A Gridified Artificial Neural Network Resource

Erich Schikuta and Peter Beran
University of Vienna
Department of Computer Science and Business Informatics
Rathausstr. 19/9, A-1010 Vienna, Austria
{erich.schikuta, peter.beran}@univie.ac.at

Abstract

We present N2Grid, a system for the usage of neural network resources on a world-wide basis. The approach employs the infrastructure of the Grid as a transparent environment to allow users the exchange of information (neural network resources, as neural network objects and neural network paradigms) and exploit the available computing resources for neural network specific tasks leading to a Grid based, world-wide distributed, neural network knowledge and simulation system. Our system uses only standard protocols and services aiming for a wide dissemination of this Grid application.

1 Introduction

The driving stimulus for development is the exchange of information and resources between researchers. This principle is just as valid for the neural information processing community as for any other research community.

As described by the UK e-Science initiative [16] several goals can be reached by the usage of new stimulating techniques, as enabling more effective and seamless collaboration of dispersed communities, both scientific and commercial, enable large-scale applications comprising of 10,000 computers, large-scale pipelines etc, transparent access to "high-end" resources from the desktop, provide an uniform "look & feel" to a wide range of resources, and location independence of computational resources as well as data.

A Grid [7] based computational infrastructure couples a wide variety of geographically distributed computational resources (such as PCs, workstations, and supercomputers), storage systems, databases, libraries, and special purpose scientific instruments, and presents them as a unified integrated resource which can be shared transparently by communities (virtual organizations).

The Grid started out as a means for sharing resources and was mainly focusing high performance computing. By the integration of Web Services as an inherent part of the Grid infrastructure the focus evolved to the sharing of knowledge to enable collaborations between different virtual organizations or subjects.

In the "Computational Intelligence" community these current developments are not used to the maximal possible extent. As justification of our hypothesis we want to put a light on the situation of artificial neural network simulators. In the last few years many different systems were proposed, but only a few got accepted by the community. There are systems developed specifically for certain network paradigms, as SOM-PAK [15], and some to deliver a comprehensive environment, as SNNS [19].

Basically all these systems, reaching from highly sophisticated interactive systems to programming language extensions, share the same common problems:

- A clumsy software environment, which mostly do not present an intuitive interface to the user.
- Most of these systems present a *proprietary system*, which is not capable to interconnect to other software systems.
- All these systems *lack a generalized framework* for handling data sets and neural networks homogenously.

These problems lead to the situation that large number of rarely used simulation systems exist, because most scientists, scared of existing programs, develop their own systems for their specific neural network applications. We believe that this situation is one of the reasons of an obstructed open information and data exchange within the Computational Intelligence community.

We see a solution to this problem by the N2Grid system [14].

N2Grid is an artificial neural network simulation environment and provides basic neural network functions like creating, training and evaluating neural networks. The system is Grid-based in order to make it open to the growing

1082-3409/07 $25.00 © 2007 IEEE
DOI 10.1109/ICTAI.2007.152

Internet/Grid community. The simulator interacts with Grid data resources (as databases) to store and retrieve all relevant data about the static and dynamic components of neural network objects, and with Grid computing resources to harness free processing cycles for the "power-hungry" neural network simulations. Further the system allows to integrate additional paradigms found on the Grid, provided by arbitrary users, into the simulator. In a former project the NeuroWeb system [13], a much simpler Internet based neural network simulator, was developed, which passes on several proven design principles to N2Grid.

The layout of the paper is as follows. In the following section we give a motivation for the development of such a system and describe the goals we are aiming for. This is followed by an overview of the N2Grid architecture, describing the design principles and justifying them. In Section 3 we present the dynamic service evolution technique, which enables a highly dynamic client-server communication in an open world scenario. Next we present a working implementation describing the use cases followed by a detailed section on the execution workflow. The paper closes with a look at future developments and research directions.

2 N2Grid Architecture

The N2Grid system is a neural network simulator using the Grid infrastructure as deploying and running environment. It is an evolution of the existing NeuroWeb [13] and NeuroAccess [2] systems. The idea of these systems was, to see all components of an artificial neural network as data objects in a database. Now we extend this approach by identifying them as resources of the world-wide Grid infrastructure.

Accordingly to the definition of the notion of "information" of Gundry we propose a layered Grid architecture based on the dimensionality of information in focus which allows to differentiate three different Grid layers:

- *Data Grid, 0-dimensional*. The Data Grid builds the basis layer and stores data which represent just facts.
- *Information Grid, 1-dimensional*. The Information Grid collects data of the Data Grid in a structured manner and attributes it with semantic contents.
- *Knowledge Grid, 2-dimensional*. The Knowledge Grid provides problem solution mechanisms on the administered information allowing a human for acting, deciding or planning.

In this architecture each layer (starting from the data layer) provides its functionality to the next layer in form of specific services.

N2Grid is based on a service oriented architecture and spans all three layers of the Grid layer architecture defined in Section 1. In Table 1 we give the mapping of the specific N2Grid services to the Grid layer architecture.

Table 1. N2Grid component mapping

Knowledge Grid	N2Grid Paradigm Service N2Grid Java Application/Applet N2GPort Web Portal
Information Grid	N2Grid Paradigm Archive Service Resource Broker, Replica Manager
Data Grid	N2Grid Simulation Service N2Grid Data Services

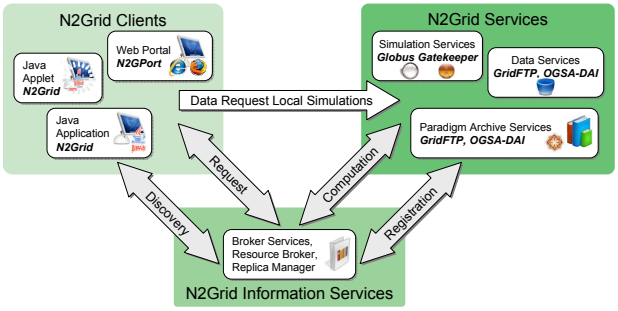

Figure 1. N2Grid application model

Figure 1 shows the overall application model of the N2Grid system in the final implementation phase. We assume a sophisticated Grid infrastructure, including independent resource brokers and replica manager. A client does not care about the execution hosts or data sources. It can control a local simulation or a simulation in the Grid infrastructure in a transparent way. Parts of the remote execution are the authentication to the Grid resources, the processing of the job descriptions and the usage of Grid data sources. For local simulation runs the client can also use local arbitrary data sources.

The components of N2Grid represent a novel, service oriented, tripartite Grid application. *N2Grid Services* realize Simulation Services, Paradigm Archive Services and Data Services. The end user can choose between three different *N2Grid Clients*. These are respectively a Java application with the possibility of direct database connections, a Java Applet and a Web portal. N2Grid services and clients are interconnected by standard Grid broker services, using standard Grid information services.

2.1 Grid Infrastructure

The N2Grid system is based on the common middleware Globus [5], and some extensions of Globus especially the DataGrid project [4] and the "Open Grid Services Architecture Data Access and Integration" (OGSA-DAI) project [3].

As independent and transparent service the *resource broker* accepts simulation jobs from the client. The job submission for the client is the same in the case of submitting it to a resource broker or directly to a simulation service. The N2Grid system uses the DataGrid [4] resource broker. Thus for huge data files, as training-, evaluation-, and propagation-data for artificial neural networks, the *replica manager* of the DataGrid project provides an optimized data file access.

The *replica manager* implements also a N2Grid key component, the N2Grid *Paradigm Service*. It gives the clients the possibility to find N2Grid paradigm archive services providing paradigm implementations. The services establish the mapping between logical paradigm names and the physical location and enables the user to search for different types of paradigms (e.g. Back propagation, Quick propagation, Jordan, ART, etc.).

2.2 N2Grid Services

N2Grid Services are Grid Services hosted by the Grid infrastructure. They execute neural network simulation tasks (as creation, training and evaluation), which are submitted to the Grid; so they do not consume local client resources. By the submission the workload is minimized on a local machine.

We can run the following tasks of neural network simulation remotely in the Grid:

1. Training of neural networks
2. Evaluation of neural networks
3. Processing of data by a neural network
4. Archiving of paradigms
5. Archiving of network objects (nodes, structures, weights, etc.)
6. Using data sources and archives

Task 1, 2 and 3 are integrated into one component of the N2Grid system, the *Simulation Service*, which accomplishes the training, evaluation and propagation function of the neural network simulator. A selected paradigm and network instantiation is executed on a Globus Gatekeeper [10]. The necessary data are provided by other N2Grid services, described below.

Task 4 is implemented as a N2Grid *Paradigm Archive Service*. As mentioned in the previous subsection, the users can find paradigm archive services over the replica manager. Paradigms are implemented as Java classes using the Java Commodity Grid (CoG) Kit [17] and are transferred by the GridFTP protocol.

Task 5 and 6 are unified by the N2Grid *Data Services*. OGSA-DAI provides the access to a database storing all training-, evaluation-, propagation-data and of course network objects (nodes, structures, weights, etc.).

2.3 N2Grid Clients

We propose three different clients as shown in Figure 1.

First, there exists a N2Grid *Java Application* client for an advanced, high end user, who can run also a local database storing his own data.

Second, we provide a N2Grid *Java Applet* having a similar user interface as the Java application but with limited functionality. Because of the sandbox principle local database and file accesses are not allowed. The user can use the built-in client functions (paradigms) and shared functions of the N2Grid system.

Third, for the purpose of thin clients a simple web browser can be used as a front end of the N2Grid system by accessing a Web portal called *N2GPort*. It provides control over running simulation jobs on the N2Grid Services and presents their results. All the functions are implemented by Java Server Pages (JSP) and Java Servlets.

3 Dynamic Client - Server communication

A service oriented architecture provides interoperability by defining the interfaces independent of the protocol, according to [8]. Further service semantics are necessary for Grid Services. This means, that service interactions needs not only an agreement over an interface but also over semantics and meaning. A main problem for using neural networks in a Grid infrastructure is that no standard exists for describing neural networks (problem domain, semantics).

In WSRF [11] the semantic data of a resource is described by a XML Schema inside a WSDL interface. However it does not enable dynamic services concerning the semantics, because semantics is a description about the service.

As for N2Grid Services also for other WSRF Services, restricting specifications of possible semantic values is contra productive. Moreover a client needs to get and implement the semantic information of the service in a dynamic way to be Grid aware. A common language (standard) is needed, to get an agreement over the semantics (service description) of a service.

Only a service and its developer know about the proper semantics. Therefore the service itself must describe its capability and semantics, functions and parameters in a client interpretable format, e.g. using a GUI Meta Description Format in XML Schema. A client can instantiate service data in a dynamic way by using this GUI description. The format of the dynamic service data can be defined in a separate service data schema, as WSRF provides by a Resource Property XML Schema [6].

By an agreement over two XML Schemas, respectively, firstly the client schema to describe the service semantics and secondly the service data schema to describe the service data, the semantics of a service can change without alterations in the client implementation. Therefore we have a more powerful, dynamic way to deal with services in a Grid. The XML schema pair builds together a common language.

We call this approach "Dynamic Service Evolution" (DSE) [18] because of two reasons.

Firstly, the service can change the semantics dynamically and the appropriate semantical description passes through an evolutionary process. In this case no adaptations are necessary on client side.

Secondly, also the language can go through an evolutionary process. A big advantage of our approach is that no strict standardization of a general and powerful semantic language is necessary. A flexible pair of two schemas defines a new language. Therefore, different languages can be built depending on the problem domain.

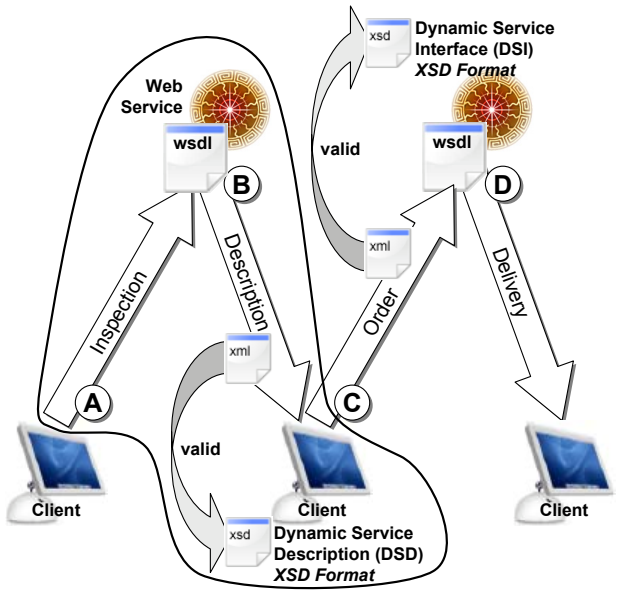

Figure 2. Dynamic Service Evolution (DSE)

Figure 2 shows the whole process with the following steps, while the new components for the novel approach compared to common Web Services are depicted by a bubble.

A. The client contacts a service found via a registry to get detailed semantic information about the service. This information is a service description beyond the pure interface description. We name the used format the "Dynamic Service Description" (DSD).

B. The service sends back the semantics by representing it in a valid client format. This respond can be processes automatically or by a GUI representation to allow user interactions.

C. The client produces out of the user input or the automated processing valid service data to get a proper service instance. This service instance can represent resources or other stateful services. We name the service data combined with the interface definition "Dynamic Service Interface" (DSI).

D. In this step the service delivers a service instance (resource) or other processing results to the client.

We applied our dynamic services approach in N2Grid for the interaction between the client and the *N2Grid Simulation Service*, which is a proof-of-concept implementation for our novel approach. We need dynamism because of the lack of a general neural network language. The semantics can not be defined strictly.

The implementation is based on a Web Service architecture and uses WSDL for the interface definition. We use the Apache Axis [1] library together with an Apache Tomcat Web container to run our services.

The service publishes its semantic description in the registry (information service), which can also be used to establish flexibility for the client. Thereupon the client can search in the registry for a specific property to find an adequate simulation service.

The header of the XML Schema to describe the semantics of a N2Grid simulation service is listed below. The N2Grid simulation services can use this schema in a proper way, for example to publish the possible parameters and available training methods of the implemented neural network algorithm. Later, after further developments of the service, the description can change dynamically.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="n2grid-types.xsd"/>
  <xs:element name="NNServiceDESCRIPTION">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PARADIGM" type="xs:string"/>
        <xs:element name="DESCRIPTION" type="xs:string"/>
        <xs:element name="TRAINSERVICE" type="xs:anyURI"/>
        <xs:element name="EVALUATIONSERVICE"
                    type="xs:anyURI"/>
        <xs:element name="STRUCTURE">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="INPUT" type="BLOCKTYPE"/>
              <xs:element name="MAXHIDDENBLOCKS"
                          type="SIZEMAX"/>
...
```

An example of a concrete N2Grid simulation service description is shown in the following listing. We get information on the available neural network structure and other characteristics of the N2Grid simulation service by the XML document. Based on the information of this document the client can produce dynamically a GUI for user interactions, allowing the user to define a specific neural network.

The GUI can be created out of our service description, but we can also apply a standard GUI language to describe our service, as e.g. XUL [12] from the Mozilla project.

```
<NNServiceDESCRIPTION xmlns=""
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="nns-description.xsd">
  <PARADIGM>Backpropagation</PARADIGM>
  <DESCRIPTION>Multi layer Back-Prop NN</DESCRIPTION>
  <TRAINSERVICE>
     http://cs.univie.ac.at/trainservice
  </TRAINSERVICE>
  <EVALUATIONSERVICE>
     http://cs.univie.ac.at/evalservice
  </EVALUATIONSERVICE>
  <STRUCTURE>
    <INPUT>
      <ID>input1</ID>
      <DIMMIN>1</DIMMIN>
      <DIMMAX>1</DIMMAX>
      <SIZEMIN>1</SIZEMIN>
      <SIZEMAX>unbounded</SIZEMAX>
    </INPUT>
    <MAXHIDDENBLOCKS>unbounded</MAXHIDDENBLOCKS>
...
```

The definition of a specific neural network is submitted to the N2Grid simulation service by the second XML Schema. This schema defines the service data for OGSI, Resource Properties for WSRF or any other service instance data depending on the service implementation. The following listing shows an example XML Schema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="n2grid-types.xsd"/>
  <xs:element name="NNDEFINITION">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NNSERVICEID" type="xs:string"/>
        <xs:element name="PARADIGM" type="xs:string"/>
        <xs:element name="DESCRIPTION" type="xs:string"/>
        <xs:element name="STRUCTURE">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="INPUT" type="BLOCK"/>
              <xs:element name="HIDDEN" type="BLOCK"
                          minOccurs="0"
                          maxOccurs="unbounded"/>
...
```

The two listed XML Schemas define a common language used in our system. A service-client pair has to agree on one schema pair. We learned that the possible dynamism is much more powerful than the usage of ordinary service data only, because of the following reasons:

- A second schema gives the service the possibility to change the semantics inside the service without adaptation on the client side.
- By decoupling two parts of one language, only a smaller part of the system has to be changed or extended in the cases of changes in one schema, or introduction of a new schema.
- The client can implement and interpret different semantic schemas and map them to one common service interface.

4 Use Cases – Scenarios

For the N2Grid system we propose several use cases, depending on the state of the dynamic and changing Grid infrastructure. Table 2 shows the categorization of the scenarios according to the Grid layers.

Table 2. Use case scenario mapping

Knowledge Grid	Search Paradigm
	Search Net Object
	Create Neural System
Information Grid	Directed Remote Exec.
	Data-driven Remote Exec.
	Computation-driven Remote Exec.
	Paradigm-driven Remote Exec.
	Stand-alone Local Exec.
Data Grid	Data Pull (GET)
	Data Push (PUT)
	Paradigm Pull (GET)
	Net object Pull (GET)
	Net object Push (PUT)

4.1 Knowledge Grid – Scenarios

4.1.1 Search Paradigm

The *Search Paradigm* scenario allows users to find a paradigm by the paradigm services (replica manager) in a paradigm archive (N2Grid Paradigm Archive Service). The available neural network paradigms are categorized and mapped to paradigm archive services offering corresponding implementations.

The user can locate the physical location of a paradigm by the paradigm archive service directory (OGSA-DAI database). Depending on its properties one of the remote execution scenarios (see *Directed Remote Exec., Data-driven Remote Exec., Computation-driven Remote Exec., Paradigm-driven Remote Exec.*) can be chosen.

This way of searching for an available paradigm is difficult and only possible if it is exactly known what to search for. Normally a neural network researcher is searching for a solution to a given problem. In this case the above described scenario is not very useful.

Therefore work is on the way to define a totally new interface for the search of appropriate neural network paradigms for specific problems. This approach is based on a pattern oriented classification of available paradigms on the Grid. We followed the vision of Alexander, who sees patterns as "A recurring solution to a common problem in a given context and system of forces". First, all of N2Grid available paradigms and, in a second stage, neural

objects are classified and described by the following structure, derived from design patterns [9]: Name and Classification, Intent, Also-known-as, Motivation, Applicability, Structure, Participants, Implementations, Example-Code, Known-uses, Related-patters, and so on.

4.1.2 Search Net Object

A specific neural network is defined by its paradigm and its describing attributes like the number/type of nodes, structures or weights of connections. Trained neural network objects are solutions to specific problems. Therefore an ever increasing number of neural network objects, each representing a specific problem solution, will be stored by N2Grid *Data Services* and, in turn, will be available as high level problem solver to the users.

For the search, description and the exchange of neural network objects we defined an xML-based Neural Network Definition Language, which we called xNNDL.

4.1.3 Create Neural System

For many applications in the area of knowledge representation and management a single neural network is not enough to fulfill the task. In these cases it is often necessary to build neural systems consisting of a number of, possibly, different neural networks interconnected by data streams, where the output of one net is the input of another network.

We developed an extension of xNNDL for the description of such systems. The language allows to describe the layout of the neural system consisting of net objects administered by N2Grid. This language provides appropriate hooks for the specification of the data streams between the networks.

4.2 Information Grid – Scenarios

4.2.1 Directed Remote Execution

If a client knows about the available remote execution hosts (Globus Gatekeeper) it can direct a simulation run directly by the *Directed Remote Execution*. No interaction with the resource broker is necessary.

4.2.2 Data-driven Remote Execution

The *Data-driven Remote Execution* scenario uses the resource broker to execute remote simulation runs. The client does not care about the used host machines. Depending on the location of the needed data the resource broker aligns the job on the computation nodes next to the data. This is an advantage for data intensive simulation runs, because transfer of data to other compute nodes is communication intensive and slow.

4.2.3 Computation-driven Remote Execution

In opposite to the data-driven remote execution the *Computation-driven Remote Execution* scenario applies another align policy. For less data intensive simulations the resource broker chooses free, power-full compute nodes. Some paradigms can also have parallel implementations. To execute these, the resource broker has to assign the jobs to front ends of clusters or other parallel architectures.

4.2.4 Paradigm-driven Remote Execution

In the *Paradigm-driven Remote Execution* scenario the resource broker and simulation client have to choose a defined host for the execution of the specific paradigm simulation. This is due to proprietary restrictions and copyrights of the paradigm, which allows the owner to sell the program with restricted access and execution rights. In turn this scenario gives the user the advantage of more economic access, because she/he has only to pay for the usage of the paradigm and not for the paradigm itself.

4.2.5 Stand-alone Local Execution

In the case of the *Stand-alone Local Execution* of the N2Grid client (Java Application or Applet) a comprehensive set of standard built-in functions and paradigms are available for local simulation runs. The user can define specific network instantiations of the built-in paradigms and execute them. At the time being the Backpropagation, Quickpropagation, Resilient propagation, Self organizing maps, Counter propagation, Hopfield net, ART 1/2, Kohonen net, Jordan net, Elman net, Cascade Correlation (with Quickpropagation) and CNN paradigms are supported by the N2Grid prototype.

4.3 Data Grid – Scenarios

4.3.1 Data Pull (GET)

Besides the neural network instantiations also the input data is stored in the Grid infrastructure. The N2Grid *Data Service* provides an interface to these data by OGSA-DAI to relational and XML databases or by GridFTP to simple files. The *Data Pull* scenario establishes the connection between a local simulation and remote input data source, which can provide training-, evaluation- or process (propagation) data.

4.3.2 Data Push (PUT)

In opposite to the *Data Pull* case the N2Grid *Data Service* can also be used to store and write output data into the Grid by the *Data Push (PUT)* scenario. A local client can save the neural network simulation results by the OGSA-DAI interface or directly to a file over GridFTP.

4.3.3 Paradigm Pull (GET)

As mentioned in Scenario *Search Paradigm* it is possible to download and install a paradigm locally on a client by the *Paradigm Pull (GET)* scenario, if the running configuration allows the download.

4.3.4 Net object Pull (GET)

Like Scenario *Paradigm Pull (GET)* also specific neural network object instantiations can be downloaded on a local client by the *Net object Pull (GET)* scenario.

4.3.5 Net object Push (PUT)

If a neural network simulation run has produced a trained network object of a given paradigm for a specific problem, it is possible to store this new network instantiation by the *Net object Push (PUT)* scenario.

5 The N2Grid Execution Workflow

In this section we will give an in-depth description of the N2Grid execution workflow of the actual implementation. During the execution of a N2Grid resource request several steps take place, which are depicted in Figure 3.

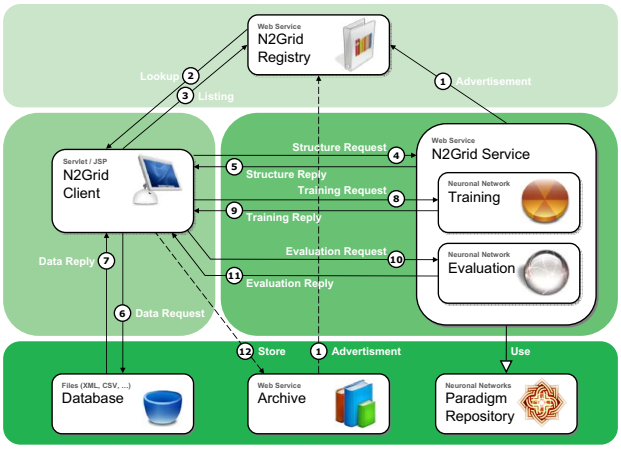

Figure 3. N2Grid execution workflow

1. *Advertisment:* The N2Grid Neural Net Web Service registers at N2Grid *Registry* by providing its self description (as laid out in Section 3) containing the *Paradigm*, *Description*, *Service Endpoints*, *Structure*, *Parameters* and *Data* information.

   Some archive services will later store trained neural nets, therefore they have to register at the N2Grid *Registry*.

2. *Lookup*: The N2Grid *Client* asks the N2Grid *Registry* for a list of available and registered Neural Net Services (see Figure 4). A lookup may contain some extra search criteria, like a paradigm or a layer (input, hidden, output) count, to allow the user to restrict the lookup range.

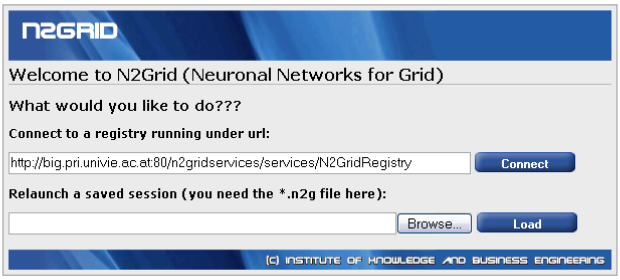

Figure 4. N2Grid lookup screen

3. *Listing*: Returns the acquired - through the *Lookup* step - list of N2Grid service endpoints to the user.

4. *Structure Request*: The client requests the service description for a particular *N2Grid Service*.

5. *Structure Reply*: The N2Grid Client retrieves the XML-based service description and parses it to dynamically generate the Web frontend and its user interaction fields like input fields, drop down lists, file upload fields and so on. By using this generated controls specifying a new neuronal net the client creates the neural net definition.

6. *Data Request*: The training data is requested from an external data source, possibly on the Grid.

7. *Data Reply*: Returns the training data.

8. *Training Request*: The client sends the service definition together with the training data to the *N2Grid Service* (see Figure 5).

   The service performs the training in an asynchronous way and returns a ticket, given by a requestId, to the client. The client has to use this ticket to poll the result of the *Training Request*.

9. *Training Reply*: After the training phase the trained net is returned as a XML document to the client, which presents the results within the Web browser (see Figure 6). This document contains two parts:

   - *service definition*: Well defined neural net as provided in step 8.
   - *neural net object*: A trained neural net instance serialized as a XML document.

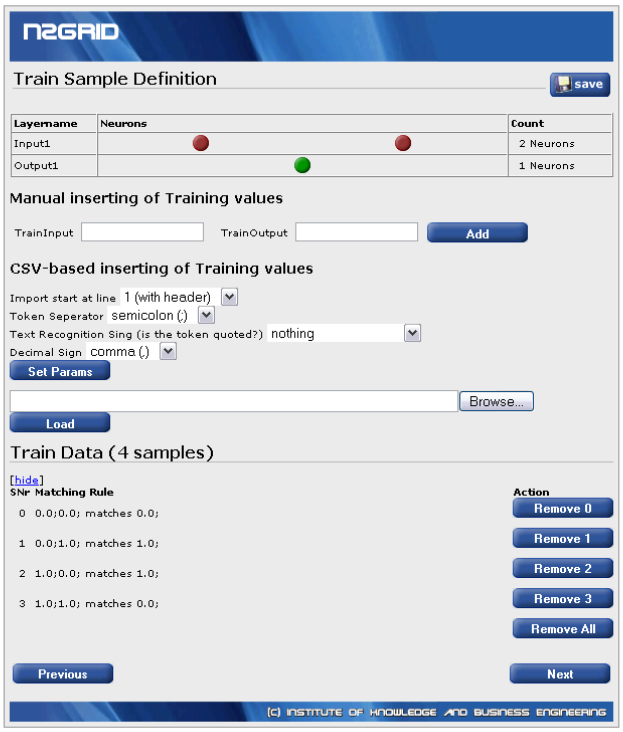

Figure 5. N2Grid training request

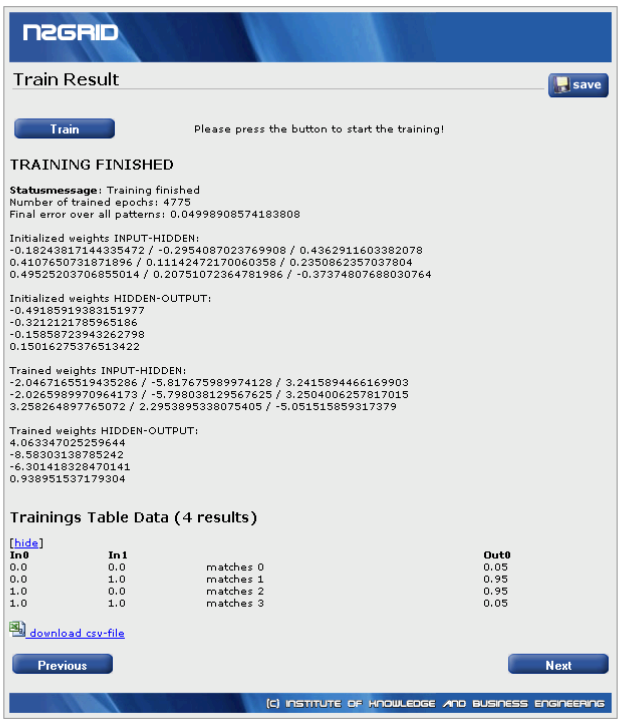

Figure 6. N2Grid training reply

10. *Evaluation Request*: In this step the trained neural net instance together with some simulation data is sent to the evaluation endpoint of the N2Grid Service. Like in the training phase the request is performed asynchronously and the user receives a ticket containing a requestId.

11. *Evaluation Reply*: The client uses the ticket to poll the service for an evaluation result of a previously submitted *Evaluation Request*.

12. *Store*: Enables the user to store a trained neural net instance.

A running N2Grid system can be accessed at `http://big.pri.univie.ac.at/n2grid/`.

6 Conclusion

We presented the N2Grid project as next step in the program evolution for neural network simulation. It is a framework for the usage of neural network resources on a worldwide basis assisted by the Grid infrastructure. Our system uses only standard protocols and services to allow a wide dissemination and acceptance.

The first final version of the N2Grid system is developed, however there is enough space for further extensions, specifically in two areas:

- The description of the paradigm has to be enhanced, to establish easier sharing between paradigm providers and customers.

- The actual N2Grid client controls single simulation runs. To allow the building of large connectionist systems consisting of several neural network instantiations (possibly of different paradigms) an extension of the N2Grid system is on the way.

At the moment a master thesis finishes which provides a new searching frontend to N2Grid, similar to Google, which allows to find a specific solution to a described problem. The user describe his/her problem in a quasi natural language form, by mapping of problem ontologies (built from a problem space with typical heuristic solutions approaches) to solution ontologies (built from known neural network solutions). As a result a set of possible neural network solutions found in the N2Grid archives is delivered.

7 Acknowledgment

On this occasion the authors want express their thank to Thomas Weishaeupl, Helmut Wanek, Stephan Wurm and Leander Krammer for their help in the design and implementation of N2Grid.

References

[1] Apache Project. *Apache WebServices Axis Project*, 2004. http://ws.apache.org/axis.

[2] C. Brunner and C. Schulte. NeuroAccess: The Neural Network Data Base System. Master's thesis, University of Vienna, Vienna, Austria, 1998.

[3] e-Science Grid Core Project. Open Grid Services Architecture Data Access and Integration OGSA-DAI. http://www.ogsa-dai.org.uk/.

[4] EU DataGrid Project. The DataGrid Project. http://www.edg.org/.

[5] I. Foster. The Globus Toolkit®: The Open Source Solution for Grid Computing. Keynote, GlobusWorld, San Diego, January 14 2003.

[6] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, I. Sedukhin, D. Snelling, T. Storey, W. Vambenepe, and S. Weerawarana. Modeling Stateful Resources with Web Services, May, 3 2004. Version 1.1.

[7] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[8] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[10] Globus. Globus Project. http://www.globus.org.

[11] Globus. WS-Resource Framework (WSRF), 2004. http://www.globus.org/wsrf.

[12] Mozilla.org. *XML User Interface Language (XUL)*, 2004.

[13] E. Schikuta. NeuroWeb: an Internet-based neural network simulator. In *14th IEEE International Conference on Tools with Artificial Intelligence*, pages 407–412, Washington D.C., November 2002. IEEE.

[14] E. Schikuta and T. Weishäupl. Artificial neural networks and the grid. In *International Conference on Computational Science*, pages 486–489, 2004.

[15] SOM Programming Team SOM-PAK. The self-organizing map program package, user guide, 1992.

[16] UK e-Science. UK e-Science programme. http://www.escience-grid.org.uk.

[17] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. A Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience*, 13(8-9):643–662, 2001.

[18] T. Weishäupl and E. Schikuta. Open language approach for dynamic service evolution. In *GCC Workshops*, pages 132–139, 2004.

[19] A. Zell, G. Mamier, M. Vogt, N. Mache, R. Hbner, S. Dring, K.-U. Herrmann, T. Soyez, M. Schmalzl, T. Sommer, A. Hatzigeorgiou, D. Posselt, T. Schreiner, B. Kett, G. Clemente, and J. Wieland. SNNS Stuttgart Neural Network Simulator user manual. Technical report, University of Stuttgart, 3/92 1992.