

# Computational Steering Revisited

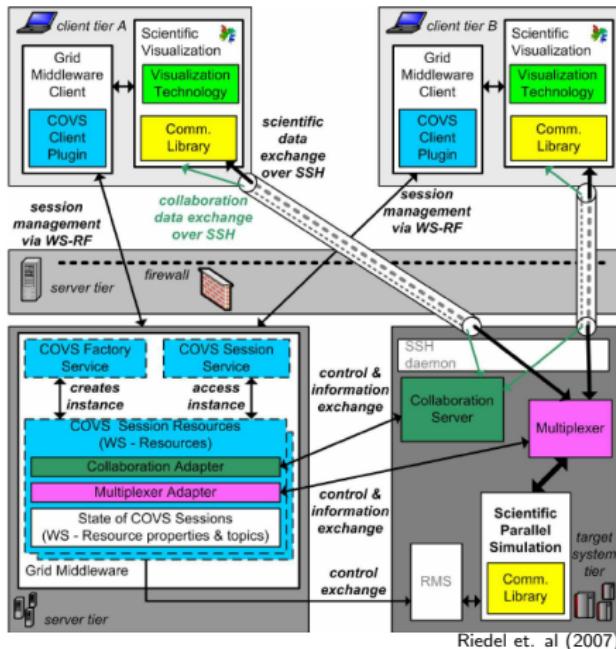
2016-06-23 | Björn Hagemeier, Christian Böttcher, Bernd Schuller

## Outline

- Computational Steering
  - COVS, UGSF
- Current Goal
  - HBP SP5 (Neurorobotics)
- Implementation
  - Bad approach
  - Good approach
- Results
- Open Issues
- Summary

# History

## Collaborative Visualization and Steering – COVS

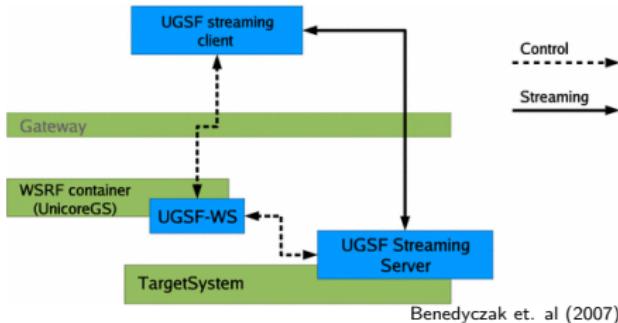


- SSH tunnels
- Collaboration
- Application instrumentation

Riedel et. al (2007)

# History

## UniGrids Streaming Framework – UGSF

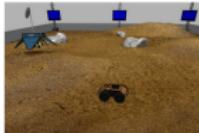


- Protocol-specific plugins
- and connection tunneling

# Current Goals

## Neurorobotics

### Workspace



### Example experiments

#### Husky Braatenberg experiment in the SpaceBotCup 2013 arena

This experiment loads the Husky robot from Clearpath Robotics and the arena from the SpaceBotCup 2013. If the user starts the experiment, the Braatenberg vehicle network is executed and the robot will...



#### Husky Braatenberg experiment with automatically switching screens

This experiment is similar to the Husky Braatenberg one (Husky robot detecting red colour and driving towards it). In this experiment the right screen is turned red automatically after...



#### iCub Visual Tracking experiment

In this experiment the iCub robot performs a Visual Tracking task in the virtual room environment.

Timeout: 00 00:14:00

Backend availability: 13 / 14

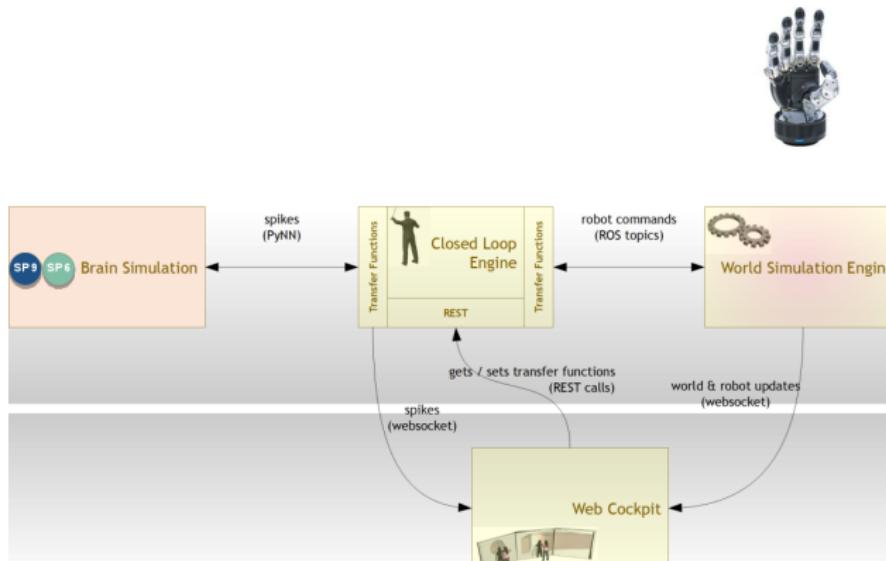
Cluster availability: 6 / 34

+ Launch

Upload custom environment

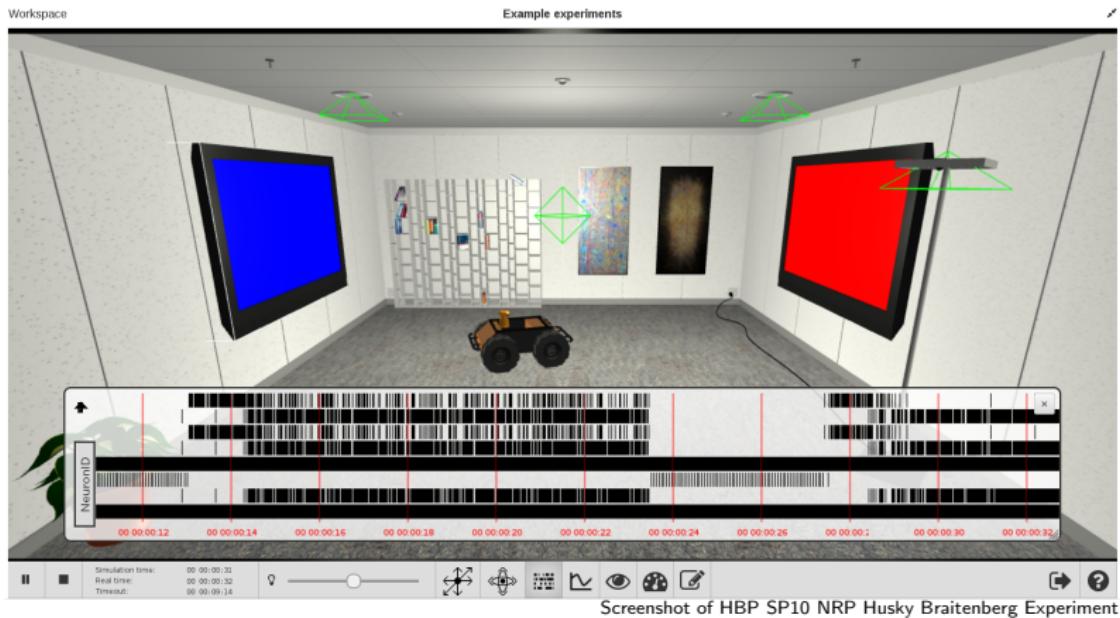
Screenshot of HBP SP10 NRP Example Experiments

# NRP Architecture



Axel von Arnim, HBP SP10

# NRP Web Cockpit

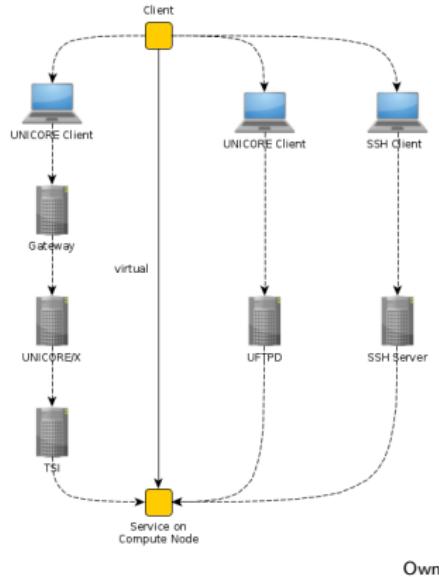


## Requirements

- Network tunnel into HPC resources
- Forwarding VNC connections
- Low latency
- Bandwidth?

# Architecture Options

- Many hops
- Higher latency expected
- Implementation difficult

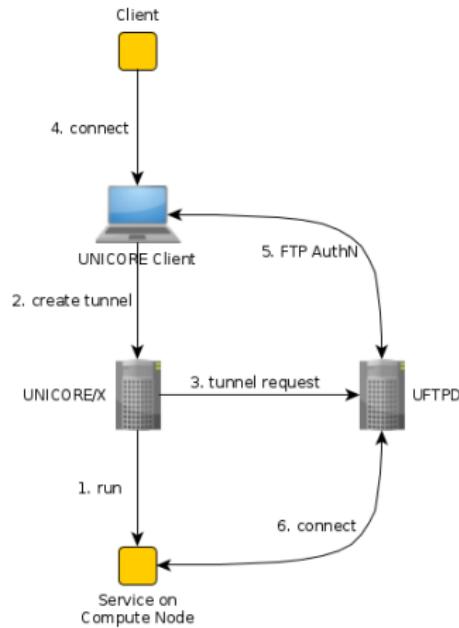


- Fewer hops
- Proven components
- Easier implementation

# Implementation

## Tunnel Setup

- Same flow as for *ordinary* UFTP connections
- Firewall friendly



# Tunnel Setup

## UCC

- New `create-tunnel` command
- Parameter `-L` resembling SSH syntax
- Source Address can be set via `-i`
  - Important for NATing or multiple interfaces
- Example:

```
$ ucc run iperf.u -a  
720208ac-7015-49c3-b0c1-9e047b4287be.job  
$ ucc create-tunnel 720208ac-*.job -L 5001:localhost:5001  
Listening on localhost:5001
```

## Results

### Some (superficial) measurements

#### Throughput/Bandwidth

- Measured using iperf (submitted as job)
- No noticeable difference to native performance or SSH tunnels

#### Latency

- Usable for VNC, no noticeable lag

# Results

## IPerf

- 6 consecutive tests, 2 each using UFTP, SSH, and direct connections
- Server side log

---

```
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
```

```
[ 5] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 41246
[ ID] Interval Transfer Bandwidth
[ 5] 0.0-10.8 sec 120 MBytes 93.5 Mbits/sec
[ 7] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 41252
[ 7] 0.0-10.7 sec 119 MBytes 93.4 Mbits/sec
[ 5] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 41253
[ 5] 0.0-11.0 sec 119 MBytes 91.3 Mbits/sec
[ 7] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 41254
[ 7] 0.0-11.0 sec 123 MBytes 93.5 Mbits/sec
[ 5] local 192.168.16.142 port 5001 connected with 134.94.168.32 port 52196
[ 5] 0.0-10.1 sec 112 MBytes 93.6 Mbits/sec
[ 7] local 192.168.16.142 port 5001 connected with 134.94.168.32 port 52200
[ 7] 0.0-10.1 sec 112 MBytes 93.6 Mbits/sec
```

# Open Issues

## The small print

- Need to deploy full NRP scenario including VirtualGL over VNC
- Currently only a single connection possible after tunnel setup
- Tunnel tear-down needs to be implemented
- It is not as straightforward to implement SSH tunnels as it may seem
- More measurements under various conditions
  - Better abstraction is needed

## Summary

- Extended UFTP-D to support network tunneling
- Promising results, comparable to other solutions
- A number of open issues to solve