SpikeStream

Version 0.2

Manual

David Gamez

CONTENTS

1. INTRODUCTION

SpikeStream is a modular simulator and analyzer of neural networks. Most of the functionality of SpikeStream is carried out by different plugins that can be used to create networks, neuron groups and connection patterns, to simulate networks and to analyze networks using different algorithms.

SpikeStream uses MySQL databases to store data about networks, firing patterns and analyses. *It will not work without these databases*. Instructions for installing the MySQL database and configuring it on Windows, Mac OS X and Linux are given in sections 2.2, 3.2 and 4.2.

In the current release the simulation functionality of SpikeStream is provided by NeMo (http://nemosim.sourceforge.net/), which can run on the computer's CPU or much faster using an NVIDIA graphics card. The hardware acceleration depends on a correct installation of CUDA hardware and drivers. Instructions for this on Windows, Mac OS X and Linux are given in sections 2.3,3.3 and 4.3.

This manual covers most of the basic features of SpikeStream. While I have endeavoured to keep this manual up to date, I apologize for any mistakes and inaccuracies that have crept in! If you have any questions, feel free to use the SpikeStream mailing list (spikestream-user@lists.sourceforge.net) or contact me, David Gamez, at: http://www.davidgamez.eu/pages/contact/. The website for SpikeStream is http://spikestream.sf.net.

2. WINDOWS INSTALLATION

2.1 Introduction

This section takes you through the steps that are needed to get SpikeStream running on Windows. The installation has the following stages:

1. Install and configure a MySQL server to hold the SpikeStream databases.

2. Install NVIDIA CUDA drivers (optional).

3. Install NeMo.

4. Install the SpikeStream application.

2.2 Installation of MySQL Server

The SpikeStream data is stored on three MySQL databases and SpikeStream will not run unless it can communicate with these. This section describes how to get a MySQL server running on your local machine. You can skip this section if you already have a MySQL server on a local or remote host that you want to use with SpikeStream.

The first step is to download the MySQL community server from: http://dev.mysql.com/downloads/mysql/. The Windows (x86, 32-bit), MSI Installer works fine, although other versions should be ok as well. Save the installer to disk and double click on it to run it. Go through the installation steps using the default options and on the last screen you will be offered the chance to configure the server as shown in Figure 1.

Figure 1. Last stage of MySQL installation dialog. Select "Configure the MySQL Server now" and click on "Finish" to launch the MySQL Server Instance Config Wizard. Registration of the MySQL server is unnecessary.

If you accidentally click past this step, you can launch the "MySQL Server Instance Config Wizard" by clicking **Start->Programs->MySQL->MySQL Server 5.1->MySQL Server Instance Config Wizard** after the MySQL server has been installed.

The MySQL Server Instance Configuration Wizard should launch as shown in Figure 2.

Figure 2. MySQL Server Instance Configuration Wizard

Click on "Next" and you will be presented with the dialog shown in Figure 3.

Figure 3. Configuration type. Select "Standard Configuration" unless you are using MySQL for other applications and want to configure it differently. How the server is configured does not currently make much difference to SpikeStream, but the Standard Configuration is easier to set up.

Select "Standard Configuration" and click "Next". You will be presented with the dialog shown in Figure 4.

6

Figure 4. Install MySQL as a Windows service. SpikeStream does not need the MySQL bin directory to be included in the Windows PATH, but it will not create any problems if you select this option.

Select "Install as Windows Service" and click "Next". This will show you the dialog in Figure 5.

Figure 5. MySQL security settings. This is the most critical part of the installation because you need to remember the password that you enter.

Select "Modify Security Settings" and enter a password for the root account on the MySQL server. *Make sure that you remember the password because you will have to provide this information to the SpikeStream Database Configuration Tool* (see Section 5)*.* Note that SpikeStream does not need to be configured with the root account and

it is possible to run each database using a different host, user and password. When you have written down or memorized the root password, click "Next" and you will be shown the dialog in Figure 6.
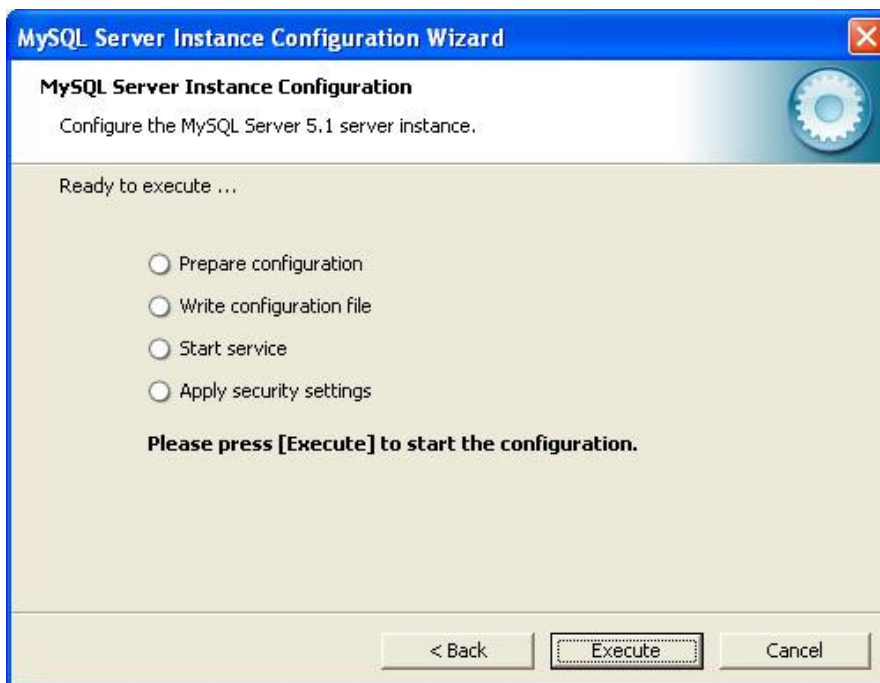
Figure 6. Execution dialog. Click on "Execute" to save your settings.

Click on "Execute" to save your settings. If the operation is successful you should see the dialog shown in Figure 7.
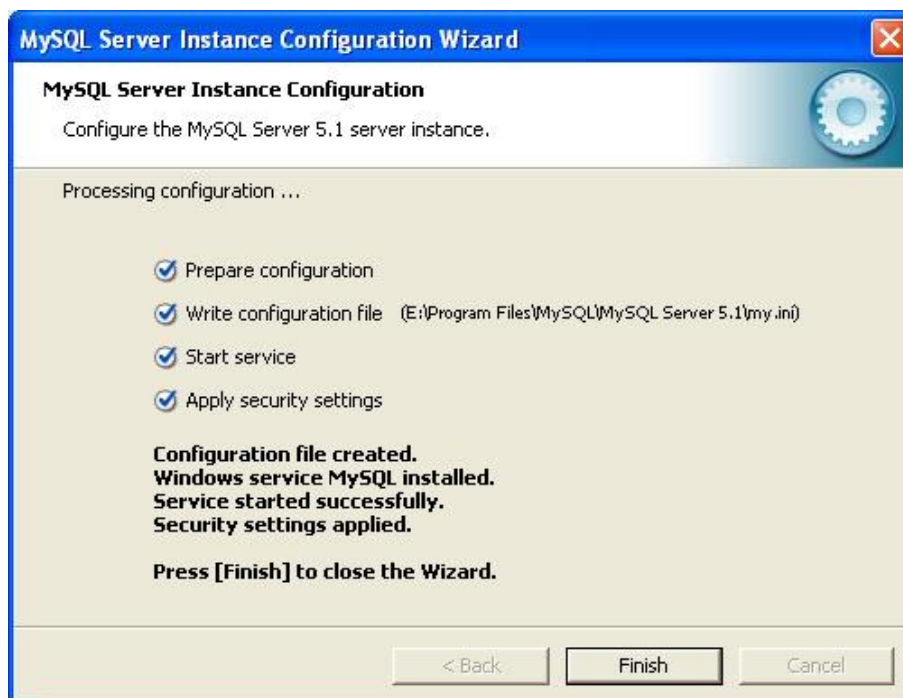
Figure 7. MySQL successful configuration.

You should now have a MySQL server running on your local machine whose root access has been configured using the password that you provided. The next step is to add the SpikeStream databases to this MySQL server and configure SpikeStream with the correct user name and password so that it can access this server. These steps can be carried out manually (see Section 9) or using the SpikeStream Database Configuration Tool (see Section 5). Before you can use this tool, you need to download and install SpikeStream.

2.3 NVIDIA CUDA Installation

This step is only required if you want to use hardware to accelerate the simulation of spiking neurons. If you do not have suitable hardware and drivers, then the simulation will run on the CPU and no further installation steps are required.

NOTE: the hardware acceleration will only work on graphics card that support CUDA 1.3 and higher. Hardware that only runs earlier versions of CUDA cannot be used for spiking neural acceleration with NeMo, and SpikeStream will use the CPU instead. The NVIDIA CUDA C Programming Guide has a list of CUDA devices and their compute capability.

NOTE: The current release of SpikeStream works with CUDA version 4.1. Higher or lower versions are unlikely to work and SpikeStream will only simulate in CPU mode. NeMo can be rebuilt with different versions of CUDA if you need them.

Full instructions for installing CUDA can be found on the NVIDA website, www.nvidia.com. The first stage is to install the NVIDIA hardware and the latest drivers. Next, download and install the CUDA Toolkit from www.**nvidia**.com/object/**cuda**_get.html. If you want to test the CUDA installation using bandwidthTest.exe, you will also have to install the GPU Computing SDK code samples.

To check that your hardware is working correctly run the bandwidthTest.exe program, which can be found at C:\Documents and Settings\All Users\Application Data\NVIDIA Corporation\NVIDIA GPU Computing SDK\C\bin\win32\Release in XP, or in a slightly different location in Vista and Windows 7.

2.4 SpikeStream Installation

Download the Windows installer and save it on your computer. Run the installer and click through the installation steps to copy the files to your computer. SpikeStream will not work until the databases have been configured using the SpikeStream Database Configuration Tool, which is described in Section 5. You can launch this tool by clicking on Start->Programs->SpikeStream->Database Configuration.

Once the databases have been configured, it should be possible to run SpikeStream by double clicking on one of the installed shortcuts or by double clicking on spikestream.exe in the bin directory of the installation.

Note: The current version of SpikeStream is installed for all users of the computer. Any changes that one user makes to the configuration of SpikeStream will affect other users of the computer. This can be avoided by creating a local install of SpikeStream for each user.

3. MAC OS X INSTALLATION

3.1 Introduction

The current release of SpikeStream and NeMo for Mac OS X only supports CPU simulation of Izhikevich neural networks. Users will find it relatively easy to build their own CUDA compatible version if this is required.

This section takes you through the steps that are needed to get SpikeStream running on Mac OS X. SpikeStream has only been tested on version 10.6, Snow Leopard, although it may work on earlier versions. The installation has the following stages:

1. Install and configure a MySQL server to hold the SpikeStream databases.
2. Install the SpikeStream application.

Brief instructions are also given about building SpikeStream from source.

3.2 Installation of MySQL Server

NOTE: There appears to be a MySQL bug with the handling of constraints on InnoDB tables, which prevents SpikeStream from working – see: http://stackoverflow.com/questions/5566991/mysql-5-5-foreign-key-constraint-fails-when-foreign-key-exists. Use version 5.1.56 to avoid this problem.

The SpikeStream data is stored on three MySQL databases and SpikeStream will not run unless it can communicate with these. This section describes how to get a MySQL server running on your local machine. You can skip this section if you already have a MySQL server on a local or remote host that you want to use with SpikeStream.

Download and install version 5.1.56 of the MySQL server from www.mysql.com.

Start the MySQL server – instructions can be found here: http://dev.mysql.com/doc/refman/5.0/en/macosx-installation.html.

You should then be able to run the MySQL program from the same directory to connect to your MySQL server.

You will need to configure a root account for your MySQL server. Follow the instructions in Section 4.2.

NOTE: You may need to restart MySQL each time that you reboot OS X.

3.3 SpikeStream Installation

SpikeStream is distributed for Mac OS X as a zipped disc image. Download and unzip the archive in the location where you want to save it. Double click on the disc image to mount it.

SpikeStream will not work until the databases have been configured using the SpikeStream Database Configuration Tool, which is described in Section 5. You can launch this tool by clicking on the dbconfigtool file in the bin directory of the installation.

Once the databases have been configured, it should be possible to run SpikeStream by double clicking on the SpikeStream file in the bin directory of the installation.

3.6 Building SpikeStream from Source

3.6.1 Dependencies

The Mac OS X dependencies are similar to the Linux dependencies described in Section 4.5.1.

You need to add CONFIG += x86 to the qwtconfig.pri file when building Qwt and use `qmake -spec macx-g++` to generate the makefiles.

The Connection Matrix Importer plugin depends on Boost. The latest version can be obtained using MacPorts (http://www.macports.org/).

3.6.2 Build Qt Plugin for MySQL

To build the qmysql plugin for the database download the source from http://qt.nokia.com. Then go to src/plugins/sqlplugins/mysql and do the following:

To the mysql.pro file add:

CONFIG += x86

INCLUDEPATH +=/usr/local/mysql/include

LIBS += -L/usr/local/mysql/lib -lmysqlclient_r

Use `qmake -spec macx-g++` to create the make file.

Type `make` and then `sudo make install` to install the plugin.

Go to the plugins directory, typically at /Developer/Application/Qt/plugins/sqlplugins

Use otool to find out which version of the MySQL library the plugin links against:

`otool -L libqsqlmysql.dylib`

Then copy this library from /usr/local/mysql/lib to /usr/lib.

3.6.3 Build SpikeStream

Build and install dependencies – typically qwt, MySQL and NeMo.

Open up spikestream.pro and comment/uncomment the components that you want to build. If you are having problems building SpikeStream, comment out everything except the core SpikeStream components and the installation tool.

Open up spikestream.pri and check that the paths in the macx section are correct. Select whether you want a debug or release build.

Change to the root directory of spikestream – typically spikstream/trunk

Type

```
qmake -spec macx-g++

make
```

You should then be able to run the database configuration tool using `open bin/dbconfigtool.app/`

And run spikestream using `open bin/spikestream.app/`

3.6.4 Deploy SpikeStream

To create a releasable version of SpikeStream:

1. Build SpikeStream.

2. Run macdeployqt tool on spikestream and the dbconfigtool: `macdeployqt dbconfigtool.app`; `macdeployqt spikestream.app`. This will deploy local copies of the Qt libraries within the applications.

3. Copy NeMo libraries into the bin/spikestream.app/Contents/Frameworks folder.

4. Change to installation/macscripts/ folder and run installation/macscripts/deploynemo.sh script. This will update NeMo libraries so that they will run locally.

5. Run installation/macscripts/pluginslib.sh script. This will alter the linking paths in the plugins so that they use the local copies of Qt etc.

4. LINUX INSTALLATION

4.1 Introduction

The installation of SpikeStream on Linux takes place in the following steps:

1. Install and configure MySQL server.

2. Install NVDIA CUDA hardware and drivers (optional).

3. Build and install SpikeStream.

4.2 Installation and Configuration of MySQL Server

MySQL typically forms part of a Linux distribution and it is also available at www.mysql.org. You need to install the development parts of MySQL as well as the server. On Ubuntu MySQL can be easily installed and configured with a root password using: `sudo apt-get install mysql-server`.

When you have installed MySQL, test to see if it is running using: `ps -el | grep mysql`. This should return a line containing "mysqld" as one of the running processes. If this is not listed, use chkconfig to enable the service. As superuser type: `chkconfig --list mysql`, which should tell you if mysql is enabled or not. If it is not enabled for your current run level, type: `chkconfig mysql on` and make sure that it is enabled.

Even when mysql is enabled, the daemon may not have started. To start the daemon go to /etc/init.d/ and log in as root. Then run the mysql command by typing: `./mysql start`, which should start up the daemon. Check that it has started, then you are ready to set up the accounts.

You need to allow external access to MySQL if you are running SpikeStream on a different machine to the database server, and your system's firewall may need to be changed to facilitate this. In SUSE this can be done by adding MySQL to the firewall configuration using YAST.

When MySQL is installed it typically comes with one unsecure root account. It is recommended that you secure the root account and create a separate account for SpikeStream. The following steps should work:

- Log in to MySQL as root using `mysql -u root`

- Display the current accounts: `SELECT user, host, password FROM mysql.user;`

- Set a password for root: `SET password=PASSWORD("secretpassword")`

- Get rid of unnecessary users: `DELETE FROM mysql.user WHERE user != "root";`

- Get rid of logins from outside machine: `DELETE FROM mysql.user WHERE host != "localhost";`

- Create accounts with the user 'SpikeStream' and the password 'myPassword' that can access the database on localhost or a subnetwork:
    - `GRANT ALL ON *.* TO SpikeStream@localhost IDENTIFIED BY "myPassword";`
    - `GRANT ALL ON *.* TO SpikeStream@'192.168.1.0/255.255.255.0' IDENTIFIED BY "myPassword";`

- If these accounts have been created successfully it should be possible to log into the database locally or from another machine on the same network using:
    - `mysql -uSpikeStream -pmyPassword` (local login with password "myPassword")
    - `mysql -uSpikeStream -pmyPassword -h192.168.1.9` (remote login with mysql hosted on 192.168.1.9 and password "myPassword")

You can create a different account for each database and the databases can be hosted on different machines. You need to remember the details for each database and enter them into the SpikeStream Database Configuration Tool (see Section 5). These details can also be manually into the spikestream.config file (see Section 10).

4.3 CUDA Installation

If you want to use the hardware-accelerated simulation capabilities provided by NeMo within SpikeStream you need to get CUDA running on your system. Slower simulations can be carried out by running NeMo on the CPU if you do not have appropriate hardware or have difficulties installing CUDA.

NOTE: NeMo will only run on NVIDIA hardware that can support CUDA compute capability greater than or equal to 1.3. Hardware that only runs earlier versions of CUDA cannot be used for spiking neural acceleration with Nemo, and SpikeStream will use the CPU instead. The NVIDIA CUDA C Programming Guide has a list of CUDA devices and their compute capability.

Full instructions for installing CUDA can be found on the NVIDA website, www.nvidia.com. The first stage is to install the NVIDIA hardware and the latest drivers. Next, download and install the CUDA Toolkit from www.**nvidia**.com/object/**cuda**_get.html. If you want to test the CUDA installation, you can install and run the GPU Computing SDK code samples.

CUDA installation on Linux can be difficult because the latest version of the toolkit is only available for particular versions of some Linux distributions. Furthermore, each version of the toolkit depends on a minimum version of the NVIDIA driver being installed, and the installation of later versions of the NVIDIA driver can lead to system instabilities. You may also have problems compiling earlier versions of the code samples using the latest version of GCC.

4.4 NeMo Installation

Instructions for installing NeMo are available at the NeMo website: http://nemosim.sourceforge.net. Follow the instructions in the NeMo manual. After a successful install the NeMo libraries should be on the system path.

4.5 SpikeStream Installation

4.5.1 Dependencies

The first step in building SpikeStream on Linux is the installation of a number of third party libraries.

Qt 4.7

SpikeStream might build against earlier or later 4.x versions of Qt, but it has only been fully tested on Qt 4.7. Type qmake --version to find out which version of Qt you are using. If it is less than 4.7, it is recommended that you download and install the Qt libraries from: http://qt.nokia.com/downloads/. On some versions of Linux you may have to install additional libraries to support OpenGL. Write down the details about these extra libraries when they are given in the Qt installer.

You will also need to install the mysql plugin for Qt. To do this change to the directory in which Qt is installed. The source code for the driver is typically located at src/plugins/sqldrivers/mysql. Change to this directory, Type the full path to the version of Qt that you are using, and then `make` followed by `make install`. If the build fails with a message about not being able to find mysql.h. you will need to install the MySQL development libraries and add, for example, INCLUDEPATH += /usr/include/mysql to the mysql.pro file.

Qwt - graph plotting library

NOTE: use version 5.2 of Qwt, not 6.0.

This is available from: http://qwt.sourceforge.net/. Use the version of Qt that you have installed to build Qwt. You need to install the library file, libqwt.so.5, in a location where the system will automatically load it or add the location of libqwt.so.5 to the LD_LIBRARY_PATH variable.

NeMo neural simulator library

Only required if you want to run simulations using SpikeStream. Download from http://nemosim.sourceforge.net. This library has its own dependencies and installation instructions are given in the NeMo manual.

GMP big number library

Available in most Linux distributions.

Boost

Available in most Linux distributions. Only the header files are required.

4.5.2 Get Source Code

Download spikestream-0.2.tar.gz from http://spikestream.sf.net/download. Move the file to the installation location and type tar -xzvf spikestream-0.2.tar.gz to extract the distribution.

You can also check out the latest version of the SpikeStream source code using SVN from http://sourceforge.net/projects/spikestream/develop. The trunk contains the most advanced code, but this may be unstable and not build properly. The release-0.2 branch contains the stable 0.2 release with bug fixes relevant to that release.

4.5.3 Set Paths

Change to the root directory of SpikeStream. This will be spikestream/trunk or spikestream-0.2 depending on whether you checked out the code using SVN or unpacked the distribution.

The build of SpikeStream is controlled by two files in the root directory of the installation:

- *spikestream.pro* controls which components of SpikeStream are built. Each path in this file points to a directory containing another .pro file with the same name as the directory. These .pro files are used to build each component. So, for example, the analysis/statebasedphi directory contains a file called statebasedphi.pro, which contains all of the information used to build the statebasedphi plugin. The building of components can be disabled by commenting them out using a hash, #.

- *spikestream.pri* contains the location of the external libraries. Before building you need to open up this file and make sure that the paths within the unix parts of the file are correct for your system, which should match the location of the dependencies discussed in the previous section. In many cases the external Linux libraries will be installed in /usr/local/lib or /usr/lib. Whilst /usr/lib is typically always on the load library path (LD_LIBRARY_PATH), some distributions of Linux do not automatically include /usr/local/lib on the load library path. If this is the case, you can add it by adding `export LD_LIBRARY_PATH=/usr/local/lib:LD_LIBRARY_PATH` to your ~/.bashrc file.

To set the build paths for NeMo, open up simulators/nemo/nemo.pro and set the location of the NeMo include files and libraries, if they are not in the standard directories. The default install of NeMo should put the files in the standard directories.

4.5.4 Build SpikeStream

In the root directory type `qmake spikestream.pro` to generate a make file. Make sure that the version of qmake that you are using is 4.7 or higher by typing `qmake --version`. Invoking the complete path to the correct version of qmake will also work, for example, /home/david/Programs/QT-IDE/qt/bin/qmake

Type `make` to build all of the components that you have enabled in spikestream.pro file. Type `sudo make install` to install SpikeStream.

NOTE: The default installation location of the SpikeStream libraries, libspikestream, libspikestreamapplication, is /usr/local/lib. In some Linux distributions this location may not be on the list of directories to be searched by the library loader. You can change the installation directories by editing the target.path variable in the files library/library.pro and applicationlibrary/applicationlibrary.pro. Alternatively you can add /usr/local/lib to the LD_LIBRRAY_PATH variable using: export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib.

Before SpikeStream can run it is necessary to set up the SpikeStream databases. This can be done manually, or using the SpikeStream Database Configuration Tool, which is described in Section 5.

Once the databases have been configured, the SpikeStream application can be run from the bin directory of the installation.

5. SPIKESTREAM DATABASE CONFIGURATION TOOL

5.1 Introduction

This tool creates the databases for SpikeStream and stores their settings in the SpikeStream configuration file (see Section 10). Before launching this tool you need to have the details of the MySQL server that will be hosting the SpikeStream databases. If you are running MySQL on a local machine, then you will just need the username (possibly "root") and password of an account that has permission to create databases.

5.2 Configuring SpikeStream Databases

WARNING: This tool completely resets the selected SpikeStream databases. Any data that is currently on these databases will be lost.

In Windows the database configuration tool can be launched by opening the "bin" directory in the SpikeStream installation and double clicking on the file "dbconfigtool.exe". A shortcut to the tool is also added in the Start Menu. In Linux, run dbconfigtool in the bin directory. In Mac OS X ..FIXME FIXME. This application is shown in Figure 9.

Figure 9. SpikeStream Database Configuration Tool. All three databases have been selected for configuration and the host, username and password entered for each. The host, username and password can be different for each database and it is possible to configure just one or two of the databases.

Click on the check box for the databases that you want to configure and enter the requested information. The settings for each database can be the same or different and only the selected database(s) will be configured. Warning dialogs will be shown if the MySQL server(s) cannot be found or if the databases already exist. When you have entered the requested information you will be presented with the dialog shown in Figure 10.

17
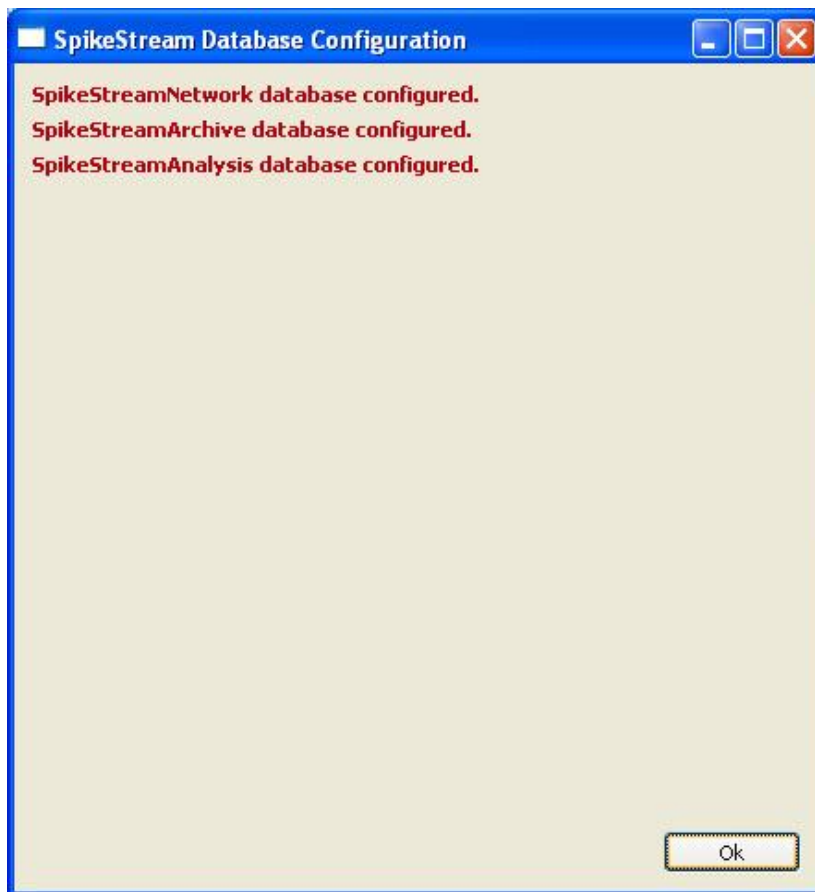
Figure 10. Results page of SpikeStream Database Configuration Tool. These results will vary depending on the databases that have been selected for configuration.

It should now be possible to run SpikeStream.

WARNING: When data already exists in the SpikeStream databases it is recommended to reconfigure all three databases at once. Otherwise there are likely to be data inconsistencies.

6. ARCHITECTURE

6.1 Databases

SpikeStream is based around three databases that hold all of its data. There is currently no way within SpikeStream to back up your data to an external file or to restore it at a later point in time. The best way to do this is using the mysqldump program, which is described in the following articles:

- http://dev.mysql.com/doc/refman/5.1/en/mysqldump.html
- http://www.devarticles.com/c/a/MySQL/Backing-Up-Your-MySQL-Databases-With-MySQLDump/

It is *strongly recommended* that you practice backing up and restoring your data with mysqldump before using it on important data. More information about the SpikeStream databases is given in Section 9.

It should be possible to use multiple instances of SpikeStream with a single database as long as each copy of SpikeStream is editing a different network. The use of multiple copies of SpikeStream on a single database has not been tested. Editing a single network with multiple instances of SpikeStream is likely to lead to data inconsistencies.

WARNING: If you wipe the database - for example, using the MySQL client, the database configuration tool or from within SpikeStream - then you lose everything, unless you have backed it up.

6.2 Networks

The current networks are listed on the Networks Tab, which is covered in Section 7.1. A network contains information about the neurons and the connections between them. Information about networks is stored in the SpikeStreamNetwork database.
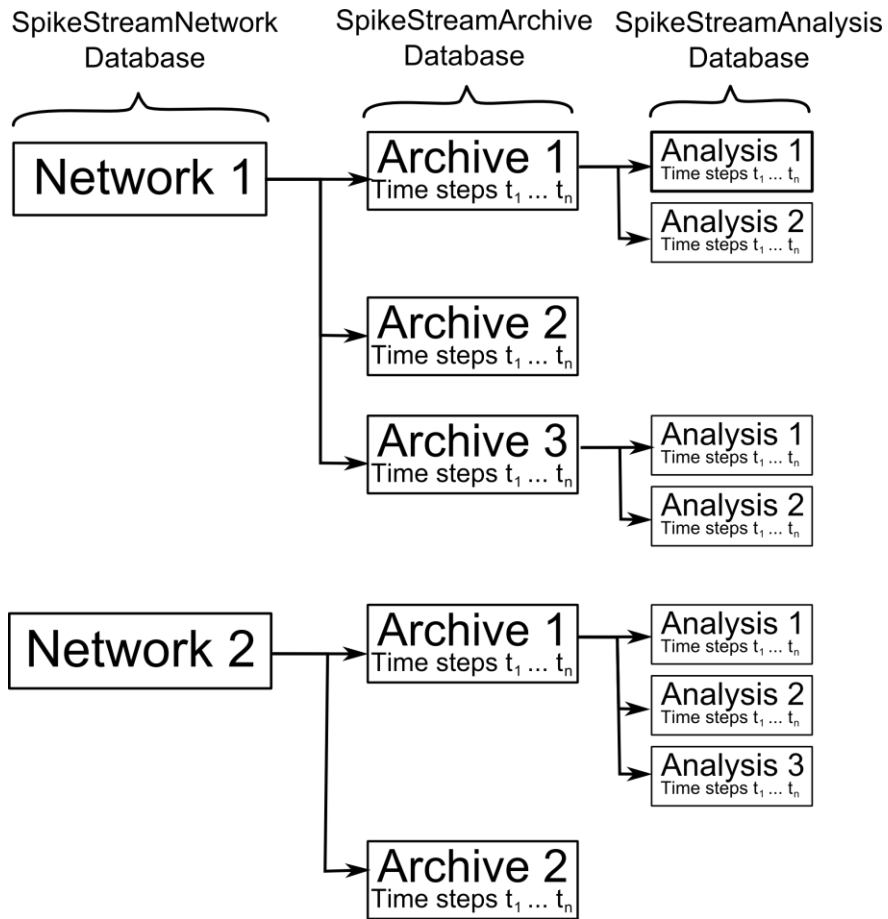
6.3 Archives

Each archive holds a sequence of one or more time steps containing the firing patterns of the neurons in the network at that time step. Archives can be viewed and played back on the Archives Tab, which is described in Section 7.6. Archives are stored in the SpikeStreamArchive database.

6.4 Analyses

Analyses in SpikeStream are handled as dynamically loaded plugins, which are managed using the Analysis Tab described in Section 7.7. Analyses are stored in the SpikeStreamAnalyses database.

6.5 Relationship Between Networks, Archives and Analyses

The structure of the data within SpikeStream mirrors that of the databases. At the top of the hierarchy is a network and only one network can be loaded into SpikeStream at a time. Each network is associated with one or more archives that contain the firing patterns of the network, and each firing pattern is associated with one or more analyses. This hierarchy is shown in Figure 11.

**Figure 11**. Relationship between networks, archives and analyses.

Each archive is an archive of a particular network and each analysis is an analysis of a particular state of a particular network.

WARNING: An archive is meaningless without its associated network, and so an archive will be deleted if its network is deleted. Similarly, an analysis is meaningless without its associated archive and it will be deleted if this archive is deleted.

7. CORE FUNCTIONALITY

7.1 Networks Tab

When you launch SpikeStream you are presented with the Networks Tab, as shown in Figure 12.
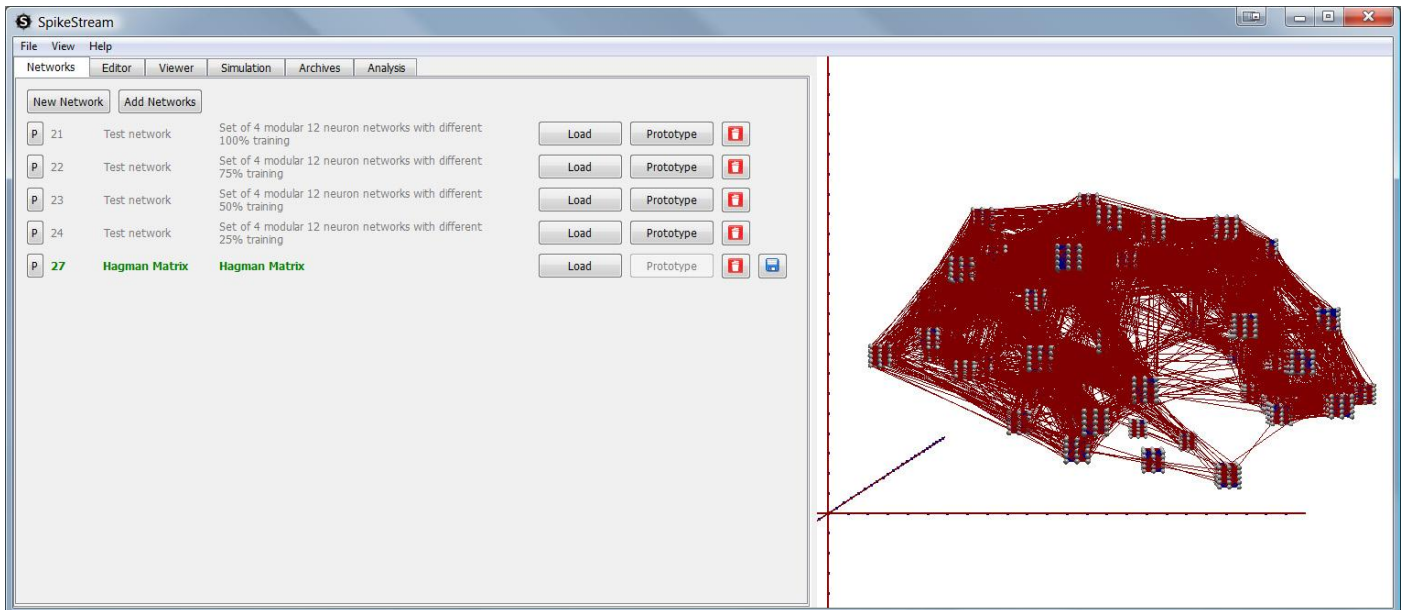
Figure 12. Networks Tab

Clicking on "New Network" launches a dialog that enables you to enter a name and description of the new network. Enter the name and description and click "Ok" to create the new empty network.

Clicking on "Add Networks" launches the Add Networks dialog shown in Figure 13. The Add Networks dialog displays the currently available network plugins. This type of plugin is designed to add entire networks to the SpikeStream database, and possibly associated archives as well. When more than one network plugin is available you can select between network plugins using the combo box at the top of the dialog. The network plugins that are currently available for SpikeStream are documented in Section 8.2.
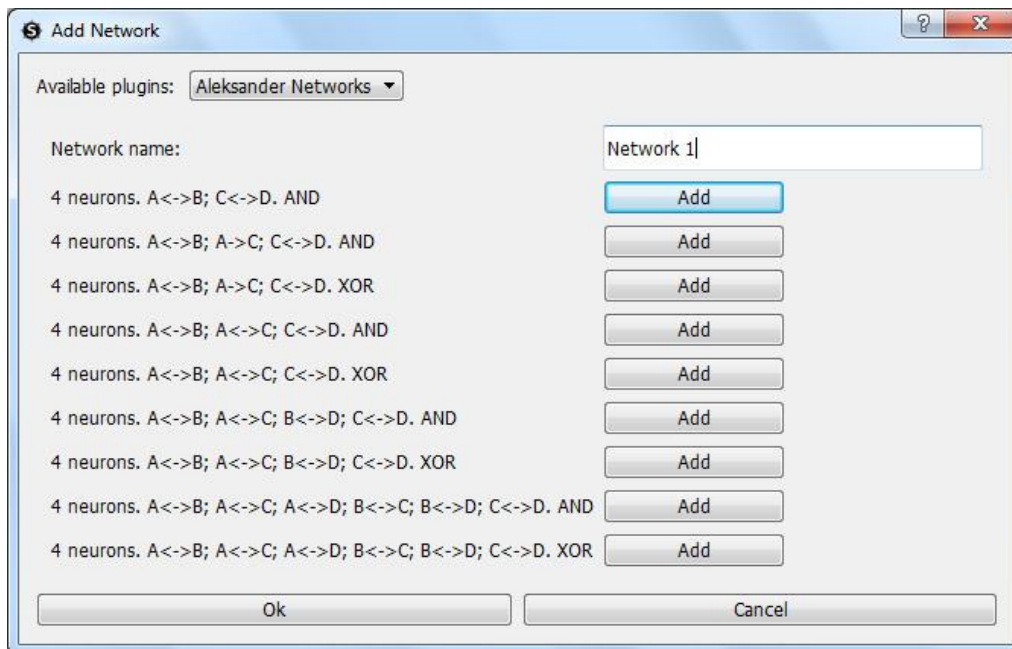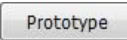
Figure 13. Add Networks dialog showing the Aleksander Network plugin (see Section 8.2.1)

Note the following features of loading, prototyping and saving networks in SpikeStream:

- Only a single network can be loaded at a time.

- When a network is loaded any changes to the network are immediately saved to the networks database. This is convenient for small networks, but it can take a long time when there are a large number of neurons or connections.

- A network can also be opened in prototype mode. In this case, some or all of the information about the network exists only in memory.

- When a prototyped network is changed a save button appears. Clicking on this button writes any unsaved changes to the networks database.

- Prototyped networks can be simulated, but their firing patterns cannot be archived. This is because the archive data references neuron IDs in the networks database, which may not exist for a prototyped network. Prototyped networks cannot be analysed for the same reason.

The following controls are available in the networks tab:

- Enables you to change the name and description of the network.

- Loads a network into SpikeStream.

- Loads a network into SpikeStream in prototype mode. Changes to the network are made in memory and will be lost unless they are saved to the database.

- Deletes a network from memory and the database.

- Saves a network that is in prototype mode to the SpikeStream database.

WARNING: Clicking on delete will permanently remove the network and all of its associated archives and analyses from the SpikeStream database. This step cannot be undone.

The shortcut **F1** can be used to switch to the Networks Tab.

7.2 3D Network Viewer

The right side of SpikeStream is taken up with the 3D Network Viewer, which displays the current network in three dimensions. The 3D Network Viewer has the following features:

- Navigation in the 3D Network Viewer is carried out using the shortcuts described in Section 11.

- Neuron and connection groups can be shown or hidden in the 3D Network Viewer using the Editor tab, as described in Section 7.3.

- The quality of the render and other viewing parameters can be set in the Viewer tab, as described in Section 7.4.

- Double clicking on a neuron in the 3D Network Viewer enables you to see the properties of its connections in the Viewer Tab, as described in Section 7.4.

- During simulations the firing patterns, membrane potential and weights can be viewed in the 3D Network Viewer as described in Section 8.5.1.

- When archives are played back neurons are highlighted in the 3D Network Viewer as described in Section 7.6.

- The results of analyses can be displayed in the 3D Network Viewer, as described in sections 8.6.1 and 8.6.2.

- Additional configuration parameters for the 3D Network Viewer can be found in the spikestream.config file, which is described in Section 10.

- To reduce the render times, the 3D Network Viewer may only show a selection of connections within a particular connection group. The Viewer tab lists all of the visible connections when a neuron has been double clicked. Parameters controlling this can be found in the spikestream.config file, which is described in Section 10.

In its initial state, the 3D Network Viewer is organized with the positive Z axis pointing upwards, the positive Y axis pointing away from the viewer and the positive X axis moving left to right.

7.3 Editor Tab

The Editor Tab shown in Figure 14 is used for editing an existing network. Clicking "Add Neurons" displays a dialog showing the available plugins for adding neuron groups. Clicking "Add Connections" displays a dialog showing the available plugins for adding connection groups. The plugins for adding neuron or connection groups are described in Section 8.3 and Section 8.4.
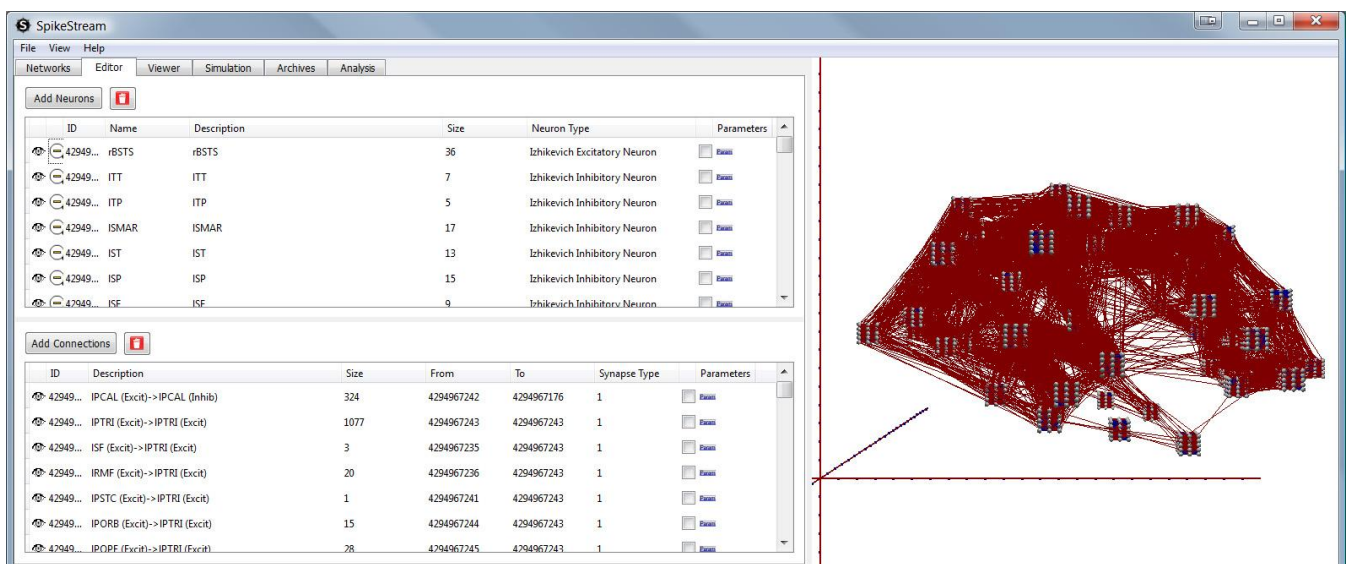

Figure 14. Editor Tab

A network is composed of one or more neuron and connection groups, which correspond to tables in the SpikeStreamNetwork database. The Editor tab has the following features:

- . Neuron and connection groups can be shown or hidden by clicking on the eye icon.

- . A single click on this icon zooms in to the side of the corresponding neuron group; a second click zooms above the corresponding neuron group; a third click zooms out to view the entire network.

- . Neuron and connection groups can be deleted by selecting them and clicking on the trash can icon. This step cannot be undone.

- Clicking on the header of the column with the eye symbol will show or hide all neuron or connection groups.

- Clicking on the header of the selection column will select or deselect all neuron/connection groups.

- Double clicking on the name or description of a neuron/connection group will launch a dialog that enables you to set its name and description.

- Clicking on the parameters column displays a dialog listing the parameters that were used to create the neuron or connection group.

The shortcut **F2** can be used to switch to the Editor Tab.
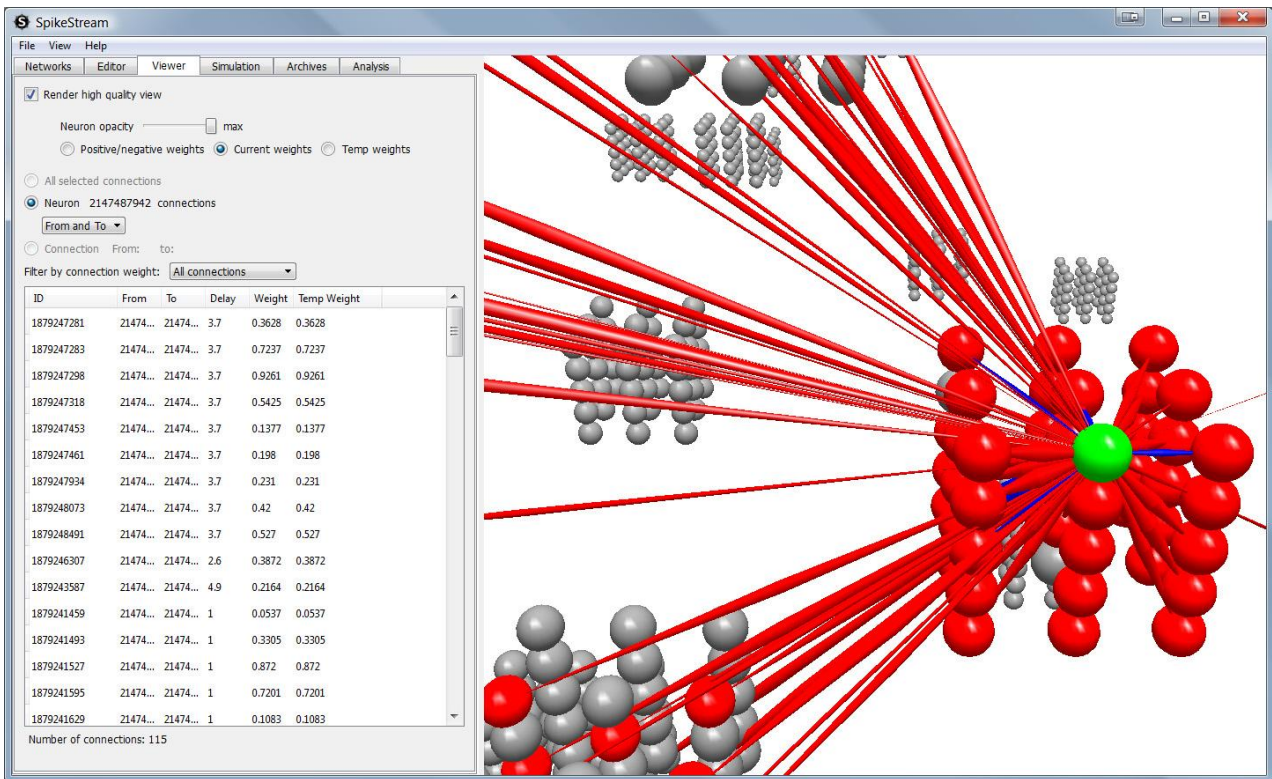
7.4 Viewer Tab

The Viewer Tab is used to set properties of the 3D Network Viewer – see Section 7.2 – and to display details about the connections TO or FROM a selected neuron.

The top of the viewer tab has controls for adjusting the rendering settings:

- **Render high quality view**. Neurons are rendered as spheres with lighting. It becomes possible to visualize the weights as the thickness of the connections.

- **Neuron opacity**. Neurons can be made partially or completely transparent to aid visualization.

- **Positive/negative weights.** Weights are rendered as lines whose colour indicates whether it is positive (red) or negative (blue).

- **Current weights.** The thickness of the line indicates the strength of the connection. The colour indicates whether it is positive (red) or negative (blue).

- **Temp weights.** Temporary weights are set as the result of a simulation or analysis. The thickness of the line indicates the strength of the connection. The colour indicates whether it is positive (red) or negative (blue).

To view the connections TO and/or FROM an individual neuron, double click on the neuron in the 3D Network Viewer. When the neuron is selected it turns bright green and only the connections TO and/or FROM the neuron are shown – see Figure 15. This feature is at an early stage of development and double clicking to select neurons is easier in full render mode and when zoomed out from the network. The connections TO and/or FROM the neuron are listed in the table. Note that only the connections that are set to visible in the Editor Tab are listed.
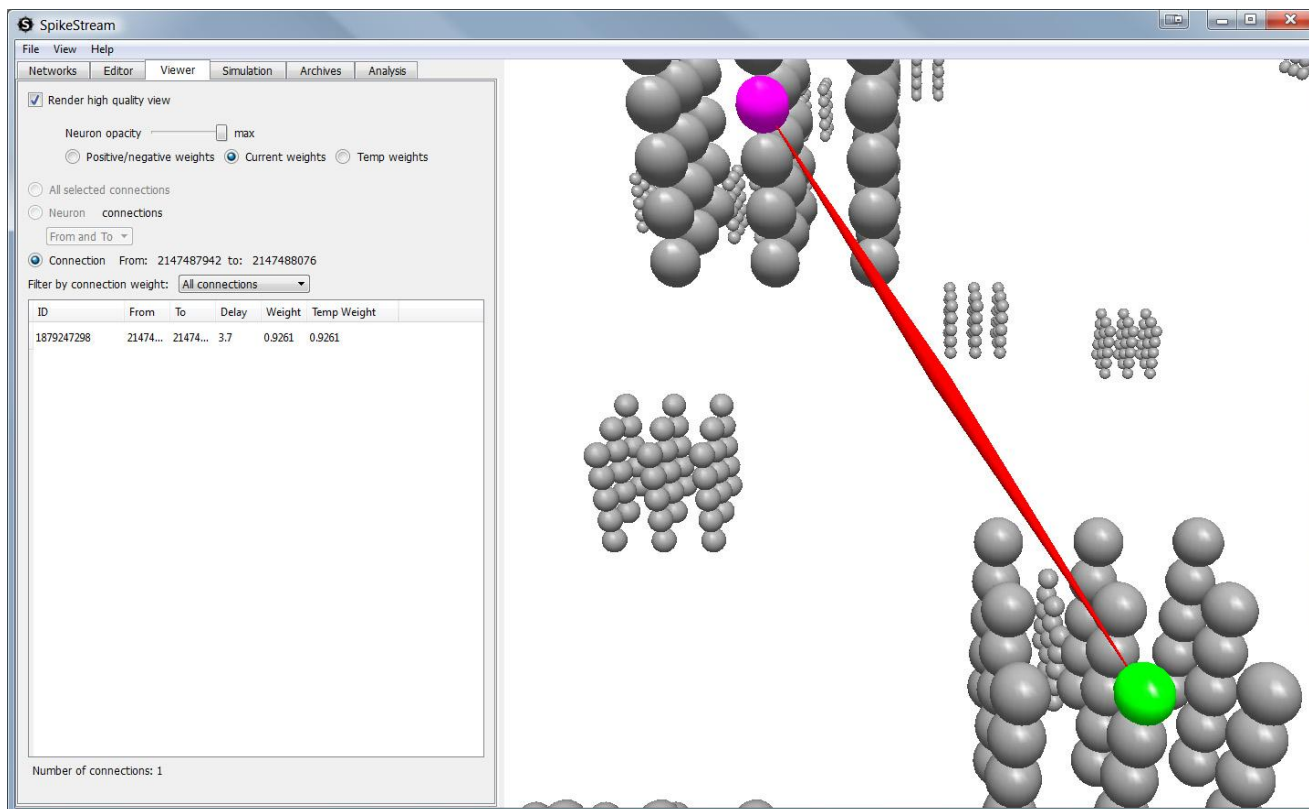
Figure 15. Viewer Tab showing connections TO a weightless neuron that was selected by double clicking

The From/To combo box can be used to select whether connections FROM and/or TO the selected neuron are shown and listed in the table. When just TO connections are shown, the truth table for a weightless neuron can be viewed by clicking the Truth Table button, as shown in Figure 16.
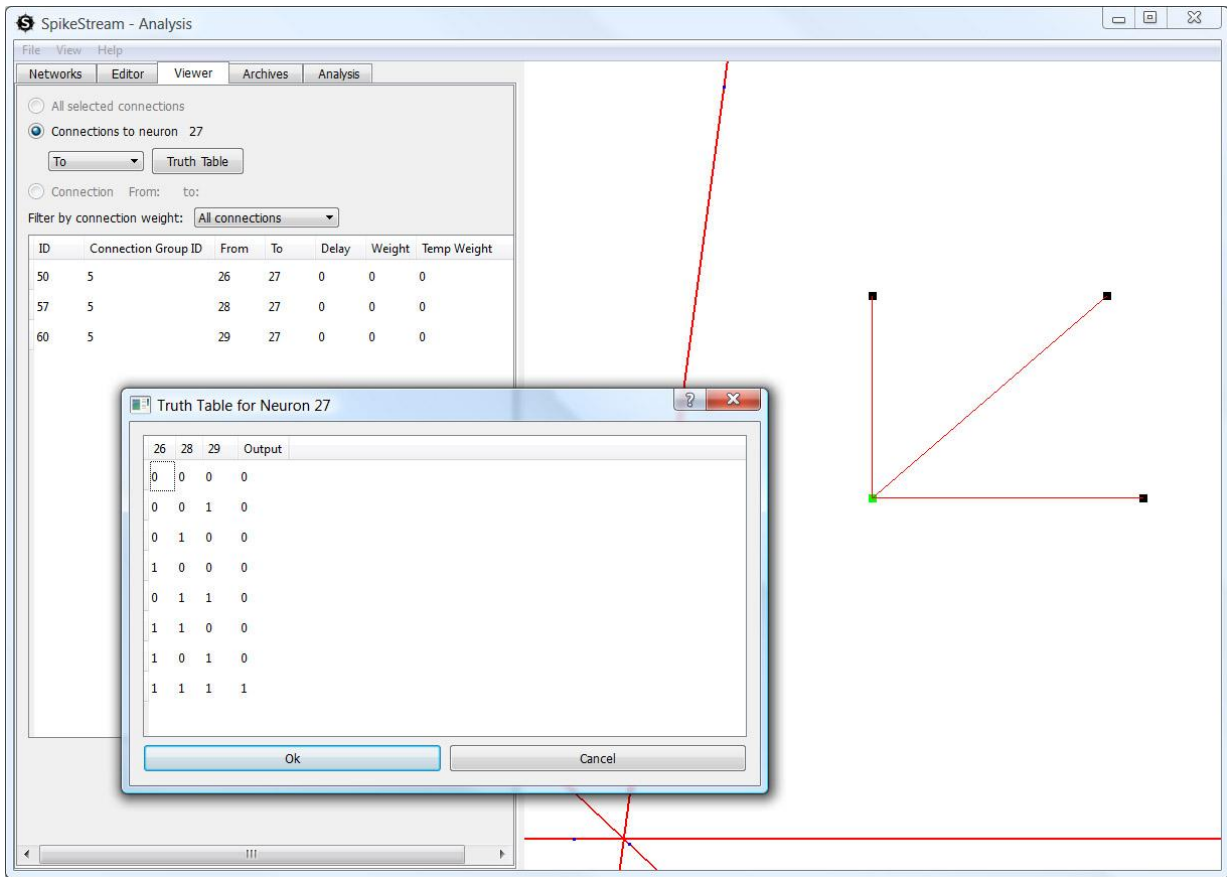
Figure 16. Viewer Tab showing connections TO a weightless neuron that was selected by double clicking, and a dialog displaying the selected neuron's truth table

When one neuron is selected, the connections between that neuron and another neuron can be viewed by holding down CTRL and double clicking another neuron. The second neuron appears coloured purple and the connections FROM the first neuron TO the second neuron are shown in the Viewer Tab, as shown in Figure 17.

Figure 17. Viewer Tab showing connections from the neuron highlighted in green to the neuron highlighted in purple.

NOTE: The connection viewing mode takes precedence over the archive playing and analysis highlighting. You need to deselect a neuron that you have double clicked before playing back an archive or highlighting a cluster.

The shortcut **F3** can be used to switch to the Viewer Tab.

7.5 Simulation Tab

The Simulation tab loads up and displays all of the available simulation plugins. When more than one plugin is available, the plugins can be selected using the combo box at the top of the Simulation tab. The current simulation plugins are covered in Section 8.5.

The shortcut **F4** can be used to switch to the Simulation tab.

7.6 Archives Tab

The Archives tab, shown in Figure 18, is used to play back archived states of the network. Each archive consists of one or a number of time steps containing the neurons that were firing at that time step.
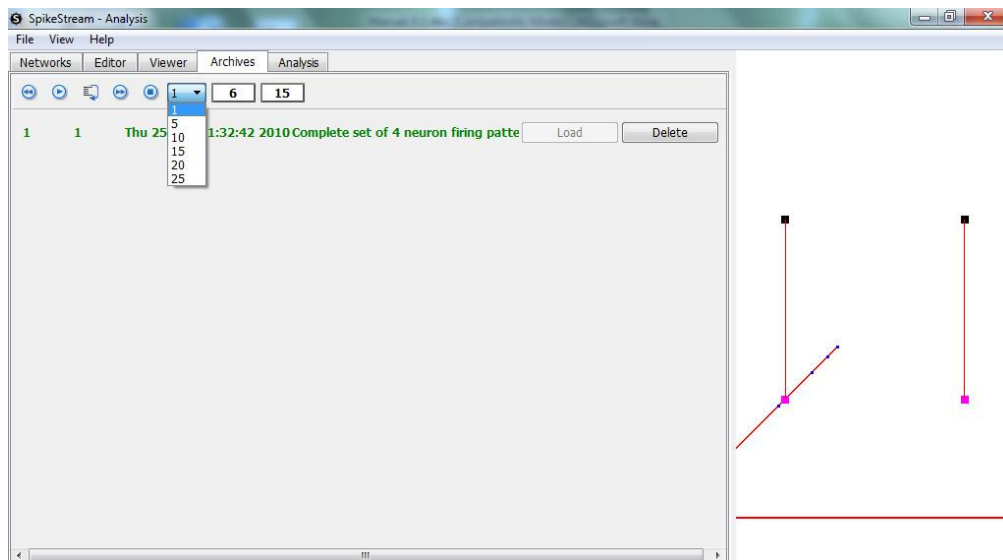
Figure 18. Archives tab

Each network can be associated with one or more archives. Clicking on the "Load" button causes the corresponding archive to be loaded up ready to play. Clicking on the "Delete" button permanently removes the archive and any associated analyses from the SpikeStream databases.

WARNING: Clicking on the delete button causes the archive to be permanently deleted from the database. This step cannot be undone.

When an archive is loaded the following controls can be used to play back the archive in the 3D Network Viewer:

Rewinds an archive back to the beginning.

Plays the archive. The combo box next to the stop button sets the speed of playback.

Steps through the archive one time step at a time.

Fast forwards through the archive.

Stops playback of the archive.

The shortcut **F5** can be used to switch to the Archives Tab

7.7 Analysis Tab

The Analysis Tab loads up and displays all of the available analysis plugins. When more than one plugin is available, the plugins can be selected using the combo box at the top of the Analysis Tab. The current plugins that are distributed with SpikeStream are covered in Section 8.6.

The shortcut **F6** can be used to switch to the Analysis Tab.

8. PLUGINS

8.1 Introduction

Many of the functions of SpikeStream are implemented by plugins, which make it easy to extend the functionality without altering the main code base. Plugins can use the spikestream and spikestreamapplication libraries or they can be entirely independent code. The minimal requirements for a SpikeStream plugin are described in Section 8.7.

SpikeStream currently supports the following types of plugin:

- **Network plugin**. Creates entire networks, including neurons, connections and archives, or import networks from files. Should be installed in the plugins/networks folder.

- **Neurons plugin**. Creates neuron groups within the currently loaded network. Should be installed in the plugins/neurons folder.

- **Connections plugin**. Creates connection groups within the currently loaded network. Should be installed in the plugins/connections folder.

- **Simulation plugin**. Carries out simulations and stores the result in the SpikeStreamArchive database. Should be installed in the plugins/simulation folder.

- **Analysis plugin**. Analyzes the currently loaded network. Should be installed in the plugins/analysis folder.

The next sections cover the plugins that form part of the current SpikeStream distribution. The functionality of these plugins reflects the research interests of the current SpikeStream developers, and other people are encouraged to write and distribute their own plugins to extend SpikeStream's functionality. Instructions for writing plugins are given in Section 8.7.

8.2 Network Plugins

8.2.1 Aleksander Networks Builder

This plugin was written to add test networks for exploring ideas about information integration. It can be viewed by clicking on Add Networks on the Networks Tab and selecting "Aleksander Networks" from the drop down combo. This plugin is illustrated in Figure 13.

The names of each network indicates the connectivity and function of the neurons. For example, the third network in the list has the name "A<->B; A->C; C<->D. XOR", which describes the connectivity shown in Figure 19.
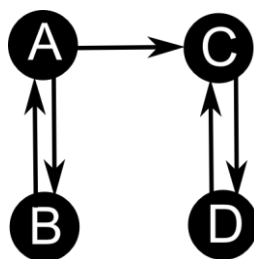
Figure 19. Aleksander plugin network: "A<->B; A->C; C<->D. XOR". Each of the neurons has an XOR function.

Clicking the "Add" next to the label adds the network to the database. If you are unsure about the connectivity and the functions of these networks, it is easy to add them and view their properties in the Viewer Tab, as described in Section 7.4.

The Aleksander Networks Builder creates networks of *weightless* neurons. It also adds an archive with 16 time steps to the database for each network that contains all of the possible firing patterns of the four neurons. These firing patterns can be played back using the Archives Tab, which is described in Section 7.6.

8.2.2 Tononi Networks Builder

The Tononi Networks Builder plugin (see Figure 20) was written to add some of the networks described in Balduzzi and Tononi (2008). These networks are composed of *weightless* neurons, and this plugin also adds an archive with a single time step that contains the firing pattern of the network as described in Balduzzi and Tononi (2008). Click on the "Add" button to add the network to the SpikeStream database.
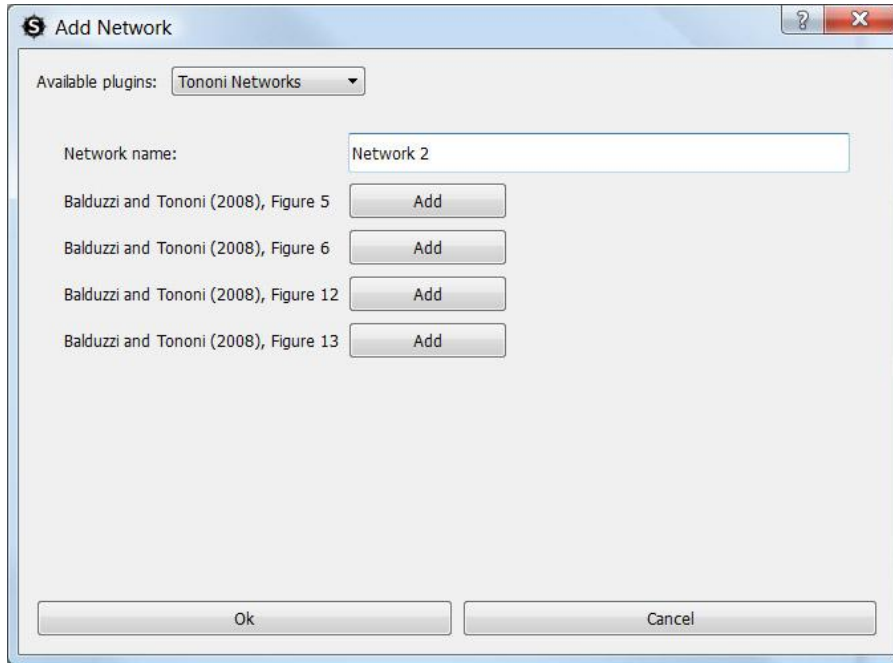

Figure 20. Tononi Networks Builder

8.2.3 Aleksander/Gamez Test Networks 2

This plugin (see Figure 20) adds a selection of 12 neuron networks that were used to compare the State-based φ measure of information integration with the liveliness measure. These networks are composed of *weightless* neurons. Click on the "Add" button to add the network to the SpikeStream database.
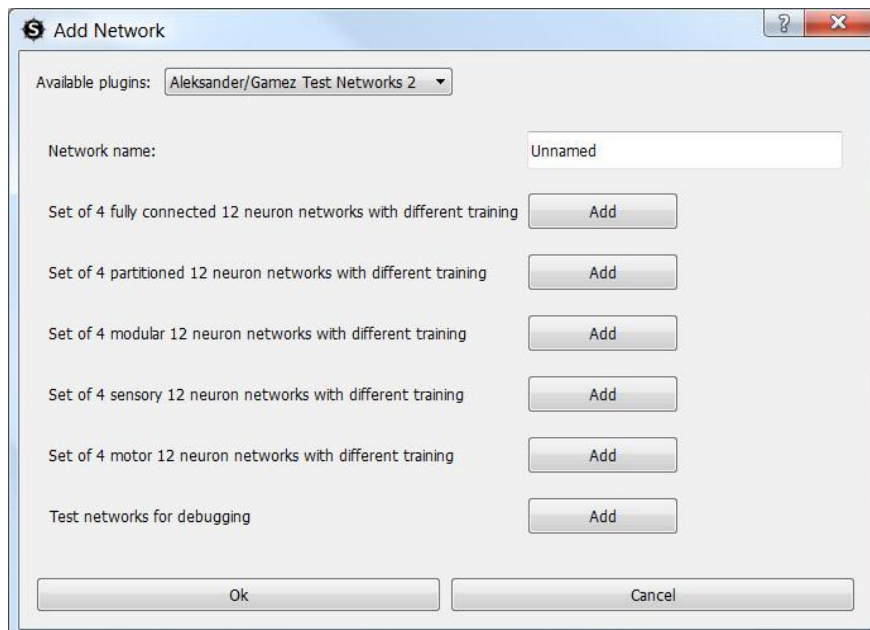

Figure 20. Aleksander/Gamez Test Networks 2 plugin

8.2.4 Connection Matrix Importer

SpikeStream can import connection matrices. The following files are used to specify the names of the nodes, the location of the nodes in Talairach coordinates, the connection weights between the nodes and the lengths of the connections:

- **Node names**. A list of labels for the nodes.

- **Talairach coordinates**. Location of the node in Talairach coordinate space.

- **C**. The connection weights between the nodes. Lines are targets; columns are sources.

- **L**. Lengths of the connections in mm. A parameter is used to convert the lengths into delays during the import.

The connection matrix importer plugin is shown in Figure 21. Select the files and click on "Import" to start the import. The network is added in prototype mode and will need to be saved if you want to archive its firing patterns or carry out analyses.
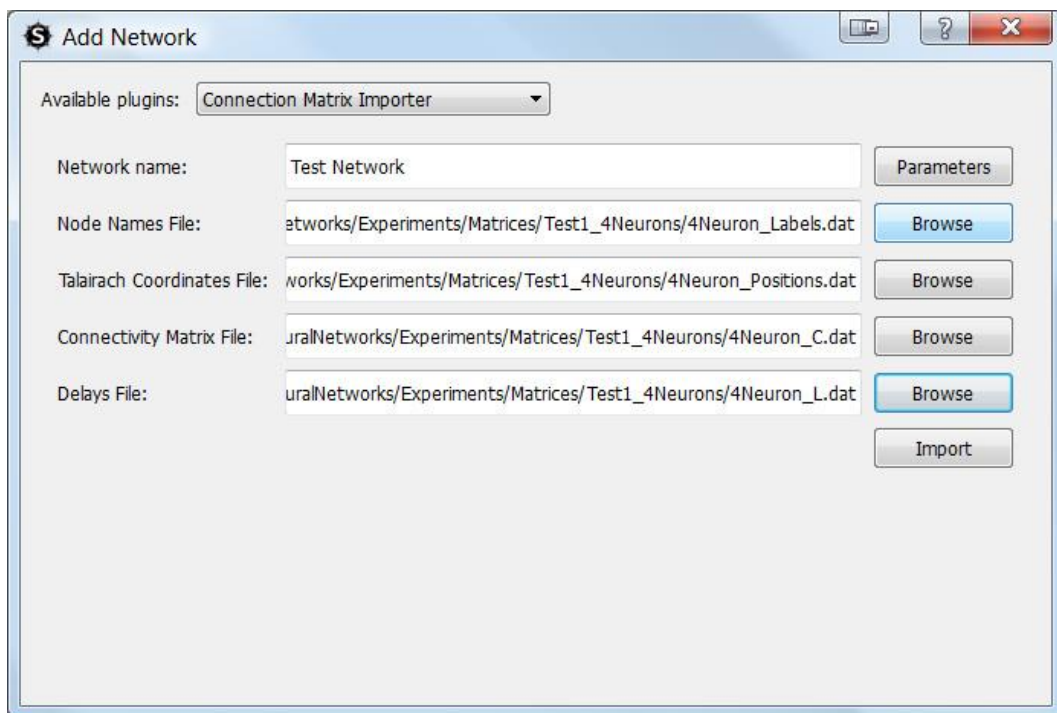
Figure 21. Connection Matrix Importer plugin

The files should all end with the extension .dat. Parameters for the import can be set by clicking the parameters button. Hover over the question mark to view a description of the parameter.

8.2.5 NRM Importer

SpikeStream has a limited ability to import files from the NRM neural simulator. Networks of weightless neurons can be simulated and trained in NRM and then imported into SpikeStream for analysis. This import functionality is pretty basic and has the following limitations:

- Only files created by the most recent version of NRM (2003) can be imported into SpikeStream.

- Only random connections are supported.

- Only neural layers with a single colour plane can be imported.

Three files are needed for an import from NRM:

- **Configuration file (*.cfg).** Specifies the network and connectivity of the NRM network.

- **Training file (*.ntr).** Contains the training of the neurons in the network.

- **Data set file (*.set).** Contains at least one firing pattern of the network.

Configuration files can be saved in NRM by selecting **Configuration->Save As**. Once a network has been built and trained, its training can be saved by selecting **Network->Training->Save network training as**. The export of data sets is more complicated because NRM does not support the saving of the state of the network at a particular time step, and so a convention has to be used in which the state of each layer in the network is added in sequence to the data set to store the network's state at a particular point in time. For example, consider the network shown in Figure 22.
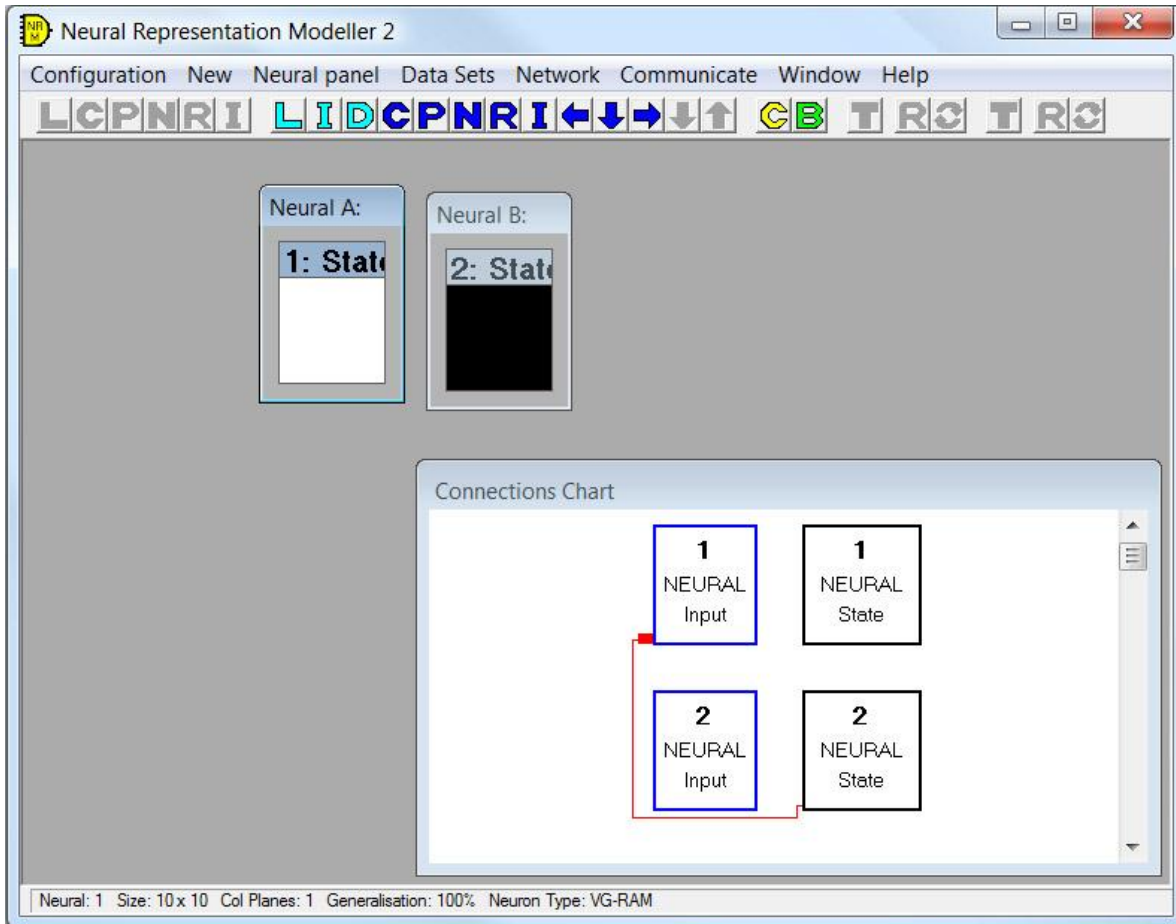
Figure 22. NRM network at time t_1

To add the state of this network, click on **Data sets->Create new**. This creates a new set and adds the state of Neural A to the set. Next, select Neural B and click on **Data sets->Add data array**. This adds the state of Neural B to the set. Suppose that at a later point in time the network is in the state shown in Figure 23.
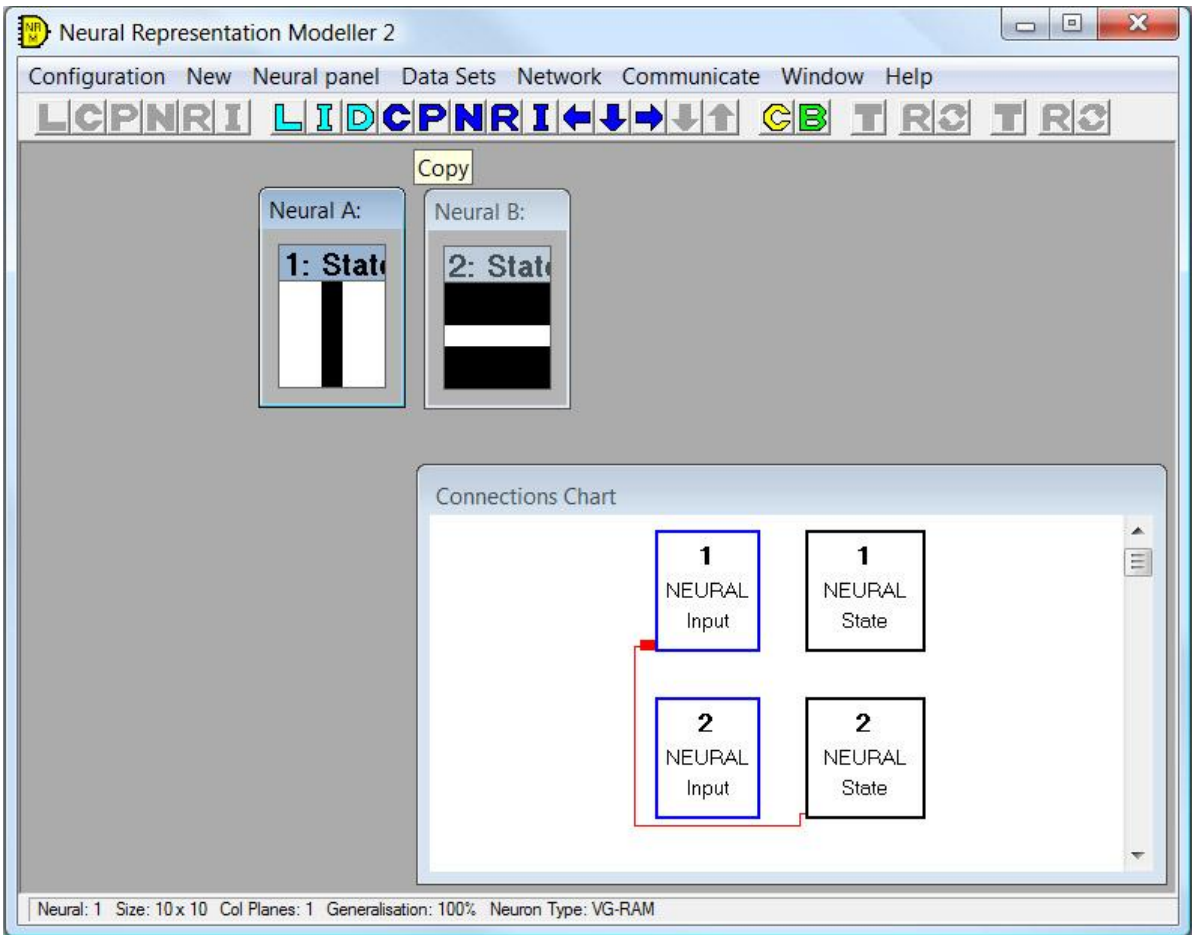
Figure 23. NRM network at time t_2

To add this state of the network to the set, select Neural A and click on **Data sets->Add data array.** Select Neural B and click on **Data sets->Add data array**. Your data set should now contain the following four entries

1. Neural A at time t_1 (all white).

2. Neural B at time t_1 (all black).

3. Neural A at time t_2 (vertical black stripe on white background).

4. Neural B at time t_2 (horizontal white stripe on black background).

Click on **Data sets->End create and save** to save this data to a .set file. When SpikeStream imports this data set it will interpret the first entry as the state of the first layer at time t_1, the second entry as the state of the second layer at t_1, and so on until it runs out of layers. The next entry will then be interpreted as the state of the first layer at t_2, and so on.

Once you have created your configuration, training and data set files, you can import them into SpikeStream by clicking **File->Import NRM Network** or using the shortcut **CTRL + M**. This displays the dialog shown in Figure 24.
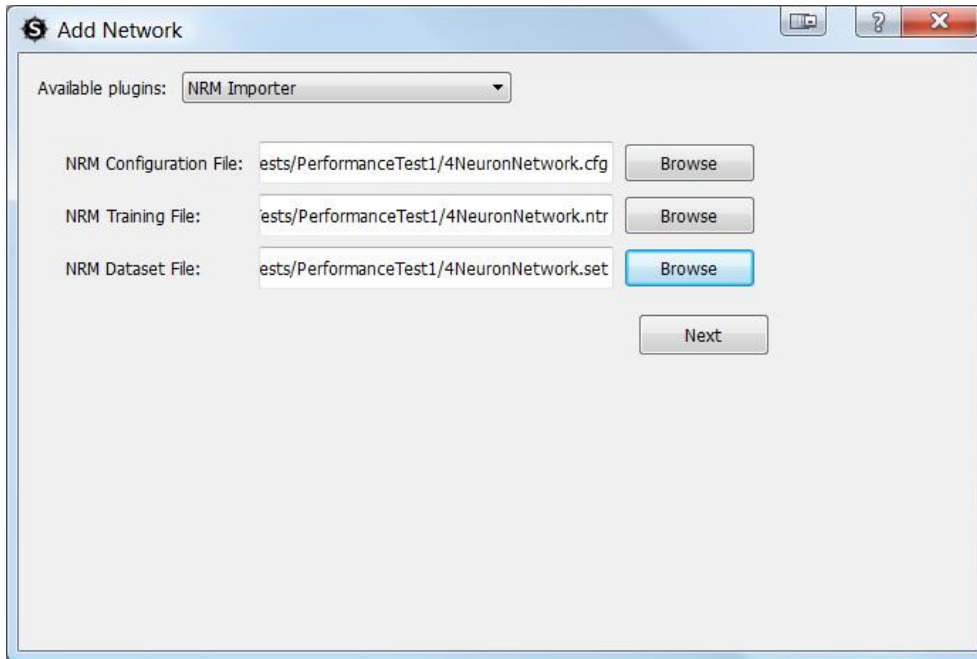
Figure 24. First screen of NRM Import Plugin

Select the configuration, training and data set files that you want to import and click "Next". If the files match and can be imported, you will be presented with the dialog shown in Figure 25.
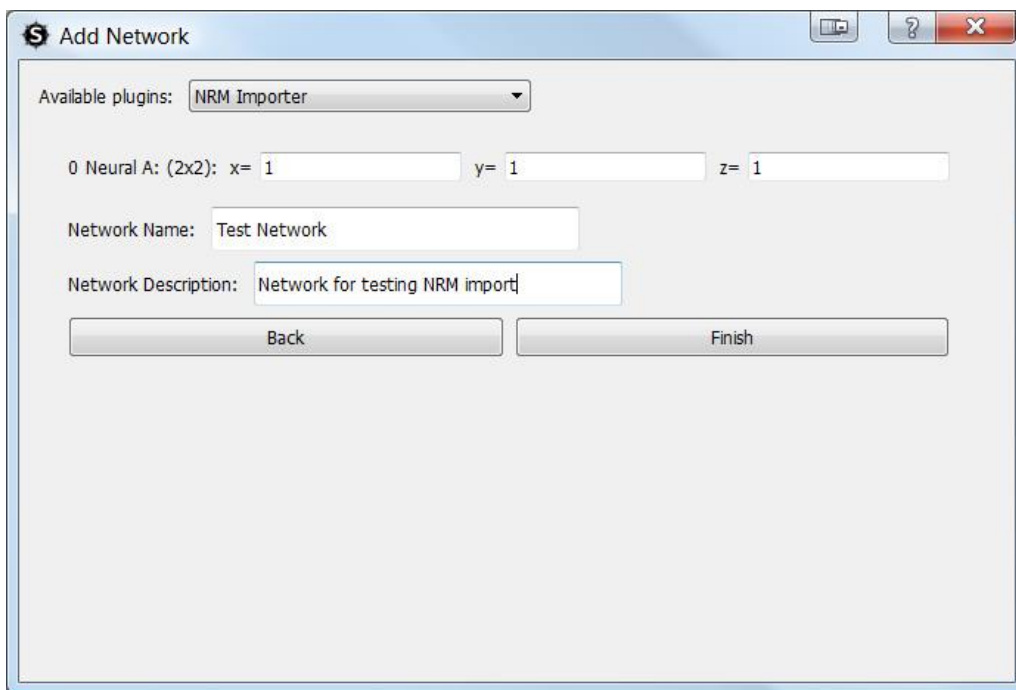
Figure 25. Second screen of NRM Import Dialog

This part of the dialog enables you to select the 3D location of the layers that are being imported and to provide a name and description of the network. When you click on "Finish", the NRM data will imported into the SpikeStream database as a new network, which will appear in the list of networks on the Network Tab (see Section 7.1).
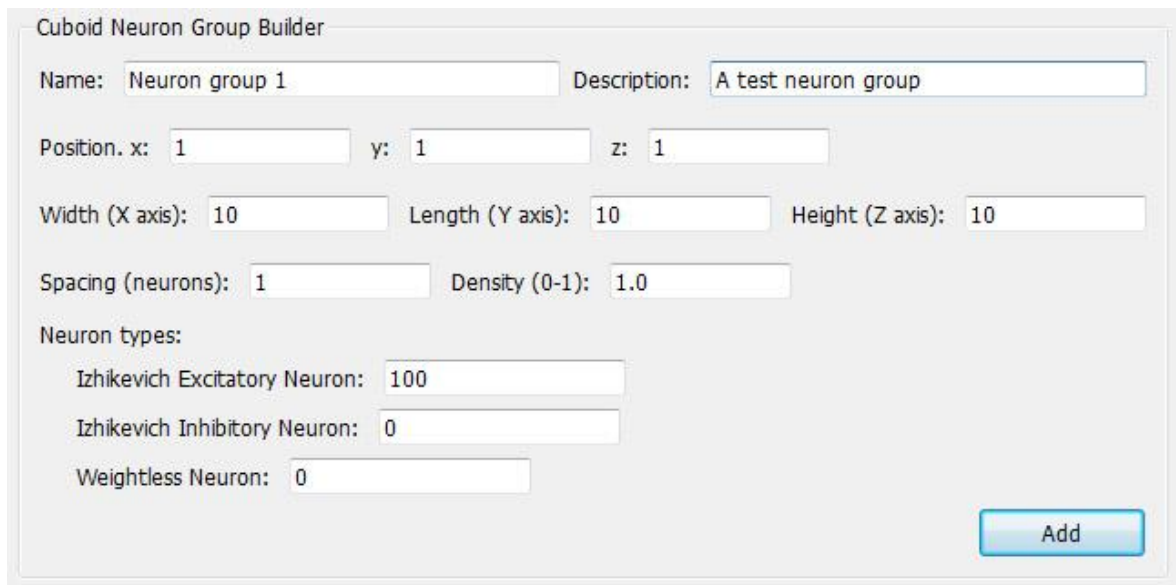
8.2.6 Izhikevich Networks

Adds the network described in Izhikevich (2006).

8.3 Neuron Group Plugins

8.3.1 Cuboid Neuron Group Builder

The Cuboid Neuron Group Builder plugin is shown in Figure 26. It enables the user to add a cuboid or planar neuron group to the network. When a number of different neuron types are included, these are added to the network as separate neuron groups.

**Figure 26**. Cuboid Neuron Group Builder plugin.

The parameters for this plugin are as follows:

- *Name*. The name of the neuron group.

- *Description*. A brief description of the neuron group.

- *Position*. The point on the neuron group closest to the origin. It is recommended to keep this value positive since it has not been tested with negative positions.

- *Width, length, height*. The width, length and height of the neuron group in neurons.

- *Spacing*. The spacing between neurons in the group.

- *Density*. The probability that a neuron will be created at a particular position.

- *Neuron types*. The proportion of each type of neuron that will be included in the cuboid. These numbers must add up to 100 and a separate neuron group will be created for each type.

8.4 Connection Group Plugins

8.4.1 Random1 Connection Group Builder

The Random1 Connection Group Builder plugin is shown in Figure 27. It enables the user to add random connections within or between neuron groups.
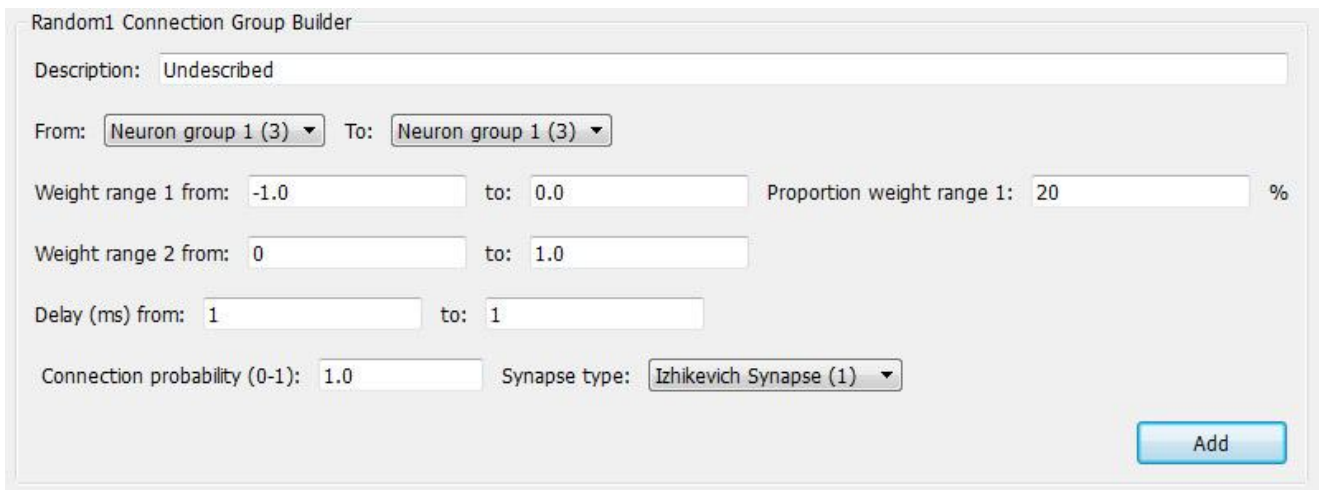
Figure 27. Random1 Connection Group Builder

The parameters for this plugin are as follows:

- **Description**. A brief description of the connection group.

- **Weight range 1, weight range 2**. Two different weight ranges can be included in the connection group. The parameter 'Proportion weight range 1' sets the proportion of weight range 1 that is used to create the weights. Connection weights are selected at random from the specified ranges.

- **Delay**. The delay of the connection. Delays are selected at random from the specified range.

- **Synapse type**. The type of synapse used in the connection.

8.4.2 Topographic Connection Group Builder

The Topographic Connection Group Builder is shown in Figure 28. It is used to create topographic connections from one neuron group to another neuron group. The topographic projection space of the source group is lined up with the centre of the destination group and may extend beyond the boundaries of the destination group. The parameters for this plugin are as follows:

- **Description**. Description of the new connection group.

- **From/to**. Source and destination neuron group for the connection.

- **Projection width, length and height**. Each neuron in the FROM layer projects into a region of space with this width height and length. The total projection space is aligned with the centre of the destination neuron group.

- **Overlap width, length and height.** The projection volumes of each FROM neuron can overlap in the TO neuron group.

- **Pattern.** Each FROM neuron can project to a Gaussian sphere, a uniform sphere or a uniform cube of neurons in the TO neuron group.

- **Density.** Set this parameter to less than 1 to reduce the connection probability between source and destination neurons.

- **Minimum/maximum weight.** Sets the range of weights for the connections.

- **Delay type.** If set to distance the delay is based on the distance multiplied by the distance factor. The delay can also be randomly selected from a range of values.
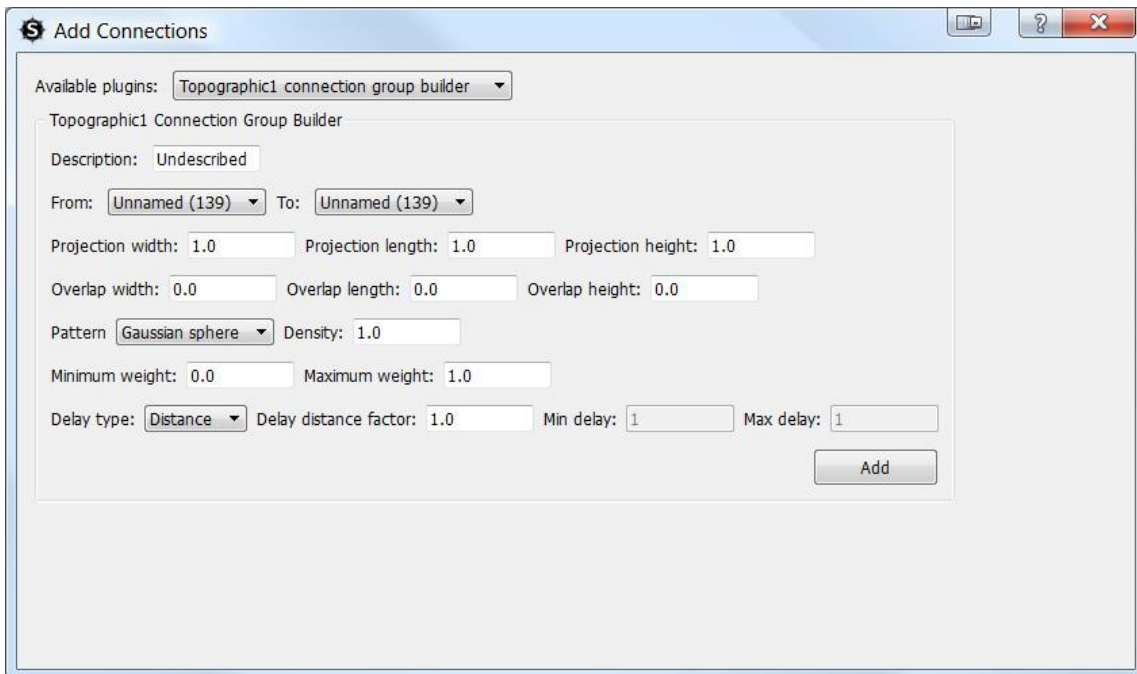
Figure 28. Topographic Connection Group Builder.

8.5 Simulation Plugins

8.5.1 NeMo Simulator

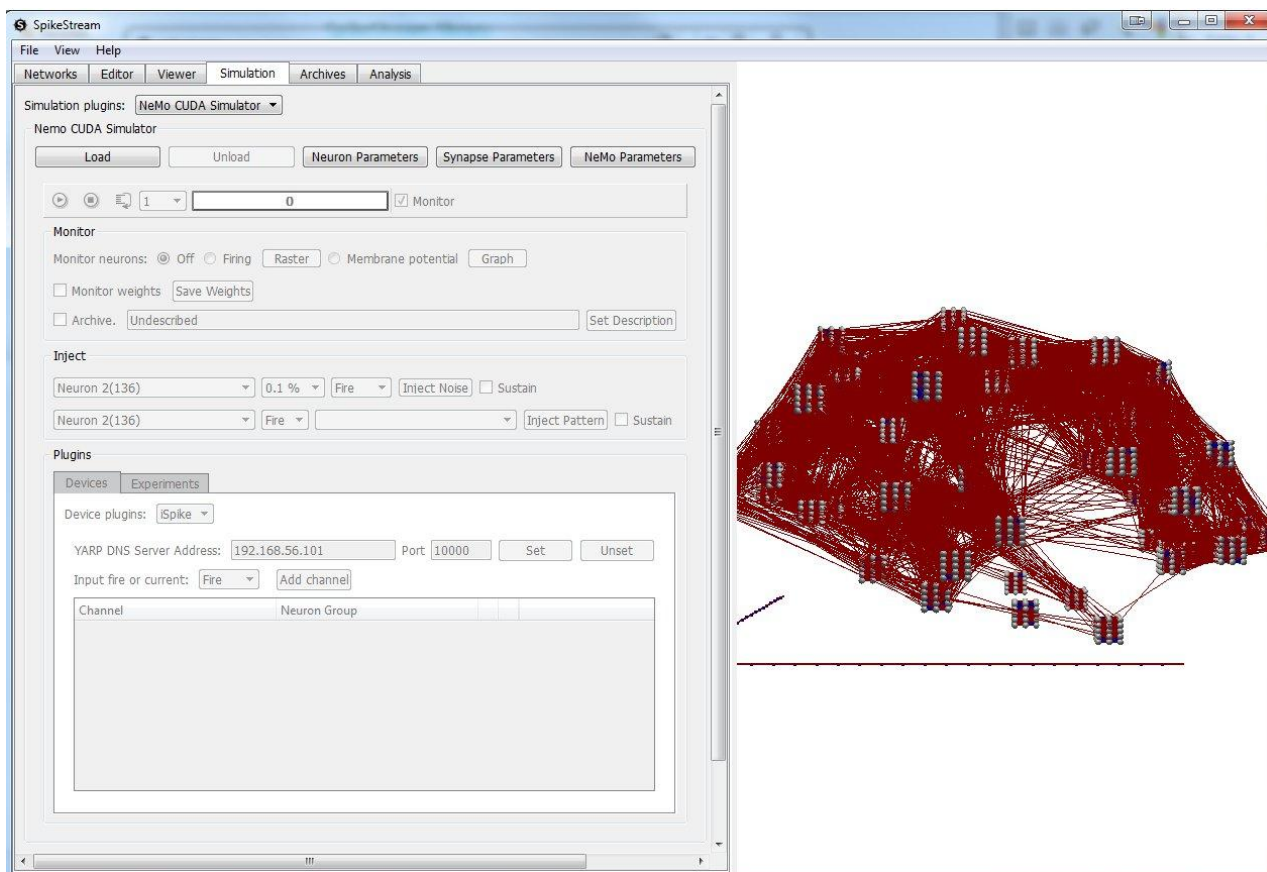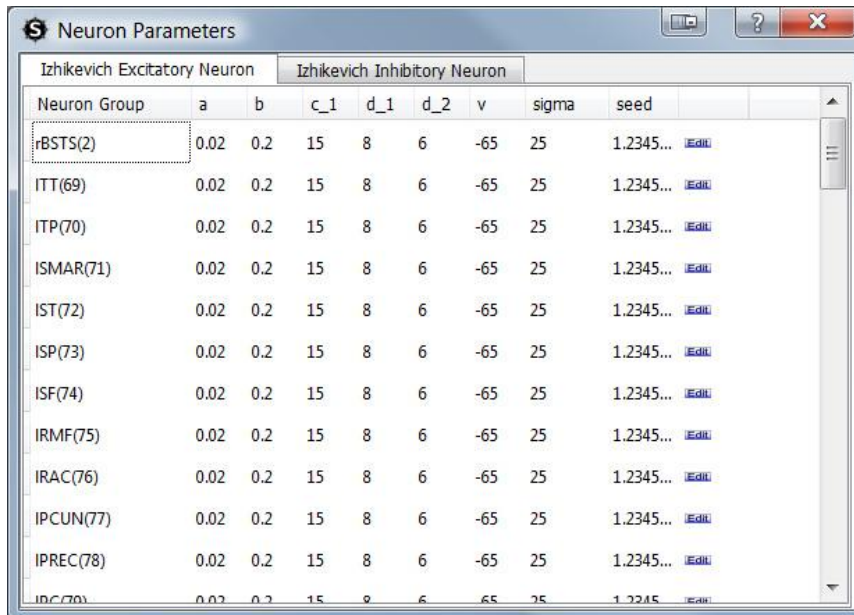The NeMo CUDA Simulator plugin (see Figure 29) wraps the NeMo simulator (http://www.doc.ic.ac.uk/~akf/nemo/index.html).

Figure 29. NeMo Wrapper plugin in unloaded state

The Neuron Parameters button launches a dialog to set the parameters for the different types of neurons, as shown in Figure30. The parameters for the Izhikevich model are documented in Izhikevich (2003).

Figure 30. Neuron parameters dialog

The Synapse Parameters button launches a dialog to set the parameters for the different types of synapse, as shown in Figure 31. Connection groups can be disabled and the STDP learning can be switched on and off. Click on Edit to change the parameters.
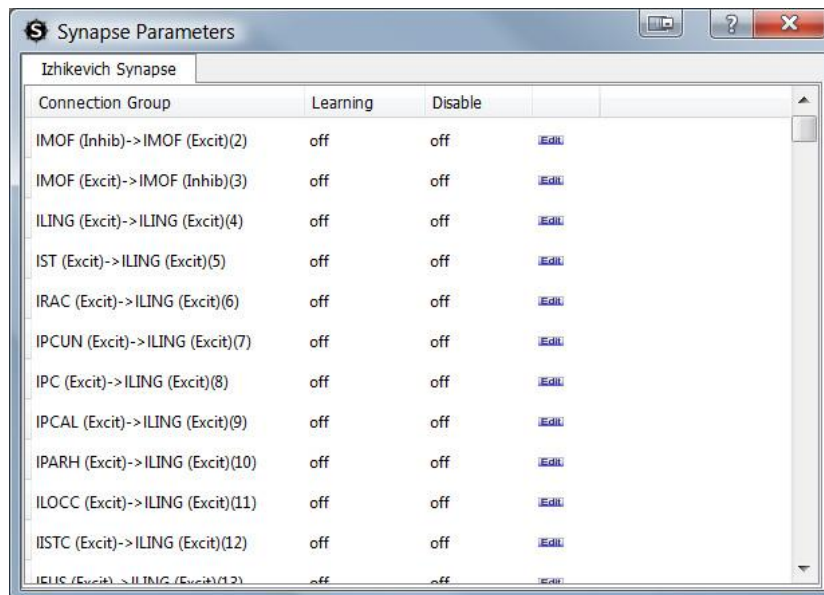
Figure 31. Synapse parameters dialog

Click on the NeMo Parameters button to set the NeMo parameters for the simulation, as shown in Figure 32. This dialog enables you to choose between using the CPU or CUDA and to set the STDP learning model. Click on the Edit button to change the parameters.

Figure 32. NeMo parameters dialog

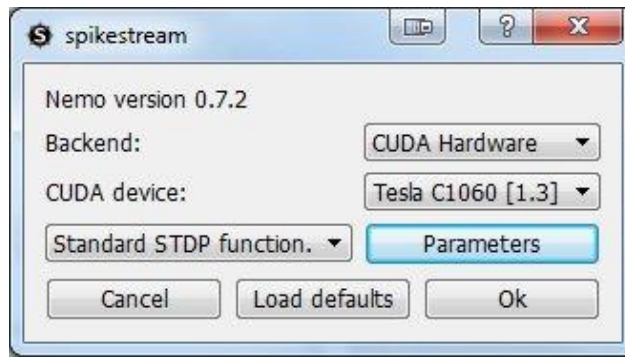When you click on "Load", the simulation is loaded and it is no longer possible to edit the synapse or NeMo parameters, as shown in Figure 33.

NOTE. It is necessary to unload the simulation before changing networks or closing SpikeStream.
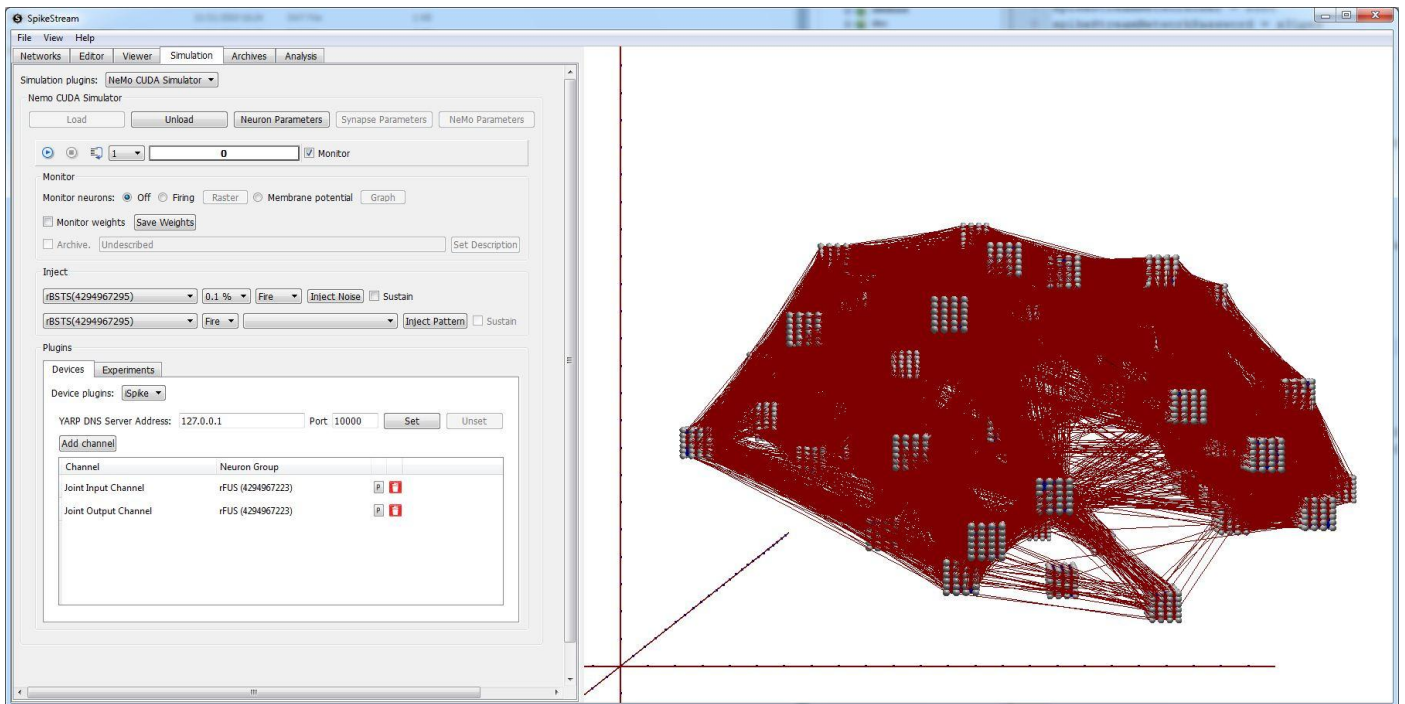
Figure 33. NeMo Wrapper plugin in loaded state

When a simulation is loaded the following features are available:

- ⊙ Runs the simulation. The combo box next to the step button sets the speed of playback in frames per second..

- ⊞ Advances the simulation one time step at a time.

- Stops the simulation.

- **Time step**. Displays the time step of the simulation in milliseconds.

- **Monitor check box**. Global control of the monitoring of the simulation. When this is unchecked no monitoring will occur, including of the time step, and the simulation will run much faster.

- **Monitor neurons radio button**. Controls whether the firing or membrane potential of the neurons is shown in the 3D Network Viewer. Click on the Raster button to view a raster plot for selected neuron

groups. Select a neuron and click on the Graph button to view a graph of the membrane potential of the selected neuron.

- **Monitor weights check box**. When using STDP the weights will change during the simulation. When this box is checked the current weights will be copied out of NeMo and stored as the Temp Weights, which can be visualized in the 3D Network Viewer (see Section 7.2).

- **Save weights button**. Saves the temporary weights to the database. This option is not available if the network is loaded in prototype mode.

- **Archive check box.** When this is checked the firing neuron patterns are written to the database. You can change the archive's description by entering a new description and clicking the Set Description.

- **Inject noise.** Select a neuron group, select a percentage of neurons, select whether the neurons are fired or have current injected into them and click on Inject Noise. A random selection of this number of neurons will fire or have current injected in the selected neuron group in the next time step. When the simulation is not running It is possible to select several different neuron group and percentage combinations by clicking Inject Noise after each selection. The sustain check box extends the current set up of noise injection over multiple time steps.

- **Inject pattern.** Patterns can be loaded from files and injected as noise or as current. Select the neuron group, select whether to fire or inject current and then load a pattern or select a pattern that has already been loaded. Patterns are defined using .pat files, and an example file can be found in the patterns folder of the SpikeStream installation. The injected pattern is aligned without scaling at the centre of the selected neuron group. It is possible to select several different neuron groups and patterns by clicking Inject Pattern after each selection. The sustain check box extends the current set up of pattern injection over multiple time steps.

At the bottom of the NeMo wrapper plugin are two tabbed windows, which display plugins for carrying out experiments and plugins for interacting for external devices. The experiments are not included in the release because they are highly specific to particular networks. An example experiment can be found here: http://spikestream.svn.sourceforge.net/viewvc/spikestream/trunk/simulators/nemo/experiments/exampleexperiment/.

The iSpike wrapper shown in Figure 34 enables spiking neural networks to interact with external devices – particularly the real and virtual iCub robots using YARP. First, type in the IP address of the YARP server and port and click on the 'Set' button. Then click on 'Add channel' to set up an input or output channel. Parameters for the channel appear in a wizard style dialog. Some of these channel properties can be altered after a channel has been created by clicking on the P button, and a channel can be deleted by clicking on the trash icon, . The "Input fire or current option" controls whether spikes from iSpike fire corresponding neurons in SpikeStream, or whether they inject current into the SpikeStream neurons.

NOTE. When using file or image readers and writers in the iSpike wrapper, full paths may need to be specified with forward slashes. For example, instead of "anglesIn.txt" use "D:/Program files/spikestream-0.2/bin/anglesIn.txt".
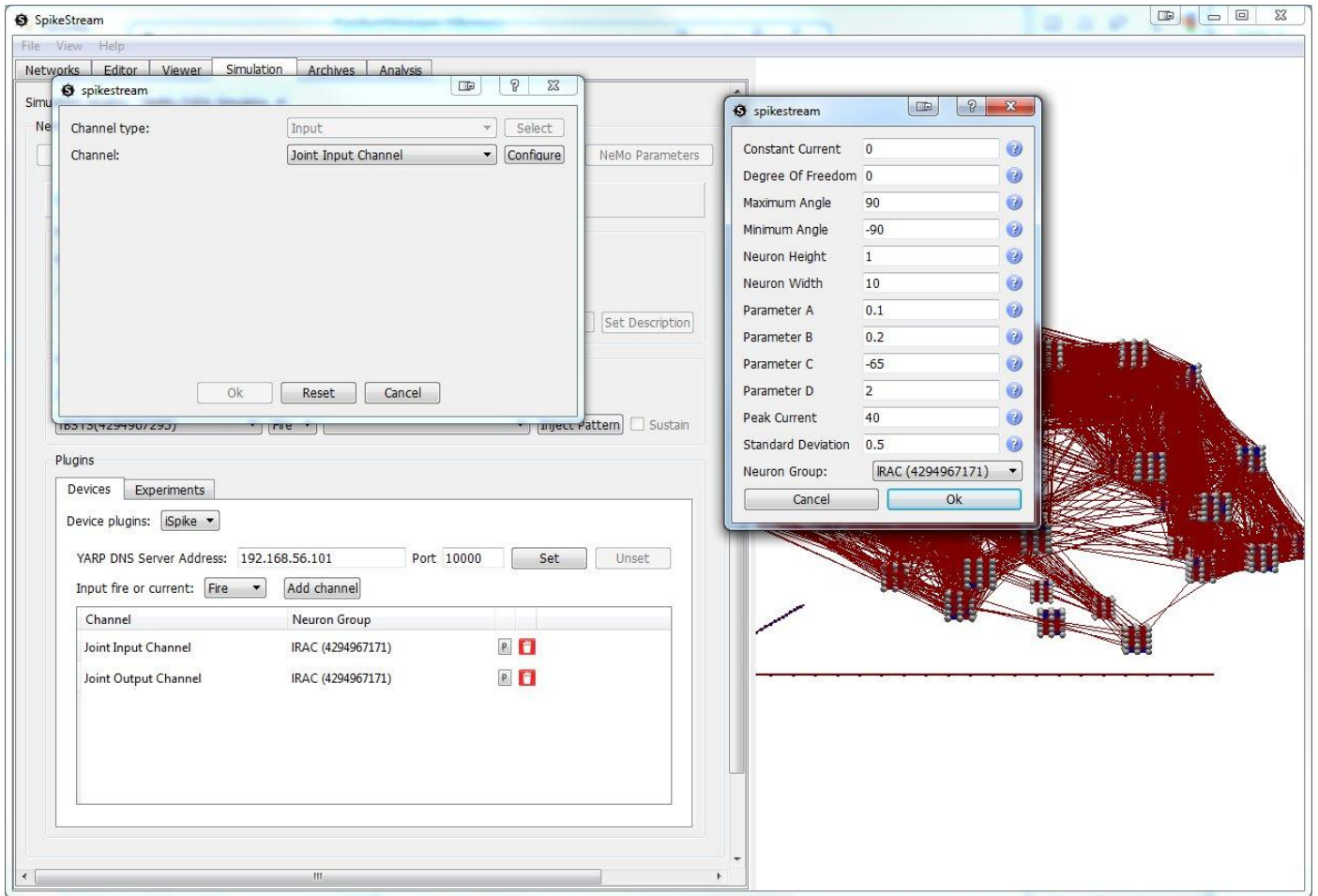
Figure 34. iSpike wrapper showing dialog to create channel and dialog to set parameters for the channel.

More information about iSpike and the channel parameters can be found here: http://ispike.sourceforge.net.

8.6 Analysis Plugins

8.6.1 Liveliness Analyzer

The Liveliness Analyzer (see Figure 35) analyzes the currently loaded network for liveliness, which is an alternative measure of information integration. The original documentation on liveliness can be found in Aleksander (1973) and Aleksander and Atlas (1973) and some recent work in this area is available in Aleksander and Gamez (2009), Gamez and Aleksander (2009) and Aleksander and Gamez (2010).

In earlier work on liveliness, the liveliness was averaged over all states of the network, whereas the Liveliness Analyzer plugin calculates the liveliness of the network for a particular state. Liveliness in this context is the probability that a particular connection will influence the firing state of the neuron that it is connected to at the next time step. This probability is 0 or 1 in a deterministic system. The liveliness of a neuron is the sum of the liveliness of the connections to the neuron, which provides a measure of the amount of information that is integrated by the neuron at that time step.

Neurons connected by lively connections at a particular time step form a *cluster*. The liveliness of a cluster is given by Equation 1.

$$\lambda_c = \lambda_{tot} \frac{\lambda_{tot}}{n^2} \ , \tag{1}$$

40

where λ_{tot} is the sum of the livelinesses of the neurons in the cluster, and n is the number of neurons in the cluster. n^2 is the maximum possible liveliness of the cluster – a situation in which all of the neurons in the cluster are connected with maximally lively connections.
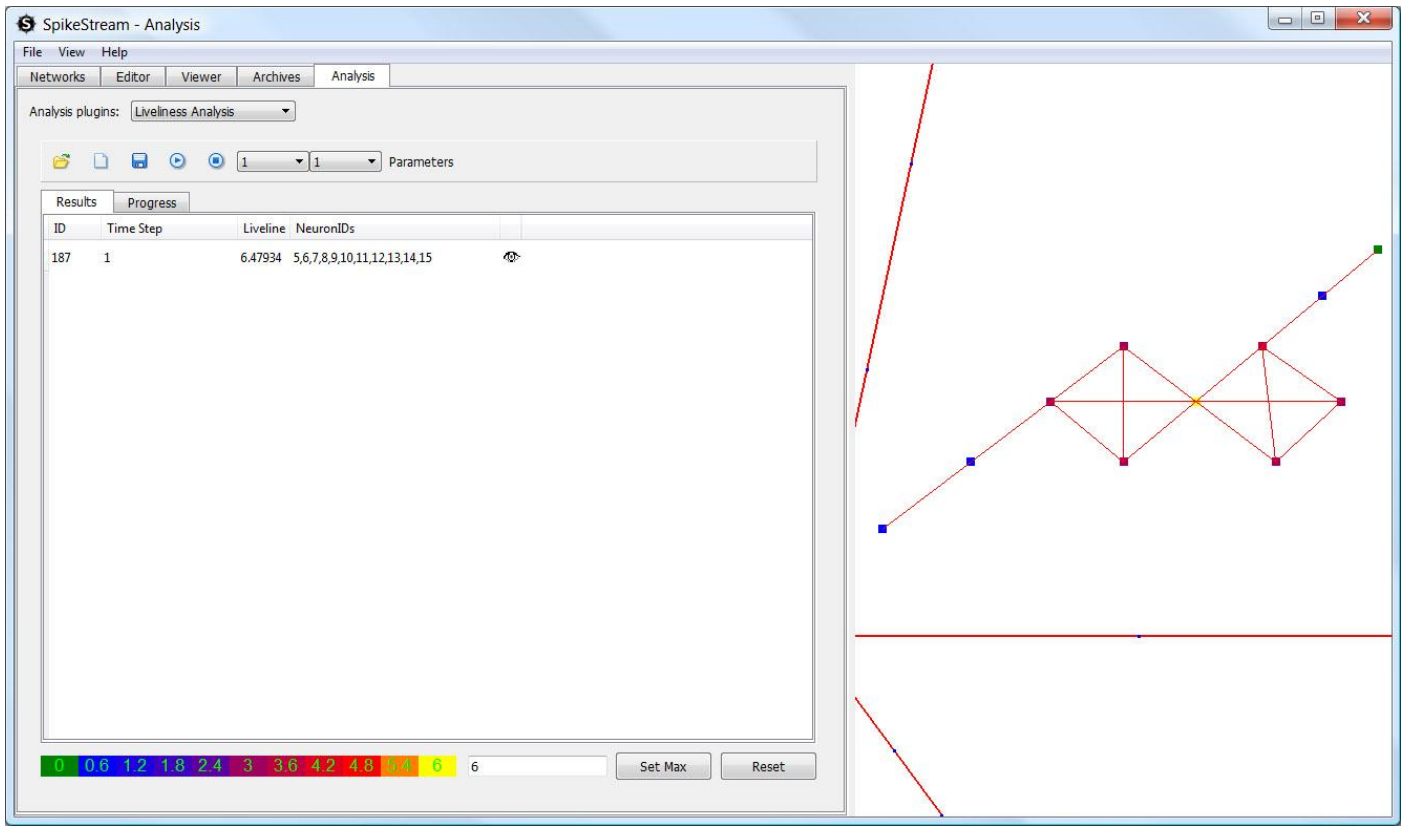
Figure 35. Liveliness Analyzer

The liveliness of each neuron is displayed as a heat map, and the scale of the heat map is displayed at the bottom of the Liveliness Analyzer plugin. This scale starts at zero and the default maximum is the maximum liveliness of any neuron in the current analysis. This maximum can be changed by entering a number next to the "Set Max" button and clicking on "Set Max". Changing the maximum value of the scale enables comparison between heat maps from different analyses.

The Liveliness Analyzer has a toolbar with the following controls:

📂 - Opens an existing analysis

📄 - Creates a new analysis

▶ - Starts the analysis running

⏹ - Stops the analysis running.

💾 - Launches a dialog that enables you to save the analysis as a tab-separated text file.

The drop down combo boxes enable the selection of the time steps from the archive that are going to be analyzed. A single time step can be analyzed, or a range of time steps. These controls are only enabled when a network and an archive are loaded.

The "Parameters" button launches a dialog that enables certain parameters of the analysis to be set:

- *Analysis description*. A description of the analysis.

- *Number of simultaneous threads*. Each time step is analyzed using a separate processing thread. This parameter sets the maximum number of threads that run simultaneously.

- *Generalization*. Sets the generalization of the weightless neurons in the network. See Aleksander (2005) for more information about this parameter.

- *Store_connection_liveliness_as_temporary_weights*. As the analysis runs the liveliness of each connection is stored as the temporary weight of that connection. These weights can be viewed using the Viewer Tab. The temporary weights are stored by each thread running for each time step, so there will be an interleaving of the temporary weights unless the analysis is only run for a single time step. This parameter is only useful if you are only analyzing a single time step and it should be disabled for large analyses.

Most of these parameters are not editable after an analysis has been run and its results stored.

The liveliness analyzer has two tabs. The "Progress" tab displays the progress of each time step that is running. The Results Tab lists the clusters that have been found. Clicking the eye symbol displays the heat map of the cluster in the 3D Network Viewer, as shown on the right side of Figure 24.

NOTE: The current version of the Liveliness Analyzer only works with weightless neurons.

8.6.2 State-based Phi Analyzer

The State-based Phi Analyzer (see Figure 36) carries out the analysis of the currently loaded network for information integration using the algorithm described in Balduzzi and Tononi (2008).
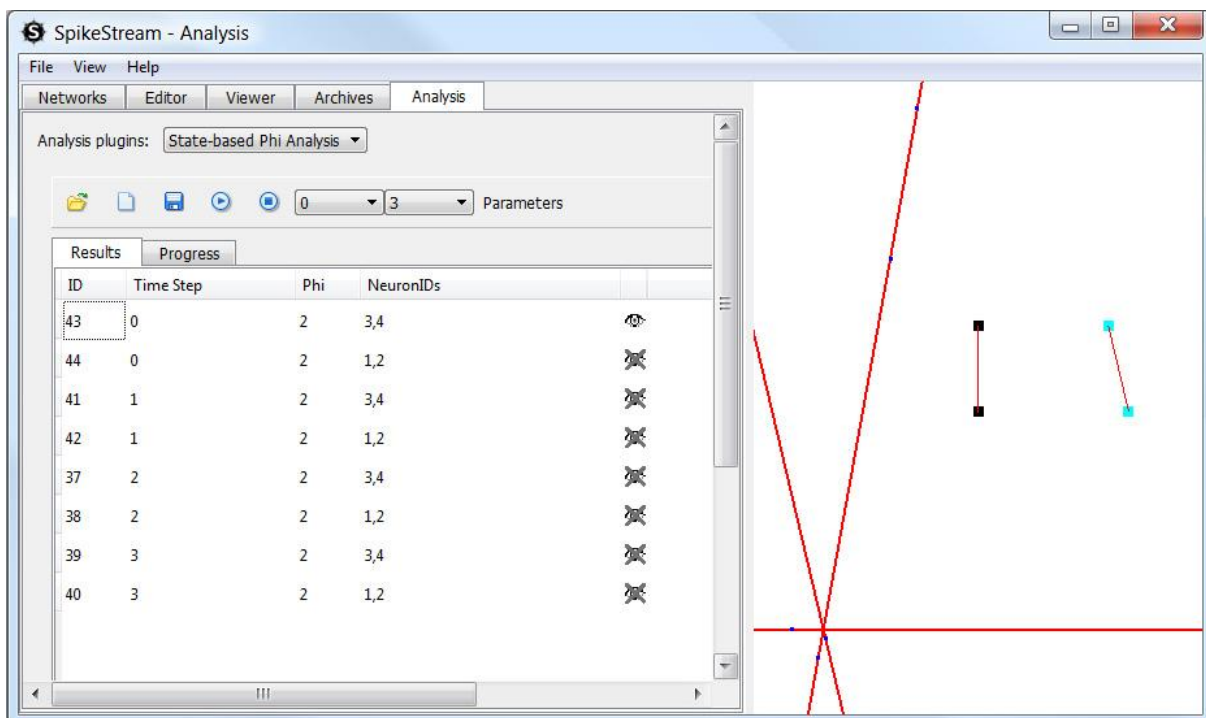
Figure 36. State-based Phi Analyzer Plugin

The State-based Phi Analyzer has a toolbar with the following controls:

📂 - Opens an existing analysis

🗋 - Creates a new analysis

⊙ - Starts the analysis running

⊙ - Stops the analysis running.

💾 - Launches a dialog that enables you to save the analysis as a tab-separated text file.

The drop down combo boxes enable the selection of the time steps from the archive that are going to be analyzed. A single time step can be analyzed, or a range of time steps. These controls are only enabled when a network and an archive are loaded.

The "Parameters" button launches a dialog that enables certain parameters of the analysis to be set:

- *Analysis description*. A description of the analysis.

- *Number of simultaneous threads*. Each time step is analyzed using a separate processing thread. This parameter sets the maximum number of threads that are running at a particular point in time.

- *Generalization*. Sets the generalization of the weightless neurons in the network. See Aleksander (2005) for more information about this parameter.

- *Ignore_disconnected_subsets*. A subset will have zero ϕ if it contains isolated neurons that are not connected to any other neuron in the subset. When this parameter is set to 1 these subsets are excluded at an early stage of the analysis.

- *minimum_complex_phi*. Many networks have a large number of meaningless complexes with low values of ϕ. This parameter enables the user to filter out complexes with ϕ less than the specified value. Complexes whose phi is greater than or equal to *minimum_complex_phi* will be included in the final results. The default setting is 1.0.

Many of these parameters are not editable when the analysis has been run and its results stored.

The State-based Phi Analyzer has two tabs. The "Progress" tab displays the progress of each time step that is running. The Results Tab lists the complexes that have been found. Clicking on the eye symbol displays the highlighted complex in the 3D Network Viewer, as shown on the right hand side of Figure 25.

NOTE: The current version of the State-based Phi Analyzer only works with weightless neurons.

NOTE: The Balduzzi and Tononi (2008) algorithm has factorial dependencies and takes an extremely long time to run on large networks. A graph showing the measured and estimated performance on randomly connected networks of increasing size is given in Figure 37.
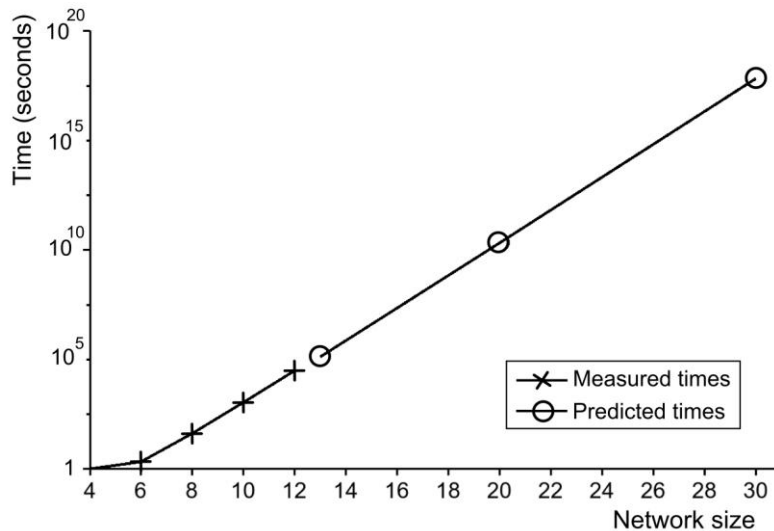
Figure. 37. Measured and predicted times for the calculation of information integration on different sizes of network using Balduzzi and Tononi's (2008) algorithm and a Pentium IV 3.2 GHz single core computer. Each neuron in the network was randomly connected to five other neurons and their truth tables had five entries. The results are for the analysis of a single time step with a random firing pattern.

8.7 Writing SpikeStream Plugins

Plugins are written in C++ and they must extend the Qt QWidget class. Each plugin should be built as a library with the extension .dll, dylib or .so, depending on the platform. This library must be placed in the appropriate folder, depending on the type of plugin. These folders are listed in Section 8.1.

Each library must implement two external C functions, getName() and getClass(). getName()returns a unique QString describing the library; getClass() returns a class that inherits from QWidget and forms the main display widget of the plugin. For example, the State-based Phi Plugin implements these two functions as follows:

```
//Functions for dynamic library loading
extern "C" {
    /*! Creates a StateBasedPhiWidget class when library is dynamically loaded. */
    StateBasedPhiWidget* getClass(){
        return new StateBasedPhiWidget();
    }


    /*! Returns a sensible name for this widget */
    QString getName(){
        return QString("State-based Phi Analysis");
    }
}
```

spikestreamapplicationlibrary contains a number of abstract classes that can be used to develop analysis plugins. More documentation will follow soon.

The SQL for each plugin should be added to the appropriate part of the database folder. For example, SQL for the SpikeStreamNetwork database should be placed in database/network/plugins, and it should be provided in both standard and test versions. This SQL will be executed by the SpikeStream Database Configuration Tool after the database(s) have been added.

The best way of understanding how to write a plugin is to look at the code for the current plugins.

9. DATABASES

SpikeStream is based around three MySQL databases and cannot run without them. These databases are used to store different types of information:

- **SpikeStreamNetwork**. Holds information about the networks, neurons and connections.

- **SpikeStreamArchive**. Holds firing patterns of the network for each time step.

- **SpikeStreamAnalysis**. Stores the results of analyses of the network.

These databases can be hosted anywhere in the world as long as the firewalls are configured correctly.

If you want to run the unit tests, then you will also need a matching set of three databases: **SpikeStreamNetworkTest**, **SpikeStreamArchiveTest** and **SpikeStreamAnalysisTest**.

More information about the structure of these databases can be found by looking at their SQL, which is in the database folder of the SpikeStream distribution. The SpikeStream databases can be manually installed by running this SQL.

SpikeStream comes with a database configuration tool that makes the setting up of the databases easier (see Section 5). Before running this tool, you need to install MySQL and configure it so that you know the username and password of an account that has enough privileges to create and modify databases. Instructions on how to do this on Windows, Mac OS X and Linux are given in section 2.2, 3.2 and 4.2.

The database host, username and password can be manually entered in the spikestream.config file, which is at the root of the SpikeStream installation (see Section 10).

10. CONFIGURATION

10.1 Introduction

The settings of SpikeStream are stored in the configuration file, "spikestream.config", which is at the root of the SpikeStream installation. Ignore the spikestream.config.template file, which holds the default settings.

SpikeStream ignores comment lines that start with the hash '#' character and blank lines. A configuration setting consists of a parameter, for example "spikeStreamArchiveHost" followed by and equals '=' sign, followed by the value of that parameter.

If you mess the configuration file up, you can copy the settings from the spikestream.config.template file or delete spikestream.config and run the SpikeStream Database Configuration Tool (see Section 5).

NOTE: SpikeStream needs to be restarted for changes in the configuration file to take effect.

10.2 Database Settings

The host, username and password of each database are stored in spikestream.config and the SpikeStream Database Configuration Tool (see Section 5) writes the settings that you enter to this file. The parameters are as follows:

- **spikeStreamNetworkHost**. Host of the SpikeStreamNetwork database.

- **spikeStreamNetworkUser**. Username for the SpikeStreamNetwork database.

- **spikeStreamNetworkPassword**. Password for the SpikeStreamNetwork database.

- **spikeStreamArchiveHost**. Host of the SpikeStreamArchive database.

- **spikeStreamArchiveUser**. Username for the SpikeStreamArchive database.

- **spikeStreamArchivePassword**. Password for the SpikeStreamArchive database.

- **spikeStreamAnalysisHost**. Host of the SpikeStreamAnalysis database.

- **spikeStreamAnalysisUser**. Username for the SpikeStreamAnalysis database.

- **spikeStreamAnalysisPassword**. Password for the SpikeStreamAnalysis database.

10.3 Other Settings

Other settings available in the configuration file are:

- **default_file_location**. Default location for loading files etc.

- **vertex_size**. The size of neurons in the 3D Network Viewer when not in Full Render mode (see Section 7.4). It can be easier to see the colour of neurons if you increase the size.

- **draw_axes.** Shows or hides the axes.

- **maximize_gui**. Controls whether the graphical interface is launched in a maximized state.

- **sphere_radius**. The radius of the neurons in full render mode.

- **sphere_quality**. The quality of the neurons in full render mode. Low quality will result in more angular spheres.

- **connection_quality**. Quality of the double cones that are used to indicate the weight of connections in Full Render mode (see Section 7.4).

- **minimum_connection_radius**. Parameter for drawing the weights of the connections in Full Render mode.

- **weight_radius_factor**. Parameter for drawing the weights of the connections in Full Render mode.

- **connection_visibility_threshold_fast**. Controls the maximum total number of visible connections that are shown in standard render mode before all of the connection groups are hidden.

- **connection_visibility_threshold_full**. Controls the maximum total number of visible connections that are shown in full render mode before all of the connection groups are hidden.

- **connection_thinning_threshold_fast**. When the number of connections exceeds this threshold in standard render mode, only a selection of the connections are shown.

- **connection_thinning_threshold_full**. When the number of connections exceeds this threshold in full render mode, only a selection of the connections are shown.

- **number_insert_neuron_buffers**. Database optimization parameter. Recommended to leave it at its default setting.

- **number_insert_connection_buffers**. Database optimization parameter. Recommended to leave it at its default setting.

11. KEYBOARD SHORTCUTS

11.1 Network Viewer Navigation

These shortcuts may vary between operating systems – particularly on Mac OS X.

Arrow Up: Move camera positively along the Z axis
Arrow Down: Move camera negatively along the Z axis
Arrow Left: Move camera negatively along the X axis
Arrow Right: Move camera positively along the X axis
Page Up: Move camera positively along the Y axis.
Page Down: Move camera negatively along the Y axis

CTRL + Arrow Right: Rotate camera anticlockwise around Z axis
CTRL + Arrow Left: Rotate camera clockwise around Z axis
CTRL + Arrow Up: Rotate camera clockwise around X axis
CTRL + Arrow Down: Rotate camera anticlockwise around X axis.

NOTE: In its initial state, the 3D Network Viewer is organized with positive Z axis pointing upwards, the positive Y axis pointing away from the viewer and the positive X axis moving left to right. Clockwise and anticlockwise are from the point of view of looking down the axis towards zero. The easiest way of learning how to navigate is try out the keys and observe their effect.

11.2 Other Shortcuts

CTRL + R: Resets the view to its initial position.

CTRL + M: Shows dialog for importing NRM network.

F1: Show Networks Tab

F2: Show Editor Tab

F3: Show Viewer Tab

F4: Show Simulation Tab

F5: Show Archives Tab

F6: Show Analysis Tab

REFERENCES

Aleksander, I. (1973). Random Logic Nets: Stability and Adaptation. *International Journal of Man-Machine Studies* 5: 115-31.

Aleksander, I. (2005). *The World in My Mind, My Mind in the World: Key Mechanisms of Consciousness in People, Animals and Machines.* Exeter: Imprint Academic.

Aleksander, I. and Atlas, P. (1973). Cyclic Activity in Nature: Causes of Stability. *International Journal of Neuroscience* 6: 45-50.

Aleksander, I. and Gamez, D. (2009). Iconic Training and Effective Information: Evaluating Meaning in Discrete Neural Networks. *Biologically Inspired Cognitive Architectures II. Papers from the AAAI Fall Symposium.* AAAI Technical Report FS-09-01, pp. 2-10.

Aleksander, I. and Gamez, D. (2010). Informational Theories of Consciousness: A Review and Extension. Submitted to BICS 2010.

Balduzzi, D. and Tononi, G. (2008). Integrated information in discrete dynamical systems: motivation and theoretical framework. *PLoS Comput. Biol.* 4(6).

Gamez, D. and Aleksander, I. (2009). Taking a Mental Stance Towards Artificial Systems. *Biologically Inspired Cognitive Architectures II. Papers from the AAAI Fall Symposium*. AAAI Technical Report FS-09-01, pp. 56-61.

Izhikevich, E.M. (2003). Simple Model of Spiking Neurons. *IEEE Transactions on Neural Networks* 14:1569-1572.

Izhikevich, E. M. (2006). Polychronization: computation with spikes. *Neural Computation* 18(2): 245-282.