The Implementation of a Distributed Hierarchical Mind on the Internet using the World-Wide-Mind

Dave O'Connor[1] and Mark Humphrys[2]
World-Wide-Mind Research Group
Dublin City University, School of Computer Applications
Glasnevin, Dublin 9, Ireland
[1] doc@redbrick.dcu.ie, w2mind.org
[2] humphrys@compapp.dcu.ie, www.compapp.dcu.ie/~humphrys

ABSTRACT

In the standard Agents paradigm, the agent "mind" (or decision-making mechanism) interacts with some environment or "world". Using this terminology of "mind" and "world", the Internet has been used to date as a means of: (a) being able to *share worlds*, and: (b) being able to construct *multi-agent* systems.

In the "World-Wide-Mind" (WWM) scheme introduced in [3], the Internet is used as a means of: (c) being able to share *minds* and *parts of minds* for use as components in large *multi-mind* systems. This is *not* multi-agent systems, but rather a "society of mind" *within* a single agent. Each sub-mind is *not* free to take actions - only the agent as a whole can take an action. One of the main reasons for this scheme is to assist in the construction of large, complex minds by teams of multiple, dispersed authors. Similar to how the Web enables publishing, the WWM scheme is designed to make it as easy as possible for an agent mind author to "publish" - to put their mind online for re-use in other societies of mind.

This paper describes the first implementation of a multi-level mind using this system. A brief description of the architecture and protocol currently used is presented. This expands considerably on the previous implementation [10].

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence - *Intelligent agents*

Keywords

Agent architectures, society of mind, Internet, agent communication protocols, agent programming environments, action selection, goal selection.

1 INTRODUCTION

In the standard Agents model, the agent "mind" (or decision-making mechanism) interacts with (and makes decisions in) some environment or "world" (real or virtual). Using this terminology of "mind" and "world", the Internet has been used to date as a means of being able to share a common world to some degree.

1.1 The World-Wide-Mind

Under the "World-Wide-Mind" scheme [3, 4], the Internet is used as a means of being able to share *minds* and *parts of minds*, for use as components in larger minds. To be precise, it is proposed that agent "minds" (and "worlds") be constructed as *servers* on the Internet. Re-use is done not by installing the software, but rather by using a remote service. The World-Wide-Mind system defines a protocol and architecture for communication with minds and worlds. The World-Wide-Mind is a term used for the system itself, and the set of worlds and minds currently online. In the lowest-common-denominator scheme we propose, minds and worlds are available online as servers, with a CGI interface, running over HTTP. They communicate using a simple set of XML requests and responses [6].

A World server may be queried for the current state of the world, and sent an action to be executed in that world. A Mind server may be sent a world state, and asked to suggest an action to be executed in this state. A large number of other queries may be defined (see [3] for many detailed examples), but the principle is that once an agent mind or world has been constructed as a server in this way, then it is possible to construct large multiple-mind systems, components of which are written in different languages by different authors on different platforms. Such systems cannot be readily constructed at present, and we suggest that only *remote server* re-use can make this possible, not local installation.

1.2 Society of Mind, not Multi-Agent Systems

It is important to note that we are not talking about *multi-agent systems*, where each agent is free to take actions in the world as a separate entity. Rather we are talking about "society of mind" systems, where only the system as a whole can take an action. In other words, for many competing sub-minds suggesting actions, only one action can actually be taken and we have an *action selection* problem.

As an example, consider what we describe as an "Action Selection" or "Mind$_{AS}$ server". This, when asked by a client to suggest an action, talks to several *other* Mind servers, and uses an algorithm defined by its author to specify which of the competing minds to "obey", given a certain set of criteria. It then returns that mind's suggested action as its own suggestion. The sub-minds exist in a "society of mind" [5], where they cannot actually take actions, but only suggest actions to a higher level in the hope that they will be executed. To the outside world, the Mind$_{AS}$ server appears and functions as just another Mind server. It may itself be used remotely as just one sub-mind among many by *another*, even higher-level Mind$_{AS}$ server. We can then run the top-level Mind$_{AS}$ server as a single agent mind in an agent world, which is also online as a server.

1.3 Issues with current AI practice

In general, we address some fundamental issues with current AI practice:

1. **Unused work** - A lot of work in AI is put into the development of agents which are intended to act autonomously in some environment. Often, these works are not reused, or easily reusable, by other researchers or laboratories, and the functionality of the agent is never reused outside the place it was developed.
2. **Doing it all yourself v. Specialisation** - It seems obvious (and yet unaddressed) that as we scale up, the AI problem should become too large for any one researcher or laboratory to do it all themselves. Thus, it is proposed that the work may be distributed between entities, allowing individuals or laboratories to concentrate on a particular aspect, or part, of an overall agent.
3. **Doing good science - objective tests** - Unless the *same* test world is reused by many different agents, how can we objectively compare different solutions? This holds for test minds as well. In practice, many results presented at conferences refer to experiments that cannot easily be duplicated by anyone else. This makes it difficult to demonstrate that one system is better than another. This slows progress in the field. This point is strongly made in [1, 2].

1.4 Making it easy for any AI researcher to "publish"

One of the unique features of this scheme is to make it as easy as possible for agent authors to "publish" their minds and worlds online for re-use remotely. In this way, the WWM is best compared to the Web infrastructure, which made it easy to publish documents online. Ideally, for example, an AI researcher would be able to publish their algorithms for re-use online without having to learn any network programming *at all*. This is in contrast to most Agent models, which assume an interest in Networks. As a result, much interesting work in AI is not online, and may never be, unless a change in approach is taken.

It is this feature (of needing to appeal to as many authors as possible) that makes the WWM work so different to almost all Agents work. All technology involving a particular programming language (e.g. Java) or requiring the learning of particular programming skills (e.g. sockets) has to be rejected. Our focus is therefore on CGI, where the Mind or World author need only know how to repeatedly read plain text from stdin and write plain text to stdout. A stock CGI program, that can be downloaded from the World-Wide-Mind portal site [11], will perform the interaction with the client, so the author does not even need to know how to write CGI scripts.

Object-based distributed systems such as CORBA do allow for multiple platforms and languages, but again seem to assume an ability in network programming, which would rule them out here. Our approach is closer to the generic XML/HTTP protocol SOAP [9], and indeed we are considering recasting the WWM as an application of SOAP (provided this does not lead to increased complexity).

2 IMPLEMENTATION - GENERAL

Minds and worlds are available online as "servers". What this means in a technical sense is that each mind or world will have a URL, which points at the CGI script used to communicate with the mind or world, something like:

```
http://www.site.edu/cgi-bin/staff/myuser/wwm/mymind
```

In order to communicate with the mind or world, HTTP requests are made to these CGI scripts. The request is in a subset of XML that we call "AI Markup Language" (AIML), defined in [6]. The CGI scripts parse the XML request, process it, generate an XML response, and exit. HTTP and CGI were chosen for their ready availability (anyone who wants can get free web hosting, even for CGI scripts) and ease of use (no new programming language to learn). XML was chosen for its simple, plain text and above all else *extendable* nature. Future changes to the definition of the XML queries should not break old servers. A sample query would be as follows. The Mind server is queried for what action to perform, given that the world is in some state:

```
<request type="GetAction" runid="RUNID">
    <data name="x">
    world state
    </data>
</request>
```

It generates the XML response:

```
<response type="GetAction" runid="RUNID">
    <data name="a">
    suggested action
    </data>
</response>
```

For more queries see [3, 6]. The format of state and action is not defined in AIML but is defined by the world server. Different definitions will co-exist. XML will allow many elegant ways of converting between different definitions. For discussion of *state* conversions see [7], and *action* conversions see [8].

2.1 The client "browser" program

In the simplest case, a client makes the requests, and passes the information between the mind and world. The mind and world servers do not contact each other directly. In other words, a client application is needed to "run" a Mind server in some World server. This is essentially the "browser" for the WWM. Such a client is easy to write, or one could download it from the World-Wide-Mind portal site [11], where work is in progress on a number of clients. The normal invocation of the client would be (from the command-line):

```
$ client (world url) (mind url)
```

to start that Mind running in that World. The world would be repeatedly queried for state, the state sent to the mind to suggest an action, the action sent to the world for execution, and so on indefinitely. A graphical display would show what was happening, with a "Stop" button to end the run.

A server may call another server, and an example would be the case of the Mind$_{AS}$ server above, which will query several mind servers based on its query criteria. For further examples of servers calling other servers see [3]. In such cases, the calling server appears as just another client, and the normal client-server model is preserved.

2.2 How does a server call a server?

In line with our need to make it as easy as possible to write servers, we would imagine that the server author, to call another server, again needs *no* network programming ability, but can call the client "browser" with certain arguments:

```
$ client -onestep (url)
```

to send a *single* XML query to that URL and get the response, and then exit. The client here would read an XML query from stdin, send it to that URL, write the XML response to stdout, and exit.

2.3 Persistent or Non-persistent server programs

There is a distinction between **the CGI script** (which must terminate after each XML query) and the actual mind or world program (we call this **the WWM server**), which may remain persistent in between XML queries.

2.4 Lockfile example

In our first implementation, the WWM server is persistent, waiting for the next query to come in. When a CGI script is called, it writes the XML query to a file, alerts the persistent WWM server somehow, and then waits for an output file to be created. When the WWM server has written the XML response to the output file, it alerts the CGI script, which then outputs this file back to the client and exits. The WWM server continues on in memory, waiting for the next query. To avoid file conflicts, a simple system of lockfiles is used. There are obviously many other possible approaches to this, such as:

1. Non-persistent WWM server. It starts up for each XML query, initialises its data structures, processes the query, and then exits. In this case there need not be any distinction between the "CGI script" and the "WWM server" - we can have just one program.
2. Semaphores / Process suspend and wakeup.
3. Instantiating a program which contacts the WWM server (possibly via the network).
4. File sharing / Queue or Stack structures.

The lockfiles approach is perhaps the one demanding the *least* amount of knowledge from the server author. In our implementation, the programs create or delete a lockfile, based on what stage in the handshake the request was at (Table 1).

Time	WWM Server	CGI script
1	Write out world state to output file	Wait for lock file to exist...
2	Create lock file	
3	Wait for lock file to not exist.	Read world state from output file
4		Write Action to Input File
5		Delete Lock File
6	Read Action from Input File	
7	Execute Action in the World.	

Table 1: The World CGI script is obtaining the world state, and giving the world an action to execute. Both the CGI script and the WWM server use the same input and output file. "Input" and "Output" is from the point of view of the WWM server, so the CGI script writes to the "input file", and reads from the "output file".

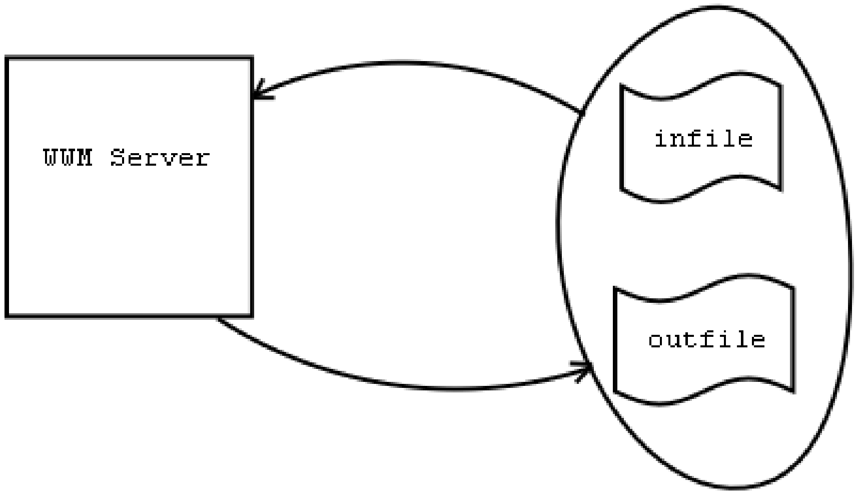

Figure 1: The WWM server does not interact with the CGI script directly, but only its own input/output files and lockfile. This is to simplify the process of writing new programs or converting existing programs to this system. A program need only write to normal files or lockfiles. It does not need any network components, and is doing only simple I/O.

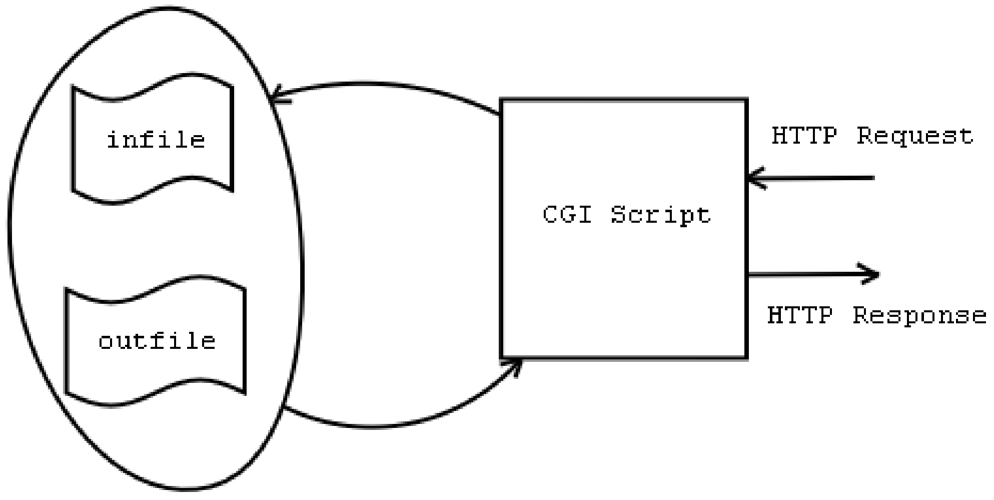

Figure 2: Only the CGI script needs a network component. The CGI script can be written by the agent author, or be one of the standard scripts downloadable from [11].

3 IMPLEMENTATION - MIND SERVER

As an initial implementation, 2 minds and 2 worlds were adapted to this system. Both worlds were simple "grid worlds" with a mobile agent, a "nest", several pieces of "food" and a mobile "predator". The agent is then given a "mind" in which to take decisions (make moves) in the world. The details of these "worlds" are not important here. What is important is that they were written by 2 different authors in different programming languages (C++ and Java), and are hosted on separate remote HTTP servers, with no agreement other than the format of the XML queries. First we show one mind interacting with one world (Figure 3). The client repeats the following for as many steps as required:

1. Client gets the world state from the World server.
2. Client sends the world state to the Mind server, and obtains the action suggested by it.
3. Client sends this action to the World server to be executed.

As well as the URL that XML requests should be sent to, the World server has a "display URL", which is an ordinary web page displaying what is happening in the world as actions are sent to it for execution. This is, of course, optional. Even if the server has no display URL, the client has the ability to output the world state as the world reports it.

3.1 Multiple languages (C++ and Java) proof-of-concept

Mind A (Java) was able to explore World B (C++), and Mind B (C++) was able to explore World A (Java). It makes no difference what language the mind or world is written in, provided each server generates the correct XML. We have not yet built a multiple-*mind* system where parts of the same mind are written in different languages by different authors, but we foresee no particular problems with this. It will still be encouraging when this is first done, though, particularly when we get the first authors from outside of our start group.

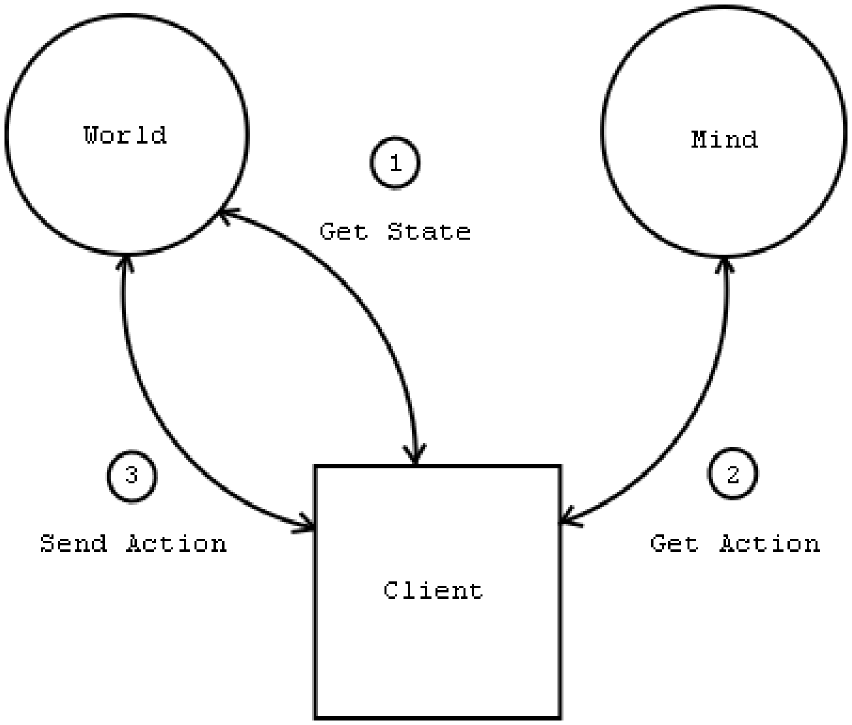

Figure 3: A Mind runs in a World.

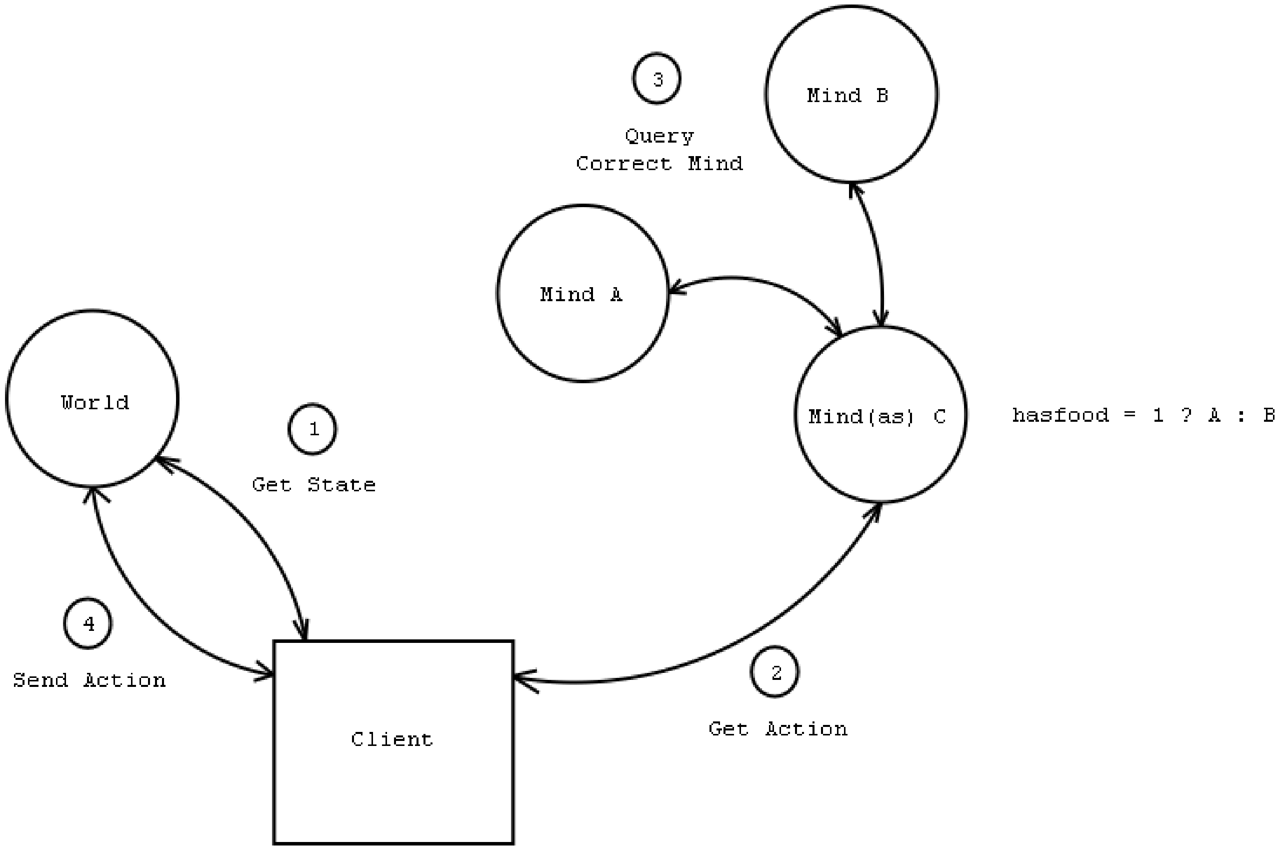

Figure 4: A Mind$_{AS}$ server runs in a World.

4 IMPLEMENTATION - MIND$_{AS}$ SERVER

The sample Mind$_{AS}$ server shown in Figure 4 is simply an abstraction of Mind A and Mind B. It uses its criterion to determine which Mind to "obey". It then returns the action suggested *by that Mind* to the client as *its* action. To A and B, the requests from the Mind$_{AS}$ server are exactly the same as requests direct from a client. In our case, the Mind$_{AS}$ server will obey Mind A if the "has food" parameter is true. Otherwise it will obey Mind B. The criteria for deciding which mind to obey can be basically anything. A server may implement any general-purpose algorithm to talk to other servers, provided that it itself responds to the XML queries expected of it.

5 DISCUSSION

The whole point of remote re-use is to enable massive re-use. Local installation of other people's AI projects involves compatibility problems with operating systems, versions, platforms, files, libraries, environments and programming languages - and as a result, generally does not happen. We view it as highly unlikely that local installation will lead to widespread re-use in this domain.

But by definition *remote* re-use has inherent speed limitations. It remains to be seen exactly which problems can be approached in a realistic time using this approach and which cannot. e.g. Problems where there is a small to moderate amount of communication, and a large amount of remote processing, should be the (large) niche in which this will work.

6 FUTURE WORK

One of the main current areas of work of the WWM project is the conversion of several well-known minds and worlds to this system, to provide an example, and perhaps an incentive for other researchers and laboratories to use this system. This will be a cumulative effort, becoming more and more useful as more servers go online.

Our other focus is on defining the WWM protocol itself - the XML queries and the CGI interface - in the simplest possible way so as to enable authors to put their work online. Our target audience is not so much the Agents community as the entire field of AI whose work cannot as of now be used remotely. We aim to change this situation, so that AI becomes a vast collective enterprise in which *no one individual* understands all of the components of the large distributed minds that are being built.

7 REFERENCES

1. J. Bryson. Cross-Paradigm Analysis of Autonomous Agent Architecture. *JETAI*, 12(2):165-189, Spring 2000.

2. J. Bryson, W. Lowe and L.A. Stein. Hypothesis Testing for Complex Agents. In *Proc. Workshop on Performance Metrics for Intelligent Systems*. NIST, August 2000.

3. M. Humphrys. *The World-Wide-Mind: Draft Proposal*. Dublin City University, School of Computer Applications, Technical Report CA-0301, February 2001. `www.compapp.dcu.ie/~humphrys/WWM`

4. M. Humphrys. Distributing a Mind on the Internet: The World-Wide-Mind. In *Proc. 6th European Conf. on Artificial Life (ECAL-01)*, Springer-Verlag LNCS/LNAI 2159, pages 669-680, September 2001.

5. M. Minsky. *The Society of Mind*. Simon and Schuster, New York, 1986.

6. D. O'Connor. *AIML - A Markup for Communication on the World-Wide Mind*. `wwm.compapp.dcu.ie/wwm/documentation/aiml.html`

7. D. O'Connor. *Subclassing of World State Data in Grid-like Worlds*. `wwm.compapp.dcu.ie/wwm/documentation/wwm_worldstate_markup.html`

8. D. O'Connor. *Representing the Action* a *in Grid-like Worlds*. `wwm.compapp.dcu.ie/wwm/documentation/wwm_action_markup.html`

9. SOAP resources. `dmoz.org/Computers/Programming/Internet/Web_Services/SOAP`

10. R. Walshe and M. Humphrys. First Implementation of the World-Wide-Mind. In *Proc. 6th European Conf. on Artificial Life (ECAL-01)*, Springer-Verlag LNCS/LNAI 2159, pages 714-718, September 2001.

11. The World-Wide-Mind project. `w2mind.org`