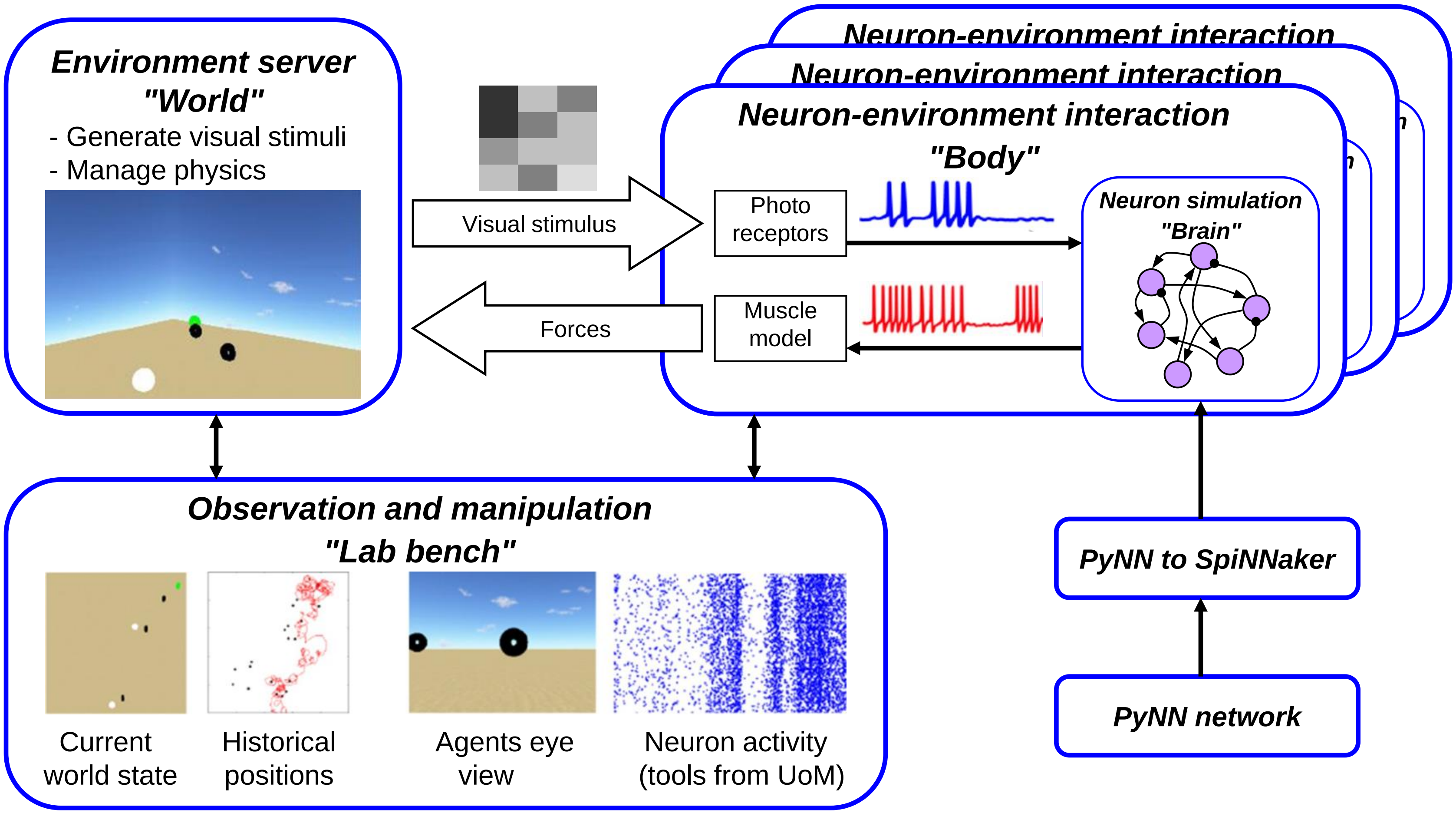
De facto technique for neural development studies

Controlled environment -

Real time interaction with :
- Other Beasties hosted on SpiNNaker
- Other Beasties hosted on conventional machines
- Humans - Turing test



Neuronscape internals



Putting it all together