SpikeStream: A Fast and Flexible Simulator of Spiking Neural Networks

David Gamez

Department of Computer Science, University of Essex, Colchester, C04 3SQ, UK
daogam@essex.ac.uk

Abstract. SpikeStream is a new simulator of biologically structured spiking neural networks 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 simulation networks or apply genetic algorithms to the SpikeStream databases. SpikeStream is available for free download under the terms of the GPL.

Keywords: Spiking neural networks, SpikeStream, event-based simulation, synchronous simulation, biologically inspired robotics.

1 Introduction

SpikeStream is a new fast and flexible neural simulator with the following key features:

- Written in C++ using Qt for the graphical user interface.
- Database storage.
- Parallel distributed operation.
- Sophisticated visualisation, editing and monitoring tools.
- Modular architecture.
- Variable delays.
- Dynamic synapses.
- Dynamic class loading.
- Live operation.
- Spike exchange with external devices over a network.

The first part of this paper outlines the different components of the SpikeStream architecture and sets out the features of the GUI in more detail. Next, the performance of SpikeStream is documented along with its communication with external devices. The last part of this paper suggests some applications for SpikeStream, compares it with other simulators and describes its limitations. Documentation and source code for SpikeStream are available for free download at http://spikestream.sourceforge.net.

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.

2.1 Databases

SpikeStream is based around a number of databases that hold information about the network model, patterns and devices. This makes it very 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:

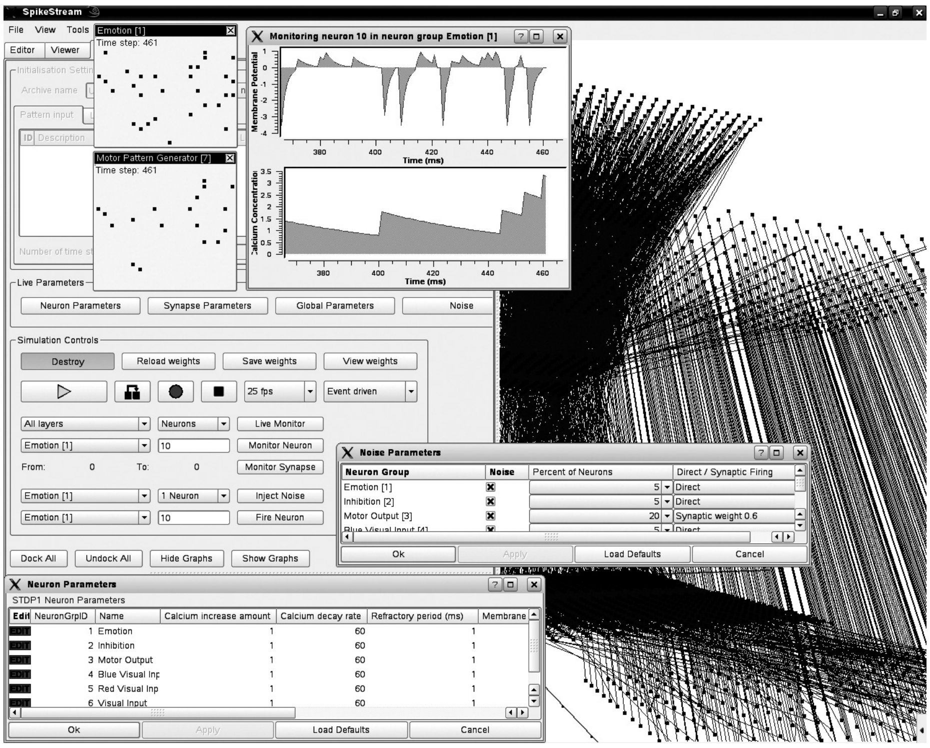
- *Neural Network*. Each neuron has a unique ID and connections between neurons are recorded as a combination of the presynaptic and postsynaptic neuron IDs. The available neuron and synapse types along with their parameters and class libraries are also held in this database.
- *Patterns*. Holds patterns that can be applied to the network for training or testing.
- *Neural Archive*. Stores archived neuron firing patterns. Each archive contains an XML representation 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 could also be edited by third party applications - to create custom connection patterns or neuron arrangements, for example - without affecting SpikeStream's ability to visualise and simulate the network.

2.2 SpikeStream Application

An intuitive graphical user interface has been written for SpikeStream (see Fig. 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 the membrane potential, graphically displayed.
- *Archiving*. Archived simulation runs can be loaded and played back.



Emotion [1] | 10
From: 0 To: 0
Emotion [1] | 1 Neuron
Emotion [1] | 10

Noise Parameters

Neuron Group	Noise	Percent of Neurons	Direct / Synaptic Firing
Emotion [1]	☒	5	Direct
Inhibition [2]	☒	5	Direct
Motor Output [3]	☒	20	Synaptic weight 0.6
Blue Visual Input [4]	☒	5	Direct

Neuron Parameters

STDP1 Neuron Parameters

Edit	NeuronGrpID	Name	Calcium increase amount	Calcium decay rate	Refractory period (ms)	Membrane
EDIT	1	Emotion	1	60	1	
EDIT	2	Inhibition	1	60	1	
EDIT	3	Motor Output	1	60	1	
EDIT	4	Blue Visual Inp	1	60	1	
EDIT	5	Red Visual Inp	1	60	1	
EDIT	6	Visual Input	1	60	1	

Fig. 1. SpikeStream graphical user interface

2.3 SpikeStream Simulation

The SpikeStream simulator consists of a number of processes that are launched and coordinated using PVM, with each process simulating an individual neuron group using a combination of event-based and synchronous simulation based on SpikeNET [7]. 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 other 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 arriving during a neuron's refractory period and alter the network dynamics (see Table 2).

One difference between SpikeStream and SpikeNET is that messages are sent rather than requested at each time step, which cuts down the message passing by 50% in a fully recurrent architecture. The spikes themselves are a compressed version of the presynaptic and postsynaptic neuron IDs, which enables each spike to be uniquely routed to an individual synapse class. Variable delays are created by copying emitted spikes into one of 250 buffers and at each time step only the spikes in the current buffer are sent. Unlike the majority of neural simulation tools, SpikeStream is designed to operate in live as well as simulation time so that it can control real and virtual robots in close to real time and process input from live data sources (see section 4). Although SpikeStream is primarily an event-driven simulator, it can be run synchronously to accommodate neuron models that generate spontaneous activity.

2.4 SpikeStream Archiver

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

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 Brader et al.'s STDP learning rule [3].

3 Performance

3.1 Tests

The performance of SpikeStream was measured using three networks based on the benchmarks put forward by Brette et. al. [4]. The size and connectivity of these networks is given in Table 1 and they were divided into four layers containing 80% excitatory and 20% inhibitory neurons. 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.

The neuron model for these tests was based on the Spike Response Model [11], with the voltage V_i at time t for a neuron i that last fired at $\hat{t}$ being given by:

$$V_i(t) = \sum_j \sum_f w_{ij} e^{-\frac{(t-t_j^{(f)})}{\tau_m}} - e^{n-(t-\hat{t}_i)^m} \mathrm{H}'(t-\hat{t}_i), \qquad (1)$$

where ω_{ij} is the synaptic weight between i and j, f is the last firing time of neuron j and H' is given by:

$$\mathrm{H}'(t-\hat{t}_i)=\begin{cases}\infty, & \mathit{if}\ 0\le(t-\hat{t}_i)<\rho \\ 1, & \mathit{otherwise}\end{cases}. \tag{2}$$

For all networks the membrane time constant, τ_m, was set to 3, the refractory parameters m and n were set to 0.8 and 3, the connection delay was set to 1 ms and the absolute refractory period, ρ, was set to 3. V_i had a resting vale of 0 and when it exceeded the threshold the neuron was fired and the contributions from previous spikes were reset. The remaining neuron and synapse parameters are given in Table 1.

Table 1. Size of test networks and their parameters

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

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.

3.2 Results

Fig. 2 plots the amount of time taken to simulate one second of biological time for each of the test networks. 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 main cause of this speed up were changes in the networks' dynamics brought about by the lower time step resolution, which reduced the average firing frequency of the networks by the amounts given in Table 2.

Table 2. Average firing frequencies in biological time at different time step resolutions

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

The differences in average firing frequency shown in Table 2 suggest that the relationship between real and biological time needs to be combined with other performance measurements for event-based simulators.

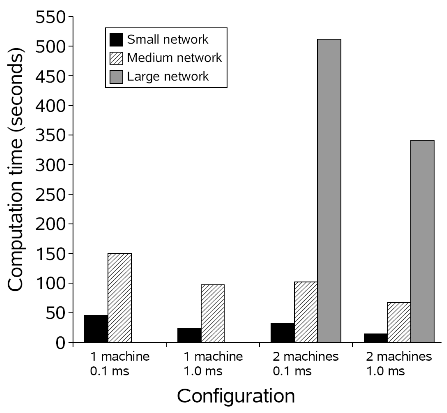


Fig. 2. Time taken to compute one second of biological time for one and two machines using time step resolutions of 0.1 and 1 ms

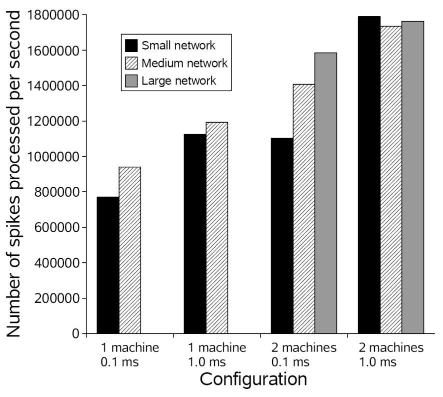


Fig. 3. Number of spikes processed per second of real time for one and two machines using time step resolutions of 0.1 and 1 ms

To address this issue, the number of spikes processed in each second of real time was also measured and plotted in Fig. 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 - an observation that should stay reasonably constant with different networks and activity levels. Fig. 2 and Fig. 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.[1]

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:

1. *Synchronized TCP*. Spikes are exchanged with the device at each time step and both only move forward when they have received their spikes.

[1] Brette et. al. [4] give the performance of other neural simulators on their benchmark networks, which can be compared with the SpikeStream results in Fig. 2 and Fig. 3. It is worth noting that the run time of a SpikeStream simulation is not affected by the speed of its databases.

2. *Loosely synchronized UDP*. Spikes are sent and received continuously to and from the external device with the rate of the simulation controlled by the rate of arrival of the spike messages.
3. *Unsynchronized UDP*. Spikes are sent and received continuously from the external device. This option is designed for live work with robots.[2]

The main external device that has been used and tested with SpikeStream is the SIMNOS virtual robot created by Newcombe [9]. SIMNOS is a humanoid robot with an internal structure inspired by the human musculoskeletal system and it is simulated in soft real time using Ageia PhysX [1]. Within this physics simulation environment SIMNOS' skeletal body components are modelled as jointed rigid bodies and its muscles are modelled using spring-damper systems.

Visual data (available with a wide variety of pre-processing methods) and muscle lengths and joint angles are encoded by SIMNOS into spikes using a selection of methods developed by Newcombe [9] and passed across the network to SpikeStream using synchronized TCP. 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. More information about SIMNOS is available at www.cronosproject.net.

5 Applications

5.1 General Application Areas

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. on the Darwin series of robots [12].
- *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 and Changeux [6] and Shanahan [17] have built models of consciousness and cognition based on the brain that could be implemented in SpikeStream.
- *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 also 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.

[2] This method of spike streaming has not been fully implemented in SpikeStream 0.1.

5.2 Recent Experiments

SpikeStream is currently being used to develop a neural network that uses analogues of imagination, emotion and sensory motor integration to control the eye movements of the SIMNOS virtual robot. When this network is online it spontaneously generates eye movements to different parts of its visual field and learns the association between an eye movement and a visual stimulus using Brader et. al.'s [3] spike time dependent learning rule. When it has learnt these associations, its emotion system is hardwired so that blue objects inhibit motor output and visual input. This switches the network into 'imagination' mode, in which it explores sensory motor patterns until it finds one that positively stimulates its emotional system. This removes the inhibition, and SIMNOS' eye is then moved to look at the red stimulus. This work is inspired by other simulations of the neural correlates of consciousness, such as Dehaene and Changeux [6] and Shanahan [17].[3]

6 Other Spiking Simulators

SpikeStream simulates biologically structured spiking networks of up to 100,000 neurons with each neuron treated as a point without the complex dendritic structure of a biological neuron. This sets it apart from simulators, such as NEURON [16], GENESIS [10] and NCS [15], that work with complex dendritic trees. SpikeStream also differs from rate-based simulators, such as Topographica [19], and synchronous simulators, such as NEST [8], which update all the neurons at each time step and often run for a fixed period of simulation time. The spiking biological aspect of SpikeStream also differentiates it from the many simulators written for conventional neural networks, which are often trained by back-propagation and have input, output and hidden layers.

The closest simulator to SpikeStream is Delorme and Thorpe’s SpikeNET [7]. This simulator runs substantially faster than SpikeStream,[4] but its extra performance comes at the cost of a number of limitations These include a lack of synaptic delay, the fact that each neuron can only fire once, a lack of recurrent connections, no graphical interface, a simple and inflexible neural model and synaptic weights that are shared between neurons in an array. All of these limitations were the motivation for creating a new simulator based on the SpikeNET architecture that was more flexible and easier to use.

SpikeStream also has similarities with SpikeSNNS [13], which is a spiking version of the Stuttgart Neural Network Simulator. This was also influenced by SpikeNET, but has a much slower performance and the SNNS interface is somewhat outdated and difficult to use.

[3] Videos of this network in operation are available at www.davidgamez.eu/videos/index.html.

[4] SpikeStream can simulate 100,000 neurons firing at 1Hz in real time with 10 connections per neuron and SpikeNET can simulate approximately 400,000 neurons firing at 1Hz in real time with 50 connections per neuron. These measurements for SpikeNET were obtained using a substantially slower machine and so the performance of SpikeNET would probably be at least 800,000 neurons per second firing at 1 Hz today.

Other spiking simulators include the Amygdala library [2] and Mvaspike [14], which lack graphical interfaces and are not designed for robotic use, and the Spiking Neural Simulator developed by Smith [18], which can simulate a spiking network for a fixed period of time, but lacks many of the features included with SpikeStream. With many of these simulators it would be possible to use parts of SpikeStream, for example its graphical interface and database, to set up and run simulations on a different simulation engine.

7 Limitations

The flexibility and speed of SpikeStream come at the price of a number of limitations, some of which will be resolved in future releases of the software:

1. Neurons are treated as points. Each connection can have a unique delay, but there is none of the complexity of a full dendritic tree.
2. The connection delay is a function of the time step, not an absolute value, and the maximum number of time steps is limited to 250. This design was motivated by a desire to keep the Connections table in the database as small as possible.
3. SpikeStream currently only works with rectangular layers of neurons. Although the editing and visualisation have been partly extended to deal with three-dimensional neuron groups, more work is required to enable three dimensional live monitoring.
4. 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.

8 Conclusion

This paper has outlined the architecture and performance of SpikeStream, which can simulate medium sized networks of up to 100,000 neurons. This simulator is modular, flexible and easy to use and can interface with real and virtual robots over a network. SpikeStream is available for free download under the terms of the GPL licence.

Acknowledgements

Many thanks to Owen Holland for feedback, support and advice about SpikeStream and to the blind reviewers for their helpful comments and suggestions. The interface between SIMNOS and SpikeStream was developed in collaboration with Richard Newcombe, who designed the spike conversion methods. Thanks also to Renzo De Nardi and Hugo Gravato Marques in the Machine Consciousness lab at Essex and to everyone at the 2006 Telluride workshop. This work was funded by the Engineering and Physical Science Research Council Adventure Fund (GR/S47946/01).

References

1. Ageia PhysX, http://www.ageia.com/developers/api.html
2. Amygdala simulator, http://amygdala.sourceforge.net/
3. Brader, J.M., Senn, W., Fusi, S.: Learning real world stimuli in a neural network with spike-driven synaptic dynamics. Neural Computation (submitted, 2006)
4. Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J.M., Diesmann, M., Morrison, A., Goodman, P.H., Harris Jr., F.C., Zirpe, M., Natschlaeger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A.P., El Boustani, S., Destexhe, A.: Simulation of networks of spiking neurons: A review of tools and strategies. J. Comp. Neurosci (in press)
5. CAVIAR project, http://www.imse.cnm.es/caviar/
6. Dehaene, S., Changeux, J.P.: Ongoing Spontaneous Activity Controls Access to Consciousness: A Neuronal Model for Inattentional Blindness. Public Library of Science Biology 3(5), e141 (2005)
7. Delorme, A., Thorpe, S.J.: SpikeNET: An Event-driven Simulation Package for Modeling Large Networks of Spiking Neurons. Network: Computational in Neural Systems 14, 613–627 (2003)
8. Diesmann, M., Gewaltig, M.-O.: NEST: An Environment for Neural Systems Simulations. In: Macho, V. (ed.) Forschung und wissenschaftliches Rechnen, Heinz-Billing-Preis, GWDG-Bericht (2001)
9. Gamez, D., Newcombe, R., Holland, O., Knight, R.: Two Simulation Tools for Biologically Inspired Virtual Robotics. In: Proceedings of the IEEE 5th Chapter Conference on Advances in Cybernetic Systems, Sheffield, pp. 85–90. IEEE Computer Society Press, Los Alamitos (2006)
10. GENESIS simulator, http://www.genesis-sim.org/GENESIS/
11. Gerstner, W., Kistler, W.: Spiking Neuron Models. Cambridge University Press, Cambridge (2002)
12. Krichmar, J.L., Seth, A.K., Nitz, D.A., Fleischer, J.G., Edelman, G.M.: Spatial navigation and causal analysis in a brain-based device modeling cortical-hippocampal interactions. Neuroinformatics 3, 197–221 (2005)
13. Marian, I.: A biologically inspired computational model of motor control development. MSc Thesis, Department of Computer Science, University College Dublin, Ireland (2003)
14. Mvaspike simulator, http://www-sop.inria.fr/odyssee/softwares/mvaspike/
15. NCS simulator, http://brain.cse.unr.edu/ncsDocs/
16. NEURON simulator, http://www.neuron.yale.edu/neuron/
17. Shanahan, M.P.: A Cognitive Architecture that Combines Internal Simulation with a Global Workspace. Consciousness and Cognition 15, 433–449 (2006)
18. Spiking Neural Simulator, http://www.cs.stir.ac.uk/~lss/spikes/snn/index.html
19. Topographica Neural Simulator, http://topographica.org/Home/index.html