6. SPIKESTREAM[1]

6.1 Introduction

This chapter outlines the spiking neural simulator that was developed to model the 'conscious' neural network described in Chapter 5. This simulator had to be fast, it had to exchange spikes with the SIMNOS robot, it needed to support different neuron and synapse models, and the ability to record a network's activity was essential for the synthetic phenomenology in Chapter 7. A substantial amount of fine tuning of the network was required, and so an intuitive graphical interface with good monitoring facilities was also desirable.

Since none of the available simulators met these requirements (see Section 5.2.4), I developed a new spiking neural simulator, SpikeStream, that can be used to edit, display and simulate up to 100,000 neurons. This simulator uses a combination of event-based and synchronous simulation and stores most of its information in databases, which makes it easy to run simulations across an arbitrary number of machines. A comprehensive graphical interface is included and SpikeStream can send and receive spikes to and from real and virtual robots across a network. The architecture is highly modular, and so other researchers can use its graphical editing facilities to set up their own simulations or use their own code to create networks in the SpikeStream databases.

The first part of this chapter outlines the different components of the SpikeStream architecture and sets out the features of the graphical interface in more detail. Next, the performance of SpikeStream is documented along with its communication with external devices. The last part of this chapter suggests some applications for SpikeStream, describes its limitations and gives details about the SpikeStream release under the terms of the GPL license. Much more

[1] An earlier version of this paper was published as Gamez (2007b).

detailed information about the features and operation of SpikeStream is given in the SpikeStream Manual, which is included as the first appendix to this thesis.

6.2 Architecture

SpikeStream is built with a modular architecture that enables it to operate across an arbitrary number of machines and allows third party applications to make use of its editing, archiving and simulation functions. The main components of this architecture are a number of databases, the graphical SpikeStream Application, programs to carry out simulation and archiving functions, and dynamically loaded neuron and synapse classes.

6.2.1 Databases

SpikeStream is organized around a number of databases that hold information about the network model, patterns and devices. This makes it easy to launch simulations across a variable number of machines and provides a great deal of flexibility in the creation of connection patterns. The SpikeStream databases are as follows:

- *Neural Network*. Each neuron has a unique ID and connections between neurons are recorded as a combination of the presynaptic and postsynaptic neuron IDs. The available neuron and synapse types along with their parameters are also held in this database.
- *Patterns*. Holds spatiotemporal patterns that can be applied to the network for training or testing.
- *Neural Archive*. Stores archived neuron firing patterns. Each archive contains an XML description of the network and data in XML format.
- *Devices*. The devices that SpikeStream can exchange spikes with over the network.

These databases are edited by SpikeStream Application and used to set up the simulation run. They can also be edited by third party applications - to create custom connection patterns or neuron arrangements, for example - without affecting SpikeStream's ability to visualize and simulate the network.

6.2.2 SpikeStream Application

An intuitive graphical user interface has been written for SpikeStream (see Figure 6.1) with the following features:

- *Editing.* Neuron and connection groups can be created and deleted.
- *3D Visualisation.* Neuron and connection groups are rendered in 3D using OpenGL and they can be rotated, selectively hidden or shown, and their individual details displayed. The user can drill down to information about a single synapse or view all of the connections simultaneously.
- *Simulation.* The simulation tab has controls to start and stop simulations and vary the speed at which they run. Neuron and synapse parameters can be set, patterns and external devices connected and noise injected into the system.
- *Monitoring.* Firing and spiking patterns can be monitored and variables, such as a neuron's voltage, graphically displayed.
- *Archiving.* Archived simulation runs can be loaded and played back.

Much more information about the graphical features of SpikeStream can be found in Appendix 1.

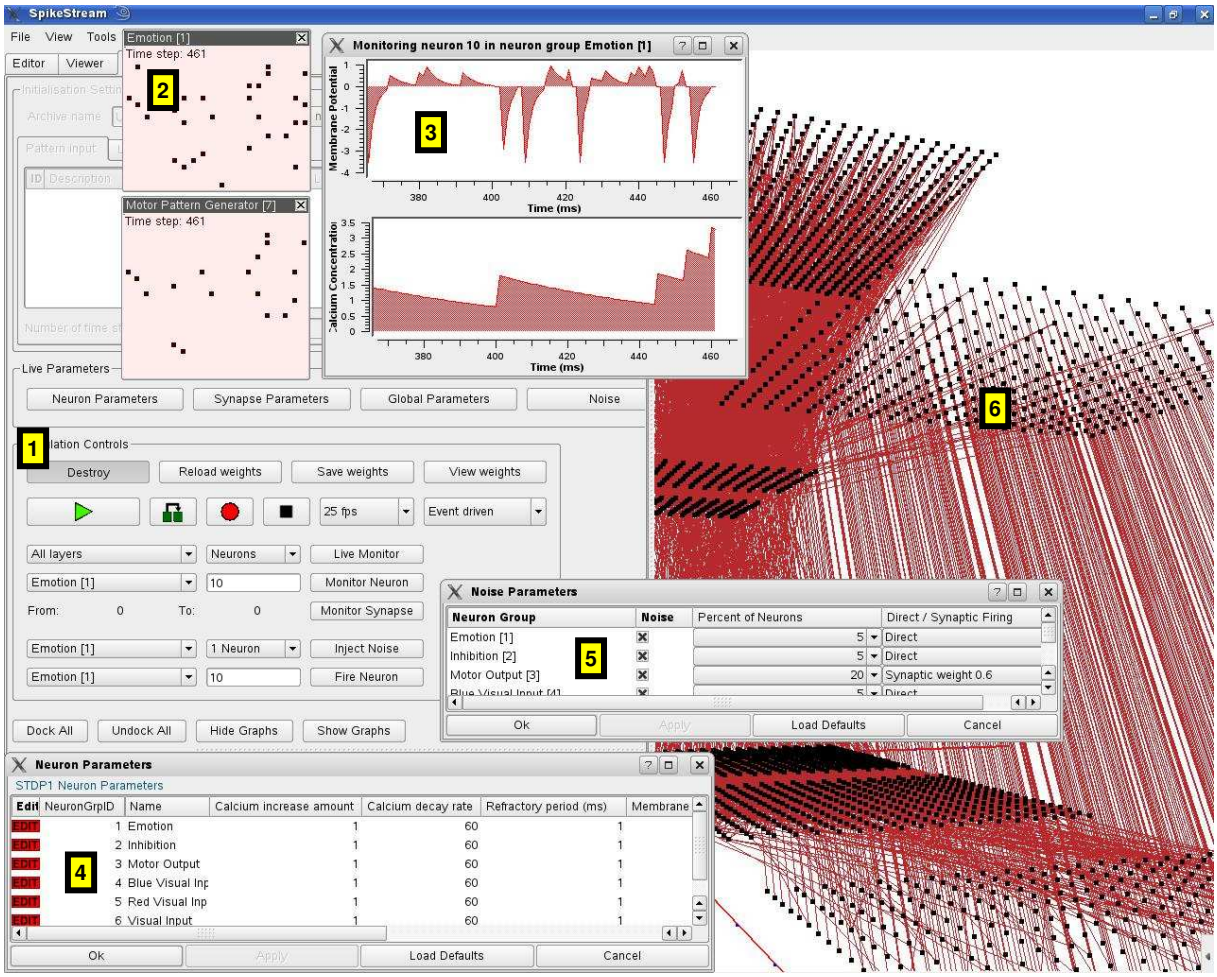

Figure 6.1. SpikeStream graphical user interface. The numbers highlighted in yellow indicate the following features: (1) Simulation controls, (2) Dialog for monitoring the firing of neurons in a layer, (3) Dialog for monitoring variables inside the neurons, such as the calcium concentration and voltage, (4) Dialog for viewing and setting neuron parameters (5) Dialog for viewing and setting the noise in the network, (6) 3D network view.

6.2.3 SpikeStream Simulation

The SpikeStream simulator is based on the SpikeNET architecture (Delorme and Thorpe 2003) and it consists of a number of processes that are launched and coordinated using PVM, with each process modelling a group of neurons using a combination of event-based and synchronous simulation.[2] In common with synchronous simulations the simulation period is divided into steps with an arbitrarily small time resolution and each neuron group receives lists of spikes from

[2] One difference between SpikeStream and SpikeNET is that messages are sent rather than requested at each time step.

other connected layers at each time step. However, only the neuron and synapse classes that receive a spike are updated, which substantially cuts down on the amount of processing required. Since the main overhead is calculating the neurons' state and sending the spikes, the simulator's update speed depends heavily on the level of network activity, and at high levels the performance becomes the same as a synchronous simulator. In theory, SpikeStream's run speed should be relatively independent of the time step resolution, since the calculation of each time step is efficient and the network should emit the same number of spikes per second independently of the time step resolution. In practice, the setting of this value can affect the number of spikes emitted by the network because higher values reduce the number of spikes that arrive during a neuron's refractory period and alter the network dynamics (see Table 6.2).

The spikes exchanged between neurons are a compressed version of the presynaptic and postsynaptic neuron IDs, which enables each spike to be uniquely routed to a class simulating an individual synapse. Variable delays are created by copying emitted spikes into one of 250 buffers, which enables them to be delayed for up to 250 time steps. This number of buffers was chosen to minimize the space required to store the delays in the database and it was found to offer enough resolution and length of delay for the time step values that were used in the experiments.[3]

Unlike the majority of neural simulation tools, SpikeStream can operate in a live mode in which the neuron models are calculated using real time instead of simulation time. This live mode is designed to enable SpikeStream to control robots that are interacting with the real world and to process input from live data sources, such as cameras and microphones. Although SpikeStream is primarily an event-driven simulator, it can also be run synchronously to accommodate neuron models that generate spontaneous activity.

---

[3] This value could be changed by editing the SpikeStream code if longer delays or higher delay resolution was required.

6.2.4 SpikeStream Archiver

During a simulation run, the firing patterns of the network can be recorded by SpikeStream Archiver, which stores lists of spikes or firing neurons in XML format along with a simple version of the network model.

6.2.5 Neuron and Synapse Classes

Neuron and synapse classes are implemented as dynamically loaded libraries, which makes it easy to experiment with different neuron and synapse models without recompiling the whole application. Each dynamically loadable class is associated with a parameter table in the database, which makes it easy to change parameters during a simulation run. The current distribution of SpikeStream includes neuron and synapse classes implementing the Spike Response Model and Brader et al.'s (2006) STDP learning rule (see sections 5.3.2 and 5.3.3), which were developed for the work in this thesis.

6.3 Performance

6.3.1 Tests

The performance of SpikeStream was measured using three test networks put forward by Brette et. al. (2006). The main network specified by this paper has 3,200 excitatory neurons and 800 inhibitory neurons that are randomly interconnected with a 2% probability. Larger networks of 10,000 and 20,000 neurons with a similar excitatory/ inhibitory ratio were also put forward by Brette et al., and for the performance tests of SpikeStream the networks were divided into four layers to enable them to be distributed across multiple machines. The neuron and synapse models for these networks could be implemented in four different ways:

- *Benchmark 1*. A network of conductance-based integrate and fire neurons, equivalent to the COBA model described in Vogels and Abbott (2005).
- *Benchmark 2*. A network of integrate and fire neurons connected with current-based synapses, which is equivalent to the CUBA model described in Vogels and Abbott (2005).
- *Benchmark 3*. Conductance-based Hodgkin-Huxley network.
- *Benchmark 4*. Integrate and fire network with voltage-jump synapses.

For these performance tests, Benchmark 4 was chosen because it was the easiest to implement using event-driven simulation strategies.

At the beginning of each simulation run the networks were driven by a random external current until their activity became self sustaining and then their performance was measured over repeated runs of 300 seconds. A certain amount of fine tuning was required to make each network enter a self-sustaining state that was not highly synchronized and the final parameters for each size of test network are given in Table 6.1.[4] The neuron and synapse models that were used for these tests were the same as those described in Section 5.3.2.

The first two networks were tested on one and two Pentium IV 3.2 GHz machines connected using a megabit switch with time step values of 0.1 and 1.0 ms. The third network could only be tested on two machines because its memory requirements exceeded that available on a single machine. All of the tests were run without any learning, monitoring or archiving.

[4] Most of the initial values of these parameters were taken from Brette et. al. (2006).

Parameter	Small network	Medium network	Large network
Neurons	4000	10,000	19,880
Connections	321985	1,999,360	19,760,878
ω_{ij} (excitatory)	0.11	0.11	0.11
ω_{ij} (inhibitory)	-1.0	-0.6	-0.6
Threshold	0.1	0.15	0.25
τ_m	3	3	3
m	0.8	0.8	0.8
n	3	3	3
Connection delay	1	1	1
ρ	3	3	3

Table 6.1. Parameters of test networks

6.3.2 Results

The amount of time taken to simulate one second of biological time for each of the test networks is plotted in Figure 6.2. In this graph the performance difference between 0.1 and 1.0 ms time step resolution is partly due to the fact that ten times more time steps were processed at 0.1 ms resolution, but since SpikeStream is an event-based simulator, the processing of a time step is not a particularly expensive operation. The performance difference between 0.1 and 1.0 ms time step resolution was mainly caused by changes in the networks' dynamics that were brought about by the lower time step resolution, which reduced the average firing frequency of the networks by the amounts given in Table 6.2.

Time step resolution	Small network	Medium network	Large network
0.1 ms	109 Hz	72 Hz	40 Hz
1.0 ms	79 Hz	58 Hz	30 Hz

Table 6.2. Average firing frequencies in simulation time at different time step resolutions

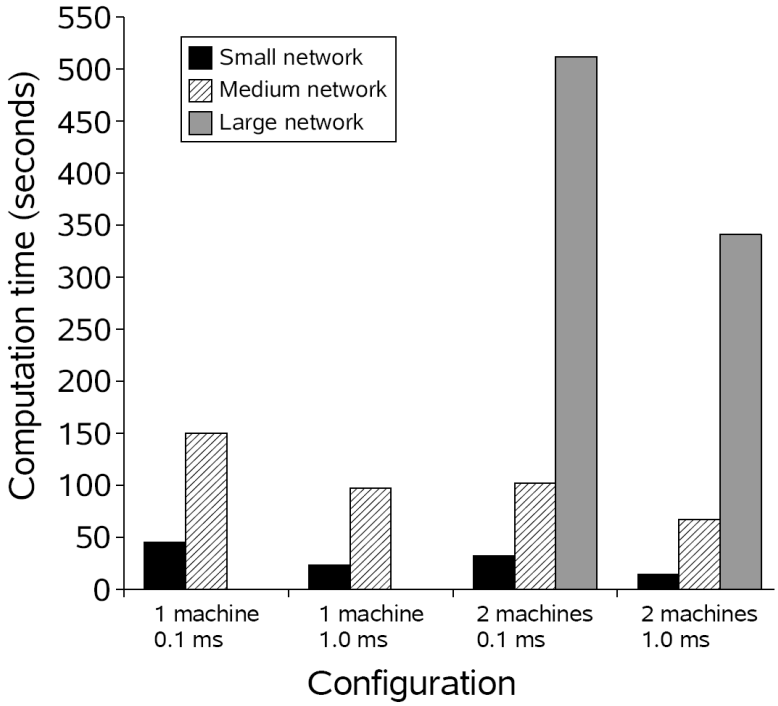

Figure 6.2.. Time taken to compute one second of biological time for one and two machines using time step resolutions of 0.1 and 1 ms

The differences in average firing frequency shown in Table 6.2 suggest that the relationship between real and biological time needs to be combined with other performance measurements for event-based simulators. To address this issue, the number of spikes processed in each second of real time was also measured and plotted in Figure 6.3. This graph shows that SpikeStream can handle between 800,000 and 1.2 million spike events per second on a single machine and between 1.2 million and 1.8 million spike events per second on two machines for the networks that were tested. Figure 6.2 and Figure 6.3 both show that the performance increased when the processing load was distributed over multiple machines, but with network speed as a key limiting factor, multiple cores are likely to work better than multiple networked machines.

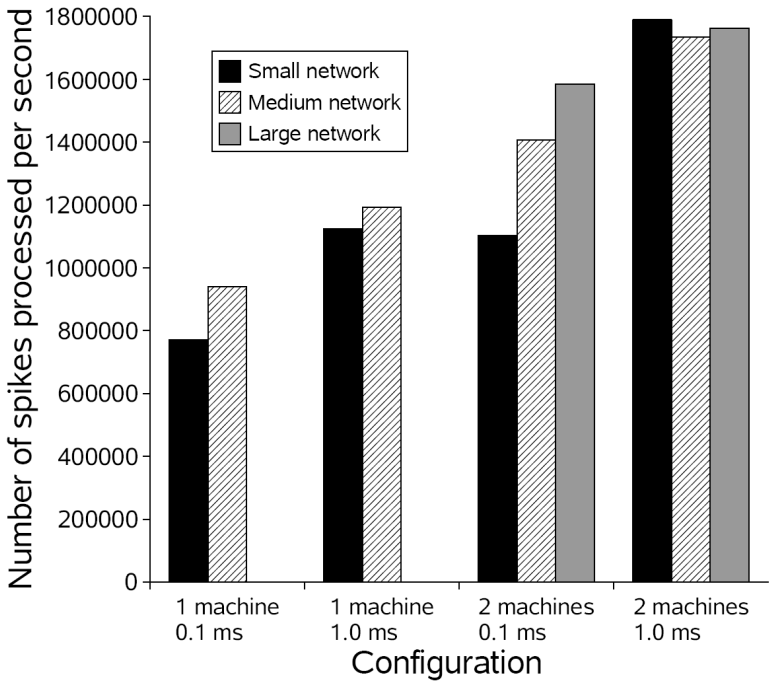

Figure 6.3. Number of spikes processed per second of real time for one and two machines using time step resolutions of 0.1 and 1 ms

Most of the performance measurements in Brette et. al. (2006) are for the neuron and synapse models specified by benchmarks 1-3, which cannot be meaningfully compared with the SpikeStream results for Benchmark 4. The only results that are directly comparable are those for NEST, which are given by Brett et al. (2006, Figure 10B) for two machines. On the 4,000 neuron network NEST takes 1 second to compute 1 second of biological time when the synapse delay is 1 ms and 7.5 seconds to compute 1 second of biological time when the synapse delay is 0.125 ms. Compared with this, SpikeStream takes either 14 or 30 seconds to simulate 1 second of biological time, depending on whether the time step resolution is 1.0 or 0.1 ms, and these SpikeStream results are independent of the amount of delay that is used.

The other point of comparison for the performance of SpikeStream is SpikeNET. The lack of a common benchmark makes comparison difficult, but Delorme and Thorpe (2003) claim that SpikeNET can simulate approximately 400,000 neurons firing at 1Hz real time with 49 connections per neuron and 1 ms time step. This works out as 19.6 million spike events per second, whereas SpikeStream can only handle a maximum of 1.2 million spike events per second

on a single PC for the networks tested (see Figure 6.3). This measurement for SpikeNET was obtained using a substantially slower machine, and so its performance would probably be at least 800,000 neurons firing at 1 Hz in real time today.

6.4 External Devices

SpikeStream can pass spikes over a network to and from external devices, such as cameras and real and virtual robots, in a number of different ways:

- *Synchronized TCP.* Spikes are exchanged with the device at each time step; SpikeStream and the external device only move forward when they have both completed their processing for the time step.
- *Loosely synchronized UDP.* Spikes are sent and received continuously to and from the external device with the rate of the simulation determined by the rate of arrival of the spike messages.
- *Unsynchronized UDP.* Spikes are sent and received continuously from the external device. This option is designed for live work with robots.

The main external device that has been used and tested with SpikeStream is the SIMNOS virtual robot created by Newcombe (see Section 1.2.3 and Figure 6.4). Visual data (available with different types of pre-processing), muscle lengths and joint angles are encoded by SIMNOS into spikes using a variety of methods and passed across the network to SpikeStream using the synchronized TCP method. When the spikes are unpacked by SpikeStream they are used to directly fire neurons or to change their voltage. SIMNOS also receives muscle length data from SpikeStream in the form of spiking neural events, which are used to control the virtual robot. Together SIMNOS and SpikeStream provide an extremely powerful way of exploring sensory and motor processing and integration.

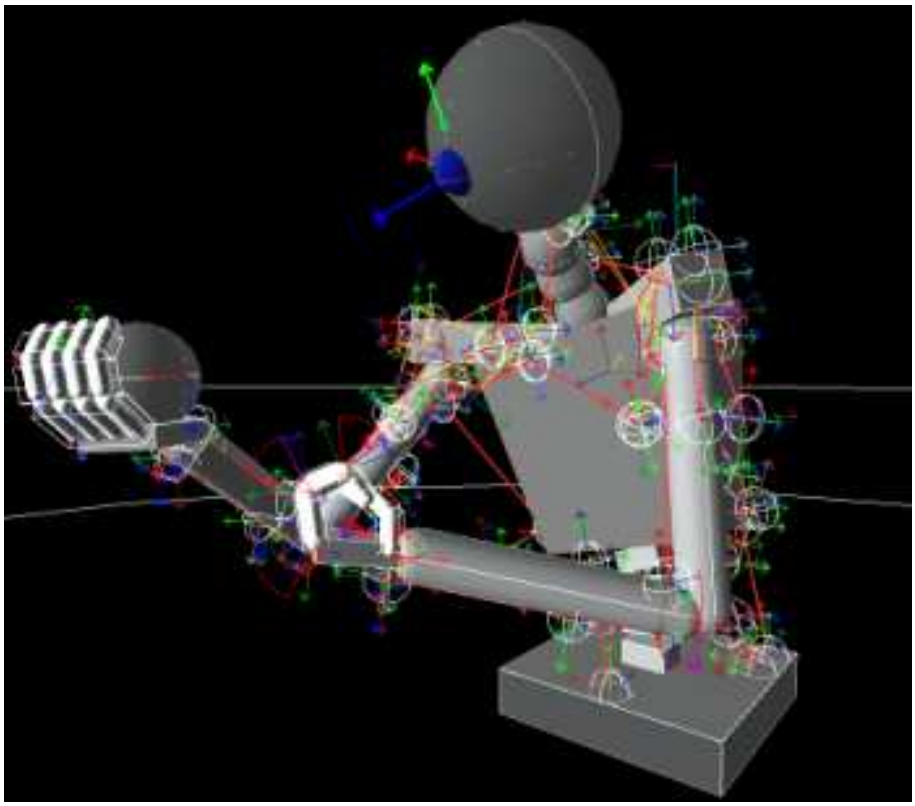

Figure 6.4. SIMNOS virtual robot. The red lines are the virtual muscles consisting of damped springs whose lengths are sent as spikes to SpikeStream. The outlines of spheres with arrows are the joint angles, which are also sent as spikes to SpikeStream.

6.5 Applications

Some potential applications of SpikeStream are as follows:

Biologically inspired robotics

Spiking neural networks developed in SpikeStream can be used to process sensory data from real or virtual robots and generate motor patterns. A good example of this type of work is that carried out by Krichmar et. al. (2005) on the Darwin series of robots (see Section 5.6).

Genetic algorithms

The openness of SpikeStream's architecture makes it easy to write genetic algorithms that edit the database and run simulations using PVM.

Models of consciousness and cognition.

Dehaene et al. (1998, 2003, 2005) and Shanahan (2008) have built models of consciousness and cognition based on the brain that could be implemented in SpikeStream (see Section 3.5.6). The neural network in Chapter 5 is also an example of this type of work.

Neuromorphic engineering

SpikeStream's dynamic class loading architecture makes it easy to test neuron and synapse models prior to their implementation in silicon. Initial work has already been done on enabling SpikeStream to read and write AER events, which would enable it to be integrated into AER chains such as those developed by the CAVIAR project.[5]

Teaching

Once installed SpikeStream is well documented and easy to use, which makes it a good tool for teaching students about biologically structured neural networks and robotics.

6.6 Limitations and Future Work

The flexibility and speed of SpikeStream come at the price of a number of limitations:

- Neurons are treated as points. Each connection can have a unique delay, but there is none of the complexity of a full dendritic tree.
- The connection delay is a function of the time step, not an absolute value, and there is a maximum of 250 delayed time steps. This limitation makes it more complicated to change the time step resolution, but it does not affect the accuracy or the performance of the simulator. The number of buffers could be changed in a future SpikeStream release if higher resolution of the delay or a longer delay was required.

[5] CAVIAR project: http://www.imse.cnm.es/caviar/.

- The full functionality of SpikeStream is only available for layers of rectangular neurons. The main work that would be needed for three-dimensional neuron groups is the extension of SpikeStream Application to enable it to monitor three-dimensional firing patterns. The editing and visualisation have already been partly extended to deal with three-dimensional neuron groups, the simulation and archiving code will work with any shape of neuron group, and the databases will also support any shape of neuron group.

- Any two neurons can only have a single connection between them. This restriction exists because the ID of each connection in the database is formed from a combination of the presynaptic and postsynaptic neuron IDs. This limitation has little impact on the ability of SpikeStream to model point neurons. Multiple connections between two neurons would only make sense if the full dendritic tree was being modelled - when a simulator, such as NEURON or GENESIS, would be more appropriate.

- Although SpikeStream's performance was adequate for the network developed by this thesis, it is likely that it could be substantially improved. Whilst the performance advantage of SpikeNET was achieved at the cost of many important features, it would be worth looking more closely at NEST to see if some its optimization strategies could be incorporated into the SpikeStream simulator. It might also be possible to use the SpikeStream databases to set up simulation runs in NEST, which lacks a graphical user interface.

- SpikeStream currently uses mysqldump to save and load its databases. In the future it would be worth extending the saving and loading functions of SpikeStream to

support the standard XML formats that have been developed for neural networks, such as NeuroML[6] and BrainML.[7]

6.7 Release

SpikeStream is available for free download under the terms of the GPL license. The current (0.1) release has 25,000 source lines of code,[8] full source code documentation, a mailing list for SpikeStream users, and a comprehensive 80 page manual, which has been included in this thesis as Appendix 1. SpikeStream is also available pre-installed on a virtual machine, which works on all operating systems supported by VMware and can be run using the free VMware Player.[9] More information about this release is available at the SpikeStream website: http://spikestream.sf.net. At the time of writing SpikeStream 0.1 has had 140 downloads from the Sourceforge website.

6.8 Conclusions

This chapter has outlined the architecture and performance of SpikeStream, which can simulate medium sized networks of up to 100,000 neurons and is available for free download under the terms of the GPL licence. This simulator is modular, flexible and easy to use and can interface with real and virtual robots over a network. SpikeStream was used to model the neural network described in Chapter 5, and the next chapter analyzes this network for representational mental states and information integration, and makes predictions about its phenomenal states.

---

6 NeuroML website: http://www.neuronml.org.

7 BrainML website: http://www.brainml.org.

8 This was calculated using Wheeler's SLOCCount software. More information about Wheeler's measure can be found here: http://www.dwheeler.com/sloc/.

9 VMware Player: http://www.vmware.com/products/player/.