ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Jonathan Furner, School of Information and Media Studies, and David Harper, School of Computer and Mathematical Studies, The Robert Gordon University, Aberdeen, Scotland. (Eds)

Information Retrieval Research

Proceedings of the 19th Annual BCS-IRSG Colloquium on IR Research,
Aberdeen, Scotland, 8-9 April 1997

Paper:

A Community of Autonomous Agents for the Search and Distribution of Information in Networks

S. Marsh

Published in collaboration with the
British Computer Society

A Community of Autonomous Agents for the Search and Distribution of Information in Networks

A Preliminary Report

Stephen Marsh,

National Research Council Canada

Institute for Information Technology

Ottawa, Ontario. K1R 7E2

Tel. +1 613 993 8553 Fax. +1 613 952 7151

steve@ai.iit.nrc.ca

Abstract

The ACORN architecture is a multi-agent based information retrieval and provision system which uses the concept of 'document as agent' coupled with a community-based approach to information sharing. In the system, with the minimum of effort, users can expect to be provided with timely information on subjects in their areas of interest, whilst being able to search for information as they wish. The system, then, can be proactive in providing information, and both proactive and reactive in searching for information. This paper describes the ACORN architecture and discusses its current implementation.

1 Introduction

Information is valuable. But not in a vacuum. What is perhaps more valuable is getting relevant information to people who would be interested in it. In other words, it is worthwhile to make sure that information is routed to pertinent destinations in a timely fashion. The flip side of this coin is the age old question of relevance: can I be sure that what I get is what I want when I search? The World Wide Web (the Web) has sullied the playing field somewhat further. The amount of information on the Web is vast and increasing, and it is, or will soon be, impossible for a human to be able to process it all [Cowie & Lehnert, 1996].

The solution is not consumer pull of information, but neither is it exclusively producer push: without a

knowledge of who is interested in what a writer to say, the audience of the writer is artificially limited, and people who may well be able to use what the writer has to say will miss it. Both writer and potential, but lost, reader will lose out. In addition, there is nothing quite so annoying as being interrupted with spurious information: 'spams', etc., or even relevant information, when concentrating on a specific task at hand.

The system we are developing provides an agent based architecture using community based approaches for information retrieval and provision across networks. It is based on the premise that a mixture of consumer pull and producer push, coupled with a tight control of information spread, will allow people to keep up to date with topics, and also allow writers, the producers of information to get their information to those who will find it relevant in a timely fashion. The agents in the system are autonomous, able to make their own decisions about what to do based on information they receive from their creators and from the data they can glean via the community of other agents in the system. They are also easily controllable and configurable to prevent unwanted interruptions whilst maintaining sensible continued behaviours. The system is called ACORN: the Agent-based Community Oriented Retrieval Network, and is currently undergoing coding with the intention of a workable version in the near future.

This paper presents an overview of the ACORN architecture, comparing its benefits to those of similar systems where possible. In the next section, we discuss related work in the field of autonomous agents and information retrieval or dissemination. Following that, we present the ACORN architecture, discussing its component agents and their workings. Section 4 presents the current status of ACORN and possible evaluation techniques, and finally we conclude with a look at possible future work.

2 Autonomous Agents and Information

In this section, we briefly outline agent technology. We present the idea of an autonomous agent, and follow with a discussion of current technology which uses multi-agent systems, or autonomous agents, that have a similarity to ACORN, presenting some of the areas where they could be augmented to provide better service to their users.

2.1 Agents: What they are

Using autonomous agents to do our work for us is not a new phenomenon. With the arrival of the World Wide Web, however, agents seem to have taken off. Search 'agents' abound to make our web browsing easier and more efficient.

While there has been much discussion about what agents are, the autonomous agent can be seen simply as a piece of software that performs a task for its user. This task may be something as simple as filtering incoming email, or it may be more complicated, for example arranging meetings. In addition, an agent may be capable of performing more than one task. The agent may be static, and remain at a single site through its life, or it may be mobile, in that the code itself can migrate from site to site, performing some action at that site before moving on elsewhere. An example of a static 'agent' is the Firefly server (see <http://www.ffly.com/>) adapted from 'word of mouth' algorithms [Shardanand & Maes, 1995], which resides at a specific site, accepting people's details and

recommending movies and music to them. Systems such as ARACHNID [Menczer, 1997] use mobile agents to perform information retrieval on the Web. Each paradigm has its benefits: static, centralised agents are easier to control, and pose no real security risks by running unwanted on remote computers. Mobile agents, however, gain the power of distributed computing, graceful degradation, and potentially greater visibility for less work, but sacrifice the implicit trust others have in their static counterparts. The following systems use both kinds of agents.

2.2 Current Systems

In this section, we discuss some of ACORN's contemporaries, giving an idea of what they aim to do, and a brief discussion of how they could be made potentially more useful.

For a short time now, the idea of multi-agent matchmaking has been germinating in the minds of a few researchers. In particular, Lenny Foner, at MIT, has been working with British Telecom on the Yenta agents [Foner & Crabtree, 1996]. In this system, agents communicate with each other on a network to discover like minded agents and introduce each other to other such agents. The result is clusters of like minded agents which can then share relevant information amongst themselves. In addition, when a person is looking for someone (an expert) the clusters can use 'word of mouth' to direct the searchers to the proper agent to talk to. Whilst this is worthwhile and powerful, it lacks the fine granularity we would wish. In ACORN, all documents are agents, so that people are not only represented on different clusters, but can share information with completely unrelated people, also without knowing them in the first place. In addition, ACORN can act as a system for secure, private communication. The limits are set by the author of the information.

Also at BT, Davies et al [Davies, Weeks & Revett, 1996] are developing the Jasper system of intelligent agents for the Web. The Jasper system can summarise web pages, making recommendations to users with similar interests automatically. This is a useful tool, since the Web seems doomed to grow for the present, and in essence, this is part of what the ACORN SearchAgent does in its 'browsing' mode, but ACORN provides somewhat more, both with directed searches and with the dissemination of information by agents sent out by the author of that information.

Kuokka and Harada's [1995] matchmaking system is a centralised system which stores information, and can forward it to those seeking it. Again, ACORN subsumes this idea with that of the topic related archive and the topic related meeting place. In addition, because ACORN is distributed across a network, we may gain the architectural benefits of mobility (meeting places may become unavailable, but others may exist) and possible speed increases (contacting a meeting place nearer you may be quicker).

On the premise that when looking for information you should be able to use the knowledge of those who have gone before you, Maltz and Ehrlich's [1995] 'active collaboration filter' is a system in which where people can send pointers to interesting information to those they know are interested (generally, in this system, their colleagues). Whilst not automated, the system has echoes of the ACORN spirit. Additionally, it uses Lotus Notes rather than the Web.

Kautz, Milewski and Selman's [1995] 'Agent Amplified Communication' is another means of aiding "person to person communication in information finding tasks." In this system, the informal person to person links within an organisation are used to form 'referral chains' using autonomous agents to hold user profiles and help to organise the process. Again, whilst related to the ACORN system's ideals, this falls short of a complete dissemination mechanism, providing as it does an augmented information finding mechanism.

In all of the above work, and more, parts of the ACORN ideas are evident, but the whole is not. In other words, we conjecture that the ACORN architecture is currently the only such system of its kind, where agents, representing units of information or single queries, interact to produce a system where information can get to the right places, and the right people are asked for help. With a final implementation due in the near future, experiments are planned to demonstrate this utility.

3 ACORN

Whilst being necessarily brief, this section will present the workings of ACORN, and how the various agents in the system work together to make it a coherent whole.

3.1 Overview

The ACORN architecture uses the idea of 'document as agent.' In ACORN, when a document (a piece of information, an image, a sound file, and so forth; for the remainder of this paper, the word 'document' can be taken to mean a single piece of information) is created, an agent is created which 'represents' that document. That agent is able to migrate around networks, presenting the document to interested parties, receiving their responses, and deciding accordingly what to do next. The agent will exist, therefore consuming resources, only as long as it is found worthwhile by those it presents itself to. When it ceases to exist, the document remains, usually at the Web site of its creator. In this manner, we attempt to maintain the life length of a document without consuming valuable resources, since the life length of an agent may be radically different from this.

In many ways, this is analogous to Dawkins' Meme [Dawkins, 1989]. The meme is a piece of information that spreads via the human mind, migrating from mind to mind when it finds like-minded people. An example of a meme may be religion, and it is easy to see how the spread of religions can equate to a migration of information. In any event, the meme would not survive if there were no like minded (or receptive) people for it to migrate to. The same goes for the document agent in ACORN. In the document agent's case, however, the information does not move from mind to mind directly (by word of mouth or example), rather via a computer network.

The following sections present a more or less detailed view of the agents making up the ACORN network. It is clear with a moments consideration that a large infrastructure is necessary for the agents to do their jobs: there must be receptive machines. We therefore have several different types of ACORN agents, each with specific tasks to do to ensure the information is spread properly. In addition, to fully make use of such a system, some kind of

‘expression of interest’ is needed on the part of a user; in other words, a search mechanism. As an extension of the ‘document as agent’ metaphor, we have the idea of a ‘query as agent’.

3.2 The ACORN Agents in Brief

In the spirit of mobile and static agents, we can split ACORN agents into both camps. The ‘Infrastructure Agents,’ those which maintain the architecture, allow migration, control access to sites and users, are static. The ‘Smart Documents’ and ‘Smart Queries,’ respectively document distribution agents and user searches, are mobile. Figure 1 shows the structure of a typical ACORN site, including each of the Infrastructure Agents and several mobile agents. From this, we can see that a site can host many mobile agents, in the form of document and search agents, and many users, ‘protected’ from the system by their individual GateKeepers. A SiteMaster controls access to and egress from the site, and MeetingPlaces exist where mobile agents can interact. In the following sections we first present a brief introduction for each, following this with more detailed examples to illustrate the agents’ workings.

Figure 1. An ACORN Site.

3.2.1 Infrastructure Agents – Static Agents

Infrastructure agents in ACORN are static agents. They are used in the architecture to control migration of mobile agents, communication between agents, and so forth. Specific infrastructure agents, the GateKeepers, also act as the user interface between ACORN users and the agents in the system. This section discusses the various infrastructure agents in ACORN.

The GateKeeper

In the ACORN architecture, the GateKeeper has two roles, one as a filter of incoming information, ensuring the user is disturbed only when they want to be, and the other as a user interface when agents are created or examined.

In the first role, that of information filter, the GateKeeper acts as a buffer between user and system. The GateKeeper agent is the first (indeed only) point of contact for any ACORN agent wishing to communicate with a user. There are at present only a few circumstances where contact is made by ACORN agents in the normal turn of events. In each of these cases, specific tasks are to be performed by the GateKeeper. The performance of the tasks is dictated by the current status of the GateKeeper with regard to its user, for example, the user may be unavailable, or may be available only for the results of a prespecified search, and so forth. The GateKeeper is expected to behave sensibly and unobtrusively in all of these states. In addition, the GateKeeper is expected to 'field' incoming agents, saving data for later consumption by the user if relevant, and refusing access to irrelevant or unwanted documents or queries. This data is gained from the user, either implicitly, through watching user feedback to received documents, or explicitly, via direct interaction and question answering sessions with the user.

The second role of the GateKeeper involves the creation of InfoAgent or SearchAgent for distribution on the ACORN network. The creation is similar for the different agents, and involves the GateKeeper gleaning information from both the user and its own 'knowledge' about the world. Using this data, the GateKeeper is able to construct the details of an agent, and pass these details to the SiteMaster in order that an agent be instantiated.

The SiteMaster

Whilst the GateKeeper acts as a user controlled information filter, the SiteMaster of an ACORN site is a site filter, amongst other things. The primary task of the SiteMaster is to protect its site from malicious or incompetent others. The degree of protection, clearly, is the decision of the site maintainer, but can include a bar on agents from specific other sites, or authors; for example, it would be possible to bar advertisements if one knew from what site they originated. Since all agents in the final ACORN version will have digital signatures, this knowledge could be verifiable. Learning about which sites or agents are malevolent is another matter, but for that, we lean on the community, since we expect agents and users within the community to share their experiences, good or bad, with others.

The SiteMaster, then, allows access to and egress from a site. It also controls all ACORN agents within a site (in the current implementation, all agents at a site are Java threads started by the SiteMaster, and under its control). All communication between ACORN agents is via the SiteMaster, so it is extremely difficult for non-registered agents to infiltrate a site for malicious purposes (regular agents will not communicate with agents who do not send messages via the SiteMaster, and will report these agents to the SiteMaster).

Site Administration

Whilst the SiteMaster provides communication and migration abilities to agents, it does not do anything else to help them in their tasks at its site. Depending on the site concerned, the value added by that site may be large or small. Some sites may provide many additional tools for agents visiting that site to use to help them navigate around the site, find the information they need, or the people or agents they should contact, and so on. Some may provide a bare minimum. In any case, these value adders are Site Administration agents. In this section, we mention some of the administration agents we envisage may be present at a site. The site owner is free to develop or use more or less of these as they choose. The final choice may dictate how popular a site becomes, or even what a site becomes.

One kind of value adder is a Local NameServer. The NameServer is able to confirm whether an agent (by ID) is present on a site. It is primarily useful for 'forwarding' agents to new sites when users (and therefore their GateKeeper Infrastructure Agents) move. The NameServer can keep a record of where they moved to and redirect mobile agents there.

The MeetServer is in control of a specific 'MeetingPlace,' a virtual space within a site, where agents are able to interact with like agents. The MeetServer acts as a mini-SiteMaster for its MeetingPlace, allowing in only agents who are related to a specific topic, for example. It also handles agent interactions within its MeetingPlace. MeetServers can also add value by introducing extra information to an agent, for example giving a Library of Congress classification to a document agent. Sites can hold more than one MeetingPlace.

The Local SearchServer keeps details of the interests and topics of all Infrastructure agents at the site. This list will naturally include the users of ACORN at the site. It will also include details of MeetingPlaces at a site. An arriving agent need only contact the SearchServer to find who may be interested in what it has to say, or who has an interest in what it wants to know, and it can be directed to them (in fact, it is given the agent ID, or the GateKeeper ID if the agent is a user) so that they can make contact. Agents need simply register with the SearchServer, informing it of their interests and wants, and it is able to direct other agents at the site to them also. Naturally, registration is voluntary, for example if privacy is required, and the SearchServer will not direct agents to unregistered others.

Kinds of Sites

Thus far, a couple of different site types have been mentioned: the archive, where information is stored, with or without processing, for access by others; and the ordinary ACORN site, with human users, and artificial agents spreading their information or queries. We also briefly mentioned the 'MeetingPlace' served by a MeetServer agent. In this section, we discuss a few of the nuances of these and other kinds of sites.

The archive is capable of storing any information. A simple way of adding value is for it to store only information relevant to a specific topic, for example, Artificial Intelligence, or Information Retrieval. Such an archive could become known as 'the place to go' to find or archive information relative to that subject. The value

added is clear: a one stop site may well serve most of your search needs. Another extension may be to process the data in some way, organising it in a sensible fashion for ease of human access, or mailing updates (or sending update InfoAgents) to interested parties, and advanced search interfaces could be added. Archives are not simply storers of information.

The idea of a MeetingPlace is also extensible to take into account whole sites: a site may contain no human users at all, simply agents seeking to interact with each other. The agents may be filtered for topic on arrival, so that the site is topic specific (much as an archive may be) so that an incoming agent is more likely to meet another agent of interest. In fact, this is akin to Foner's matchmaking clusters [Foner & Crabtree, 1996]. A site may be subdivided into several such meeting places, each concerned with specific topics or subtopics. The idea of such meeting places is clear: if an agent does not know where to go next, random meetings may not pay off, but more directed meetings may well. Finally, however, a completely random, 'all-comers' meeting place may have its place too, for agents to glean potentially useful information from other contacts.

Sites in ACORN are value adders: in some way, they add value to the information people produce. This value added may be as simple as long term storage, but is nonetheless useful in the ACORN structure. It is not possible to envisage all of the different types of sites that may arise to fill niches. The ACORN architecture can grow to a complex system, and it can adapt to the changing needs of those who use it [Holland, 1995]. New value adders may appear spontaneously to fill those needs, and may vanish as quickly. The architecture has been designed to be extensible and flexible enough to meet these requirements and cope with the changes.

3.2.2 Information Carrying Agents – Mobile Agents

In ACORN at present we have two 'information carriers,' the document agent and the query agent. These agents are (conceptually) mobile, in that the information they carry can be transferred from site to site. This section presents the document (InfoAgent) and query (SearchAgent) agents.

The InfoAgent

The InfoAgent is the embodiment of 'document as agent'. It is the InfoAgent that is created to represent a specific document, and it is the InfoAgent that migrates around the ACORN network looking for people who would want to read that document (listen to the sound, look at the image, and so on). It is, in today's terms, an example of 'producer push' (see [Wired, March 1997]). InfoAgents are created on the user's demand from a finished distributable document by the GateKeeper in conjunction with the SiteMaster at a site. They are mobile, in that the information (their state) can be moved from site to site, and at each new site, a new InfoAgent is created to handle the information at that site. In other words, the code of the agents itself is not mobile, rather the information it carries. In this way, we contain the security risks inherent in mobile code whilst still benefiting from mobility.

The SearchAgent

Given the InfoAgent, we have a degree of producer push. The SearchAgent, as the embodiment of a user query, or search, introduces consumer pull to the system. A SearchAgent is generated from a user query by the GateKeeper in conjunction with the SiteMaster at that site. As with the InfoAgent, the SearchAgent code is not in itself mobile. Rather, the information contained in the query itself is moved from site to site, and new SearchAgents created at each site to manage the data.

3.3 The Details: How it all fits together

The previous section provided a glimpse of the agents ACORN uses. We said nothing of how they worked together to produce the community-based approach we are looking for, neither did we provide any details about how InfoAgents or SearchAgents interacted, migrated, or were created. This section corrects that omission with the knowledge that the reader, having read this far, has an idea of the agents involved which more detailed discussions can rely on.

3.3.1 Mobile Agent Creation and Life Cycle

To a large extent, the structure and creation of both InfoAgents and SearchAgent is similar. Both carry details about the user they represent, both have lists of places to visit, places they have been, and so forth. Both hold their document or query data in a standard metadata format: the Dublin Core [Weibel *et al.*, 1995]. This section describes the life cycle of a mobile agent from creation to extinction. Where InfoAgents and SearchAgents differ, this will be pointed out in the text.

The creation of a mobile agent is initiated by the user. If a search is needed, the user informs the GateKeeper of a new search, if a new document is ready to be disseminated, the user informs the GateKeeper of this. In both cases, the GateKeeper is informed via its user interface, which at present is displayed via a Java application thread. The GateKeeper's task is to gather the information necessary to create a useful agent that is capable of visiting sensible sites or users in order to distribute information to them or to query them for information. This data can come from several areas: the user, previous interactions with the user remembered by the GateKeeper, and data from returned mobile agents. The first case is the most straightforward for the GateKeeper, but is not the most efficient on the user's behalf: we aim to limit as much as possible the amount of extra work a user has to do to distribute information or perform a search, since any extra work will inevitably lead to a negative impression of the system, and take-up will be slow, making a critical mass of users hard to achieve. From the user, then, we require, in the case of a search, the search terms. In the case of a new document or piece of information to disseminate, we can often get away without asking for any additional information since what we need may be stored in the information itself. However, given that the user may know of the community for which the information is destined, it is sensible to query the user for a list of other users who may be interested in it. This will form the basis of a list of Infrastructure Agents to visit during the life cycle of this mobile agent.

As was briefly mentioned above, we use a metadata set, the Dublin Core, for agent knowledge of query or information being disseminated. This is a reasonably powerful metadata set proposed at the Dublin Metadata workshop in Dublin, Ohio [Wiebel *et al*, 1995] which was attended by fields such as library science and internet development. Rather than having an InfoAgent carrying around a potentially large piece of information (the complete document), the Dublin Core metadata for the information can allow less bandwidth use whilst still retaining useful data. The Dublin Core set includes sections for title, author, time and date, type, language, and so forth. For a document, most of this can be relatively easily obtained by the GateKeeper without resorting to questioning the user. Keyphrases can be extracted from a document if applicable, or a short summary and keywords can be requested from the user for non-textual items such as images or sound. The InfoAgent also carries an abstract as part of our own extensions to the Dublin Core for ACORN.

For the SearchAgent, the Dublin Core provides an extremely powerful search mechanism, since a user can specify searches using any combination of the Dublin Core elements: it is possible, for example, to search for images covering Northern Europe taken between December 12th and 13th 1996, or for documents in French related to German Reunification written by someone from the University of Ottawa (a very short list of documents).

The GateKeeper puts this data into a file, along with user data such as address, the ID of the GateKeeper and its address, and so forth. In addition, a time limit may be placed on the agent's existence (this is especially useful for SearchAgents which the user wishes to return with results within a set time). Both kinds of agents are given time intervals at which they must send reports back to the creating GateKeeper, so that they can be updated and tracked as necessary. SearchAgents are given a user generated Search Tag (a simple mnemonic string) so that a user can keep track of extended searches or browsing sessions of their agents, and are able to see search results from distinct searches as they arrive.

Finally, the agent has to be *seeded*. In other words, the GateKeeper has to build a list of Infrastructure Agents the agent should go to to disseminate its information, or to present its query. These Infrastructure Agents may be other GateKeepers (so the agent is effectively visiting another user), or MeetingPlaces, or Archives, and so on. Anything which may be of use to the agent can be included in this list, which we call the *Recommended* list. This data can come directly from the user, or it can be given by the GateKeeper from data which it has gathered from agents visiting it or returning to it at the end of their life cycles. It follows that the longer a GateKeeper has been in existence, the more data it can potentially give to the agent. This list is put into the agent's data file along with the metadata and user data. In addition to this, the mobile agent is given two other lists, which are at present empty. These represent the Infrastructure Agents the agent has visited, and the results of these visits (the *Visited* list) and a list of Infrastructure Agents the agent knows of which are not directly of use to it (the *Known* list) but which may potentially be of use to other agents it interacts with in future. Figure 2 shows the outline structure of a mobile agent.

The file containing this data is passed to the SiteMaster, and the SiteMaster creates a new mobile agent (InfoAgent or SearchAgent) with a unique ID from the data in the file. That agent is now free to proceed with its task.

The mobile agent's algorithm is relatively simple: visit all the places on the *Recommended* list, trying to build it up via recommendations from those who have seen the data or query carried, until the list is empty (at this point, the *Visited* list will have grown considerably. When this happens, return to the creating GateKeeper, report what was learned, who saw the information or query and their responses, and then die. At various points along this life cycle, the agent (its data) will migrate from site to site and will interact with others, either at MeetingPlaces (mostly for other mobile agents) or directly (for example when the other agent is a GateKeeper). In such interactions, the *Recommended* list may be augmented or reordered to change the subsequent movements of the agent, to direct it to other users, or to improve its traversal of the network. The following sections describe these processes.

Figure 2: Mobile ACORN Agent Structure

3.3.2 Migration to other Sites

This section describes the ACORN migration process in its current form. As may be expected, the decision to migrate comes from the mobile agent itself. This may be because the site is where the agent was created, and it is now moving off site to start its search or dissemination tasks, or it may be because the agent migrated to this site previously and has completed its work here. Whatever the case, the agent informs the SiteMaster that it wishes to migrate. The migration process happens as follows:

1. Agent informs its creating GateKeeper (via inter-site messages if necessary) of impending migration. It must wait for reply. GateKeeper can use this for giving feedback to the agent, to recall the agent, or to allow or forbid migration to the new site;
2. Agent informs Local SiteMaster of desire to migrate;
3. Local SiteMaster suspends migrating agent, checks if any messages waiting for it. If yes, the agent must first respond to these messages, so it is reawaken to do so;
4. If no messages waiting, agent details are extracted and stored in a file;
5. The local SiteMaster decides itself whether it is going to allow such migration;
6. Assuming migration is allowed, it then contacts (via the normal TCP/IP protocol mechanisms built into the Java language) the remote SiteMaster to inform it of the migration request. The Remote SiteMaster is given the address (URL) of the agent details file;
7. Remote SiteMaster grabs file, scans it, decides on allowing the agent in;
8. If the agent is allowed in, the Local SiteMaster is informed and it kills the agent at this site. Remote SiteMaster starts up a new agent at the remote site and gives the agent details file to that agent;
9. The agent has now migrated. The details file is kept at the original site until the next migration, when the Remote SiteMaster informs this site of migration and it is deleted. We use this as possible backup in the event of site failures.

One major point should be reemphasised from this description: the code of the agent does not migrate. The agent *code* at a specific site is entirely under the control of the administration of that site. Indeed, the code for all ACORN agents can be implemented in the language of choice of the site administration (we are using Java to allow for ease of integration to new systems). The *data* that is passed between agents (on creation, migration, and so forth) is fixed in structure, and indeed agent behaviours are largely proscribed by the system, but this system gives site owners a large amount of control over what runs at their site. Of course, this approach greatly simplifies many of the aspects of the ACORN architecture, but it is the *information* that is important rather than the agents themselves, which act as the medium for transporting that information. As long as the information reaches its goal, the means of transportation are relatively unimportant. The use of autonomous agents, however, does give us advantages over other means like mailing lists, such as an inherent scalability. In addition, the creator of an agent can find out where the agent is at any time via their GateKeeper (to which regular updates are sent by an agent, and to which migration details are sent).

It should be clear from this discussion that the system is relatively secure in that malicious code should not be able to be transmitted. Even if the data that is transmitted contains such code (which is unlikely, given its fixed structure), the agents at the remote site should not be able to instantiate it. As may be expected, although we have aimed at security, there are always going to be problems in a system of migrating agents. We are continually

examining the developments in this field, and the ACORN architecture will be upgraded accordingly.

3.3.3 Interaction

MeetingPlaces

ACORN agents carry information about sites they have visited and the Infrastructure Agents they encountered at those sites. They also hold information about users they have encountered. This information may be valuable to other agents in the community as they seek to find users to examine them or to help with their queries. It is through interactions with other agents that this community knowledge can be shared. This is primarily done in an ACORN site's MeetingPlace.

A MeetingPlace is a virtual space within a site which is controlled by a MeetServer. As discussed above, MeetingPlaces can be dedicated to specific topics, disallowing non-related agents, or they can be more general. We see them as a means of augmenting an agent's knowledge of itself as much as of other agents. For example, MeetingPlaces could add different classifications of agents' data as they process the agents. The primary task of a MeetingPlace, however, is to exchange agent knowledge. To do this, an agent gives its *Visited*, *Recommended* and *Known* lists and its owner address to the MeetServer. Once a critical mass of agents has come to the MeetingPlace, the lists of each are examined by the MeetServer, which can then add new details to an agent's *Recommended* list according to the lists of the other agents present. It can also add data to the *Known lists* of agents. In other words, relevant information known by other agents can be given to agents who will find it of use, so that the newly augmented agents now have more places to visit (in the *Recommended* lists) and leave the MeetingPlace with that extra knowledge. Note that at no time in the MeetingPlace (or indeed in a mobile agent's life cycle) does the agent process these lists itself: it simply carries the lists around, visiting from place to place on the *Recommended list* until it is empty.

Figure 3 illustrates this conceptually using an example with three agents. In their original state, agents W, X and Y arrive at site 23 and enter the MeetingPlace there. In that MeetingPlace, their respective details lists are taken by the MeetServer and combined. The end result is that W has two more Infrastructure Agents it can visit (has been recommended): 1 and 12. Agent 1 because SearchAgent X's search topic is cats, and therefore its user (the GateKeeper has ID 1) will want to see the data, Agent 12 because SearchAgent Y 'knew of' an Infrastructure Agent from its own seeding whose interest user's interest) was cats. The result for the other two agents is similar. Not that all the agents benefit from the interaction. In addition, the agents' *Known* lists have increased in breadth, so that they may be of more benefit to others in future. The *Known* list may have a finite size (it is less important than the other two lists) and not all the data need be stored. Note also that MeetingPlace 23 is stored in the *Visited* lists of each agent, possibly along with additional information about the agents that the MeetServer has added.

Individual Agents (GateKeepers)

When an agent approaches a GateKeeper to present information or query, the process is similar but more tightly controlled, since the interaction is one-on-one. The agent gives the GateKeeper copies of its three lists and its Dublin

Core set, and the GateKeeper (using a comparison agent of some sort) compares the data to its own user data and what it knows of the outside world (this data is bought back by returning mobile agents, and is augmented by interactions such as this). It can add details to the *Recommended* list and grab details from all three lists, so as to better seed its own agents in future. It can examine the metadata elements and store them for future perusal by the user. Later, the user is able to examine the document, giving feedback to the GateKeeper which it can forward to the document author.

Figure 3: Interactions between agents at a MeetingPlace

3.4 Finally

The preceding sections have described in more detail the workings of the ACORN architecture. The questions that remain are related to whether or not it works in practice and how to evaluate the resultant system. The latter question is addressed in the next section. Here we discuss what we can gain if it works.

We can imagine an InfoAgent to be little more than a fancy email forwarding system. However, the system adds somewhat more to such a process than such a cursory look reveals. At each migration, agents inform their creating GateKeeper (and thus, indirectly, their creating user) where they are proceeding to. At the end of every GateKeeper-InfoAgent interaction, feedback is sent to the user, and at preset intervals, the InfoAgent reports back to the GateKeeper to be given such feedback and updating as is available. The inherent power in such a system is clear: a writer (in the case of the InfoAgent) will always know who has seen what they have written, and where it is now. They will always be in control of where it can go next, and they will know continually what kind of impact their writing is having on the community. A simple email redirection does not give this kind of data to a writer. For a SearchAgent, the searcher will always know the current search results, which might be from a wide range of distributed sites that they initially did not know of. They will continually be in control of their search, able to refine it on the fly, and to make use of results as they come in whilst the search is still under way.

The one caveat to all this is user acceptance. Any community-based approach relies heavily on attaining a critical mass of users. Without users willing to recommend interested parties or items of interest, the system will not work. This is something we need to address in the creation of agents, the reporting of user feedback, and the general ease of use and potential utility of the system, and it is something we are investigating at the moment.

4 Current Status and Proposed Evaluation Techniques

The ACORN architecture is now 11 months old. In that time, it has progressed from the germ of an idea to a full blown system design. Currently, the system is being developed using Sun's Java programming language, which offers us superior communication techniques, whilst being ostensibly portable enough to allow installation of several classes of machine. The implementation has been under way for a couple of months now and is progressing smoothly. We have a rudimentary SiteMaster up and running, along with simple agents to test messaging and migration. The next step is the development of GateKeepers and more advanced Info- and SearchAgents, which we envisage will be ready and working in the near future. This will give us the complement of agents necessary to begin an evaluation of the system. In particular we are looking for sensible delivery of information to those who have expressed an interest, the sensible querying of those who may hold answers, and a good and useful speed of query and dissemination of information.

Initial evaluation is planned with artificial users, testing viability, scalability within a constrained system, and timeliness. Following this, we will expand to human users, testing the system in our group. We then plan to expand the system base to test scalability and usefulness to a large group of users, where both speed and privacy will become important issues. Following possibly redevelopment, the system is planned for a beta release later this year. Interested readers should contact the author for more details.

5 Conclusions and Further Work

The ACORN architecture is an agent based information retrieval and provision architecture. Information and queries within the system are represented by autonomous agents that are (effectively) able to migrate around networks in order to best attain their goals of sharing what they know with interested parties and finding out what they need to know from knowledgeable parties. The community of agents in the system shares information between the agents so that, with little or no prior knowledge about who to visit, they can build up a picture of the community and who or what is best to visit. This system, currently under development using the Java programming language, provides a unique addition to currently available tools. It helps make systems such as the World Wide Web more manageable because it allows users to automatically browse for relevant information, and to search for it in a sensible, informed fashion. In addition, it allows authors of information to notify interested parties of the existence of new or updated information without any initial knowledge of who those parties may be.

There are still problems with the architecture. In particular, we are concerned with the aspects of privacy and security. In order to maintain a user's right to privacy, we are building in a facility for the encryption of the data, along with the already present directions to an agents about who it can share its information or query with. On the security front, we are considering digital watermark technology amongst other things.

Further work is also in hand to extend the architecture to Personal Digital Assistants, so that users can still take advantage of its benefits whilst mobile, with less powerful handheld computers and slow (and expensive) phone lines. To that end, we are considering an initial solution using Apple's Newton architecture.

Acknowledgements

The author would like to thank colleagues in the Interactive Information Group at NRC Canada for interesting and enlightening discussions on this topic. Part of the research was carried out while the author was an NSERC Postdoctoral Fellow.

6 References

- Cowie J. & Lehnert, W. Information Extraction. Comms. ACM, January 1996, Volume. 39, Number. 1. Pages 80-91.
- Davies, N.J., Weeks, R. & Revett, M.C. Information agents for the World Wide Web. BT Technology Journal, October 1996, Vol. 14, No. 4. Pages 105-114.
- Dawkins, R. The Selfish Gene –New Edition. Oxford University Press. 1989.
- Foner, L. & Crabtree I. B. Multi-agent matchmaking. BT Technology Journal, October 1996, Vol. 14, No. 4. Pages 115-123.
- Holland, J. Hidden Order. Addison Wesley, MA. 1995.
- Kautz, H., Milewski, A. & Selman, B. Agent Amplified Communication. In: AAAI Spring Symposium Workshop Notes on Information Gathering in Distributed Heterogeneous Environments, Stanford, CA. 1995.
- Kuokka, D. & Harada, L. Matchmaking for Information Agents. In: Proceedings International Joint Conference on Artificial Intelligence, Montreal, Canada. pages 672-678. Morgan Kaufmann, CA. 1995.
- Labrou, Y. Semantics for an Agent Communication Language. PhD Thesis, University of Maryland, Computer Science and Electrical Engineering. 1996.
- Maltz, D. and Ehrlich, K. Pointing the way: active collaboration Filtering. In: Proceedings CHI'95. 1995. Available on the World Wide Web at:
http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/ke_bdy.htm
- Menczer, F. ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. Submitted to ICML-97.
- Shardanand, U and Maes, P. Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In proceedings ACM CHI'95. Available on the World Wide Web at:
http://www1.acm.org:82/sigs/sigchi/chi95/Electronic/documnts/papers/us_bdy.htm
- Weibel, S., Godby, J. , Miller, E. & Daniel, R. OCLC/NCSA Metadata Workshop Report. 1995. Available on the World Wide Web at:
http://www.oclc.org:5046/oclc/research/conferences/metadata/dublin_core_report.html
- Wired, March 1997. Special Issue on Producer Push.