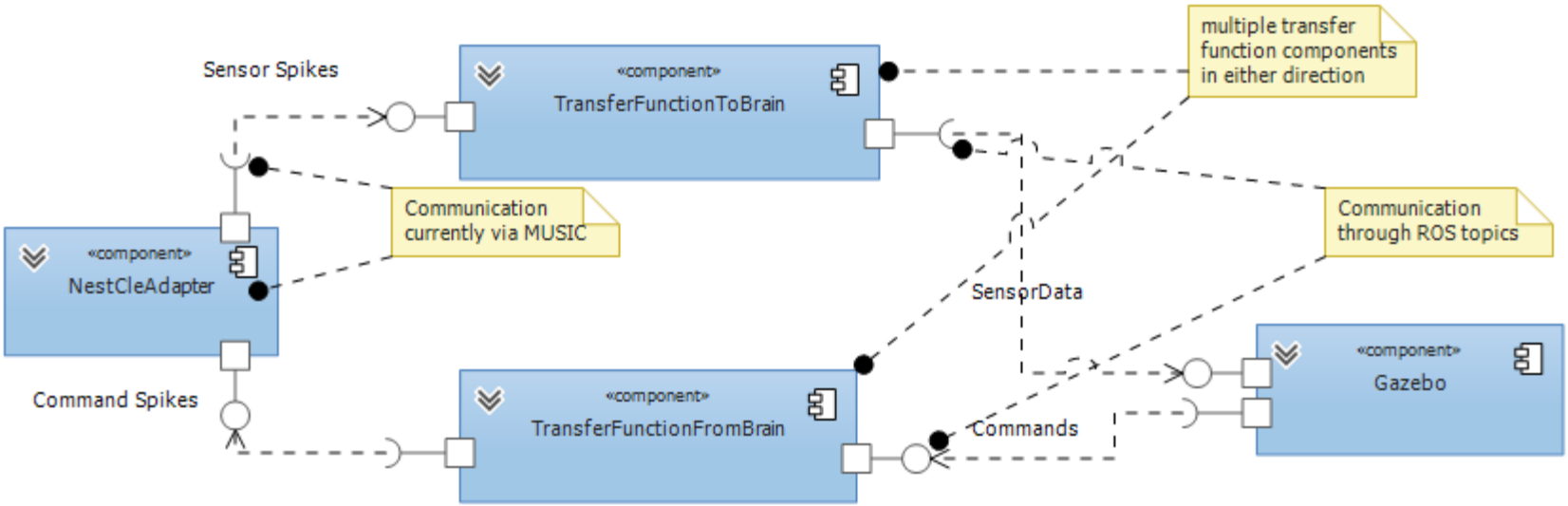
Transfer Functions Architecture

HBP – SP10 Neurobotics

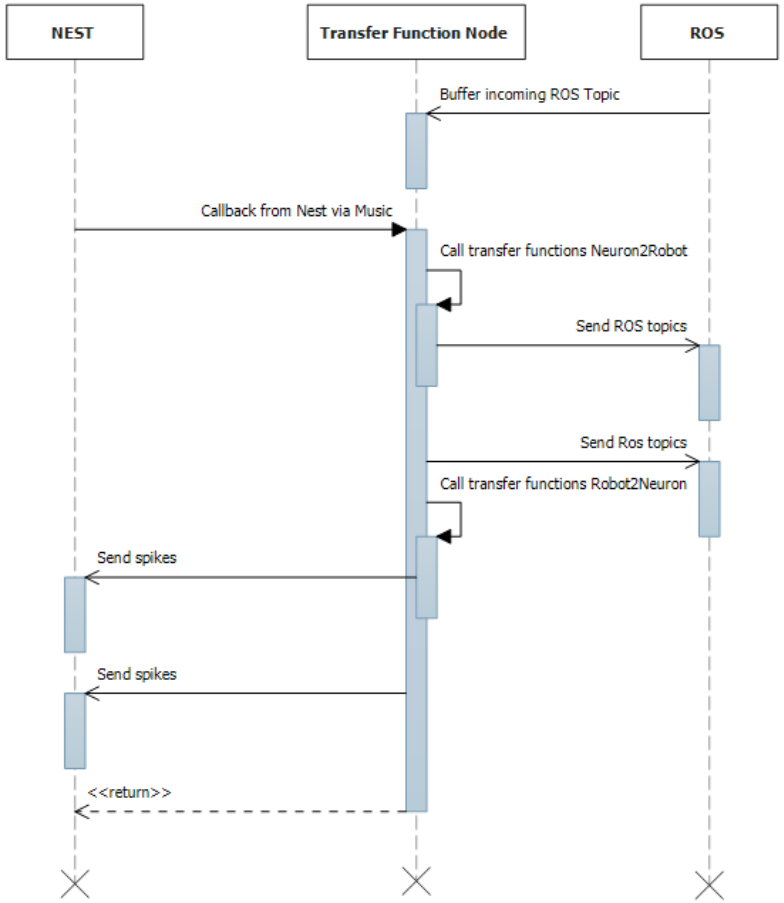
Requirements

- Neurons occasionally send spikes
  - Transfer into continuous robot messages
- Robot continuously sends messages
  - Transfer into spikes for set of neurons
  - No assumptions on spike generation patterns
  - Support for common spike generation patterns such as fixed frequency or Poisson-based
- Transfer functions in both directions may hold a state
- Specification of Transfer Functions in Python or C++

Architecture Overview



Iteration of a TF node



Python Prototype

```
import NeuroboticsFramework as nrp
from husky import Husky

right_arm_v = 0


@nrp.Neuron2Robot(Husky.RightArm.pose)
def right_arm(t, neuron0):
    global right_arm_v
    if neuron0:
        right_arm_v = 1
    else:
        right_arm_v /= 2
    return right_arm_v


@nrp.MapNeuronParameter("neuron2", nrp.spikes("neuron2", 10))
@nrp.Neuron2Robot(Husky.LeftArm.twist)
def left_arm_tw(t, neuron1, neuron2):
    if neuron1:
        if neuron2[0]:
            return 0
        else:
            return 1
    else:
        if neuron2[1]:
            return 0.75
        else:
            return 0.25
```

```
camera_spike_generator = nrp.CameraSpikeGenerator(200, 300)

@nrp.MapRobotParameter("camera", Husky.Eye.camera)
@nrp.Robot2Neuron(nrp.spikes("spike45", 200000))
def transform_camera(t, camera):
    global camera_spike_generator
    if camera.changed:
        camera_spike_generator.updateImage(camera.value)
    return camera_spike_generator.tick(t)


if __name__ == "__main__":
    nrp.initialize()
```

C++ Prototype

```
/*
 * MyTransferFunctions.cpp
 *
 *  Created on: 11.08.2014
 *      Author: GeorgHinkel
 */

#include "NeuroboticsFramework.h"
#include "husky.h"
#include <string>

#include "ComplexImageProcessing.h"
#include "ObstacleDetection.h"

using namespace Neurobotics;

double left_arm_v = 0;
bool* spike1;
ImageSpikeGenerator camera_spike_generator(300, 200);


double moveLeftArm(const int t, const bool spike0) {
	if (spike0) {
		left_arm_v = 1;
	}else {
		left_arm_v /= 2;
	}
	return left_arm_v;
}
```

```
double moveRightArmMuscles(const int t, const bool* right_arm_spike) {
	if (spike1) {
		if (right_arm_spike[1]) {
			return 1;
		}else {
			return 0;
		}
	}else {
		if (right_arm_spike[2]) {
			return 0.75;
		}else {
			return 0.25;
		}
	}
}


bool* sendCameraData(const int t, const CacheValue<char*> sensor) {
	if (sensor.Changed) {
		camera_spike_generator.updateImage(sensor.Value);
	}
	return camera_spike_generator.tick(t);
}


int main(int argc, char** argv) {

	spike1 = RegisterReadNeuron("spike1");

	TransferNeuronToRobot("spike0", husky::left_arm::elbow::pose, moveLeftArm);
	TransferNeuronsToRobot("right_arm_spike", husky::right_arm::muscles,
moveRightArmMuscles, 10);

	TransferRobotToNeurons(husky::eye::camera, "eye_spikes", sendCameraData, 2000);

	StartNode("MyTransferFunctions", argc, argv);
}
```

API Proposal

Purpose	Python	C++
Register a transfer function to transform spikes into messages for the robot	Neuron2Robot decorator	TransferNeuronsToRobot, TransferNeuronToRobot
Register a transfer function to transform robot messages into spikes	Robot2Neuron decorator	TransferRobotToNeuron, TransferRobotToNeurons
Send spikes to a neuron	sendSpikes	Via object obtained from RegisterWriteNeuron(s)
Send a message to the robot	sendRobot	Via object obtained from RegisterWriteTopic

API Proposal

Purpose	Python	C++
Register that a transfer function will read spikes from a neuron	Inferred by the parameter name of a function, or explicit MapNeuronParameter annotation	One neuron with the registration, others via RegisterReadNeuron
Register that a transfer function will generate spikes to a neuron	Parameters of Robot2Neuron annotation, first neuron(s) is/are connected to function return value	One neuron with registration, others via RegisterWriteNeuron, RegisterWriteNeurons
Register that a transfer function will read robot topics	MapRobotParameter annotation	One robot topic with the registration, others via RegisterReadTopic
Register that a transfer function will write robot topics	Parameters of Neuron2Robot annotation, first topic is connected to return value	One topic with registration, others via RegisterWriteTopic

Limitations

- Robot interface description generated from ROS information model of the robot
  - Husky.py or husky.h
- Technical interface to Nest as exchangable component

Python prototype output

- <function right_arm at 0x2081af0> transfers to robot /husky1/joint325/pose : float () using [spike0(1)]
- <function left_arm_tw at 0x2081e20> transfers to robot /husky1/leftArm/twist : float () using [spike1(1), spike2(10)]
- <function transform_camera at 0x20e4490> transfers to robot spike45(200000) () using [/husky1/sensors/camera1 : list]