The ACORN Multi-Agent System

Stephen Marsh (steve.marsh@nrc.ca)
Institute for Information Technology
National Research Council
Ottawa, ON, K1A 0R6, Canada

Ali Ghorbani (ghorbani@unb.ca) and Virendra C. Bhavsar (bhavsar@unb.ca)
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, E3B 5A3, Canada

Abstract. ACORN (Agent-based Community Oriented Routing Network) is a distributed multi-agent architecture for the search, distribution and management of information across networks. ACORN utilises the concept of 'information as agent' together with an application of Stanley Milgram's Small World Problem (the idea of Six Degrees of Separation) in order to route individual items of information around a network of people and agents. The ACORN ideal is to achieve a state where a web of users is created such that information distribution, queries and search, and browsing behaviour is encapsulated in a single adaptive architecture which learns community behaviour and knowledge in order to route agents to relevant destinations (users).

This paper describes the ACORN architecture and its implementation. We introduce a novel idea of agent meeting places, or Cafés, to carry out community-based information sharing among mobile agents in ACORN. ACORN is compared with similar work, and evaluations of ACORN for information sharing among mobile agents are described.

Applications of ACORN include Business to Business and Business to Consumer based e-Commerce solutions, virtual community creation and support systems, peer reviewing systems, and personalized directed information handling.

Keywords: Multi-Agent information architectures, autonomous agents, mobile agents, keyphrase matching, multi-agent architecture, community based information handling, e-Commerce.

1. Introduction

Information is playing an increasingly important role in the networked world. Great changes are taking place in the area of information supply and demand due to the widespread application of computers and the exponential increase of computer networks such as the Internet. The main problems we are facing right now are how to extract relevant, useful, and interesting information from many diverse sources and how to distribute our information to relevant people. Currently, two different technologies are commonly used to address the information demand problem, but fewer systems exist for distributing information. The existing solutions for the information demand problem can be either information retrieval, such as is used in the current crop of Web search engines, or information filtering, for example in services such as SDI (Selective Dissemination of Information). One embodiment of the information filtering technique is the software agent. Software agents exhibit a degree of autonomous behaviour, and attempt to act intelligently on behalf of the user for whom they are working. Agents maintain user interest profiles by updating them based on user feedback. Examples of such systems are the Kasbah framework for e-commerce [7], and matchmaking systems such as Yenta [8, 9, 10] and ACORN [22].

If we accept that all the Web cannot be indexed [3, 17], either because there are always pages that are new, others are not reachable by any known links, and still others are private (as is the case with many intranets), then the problems discussed above are almost certainly compounded. That is, there are some pages that are not connected to any others (and thus can't be indexed) for one reason or another. An additional problem is that some knowledge cannot be represented on web pages without considerable work – this knowledge is social network knowledge, what we call 'Who Knows Who Knows What', or WKWKW. Moreover, the contents of the web are constantly changing. All these factors limit the performance of web search engines and information filtering solutions. In this paper we propose an alternative to such solutions.

In any reasonably sized network of people, it is fair to assume that the chances of any one individual knowing everything relevant about every other individual in the group are slim. Any one individual, however, may know others in other social or interest groups, so the net result is a collection of interconnected groups of people, linked by 'middle men' who know others in different groups. In this sense, we are all 'middle men[1]' in some form or another.

In large organisations, too, such collectives of people exist. The Knowledge Management problem in such organisations lies in trying to find out who is doing what, or knows what, in any particular area. Corporate Intranets or employee databases are legitimate attempts to solve this problem, but have to be kept consistently up to date for them to be of any use to the members of the organisation. Moreover, they have to be *useful* and *used.*

The academic world is a huge web of interconnected practitioners. In this web, information and knowledge sharing is perhaps more open than in other arenas, but still, there are times when it is impossible to

[1] Naturally, we don't aim to be exclusive in our terminology. The term 'middle person' does not, however, seem to roll easily off the tongue.

know who is doing what in which area. Again, as in Milgram's Small World Problem [23] and see Section 3.2), chains of people exist so that networking, for example at conferences, really can help find people.

The ACORN (Agent-based Community Oriented Routing Network) system provides an agent-based peer to peer architecture using community based approaches for information retrieval and provision across networks. It is based on the assumption 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 will allow the producers of information to get their information in a timely fashion to those who will find it relevant. The agents in the system are autonomous; they make their own decisions about what to do based on information they receive from their creators and from the data they get from other agents in their community. One of ACORN's goals is to facilitate an architecture whereby social networks of computer (ACORN) users could share information based on 'people chains.'

1.1. Novelty and Contribution of ACORN

ACORN embodies the concept of autonomous mobile information in a peer to peer community of users. Within this community human users are able to dissiminate information in a timely and accurate fashion. In addition, users and agents are able to provide valuable recommendations to non-community members (potentially even non-ACORN users).

Information within ACORN is able to be rated and filtered by community members and agents, thus providing a valuable means of attaining peer-review of information that may have been previously unrated (and possibly unrateable given current standards of technology). The system uses chains of people and agents to build paths through which mobile agents can disseminate information to like-minded individuals without necessarily requiring those individuals to expressly indicate their interests. In addition, because of its novel information mingling facilities via multiple agent meeting places, or 'Cafés,' agents can learn recommendations not only from human users but from each other.

ACORN provides an architecture for security which, while conserving user anonymity if necessary and required, nevertheless allows for this peer review and recommendation capacity. The architecture uses the innovative approach of separate agent 'brain' and 'body,' wherein agent code does not migrate, while agent data does, coupled with a secure agent directory system to provide reliable mobile agent security measures. These measures protect not only the server on which mobile agents may be running, but also the mobile agents themselves, both from inadvertent and malicious attacks on their code and/or the data they may be carrying.

All of this is managed in a complete, readily available architecture that is implemented in a relatively small footprint (in Java), is extensible to take into account new developments in information sharing and handling, and makes full use of readily available communications and metadata standards such as XML and the Dublin Core [13, 30]. In addition, the system provides for readily adaptable user interfaces because of its adoption of the JSP libraries.

1.2. Potential ACORN Applications

ACORN's potential applications include people finding via virtual communities and also via extension to standard web searching approaches, the dynamic building and maintenance of virtual communities of like-minded individuals, and adaptive messaging through intelligent information dessimination.

On the e-business front, ACORN can provide facilities such as user preference directed information (i.e., directed advertisements), the dynamic building of business and consumer coalitions, and novel knowledge management tools for larger organizations [19].

Finally, we see ACORN as a suitable replacement for or extension of standard email messaging systems, chat and web search architectures. In this context, to coin a phrase, we see ACORN as 'email with attitude.'

1.3. Paper Organisation

This paper is organized as follows. Section 2 briefly reviews related work. An overview of ACORN is given in Section 3 along with a description of Milgram's Small World Problem. The architecture of ACORN is presented in Section 4. In Section 5, the various types of agents in ACORN are described. We propose a novel idea of Café as a virtual meeting place for mobile agents of ACORN in Section 6, and some relevant methods for information sharing are outlined. We also explore a dynamic clustering method for ACORN Cafés. Salient features of the ACORN implementation, along with performance evaluations of ACORN are described in Section 7. We conclude with pointers to further work and a summary of ACORN's capabilities in Sections 8 and 9.

2. Related Work

Community-based navigation and information sharing are not new [12, 16, 18] and interesting twists on the themes exist (or have existed), such as the now extinct Firefly and the vibrant Epinions [14], which take the views of a community, aggregate them, match individual interests, and recommend to their members movies, music or whatever that related members are into. These services work, but in a limited sense: people need to go to some non-minimal effort to describe their interests, and they need to keep coming back to recommend new movies, recordings, books, or whatever. Amazon (www.amazon.com) has another way of doing this, by suggesting books to people that others have bought, based on the title they are looking at now. This requires no effort on the part of the 'recommender,' and works surprisingly well. The concept of effort is important: people aren't going to use something they have to do extra work for unless they get something more in return.

Agents have been used to provide or discern links between people. Foner's Yenta [10] is an example of this. It takes people and their interests and tries to match them with others with similar interests. This is of particular use in a community-building situation, but can also help when looking for information (or possible interested receivers of information you have created); similar systems are described in [15, 16].

3. Overview

ACORN embodies the principle of 'Social Knowledge Management,' wherein the knowledge may lie in private yet easily obtainable places such as intranets, or may lie only in the heads of the people in the society. ACORN is a multi-agent based system which uses the concept of 'information as agent' to route information around networks (or communities) of people. Information in ACORN's context is anything that can be transported electronically, such as documents, in whole or in part, queries, images, sounds, and so on. The implementation of ACORN referred to in this paper uses mobile agents that are capable of performing both search and distribution of information.

3.1. Motivation

Information is hard to work with, particularly if there is a lot of it and it is disorganized, such as on the Web. Current solutions are straining at the seams, and a new paradigm is required to handle the volume and noise. Agents present only a possible solution, but one which bears closer investigation. Conceptually, an agent can be sent out onto networks to scour web sites or corporate information sites and databases available to it for information relevant to a search topic it has been given. This is an extension to the 'spiders' that the original search indexes used, with directions not to grab every file at a site, but to get relevant pages. Such 'searchbots' exist and work relatively well, for all their simplicity. However, humans employ other search strategies, via their communities: asking questions of people you know might result in them going elsewhere to ask another person they know (and so on) resulting in, eventually, an answer coming back to them. This is community-based searching, and requires only that you are a member of a community, and since communities overlap, it is easy to grab information across community boundaries. ACORN uses such a community-oriented approach.

The basic questions that ACORN seeks to address are those we live with everyday as members of communities:

- Who can help me with a specific problem?
- Who would be interested in this information?
- Who knows the answer to this question?

In addition, we attempt, via ACORN, to answer other questions that might be asked in some situations:

- Who has read what I wrote, and what did they think of it?
- How can I control who sees what I write, yet still use an open network to distribute it?
- Can I build up a team or community of like-minded people quickly?

3.2. Milgram's Small World Problem

ACORN's primary approach is to use mobile agents to distribute information. ACORN's agents move from person to person, following trails of recommendations. The users the agents visit can recommend other users, thus increasing the length of the chain and the number of people the agent can visit. At the end of its journey, the agent should have learned more about who exists in the community and who is interested in what it has to present.

ACORN's behaviour is in fact very close to the effects seen in Stanley Milgram's Small World Problem [23]. In Milgram's experiment, several letters were sent to randomly selected people in the continental US, with instructions that the letter had to find its way back to a named person (in Boston, since that's where Milgram was). However, the caveat was that the letter could only be forwarded via people whose first name the forwarder knew (in other words, people they knew socially).

Taking into accound blind alleys, closed worlds, and the occasioal fluke (for example, one of the randomly selected people just happened to know the postmaster in the town the final recipient lived, so the chain was very short), the average length of referral chains was six people. Hence, the concept of Six Degrees of Separation between any two randomly selected people in the continental US at the time of the experiment.

Although things have changed – for one thing, there are more people out there – certain assumptions about the Small World Problem can still be made. That people still interact with each other, even over large geographic distances, is clear, for example. In addition, people are becoming more, not less, networked [31]. The number of links between people has probably grown, although a new experiment would be required to prove that assumption.

Recently, attention has turned naturally to the World Wide Web. The Web is a very connected network of pages, and hence people or organisations. In addition, the Web is very information centric – the resulting ties of linked pages are often more akin to citations in scholarly works than social links, although these, too, exist. Recent work in this area has found that the average number of links from one random web page to another is in around 19 [1]. However, the actual shape of the Web may in fact be less straightforward than such simple figures suggest [3], leading to additional problems with the traditional search engines, on top of what they already have to deal with [17]. The Web, however, is almost certainly a Small World [1, 29], or at least a collection of Small Worlds.

We conjecture that the people connected through the Web and through networks in general are part of a Small World in themselves, and it is on this assumption that ACORN operates.

4. Architecture

Figure 1 shows an overview of the ACORN architecture as it stands today. As can be seen, ACORN is, from the top level, a Client/Server architecture. Clients, the user's interface to the ACORN system, present their user interface via JSP pages and displayed to the user on any standard web browser. Clients provide the ability to create agents, organize received agents, describe a user's interests, and input details of other users and their interests. This information is used in the agents to better distribute themselves.

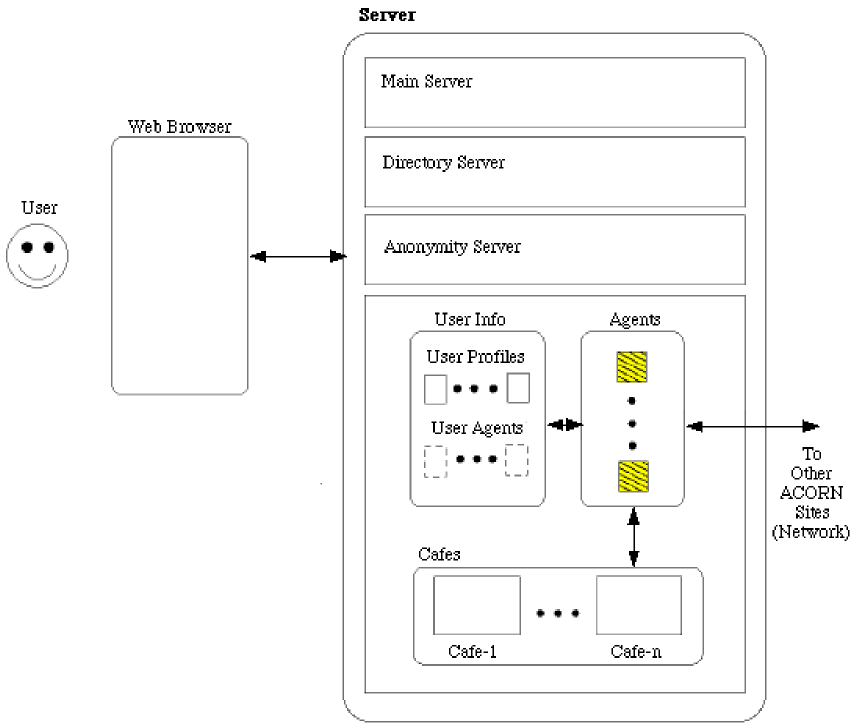

Figure 1. ACORN Architectural Overview

All community information the Client has is stored in a separate user profile file on the server side. This core contains user information, references to agents received and sent, user preferences, and also information about the people in the user's database. Data about other users is stored as email addresses allied with keyphrases and associated weights.

The Main Servers in ACORN provide mobility capabilities to agents, and also give ACORN a degree of persistence in information. All Client data is stored (as a client core) on the server side so that, for example, if the client shuts down, incoming agents can still be processed according to user requirements. By the same token, we can provide automatic redirection and recommendations to agents because the server can access the client cores to match agent topics with known user interests in the client core. Because of this structure, the user is able to log on to the ACORN system from any machine with a simple web browser.

Agent migration is achieved by sending agent cores from server to server. Like client cores, the agent core contains all the information the agent needs to accomplish its tasks: addresses of people to visit, user interests, document information and so on. When an agent reaches a new site, the server at that site instantiates a new agent with the core that has just been received, and lets the agent go on its way. At a site, the agent can visit users, and communicate with (mingle with) other agents at a central meeting point (which we call a Café) to share addresses and interests with others, thus aiming to extend the list of addresses for people to visit.

Recent additions to the architecture are the Directory Server, which provides the ability to track and control agents which are off site, and the Anonymity Server, which provides routines to anonymise InfoAgents before they are sent to the network, and to re-instantiate them as necessary on their return.

5. Agents — Client, Server, InfoAgent, Anonymiser, Directory Server

In order to provide the infrastructure for the mobile agents to perform their tasks, the architecture uses static agents. These are the *Client* and the various *Servers*. In addition, the *Café*s, with their managers and blackboard agents, provide for information sharing between agents and persistence of information. The following sections describe the Client, Servers and Mobile Agents, how they interrelate and the tasks they are designed for. Section 6 describes the Cafés in more detail, since we consider these to be of paramount importance in the architecture as a whole.

5.1. Client

ACORN's Client serves a dual purpose. Its primary role is to provide a user interface to the ACORN system. It allows the creation, browsing, control, tracking, and deletion of incoming and outgoing ACORN agents, and in this sense is much like any email client.

The second role of the Client is to maintain a user profile and perform information handling tasks. In this role, the Client acts as both a filter to incoming information (agents) and a form of 'community browser.' In its filtering tasks, the Client can scan, prioritise and route incoming agents according to the preferences based in the user profile (which is controlled by the user and includes priority information, user interests, contact information, and so forth). For more information on the Client and user profile, see [22].

As a community browser, the Client tracks incoming agents and files information about these agent's owners (that which is made public by the owner's Client on agent creation - see [22]). In addition, the Client's user can input similar information, which takes the form of 'person X is interested in topics $y, z, \ldots$' This information can be used at agent creation to suggest possible recipients for an agent. Over time, then, the Client can build up a fairly sophisticated world view of the communities its owner belongs to. Again, this can be used in filtering, prioritising, and recommending tasks.

5.2. The Main ACORN Server

Much like a Web server, the ACORN Server resides at the point of entry from a network to a site. All mobile agents must enter the site through the Server. While the Client acts as a user-controlled information filter, amongst other things, the ACORN Server is a site filter whose primary task is to protect the site and decide whether a particular mobile agent is allowed in. It also controls mobile agent migration carried out via server-to-server communication. In addition, it acts as a permanent repository for Client and resident mobile agents at a site, in order to achieve persistence in agents.

When a mobile agent arrives at a site, the Server saves its state. Clients communicate their state to the Server whenever they are started up and at specified intervals while they are running. The Server stores the Client data and can augment it with messages for the user if any are sent.

As well as the Main Server, the ACORN site architecture contains two additional Servers — a Directory Server and an Anonymity Server. Because of their direct application to the workings of the mobile InfoAgents in ACORN, these are described below in Section 5.4.

5.3. InfoAgents — Mobile ACORN Agents

The primary distribution of information in ACORN is carried out by its Mobile Agents, or InfoAgents. An InfoAgent in ACORN represents a piece of information of whatever form its creator requires (image, sounds, documents or sections of documents, etc.). In fact, the InfoAgent need not know anything about the information structure or type since it does not attempt to do anything with the information beyond distribute it, or at least, its metadata.

As shown in Figure 2, InfoAgents are in fact 'thin' representations of pieces of information, since they do not contain the information itself, just a link to it and a set of metadata elements that describe it. In addition, they contain a 'thin' user representation in the form of a list of user interests specified with a set of user-defined keyphrases. In addition, the InfoAgent carries three distinct lists:

- *Recommended* — a list of users *to be visited*
- *Visited* — a list of users *that have been visited*
- *Known* — a list of users who are not going to be visited but whose data may be of interest to the agent's creator or other agents

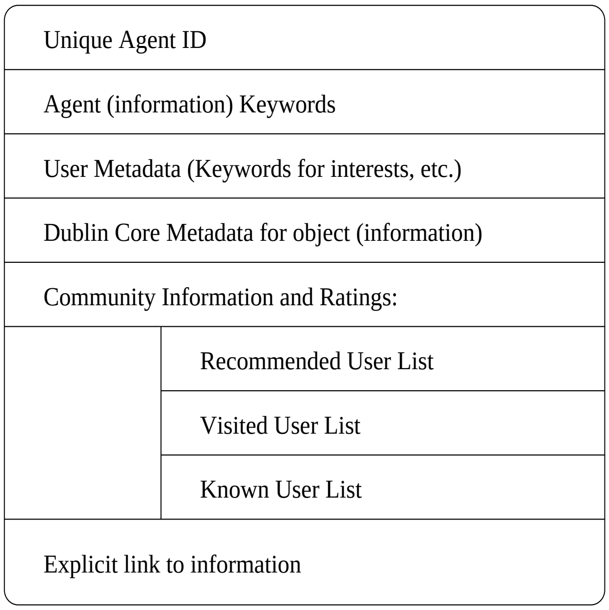

Figure 2. ACORN Mobile (Info)Agent

The first two lists are relatively straightforward. The *Recommended* list is a set of users the InfoAgent has been recommended to visit. These recommendations can come from its creator at creation time or afterwards (Section 5.3.2), from other users the InfoAgent has visited (Section 5.3.4) or from other agents via mingling in Cafés (section 6). The InfoAgent proceeds through this list in a linear fashion (although we are working on an optimisation of this process – see Section 8).

On visiting a user, the user's details are taken from the *Recommended* list and are added to the *Visited* list. The process of visiting (Section 5.3.4) may result in additional information being given to the InfoAgent about the user and their interests. This is stored in the *Visited* list, and will be used to update the Client's user profile data on the agent's return.

5.3.1. *Community-useable information*
The final list the InfoAgent carries is an oddity at first, but in the context of virtual communities is in fact extremely powerful. We call it the *Known* list. Users in the *Known* list are not readily of interest to the agent as it stands. In fact, any time the agent comes across others in Cafés, the chances are that some of these will be from users who are not of interest. In these circumstances, the users and their interest details are stored in the agent's *Known* list. This list is useful in two major ways:

- Firstly, whenever the agent mingles with others, if a user in the *Known* list relates to some other agents and their information, they can be given to that agent to add to its *Recommended* list — this is community based information sharing at its best.

- Secondly, when the agent returns to its home base, this information can be given to the Client to add to the user's profile. Visiting agents can be recommended to this user, or the Client's owner can be advised of a new person to send information to if their interests match in other ways.

The *Known* list is likely to become much more valuable in time than either of the other two lists - it provides a source of community information unavailable in any other way to mobile agents and ACORN users, and radically enhances ACORN's flexibility.

5.3.2. *InfoAgent Creation*
Agent creation is in fact a relatively straightforward process. We have attempted in our implmentation to automate the process as much as possible. However, there remain areas where user input is required, as with all messaging systems.

On creation, the InfoAgent needs to gather metadata about the information which it is to represent. This metadata will be stored in a Dublin Core metadata element set [13]. One of the benefits of the Dublin Core in this instance is that no element is absolutely required, and all elements can be duplicated. Thus for more than one document name, or language, for example, we are able to represent the document adequately. More importantly, the user is not required to give more data than they feel is absolutely necessary and ACORN will still do its work. One thing we do require is a set of keyphrases to describe the information, and this is stored both in the Dublin Core and separately in the agent itself. This keyword list is what the agent uses to match relevant users to which to route itself, hence its importance. Given this importance, we are attempting to automate the process of keyphrase discovery, at least for textual documents, and are able to use automated techniques such as Extractor [28]. However, there remains no adequate method of attaining keyphrases from, for example, music files or images, thus we require the user to do some additional work. Our hope is that this work is not too onerous considering the fact that the more keyphrases, and the more accurate they are, the better the document will be matched to and thus routed to other users, to the benefit of the author of the information.

Other metadata that the agent can grab automatically and that is extremely useful for community navigation is the owner's interest, in keyphrase form, and potentially a list of recommended users to route itself too. This list can be obtained from the Client by examining and matching other users' interests in the owner's user profile. The owner is also able to recommend users not in their profile (given these recommendations it's a dilemma as to whether we add these users to the Client's user profile. We have decided against since, on returning, the agent should have more accurate information about these individuals that we can add as necessary).

The user is able to give the agent a time limit, at which limit the agent will return to the user whether or not all of the recommendees have been visited. In addition, an anonymous bit can be set, and if it is the agent will proceed to an anomymiser to strip all information that can potentially identify the user (see Section 5.4.2). Other information, such as a subject for the agent, additional text messages, and so forth, are also possible. Finally, the user can specify that the agent stay in the Café when empty — this is useful in allowing an agent to discover potentially interested parties by agent mingling when the owner has no suggestions for who to visit, but is also under the ultimate control of the Café Controller (see Section 6).

Once this information has been given (and we have a simple single web page to allow this in our current ACORN implementation) the agent is free to proceed in its lifecycle, the next step of which is to migrate to the first user on its recommended list (if there is one — if not, the agent migrates to the local Café in the hope of finding one or more users before visitng them once found).

5.3.3. *InfoAgent Migration*

Migration in ACORN is very important to the system. Without migration, one agent cannot reach other agents to interact with them. Hence information cannot be shared. In fact, as was mentioned above, only information or queries carried by agents migrate. The agent's codebase does not migrate. More recently we have been exploring an even thinner agent, which moves only very minimal information from site to site and stores the majority of an InfoAgent's data at its origin site (or at its Directory Server — see Section 5.4.1 below.

When a mobile agent wants to migrate to another site, it informs the Server at the present site of this fact. The site's Server first ascertains that the mobile agent is allowed to move from this site via consultation with the remote Server. If so, the agent's state is passed to the remote Server via ACORN Server-to-Server protocol. If not, it informs the mobile agent, and the agent reorganizes its migration strategy (at present, going to the next person on its list at another site).

5.3.4. *InfoAgents – Visiting Users*

When an agent arrives at an ACORN site, two types of interactions between agents take place. The first interaction is that the mobile agent may have a specific person (his/her Client) to contact. In this way, information can be shared between the person and the mobile agent. The second type of interaction takes place between mobile agents in a Café. Once all the recommended users on a site have been visited, the agent should have an enlarged list of users to visit, and possibly a set of query responses (which it can carry with it or send back to its own user). Either way, it can be seen that the agent obtains more community information as it migrates, amongst other things, and that this information can be made use of at the end of the agent's life (when it returns home to its user's Client to upload the community information it has gathered).

5.4. Additional Services

5.4.1. *The Directory Server*

One of ACORN's main problems in the past has been the lack of complete user control over their InfoAgents. Clearly, ACORN has been designed such that the InfoAgents are capable of carrying out their tasks with minimal, if any, user interaction beyond the initial agent creation phase (see Section 5.3). However, it has long been a concern that in some circumstances user involvement may well be of worth. For example, there may be circumstances where the user wishes to recall the agent altogether, or where the user does not wish the agent to visit a specific recomendee. In these instances, the user needs to be able to control the agent's path. In addition, we feel that it is of some worth for the user to be able to know at any instant where their InfoAgents may be.

To accomodate these requirements, we have adopted a Directory Server approach [6]. In this approach, agents register with the Directory Server on creation and before migration. Note that this registration is not required — the user can choose not to have this additional overhead. The Directory Server maintains a log and database of agents registered, and every time a migration is planned, the agent contacts the Directory Server to update its database entry.

The Directory Server has its own interface through which users and system administrators can check the location and status of:

- Their own specific InfoAgents,
- Any public InfoAgents sent from this site,
- All InfoAgents sent from this site (system administrators only)

Extensions to the Directory Server architecture are either planned or are currently in progress to allow users to recall agents at any time, and to examine and alter agent path information (the Recommended list, as described in Section 5.3) on the fly. The Directory Server architecture also gives us the capacity to further enhance ACORN's security and control mechanisms — for more information see Section 8.1.3.

5.4.2. *The Anonymity Server*

In ACORN's origins, information, although considered valuable, was also considered to be public. Although not a comment on the socio-political climate, this requirement of public information was sadly naïve (or at least over-enthusiastic on the part of the designer...) It is clear that people value their privacy, at least in some fashion, in networked applications (including e-Commerce applications). In order to accomodate this requirement of privacy we considered several options. One of these was the introduction of encryption of InfoAgents, but this was rejected as being overly problematic for both political and implementational reasons. In addition, encrypted agents are still problematic in that they are obviously *from* a specific place. Knowing where something is from is one step towards knowing who it is from (especially with regards to organisations rather than people). What is more, when an agent is decrypted, all the information is once more available – the solution is an all or nothing one.

Ultimately, we have decided to focus on the concept of anonymity. The Anonymity Server thus serves to create a situation where InfoAgents are able to do their jobs (as described above) but without giving away anything which may give a clue to who may have sent them. Naturally, this raises issues of trust (for example in information sharing) and our solution reflects this in certain sharing algorithms. However, the solution is an extremely powerful one, giving ACORN considerable new capabilities (for which, see Section 8.1.1).

The Anonymity Server quite simply strips an InfoAgent of any information which could tie it to a person or geographic location. Anonymity Servers can reside at any site, and indeed could be the *raison d'être* for a specific site (a worthwhile service in itself). In other words, just because an agent comes via a specific Anonymity Server does not mean it came from that site originally. The Anonymity Server operates by first saving all InfoAgent details, then creating a new unique ID for the InfoAgent. Once done, a new Anonymous InfoAgent is created which has all of the contents of the source InfoAgent (including Recommended, Known, and Visited lists, and so forth) but with no identifiable information. The Anonymous InfoAgent is free to proceed on its path, acting and interacting exactly as any other InfoAgent but without giving any personal data away. Thus the address of the Anonymous InfoAgent is that of the Anonymity Server with a unique ID for agent identification at that Server. The path to the original user is now via the Anonymity Server (and any path could go through several such Anonymity Servers. . .)

Once the Anonymous InfoAgent's task of distribution/search is done, it returns to the Anonymity Server, where it is merged into the source InfoAgent, which is then returned to its origin.

The final stage of this process, as yet unimplemented, is a secure path between origin and Anonymity Server. It is here that encryption can become of some use to us, and we are examining its possibilities.

6. Café

One of ACORN's main strengths lies in the ability of InfoAgents to share information amongst themselves. This allows for automated filtering and forwarding of information without human intervention. The process of information sharing is carried out in agent meeting places, which we call Cafés.

In the Café, at certain times (explained later), all InfoAgents present in a particular Café give their community and personal data to the Café manager, which compares all data, sharing out any relevant community data to agents who would find it of use, before sending out the agents. The net result is that, on exit, agents potentially know more about the community than when they came into the Café.

For example, if two agents enter the Café, one representing a document about programming in Java (keyphrases: OOP, Objects, Java, etc.), the other representing a query about Object Oriented programming (keyphrases: Objects, OOP, Software Engineering, etc.), the Café will match the keyphrases (besides such an exact match, other techniques can also be used, see Section 6.4) and take the relevant user information from each agent, giving it to the other. The net result is that the first agent will now visit the second agent's owner to present its information, which may be of relevance, and the second agent will visit the first agent's owner, hoping to learn more. Of course, we envisage that many more than two agents will be present in each Café Information Exchange, with a corresponding increase in the utility of exchanged information. At present, each agent will visit the Café at every site it visits thereby spreading and gaining relevant information. A site may have from zero to many Cafés, each of which could be dedicated to a specific genre of information. The decision about which Café to visit can be based on a similiarity measure or other criteria, and is controlled by the Café Controller, as can be seen below. In this manner, the Café construct automates community based information sharing for agents.

6.1. Café Architecture

The structure of a single ACORN Café is shown in Figure 3. Simply, each Café has two main agents, the Café Manager and the Blackboard.

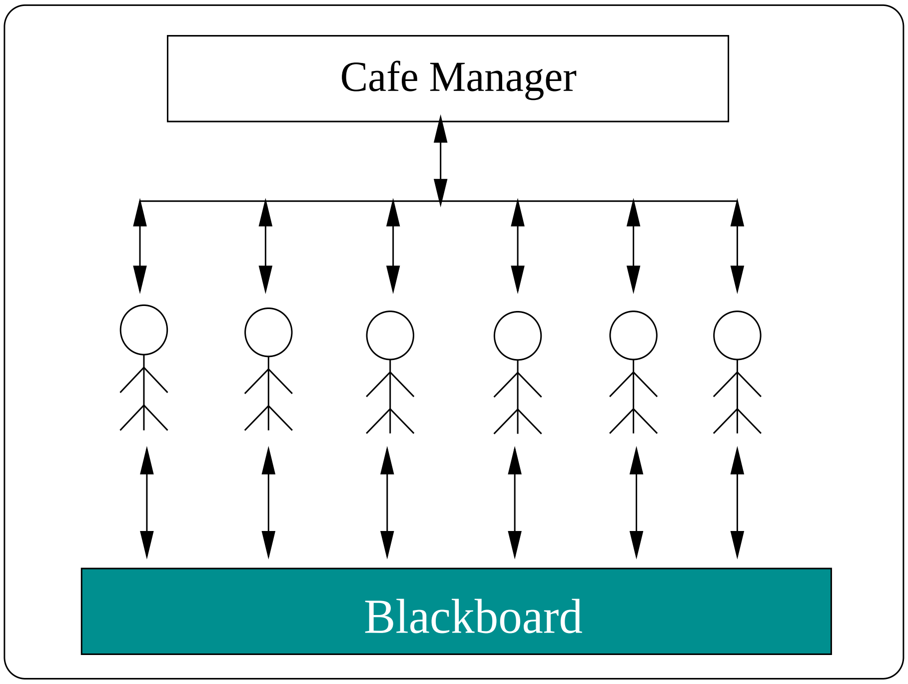

Figure 3. Single ACORN Café

6.1.1. *Café Manager*

The Café Manager is responsible for the mingling of agents in the Café at any given time. Each agent entering a Café has a certain amount of *patience* — a time limit beyond which they wish to have left the Café. Each Café also has a maximum number of agents permissible (this is manually set by the site owner at present). When one or both of these limits is reached, mingling takes place (see Section 6.4). Once mingling is finished, all agents in the Café at the time of mingling are ejected. The exception to this rule is if an agent has been designated as a permanent resident in the Café on creation. Such agents can be used to express an interest in a specific topic, for example, or to perform advertising roles. Naturally, the Café manager ultimately controls if they can stay or not (since they take up space and time).

6.1.2. *The BlackBoard*

The second Café agent is the Blackboard. The Blackboard serves the purpose of maintaining some degree of persistence in agent information. It is clear from the above discussion that the mingling process followed by agent ejection with no history means that there is a high probablility that some information that may be of use is lost. For example, agent Y may have information relevant to agent Z, but unless Y and Z are in the same Café at the same time, they will never share that information, despite going to the same Café at different times.

The Blackboard solves this dilemma to some extent. Any agent, on entering the Café, may choose to interact with the Blackboard agent. The Blackboard is given the agent's list of interests and contact information (its owner's ACORN address). This information is prioritised according to Blackboard rules (see below) and stored if applicable. In addition, the Blackboard gives a list of matching topics and the addresses of the users allied with them. The agent can add these to its distribution list. Thus, agent information in Cafés is not lost when agents leave the Cafés.

6.1.2.1. *BlackBoard Rules*

The BlackBoard agent follows strict rules on how to store the information that is given to it by agents to achieve persistence. These rules serve to limit the amount of space used at a Server site, and to ensure that information stays current in Cafés. There are in fact several strategies that can be used here, and which follow classic rules for Queues, Deques, and so forth. ACORN's BlackBoards are at present using a FIFO methodology, with space limit of 50 agent lists. These lists are not limited in size, however. Current ACORN setup allows the administrator of a Server Site to change this value on startup.

Prioritisation of lists is based on order of entry into the system. Note that there can be several ways of dealing with prioritisation, however. These rules are at present only able to be hard-coded into the current system. We are investigating different prioritisation methods, and are also investigating adding more 'intelligence' to the BlackBoard, to allow communication between BlackBoard agent and InfoAgent such that more relevant information is returned in response to BlackBoard queries from InfoAgents.

In addition to BlackBoard rules, there are concerns about more sensitive information or information that is entirely in flux as far as the creator of the InfoAgent is concerned. It should be noted that InfoAgents are free not to put information on the BlackBoard at all — it is entirely the choice of the InfoAgent.

6.2. Multiple Cafés

An ACORN site can in fact have any number of individual Cafés (including none at all, although this removes one of ACORN's main strengths). This is shown in Figure 4.

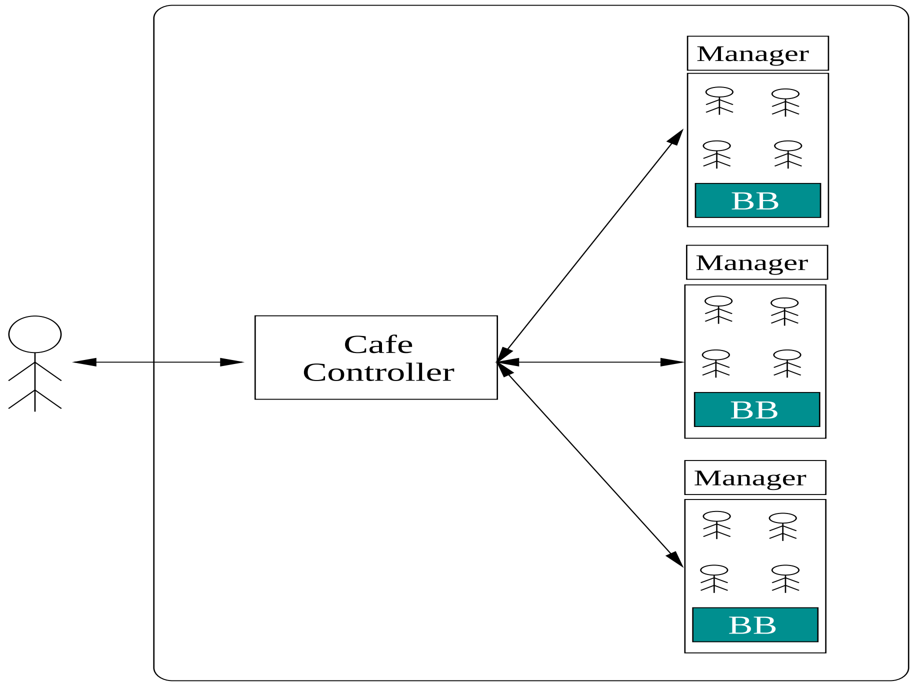

Figure 4. Multiple ACORN Cafés with Café Controller

Each Café at a site has associated with it a set of topics for agents which it is willing to accept. Thus, agent X can enter a specific Café

iff one or more of X's keyphrases match one or more of that Café's topic list. In this way, we are able to provide a more specific matching between similar agents. All of the Cafés at a site are monitored and controlled by the site's *Café Controller*. The Controller keeps track of all Cafés and their topics and is responsible for routing agents to and from Cafés.

On entering a site and choosing to enter a Café, an agent first interacts with the Café Controller for that site. The Controller compares agent and Café keyphrases and routes the agent to the most relevant Café. Note that the agent can choose to go to any number of Cafés if its topics match. The Controller manages this process, sequentially routing the agent from Café to Café as necessary. Within individual Cafés, the process is as described above and in Section 6.4.

6.3. Dynamic Cafés

In the course of our experiments and usage of ACORN it became clear that, initially, a single Café per server was not enough: single Cafés are too generic and result in information sharing (mingling) between agents that is clumsy and not specific enough for specialised InfoAgents. Our first solution to this problem was the introduction of multiple Cafés as described above. However, this solution suffers from its own set of problems, not least that it is also far too constraining when we consider that the agents may well have extremely diverse contents as regards the information they carry. It is worth noting, however, that dedicated Cafés are indeed an extremely powerful added value to any site, since they ultimately allow that site to provide a specialised service to chosen classes of InfoAgents. However, the problem remains for less specialised InfoAgents, or those that do not fit the mould.

A simple solution to this problem is to include a generic Café on any server, such that, if there is no specific Café for an agent to join sensibly, there is always at least one Café for it to potentially mingle with others. With some thought, it seems clear that this solution simply brings us back to the original problems we experienced with a single Café per server.

Our most recent innovation within ACORN is to introduce the concept of dynamic clusters of InfoAgents for information sharing. This can also be thought of as just-in-time Cafés. As can be seen in Figure 5, in this system, the InfoAgents register with the Café Controller as usual, but once in the Café Space on the server they become part of a community in flux. Continually, the agents are examined and clusters of agents are formed dynamically such that those whose interests closely match are clustered together. Once an agent enters or leaves the Café space, the clusters are re-examined and potentially reorganised to take into account the addition of, or loss of, the new information. The result is a much more closely fitted solution where agents are guaranteed to be able to find similar others if they exist in the Café Space.

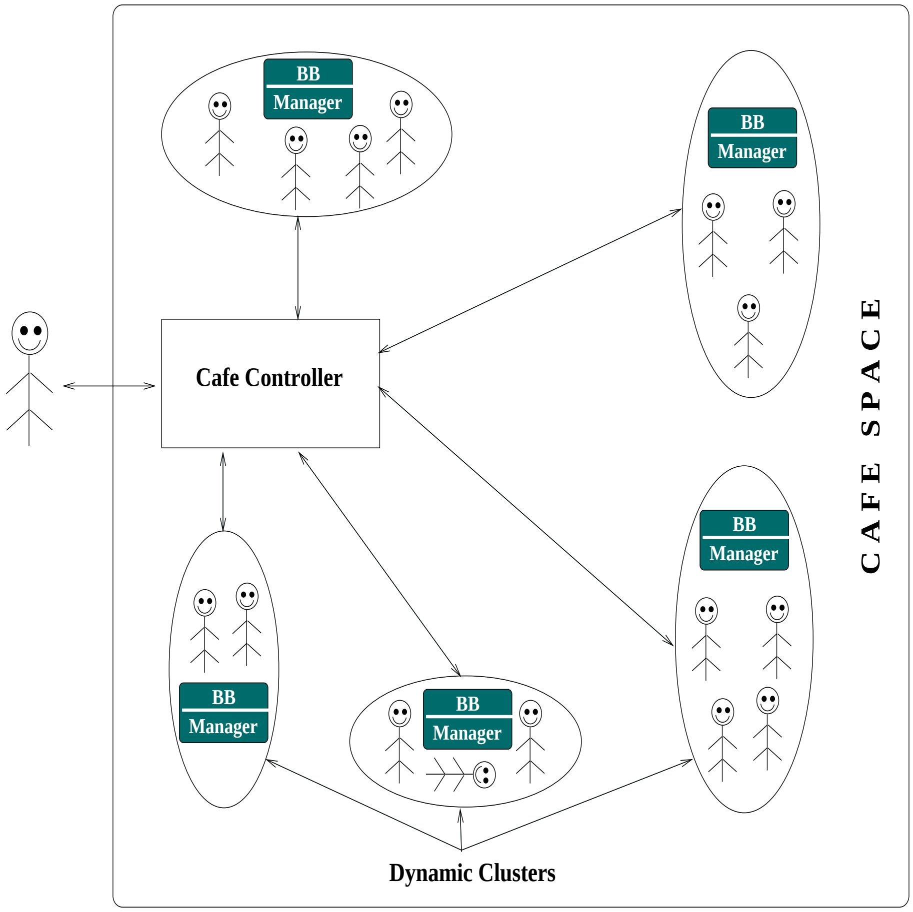

Figure 5. Dynamic ACORN Cafés — Just-in-time Café Creation

In order to dynamically create Cafés that unite specialised InfoAgents, a systematic way of comparing and assigning agents to Cafés must be proposed [5]. In this case, it is suggested that the dynamic K-means algorithm (see for example [27]) be applied to compare and assign these agents to new or existing clusters. Each cluster corresponds to a unique Café.

Each cluster is defined as having a center composed of a normalized vector of keywords and corresponding weights that reflect the interests of all members of the given cluster. It is calculated by averaging the vector of keywords and corresponding weights of each member of the cluster. InfoAgent membership to a particular cluster is dependent on the Euclidean distance from the InfoAgent's keyword vector to the given cluster center. In this case, the InfoAgent will join the cluster whose center is geometrically closest to its own. In the event of an unacceptably large distance to the nearest cluster, the InfoAgent is assigned to a new cluster that contains a center equal to the InfoAgent's keyword vector (i.e., a new cluster is created).

Upon joining and leaving the cluster, the cluster's mean center is adjusted to reflect the change in the members' keyword interests. As such, clusters have a dynamic nature that allow for the interests of the cluster to change with time as the members' interests change. At the same time, the cluster population is dynamic to reflect the variety of interests of the InfoAgents.

Within the implementation, each InfoAgent is assigned to one unique cluster. This unique assignment to a highly specialized subset of the InfoAgent population will minimize the quadratic mingling that occurs amongst InfoAgents in any given Café.

It is clear that the addition of dynamic clusters to the Café Space is valuable and provides for a powerful information sharing mechanism. However, we feel it appropriate to couple the dynamic clusters with the already extant specialised Cafés, since we conjecture that specialised Cafés allow agents to select sites to approach simply for their recognised 'expertise' in a given topic, and allow sites to successfully position themselves as specialised information servers. At the same time, adding dynamic clustering to this facility allows more genericity and therefore utility to non-specialised InfoAgents.

6.4. Information Sharing

People create InfoAgents to work for them, but their purposes may not be the same. For example, some people may create InfoAgents for searching for information, while others may create them for distributing information to relevant people. One of the important aspects of the information retrieval and provision is to share information among InfoAgents. In information sharing, an InfoAgent can receive information from and provide information to other relevant InfoAgents. We use keyphrases to represent concepts such as the user's interests, the contents of a document carried by an InfoAgent, and the information being searched. InfoAgents use these keyphrases to find other relevant agents to exchange information. In addition, each of these keyphrases is weighted for importance (on a scale from 0 to 100) by the user. These weights can be used in matching process (and indeed in the filtering process when an InfoAgent is delivered to a Client).

There are several different methods available to the Café Managers when mingling the InfoAgents in the Café, from extremely simple to relatively complex. The following sections describe the matching techniques we have implemented for ACORN to date. Section 7 presents evaluation experiments and timing results.

6.4.1. *Exact Matching*

In ACORN's original incarnation, two InfoAgents shared information whenever there was at least one keyphrase match between the two of them [22]. As was stated above, InfoAgent document metadata and associated owner interests are presently described in terms of simple keyphrase lists with possible associated weights. Thus, each InfoAgent carries two lists: one to describe the information it represents, and one to describe owner interests. In Café mingling, both of these lists are consulted in order to match InfoAgents. This is because we make the assumption that the creator of a document is interested in what the document contains, even should a keyphrase only be listed in document keyphrases, not in owner interests. Although a simplistic assumption, we feel it can potentially increase the validity of InfoAgent recommendations.

The mingling process is in fact relatively straightforward and in its simplest form is an exact keyphrase matching exercise, where each InfoAgent's keyphrase, from both owner and information lists, is compared with each other InfoAgent's keyphrase. If any match, the document in question is deemed of relevance to the other owner, and the other owner's address is added to the Recommended list, along with a list of their interests (derived from combinng owner and information keyphrase lists). If there is no match, the information can still be added to the Known list for potential future use, and to learn community knowledge. Note that, if an address has already been visited or recommended, or is on the Known list, the information is not duplicated.

Figure 6 shows this process in a slightly simplified form. For the sake of simplicity, we have neglected to add the keyphrase lists to the agents at the end of the mingling process. Ordinarily, at the end of the process, the lists would hold both address and interest information. For the sake of clarity, we describe the mingling process shown in Figure 6 here.

At the onset of the mingling process, there are three InfoAgents in the Café. These belong to users *A@addrp1*, *B@addr-2*, and *C@addr-1*. For simplicity's sake, we refer to these as A, B, and C. Note that,

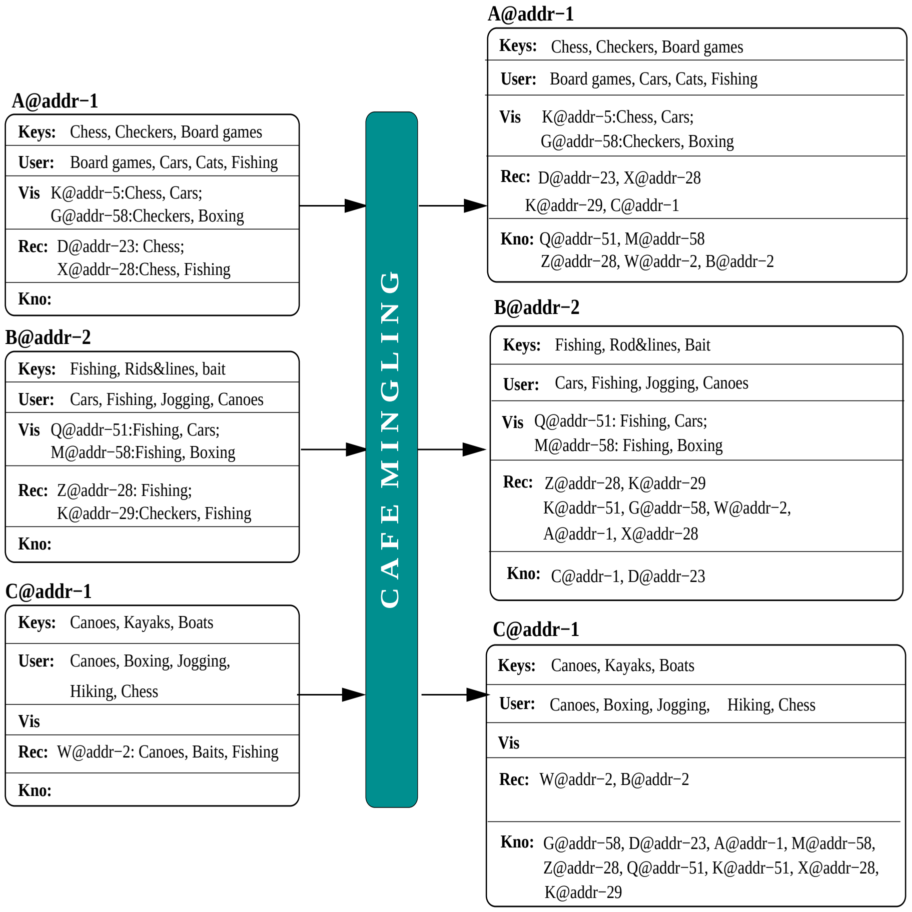

Figure 6. Information Sharing between Agents — Mingling in the Café

although A and C come from the same local Server, they need not be aware of each other in the first place, as is the case here. Note also that it is possible for one user or InfoAgent to know of another while the reverse need not be the case (at least until an InfoAgent visits the known other). Finally note that the addresses (e.g., *A@addr-1*) serve in this example as placeholders for real ACORN addresses, which are in fact exactly the same in format as standard email addresses.

In this example, A's InfoAgent represents a document whose keyphrases (**Keys**) are *Chess, Checkers, and Board Games.* Also, A's interests (**User**) are listed as *Board Games, Cars, Cats, and Fishing.* As can be seen, B's and C's InfoAgents hold similar lists.

As with all InfoAgents, A's, B's and C's InfoAgents have three lists: Visited (**Vis**), Recommended (**Rec**) and the Known (**Kno**) list. Here, A's InfoAgent has already visited *K@addr-51* and *G@addr-58.* From these visits, the InfoAgent has learned that G has interests in *Chess* and *Cars*, and G has interests in *Checkers* and *Boxing* (this is a good thing for both since the InfoAgent represents information about Chess, Checkers and Board Games. . .). The InfoAgent also has two users on its Recommended list: *D@addr-23*, whose interest as far as the InfoAgent knows is only in *Chess*, and the slightly more interesting *X@addr-28*, who at least as well as *Chess* has another interest in *Fishing*. The Known list is empty. Again, similar situations, with different users and interests, pertain to B's and C's InfoAgents. The Known list is empty for all the InfoAgents.

The mingle process simply matches keyphrases in its Exact Matching incarnation. In this example, A and B InfoAgents are compared first, then A and C, followed by B and C. In fact, there's no real reason to the order as long as every InfoAgent gets compared with every other. In the first comparison, A with B, each keyphrase in A's lists of **Keys** and **User** keyphrases are first matched with B's **Keys** list. Note that the **User** list is not compared since we are trying to ascertain if the B InfoAgent should visit A. In this instance, there is a match — A likes *Fishing* and B's InfoAgent has a keyphrase for *Fishing* in its document description list. As a result, *A@addr-1*, along with her interests: *Chess, Checkers, Board Games, Cars, Cats,* and *Fishing* is added to B's InfoAgent's Recommended list. in due time, B's InfoAgent will visit A. The reverse does not attain though — there are no matches between B's interests and the information A's InfoAgent represents. However, in the spirit of building community knowledge, *B@addr-2*, along with a list of B's interests: *Fishing, Rods & Lines, Bait, Cars, Jogging, Canoes* are added to the Known list of A's InfoAgent. Somewhere down the line, this could be of use to A (for example were A to write a missive on Fishing by Canoe) or to another InfoAgent that A's current InfoAgent might bump into. Again note that, although the interest lists are copied, they are not shown in the diagram.

The process continues with more comparisons and matching until all InfoAgent details are compared and exchanged. Figure 6 shows initial and final states for the interested reader to work through.

With a little thought, it soon becames clear that using such exact keyphrase matching, an InfoAgent may collect extremely large amounts of information from other InfoAgents and finally deliver this information to its owner. In our opinion, this is not an *efficient* way of searching and distributing information. In addition, the comparisons between keyphrases are always $O(n^2)$ where n represents the number of keyphrases, and for large numbers of keyphrases in large numbers of InfoAgents, this is clearly a problem.

This matching method makes it almost impossible for people to extract *relevant*, useful, and interesting information from information sources. In the same vein, it is also impossible for people to distribute their information to other *relevant* people.

The next section describes techniques we used to alleviate some of these problems - Substring Indexing and Cosine Similarity Measures.

6.4.2. *Two Different Similarity Measures*

Similarity measures are widely used in the information retrieval (IR) community and are sometimes referred to as the matching functions, the correlation coefficients, or the selection algorithms. In IR, similarity measures are the mechanisms through which the retrieval software makes a comparison between document and query representations to effect retrieval [11]. A similarity measure is any function that assigns a value of matching coefficient to a pair of vectors. Each vector includes a set of attributes that characterizes an entity. Similarity measures have been used to cluster documents and determine similarity between a query and a document. When the similarity measures are used to cluster documents, they are designed to quantify the likeness between documents. If one assumes that it is possible to group objects in such a way that an object in one group is more like the other objects of that group than it is like any object outside that group, then a clustering method enables such a group structure to be discovered. When these measures are applied to determine the similarity between a query and a document, they serve in matching or ranking. In this section we adapt two similarity measures used in IR to share information among InfoAgents in the Café, the substring indexing method [24] and the Cosine measure method [25]. For more information, see [33].

6.4.2.1. *Substring Indexing*

Substring indexing is simply the proportion of common terms in two documents (or InfoAgents). It is expressed as follows.

Let a_i represent i-th agent and $K^{(a_i)}$ represent the set of keyphrases associated with agent a_i. Further, $K^{(a_i)} = \{k_1^{a_i}, k_2^{a_i}, \ldots, k_m^{a_i}\}$ where $k_j^{a_i}$ represents the j-th keyphrase of agent a_i. Adapting the substring indexing method, the similarity of two agents, S_{a_1,a_2}, is the proportion of common keyphrases carried by the two agents and is given as follows:

$$\mathbf{S}_{a_1,a_2} = \frac{2.|K^{(a_1)} \cap K^{(a_2)}|}{|K^{(a_1)}| + |K^{(a_2)}|} \tag{1}$$

where $|.|$ represents the cardinality of a set. $|K^{(a_1)} \cap K^{(a_2)}|$ is the number of keywords shared by both agents. S_{a_1,a_2} is equal to 1.0 for agents with identical sets of keyphrases and 0.0 for agents with no common keyphrases.

6.4.2.2. *Cosine Measure*

In the substring indexing similarity measure, all keyphrases are considered to be of the same importance. However, in reality in most of the cases, the keyphrases have different weights representing their importance.

Let $w_j^{(a_i)}$ represent the weight of the keyphrase $k_j^{(a_i)}$ and $\mathbf{K}^{(a_i)}$ represent the keyphrase vector defined as follows:

$$\mathbf{K}^{(a_i)} = \left\{ \left(k_1^{(a_i)}, w_1^{(a_i)}\right), \left(k_2^{(a_i)}, w_2^{(a_i)}\right), \ldots, \left(k_m^{(a_i)}, w_m^{(a_i)}\right) \right\} \tag{2}$$

With the Cosine measure method [25] we define the similarity of two InfoAgents as

$$\mathbf{S}_{a_1,a_2} = \frac{\sum_i w_i^{(a_1)} w_i^{(a_2)}}{\sqrt{\sum_i \left(w_i^{(a_1)}\right)^2} \sqrt{\sum_i \left(w_j^{(a_2)}\right)^2}} \tag{3}$$

The Cosine measure method finds the cosine of the angle between two vectors defined in the space of keyphrases. Note that each keyphrase represents a unique dimension in this space. S_{a_1,a_2} is equal to 1.0 for InfoAgents with identical keyphrase-weight pairs and 0.0 for agents with no common keyphrases.

7. Evaluation

ACORN has been implemented in Java from its conception, and has passed through several incarnations since then. Presently, ACORN is implemented using J2SE, and uses JSP for its user interface requirements. We have found that the combination of Java's powerful communications and data abstraction capabilities coupled with the flexibility and portability of JSP and HTML as interface description languages provides for an efficient and easily extensible modular system.

This section presents some of the experiments we have carried out with ACORN in terms of timing and evaluation in the past [2, 33]. It also presents some of the changes we have made to the system in order to facilitate such testing, and discusses further evaluation possibilities.

7.1. Information Sharing

7.1.1. *Comparison of Similarity Measures*

This section describes the evaluation of two of the similarity measures explained in Section 6.4.2, the sub-string indexing method and the Cosine measure method. For comparing the methods we use a sample of 49 article abstracts from technical magazines and web pages. We consider 49 ACORN agents each carrying keyphrase-weight pairs of only one of these articles.

The 49 article abstracts are classified into five categories based on their content: Category 1 (articles 1-11), Category 2 (articles 12-22), Category 3 (articles 23-32), Category 4 (articles 33-42), and Category 5 (articles 43-49). Each category represents a specific topic. The text document indexing software Extractor [28] is used to obtain keyphrases and their weights. The details of these articles and the output obtained using Extractor are given in [32]. One article from each category was selected and the similarities between this article and the entire set of articles from all categories were calculated using equations (1) and (3). Note that the similarity between two articles is commutative.

Figures 7 and 8 show the similarity measure results using the Cosine measure method and the substring indexing method. It is seen from Figure 7(a) that for the article 12 of Category 2, the $S_{a12,a12} = 1.0$. Further, the degree of similarity of article 12 with the rest of the articles of its category is much higher than the similarity with articles from other categories. For the substring indexing method, as seen in Figure 7(b), the $S_{a12,a12} = 1.0$. However, there is not much difference seen in the degree of similarity with articles from its own Category 2 as well as Category 1. Thus, the Cosine measure method is found to be superior than the substring indexing method. Similar observations can be made from Figure 8 and additional results given in [32].

7.1.2. *Recall-Precision with Cosine Measure*

In information retrieval, precision and recall are commonly used to measure the effectiveness of the retrieval methods. Recall measures the ability of a method to retrieve useful documents. It is defined as the proportion of relevant material retrieved. Precision, conversely measures the ability of a method to reject useless materials, and is defined as the proportion of retrieved material that is relevant to a query. In the following discussion, we adapt the definitions of precision and recall used in information retrieval to explore the quality of information exchange among InfoAgents.

The InfoAgents in a Café are divided into two sets, the set of InfoAgents, A, that carry relevant information and those that do not,

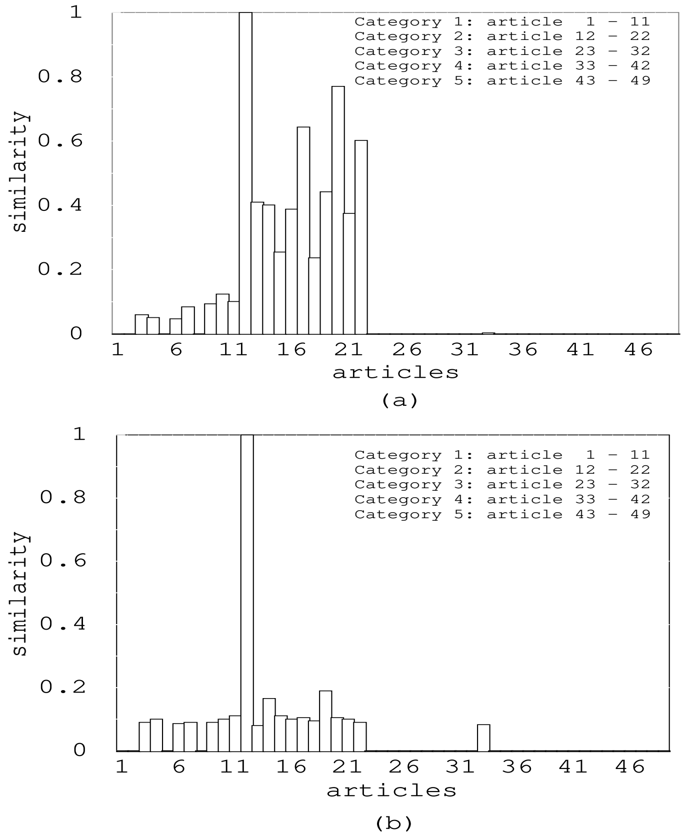

Figure 7. Similarity of article 12 (from category 2) with all articles: (a) the Cosine measure method, (b) the substring indexing method

$\bar{A}$. Let a represent the number of InfoAgents from A that are selected by the similarity measure method for information exchange. Further, $b = |A| - a$. Let c denote the number of InfoAgents also selected from $\bar{A}$ for information exchange and finally $d = |\bar{A}| - c$. Based on these notations, the precision (p) and recall (r) measures are defined as follows:

$$p = \frac{a}{a+c} \tag{4}$$

$$r = \frac{a}{a+b} \tag{5}$$

The above precision and recall definitions were sufficient and appropriate for the early IR systems, which were merely capable of boolean searching. In the early IR systems, a user's query was expressed as a boolean combination of keywords, and the systems retrieved the doc-

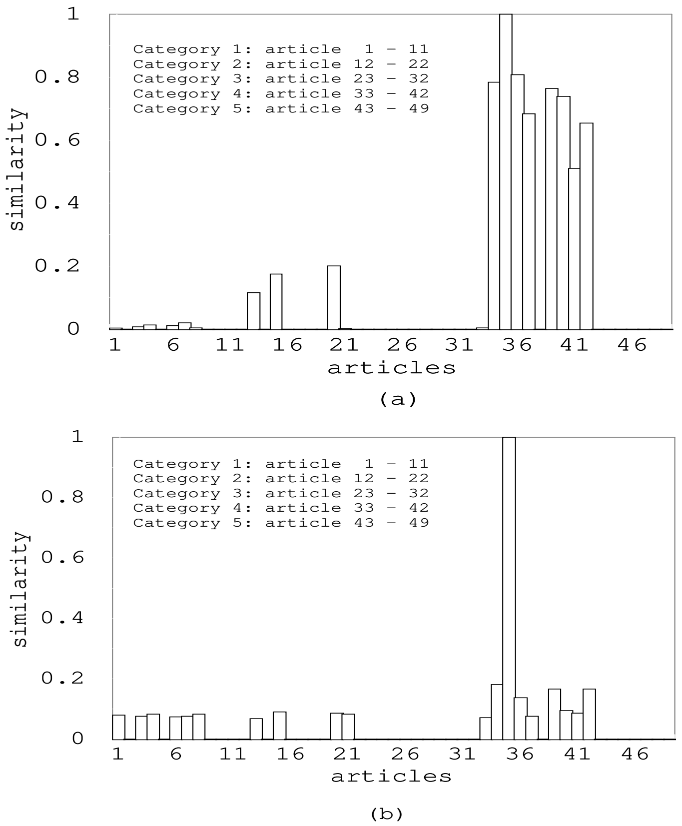

Figure 8. Similarity of articles 35 (from category 4) with all articles: (a) the Cosine measure method, (b) the substring indexing method

uments matching the constraints represented by the query [4]. Unlike early IR systems, our current information sharing system uses vector-space model based on keyphrase-weight pairs. It is used to calculate the degree of similarity between two InfoAgents. The owner of an InfoAgent can assign a similarity threshold, η, to his/her agent. An InfoAgent a_1 will share information with another InfoAgent a_2, if $S_{a_1,a_2} > \eta_{a_1}$.

The choice of the threshold will greatly influence the precision/recall of the system. For example, when an InfoAgent communicates with other InfoAgents, if it shares information with a greater number of relevant others, the system's recall value increases. However, when it shares information with larger number of irrelevant others, the precision decreases. In order to take this trade-off into consideration, we evaluate the keyphrase-based information sharing system using average of precision and recall values over the set of 49 articles for different thresholds.

Table I lists the average precision and recall values for a test data set (see [32] for details). Figure 9 depicts the average precision versus average recall for various similarity thresholds for the test data set given in Table I. It is seen that as the similarity threshold is increased, the precision increases whereas the recall value decreases.

Table I. Average precision and recall for various similarity thresholds.

	Similarity Threshold									
	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Precision	0.82	0.9	0.98	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Recall	0.95	0.95	0.93	0.85	0.75	0.64	0.41	0.38	0.21	0.1

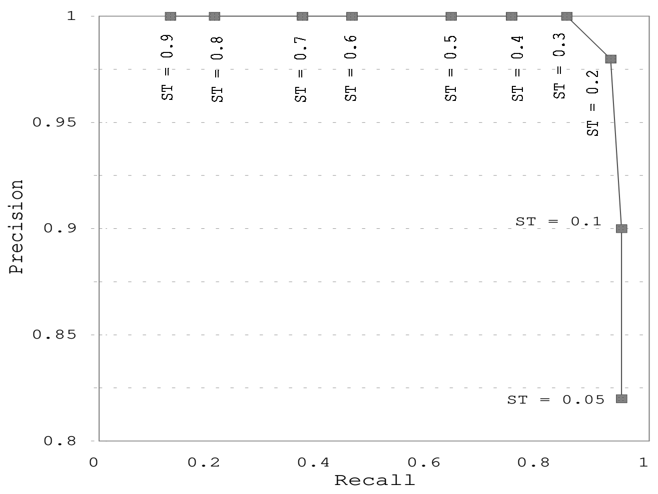

Figure 9. Average precision vs. average recall for various similarity thresholds (η).

7.1.3. *Execution Time with the Cosine Measure*

Experiments were carried out to study how the Cosine measure method affects the execution time of ACORN. For each InfoAgent a set of keyphrases with random weights was generated. We define the *mingle-time* as the time needed for information sharing in the Café. Ten experiments with number of agents ranging from 10 to 100 were conducted for ACORN, with and without the Cosine measure method.

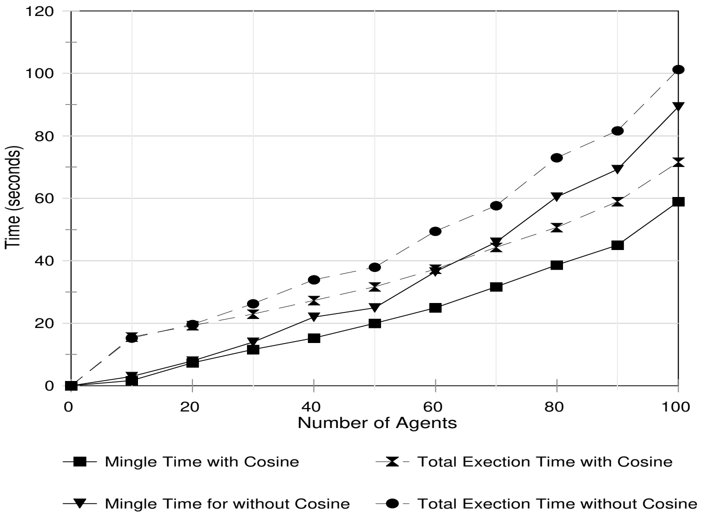

Figure 10. The execution times for ACORN with and without the Cosine measure method.

Figure 10 shows the experimental results. The execution time is the average of three runs for each experiment. It is seen that the mingle time of ACORN with Cosine measure is smaller than that of ACORN not using the Cosine measure. This is due to the selective information sharing carried out in the Cosine measure method. Such information sharing also reduces the total execution time, as can be seen in Figure 10.

7.2. Dynamic Clustering

As documented in Section 6.3, we have adopted a just-in-time approach to Café creation in which we use dynamic K-means clustering to create clusters of similar InfoAgents on the fly. This should result in a more closely matched sharing of information in the mingling of the InfoAgents in the cluster. There are, however, issues which need to be measured in this process, and this section presents some of the experiments and associated results we have obtained. This portion of the evaluation of ACORN should be considered to be preliminary since we are in the process of optimising the clustering process (which will result in better timing results if nothing else). Note also that once InfoAgents are in their clusters, the likelihood of finding similar others is increased over single Cafés and at least as good as multiple Cafés, so we expect a dramatic improvement in mingling results and speed given the already similar InfoAgents.

Our experiments concern the timing and efficiency of clustering for the dynamic K-means clustering process we have adopted.

7.2.1. *Experimental setup*

In order to test the effectiveness of the K-means algorithm, a population of virtual InfoAgents and interests must be established. In this experiment, one hundred randomly generated keywords were used. Each keyword was assigned a random weight value. The virtual InfoAgent is represented as a normalized vector of randomly chosen keywords and corresponding weights. Each established virtual InfoAgent is allowed to enter the Café Space in a serial fashion. As each InfoAgent enters, the Café Controller must decide to generate a new cluster or simply assign the incoming agent to a given cluster. These experiments aim to study the timing associated with the process along with the minimum cluster distance limit parameter of the algorithm.

The controlled dimensions of the study include the following: virtual InfoAgent population, keywords per InfoAgent, and minimal cluster distance. The virtual InfoAgent population represents the total number of InfoAgents that will be assigned to clusters. It is allowed to vary betwen one and fifteen hundred with increments of sixty-four. The keywords per InfoAgent are the total number of keywords that will be assigned to each InfoAgent. They vary from two to twenty with increments of two. Lastly, the minimum cluster distance limit represents the minimum distance that an InfoAgent must be from its closest cluster in order to generate a new cluster. It is allowed to vary from 0.2 to 0.8 with increments of 0.02.

Through the incremental changing of these variables, various times for process completion are noted and conclusions are drawn concerning an optimal value for the minimum cluster distance limit.

7.2.2. *Results*

Initially, the process completion time was studied for the case of all InfoAgents having five random keywords. As depicted in Figure 11, the cluster assignment time increases quadratically as the InfoAgent population and minimum cluster distance limit increase. A curious valley develops under the condition of a minimal cluster distance of 0.4. Figure 12 denotes a slice of the surface graph with the minimum cluster distances of 0.2 to 0.8 with increments of 0.2. Both graphs suggest that a value of 0.4 is ideal for the fixed case of all InfoAgents having five keywords.

Figure 13 depicts the scenario of allowing the keywords per InfoAgent to vary as the InfoAgent population remains at its maximum of fifteen hundred. The valley of completion time exists so long as the

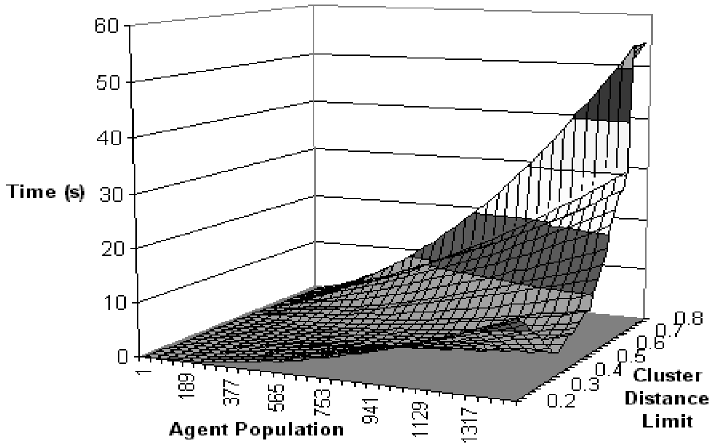

Figure 11. Completion time for different minimum distances with 5 keyphrases per InfoAgent

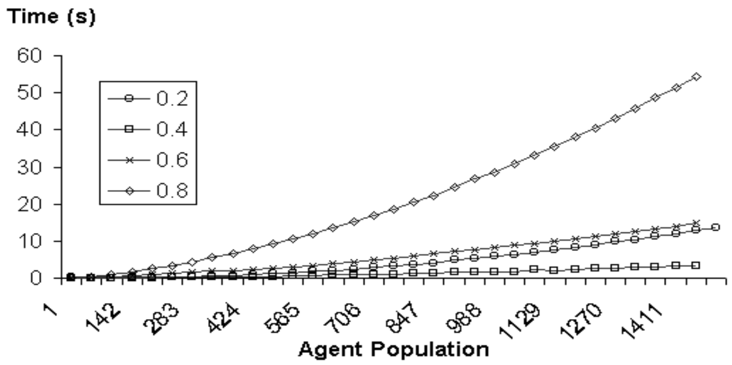

Figure 12. Completion time for different minimum distances with 5 keyphrases per InfoAgent

number of keywords per InfoAgent varies between three and seventeen. As such, the optimal value for the minimum cluster distance limit is dependent upon the average number of keywords per InfoAgent.

In Figure 13, it is assumed that InfoAgents will have an average of five keywords of interest. Hence, the determination of an optimal value of 0.4 is appropriate. In the event that five keywords is not the average, there will exist an optimal value for the minimum distance that can be ascertained via experimentation. Further work will be devoted to automating this process.

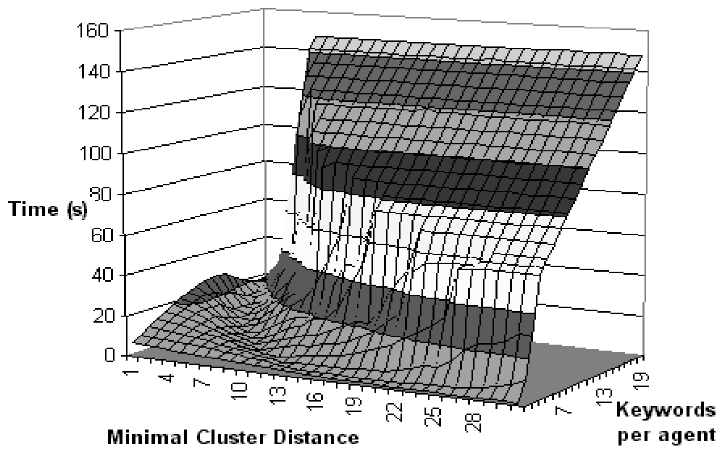

Figure 13. Completion time for varying number of keyphrases per InfoAgent

7.2.3. *Comments on Dynamic Clustering*

The advent of just-in-time Café building, or dynamic clustering, has given a boost to the quest for more relevant information sharing in ACORN. Although our initial implementation is relatively straightforward and does not attempt radical efficiency measures to improve speed, we are nevertheless pleased with the results we have obtained in our performance evaluations. It seems clear that given more efficient algorithms the clustering process would add little or no overhead to the previous systems in ACORN where the Café Controller controlled agent access to Café Space in a different fashion (via boolean keyphrase comparisons).

8. Ideas for Further Work

Clearly, given the complexity of the social structures we are hoping to leverage with ACORN, and the obvious adaptibility and power of the architecture itself, there is much to do. A currently working version of ACORN contains all of the capabilities documented in this paper and we consider it to be the first publically viable ACORN version. However, we also envisage continuing work on ACORN in the future, both short and medium-long term. This section documents some of our ideas for the future extension and application of ACORN.

8.1. Short Term Ideas

8.1.1. *Extending Anonymity*

One of the more powerful additions to ACORN we have applied recently concerns anonymity. Our current implementation, however, leaves something to be desired. Put simply, the way we deal with trust and transitivity of information sharing does not fit our view of how information should behave, especially in more sensitive applications (be they personally or commercially sensitive). Consider, for example, a situation where an anonymous agent, A, seeks some information from another agent, X, that a third party, Y, knows (and X knows Y knows this, while Y knows X knows, and so on...) In the current implementation, X would just give Y's address to the anonymous agent, given certain caveats as mentioned above. Ordinarily, this may be a reasonable thing to do, but in certain circumstances where the information may be sensitive in some way, Y may be justifiably upset with X for revealing his knowledge of this piece of information. Clearly we need a solution to this problem. We are currently working on extending the anonymous server to provide for anonymous forwarding of information, while extending the agents themselves to include a more private list of known community members, and a private list of potential recommendations. The solution to this problem is then as follows: X is free to tell A that she knows of someone who can help, but is *not* free to announce that it is Y. X then contacts Y by sending a *Notification Agent* (simply a new kind of InfoAgent with limited contents) including a reference to A (which is in fact an ID on the Anonymity Server and the address of that Server). Y is then free to contact A by sending an ordinary InfoAgent to the Anonymity Server which can forward it on to the relevant anonymous user, the owner of A. This user is free to contact Y in person or through another agent.

The power of this solution becomes clearer when one considers that Y need not expose himself at all, since he is capable of sending an anonymised agent via a different Anonymity Server (or even the same one if he trusts it enough), which can be replied to by A's owner with another anonymous agent. Neither party need know who the other is whilst still being capable of entering into a useful communication. We believe this is one of the most powerful aspects of ACORN, possibly second only to that of the Café mingling and clustering techniques we use.

8.1.2. *Notification of Recommendation*

The *Notification Agent* is another extension which in fact does not need anonymity to be of use. In any recommendation, it should be possible to inform the recommended party that they have been recommended, to whom, and in what context. This is worthwhile not only because the recommended party has time to formulate appropriate responses to the recommendee, but also because, for example in a commercial environment, suitable consideration can be given to the recommender for their recommending services (a commission, in other words). The Notification Agent allows this process to take place. At every recommendation, the parties concerned have the option of requesting the Café or Server at which they are working to send the Notification Agent to the recommended party — this agent consists of no more information than recommender, recommendee, context and subject recommended (along with usual data such as timestamps, for example). This requires no more than a minimal adjustment to the Café/Server architecture and almost no change to the ACORN InfoAgents (in fact, the Notification Agent can be very easily represented by an InfoAgent).

8.1.3. *InfoAgent Control*

The current Directory Server structure allows system administrators, and to a lesser extent users, to examine the current status of InfoAgents in the network. The Directory Server has its own user interface (implemented via JSP) to enable this examination. However, the Directory Server structure clearly has much further ramifications to the control of InfoAgents, however. At the very least, we will be incorporating the user interface into the Client interface, such that we have a single interface to the ACORN system. Secondly, we will be implementing more InfoAgent control into the system, via various means, to achieve the following:

- Ability of user to recall InfoAgent at any time,
- Ability of user to examine potential path of InfoAgent (Recommended list),
- Ability of user to adapt InfoAgent path (Recommended list) on the fly. This enables a user to control who the InfoAgent visits and who it does not, but note that this control is not forced on the user, since the InfoAgent is capable of making its own decisions.

The Directory Server structure also allows for:

- More intelligent planning of InfoAgent paths, both within and between InfoAgents and users, and the Directory Server,
- Sharing of completed paths for future searches/distribution of similar information – this is in fact a potentially very powerful aspect

of ACORN that can allow information dissemination to speed up based on already discovered communities,

– (Pseudo-) intelligent information sharing between InfoAgents and users based on the thin/fat agent structure (see [6] and Section 8.2.1).

8.1.4. *infoDNA*

The InfoAgents in ACORN are little more than naïve searchers. They accept any information given to them and do not question it – such questioning is for their owner to do.

However, we believe that even the agent owner can be assisted in this task, and for this we are incorporating the concept of infoDNA [21, 20] in our agents. infoDNA is in fact a simple trust handling mechanism which takes its information from community ratings. In this it is similar in concept to the system described in [26], although applied to information. Within the infoDNA system, each agent carries extra information which contains the ratings and signatures of all those who have seen the information. These ratings can be averaged to arrive at a community trust rating for the information. This, allied with InfoAgents' potential recognition of each other (or each others' owners) can allow an InfoAgent to at least prioritise recommended or given information. This solution is a simple addition to ACORN which provides a worthwhile addition to the simple reasoning capabilities of the agents, while giving users additional tools for prioritisation and filtering of incoming information.

8.2. Longer Term Ideas

8.2.1. *Smart Information Sharing*

The current Directory Server structure has allowed us to implement the concept of thin/fat InfoAgents within ACORN. The thin/fat InfoAgent structure (see [6]) allows for only a small amount of information to be sent from Server to Server, and for the Server to request the information it needs (from the Directory Server) at any time. In fact, the current thin/fat InfoAgent structre needs send only the agent unique ID and the Directory Server address. On arrival at a Server, the rest of the InfoAgent data is sent to the Server on request. InfoAgent lifecycle on a Server is then unchanged until migration, then the data is sent back to the Directory Server which acknowledges with the address of the next Server to migrate to. This gives a large amount of control to the user should they require it, while allowing the system to proceed as before to all external appearances. However, the solution is non-optimal and does not take into account the available power of such an architecture. In future, we will be implementing a smart Directory Server-Client communication structure such that the information (InfoAgent contents) that is sent to the requesting Server is based on user requirements, the Server doing the requesting, the context of the request, trust, and so forth. Clearly this raises substantive questions about the nature of trust and the sharing of information, and these are longer term research considerations. As a beginning, the current structure provides more power as well as allowing multiple copies of an InfoAgent to exist and be controlled from a sensible point (the Directory Server) with or without user intervention.

9. Conclusions

ACORN is a multi-agent peer-to-peer system where mobile agents represent pieces of information and are capable of learning and using community knowledge, both accesible and solely within the heads of the community members, in order to route relevant information around a network.

ACORN has progressed from a simple information sharing system in its first incarnation to a sophisticated information architecture capable of supporting privacy, security, multiple information types, an extensible user interface and heterogeneous platforms, and the requirements of the information rich society we live in today. It is able to be easily extended and adapted for purposes we did not envisage at first blush, and remains a powerful tool for the purpose for which it was first designed. Recent additions to the ACORN architecture serve to increase its power via dyanmic clustering of InfoAgents, the ability to more properly track and control InfoAgents from the user's side, and the ability to use secure and powerful privacy techniques.

This paper described the ACORN information system as it stands today and pointed towards future developments (of which we envisage the short term ideas completing within the next six to twelve months). We do not see the completion of these works as the completion of ACORN, which is clearly an architecture with enormous capabilities in directions we cannot foresee. We look forward to working with ACORN further and to seeing it in use. As a first step, we will be releasing ACORN services (at least as they are documented in this paper) in the near future.

For up to date details on ACORN, see

http://www.iit.nrc.ca/~steve/acorn.html

Acknowledgements

The authors would like to express their deepest thanks to the people who have worked with ACORN in the past, without whom the architecture would not exist in its current form. Accordingly, we thank to Youssef Masrour, Leigh Wetmore, Hui Yu (especially for the work on information sharing algorithms documented in Sections 6.4 and 7.1), and Jonathan Carter (especially for the work on Directory Servers, Anonymity Servers, Dynamic Clustering as described in Section 6.3 and the thin/fat InfoAgent). This work has benefitted from consultation with many of our colleagues at the NRC and the University of New Brunswick, to whom, our thanks.

Dr Ghorbani's and Dr Bhavsar's work was partially funded through grants RGPIN 227441-00 and OGP0089 respectively from the Natural Science and Engineering Research Council of Canada.

References

1. Albert, R., H. Jeong, and A.-L. Barabasi: 1999, 'Internet: Diameter of the World-Wide Web'. *Nature* **401**(6749), 130–131.
2. Bhavsar, V., A. Ghorbani, and S. Marsh: 2000, 'A Performance Evaluation of the ACORN Architecture'. In: *Proceedings of the 14th Annual International Symposium on High Performance Computing Systems and Applications (HPCS 2000), Victoria, British Columbia, Canada.* http://www.cs.unb.ca/profs/ghorbani/ali/papers/hpcs00.ps.
3. Broder, A., R. Kumar, F. Maghoul, P. Raghavan, R. Stata, A. Tomkins, and J. Wiener: 2000, 'Graph structure in the web'. In: *Proceedings WWW9.* http://www9.org/w9cdrom/160/160.html.
4. Brüninghaus, S. and K. D. Ashely: 1998, 'Evaluation of Textual CBR approaches'. In: *Proceedings of the AAAI-98 Workshop on Textual Case-Based Reasoning, Madison, WI. (AAAI-Technical Report WS-98-12).* pp. 30–34.
5. Carter, J., A. A. Ghorbani, and S. Marsh: 2001a, 'Just-in-Time Information Sharing Architectures'. In preparation.
6. Carter, J., A. A. Ghorbani, and B. Spencer: 2001b, 'Agent Design Considerations within Distributed Information Retrieval Systems'. In: *Proceedings of the Workshop of Novel E-Commerce Applications of Agents, (in conjunction with the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence – AI 2001), Ottawa, Canada.* pp. 23–28.
7. Chavez, A. and P. Maes: 1996, 'Kasbah: An Agent Marketplace for Buying and Selling Goods'. In: *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96). London, UK.* http://agents.www.media.mit.edu/groups/agents/publications/.
8. Foner, L. and I. B. Crabtree: 1996, 'Multi-agent matchmaking'. *BT Technology Journal* **14**(4), 115–123.
9. Foner, L. N.: 1996, 'A Multi-Agent Referral System for Matchmaking'. In: *The First International Conference on the Practical Appli-*

cations of Intelligent Agents and Multi-Agent Technology, London, UK. http://foner.www.media.mit.edu/people/foner/yenta-brief.html.

10. Foner, L. N.: 1997, 'Yenta: A Multi-Agent, Referral Based Matchmaking System'. In: *First Conference on Autonomous Agents (Agents '97), Marina del Rey, California.* pp. 301–307. http://foner.www.media.mit.edu/people/foner/yenta-brief.html.
11. Gerrie, B.: 1983, *Online Information Systems.* Arlington, VA.: Information Resources Press.
12. Hill, W., L. Stead, M. Rosenstein, and G. Furnas: 1995, 'Recommending and Evaluating Choices in a Virtual Community of Use'. In: *Proceedings CHI'95.* pp. 194–201.
13. http://dublincore.org/.
14. http://www.epinions.com/.
15. Kautz, H., A. Milewski, and B. Selman: 1995, 'Agent Amplified Communication'. In: *Proceedings AAAI-95 Spring Symposium on Information gathering from Heterogeneous, Distributed Environments, Stanford University, CA.* pp. 78–84.
16. Kuokka, D. and L. Harada: 1995, 'Matchmaking for Information Agents'. In: *Proceedings International Joint Conference on Artificial Intelligence, Montreal, Canada.* pp. 672–678.
17. Lawrence, S. and C. Giles: 1998, 'Searching the World Wide Web'. *Science* **280**(3), 98–100.
18. Maltz, D. and K. Ehrlich: 1995, 'Pointing the way: active collaboration filtering'. In: *Proceedings CHI'95.* pp. 202–209.
19. Marsh, S.: 1999, 'Agent Oriented Information - Social Knowlewdge Management'. Invited talk at IKMS Canada '99, International Knowledge Management Summit, Toronto, November 22-24, 1999. Available from http://www.iit.nrc.ca/~steve/Publications.html.
20. Marsh, S.: 2001a, 'And Introducing... SociAware. A New Construct for Social Awareness Utilising Trust'. In: *Proceedings MICON 2001 - In Press.*
21. Marsh, S.: 2001b, 'An Open Standard of Trust'. Presentation given at ENTER 2001 conference (www.enter2001.org), Montreal, May 2001. Available from http://www.iit.nrc.ca/~steve/Publications.html.
22. Marsh, S. and Y. Masrour: 1997, 'Agent Augmented Community Information — The ACORN Architecture'. In: H. Johnson (ed.): *Proceedings CASCON 97: Meeting of Minds, Toronto.* pp. 72–81.
23. Milgram, S.: 1992, 'The Small World Problem'. In: S. Milgram, J. Sabini, and M. Silver (eds.): *The Individual in a Social World: Essays and Experiments. 2nd Edition.* New York. NY.: McGraw Hill.
24. Morita, M. and Y. Shinoda: 1994, 'Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval'. In: *Proceeedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval.* pp. 272–281.
25. Salton, G. and M. McGill: 1983, *Introduction to Modern Information Retrieval.* New York, NY.: McGraw-Hill.
26. Schneider, J., G. Kortuem, J. Jager, S. Fickas, and Z. Segall: 2000, 'Disseminating Trust Information in Wearable Communities'. In: *2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K), Bristol, UK.*
27. Tou, J. T. and R. C. Gonzalez: 1974, *Pattern Recognition Principles.* Reading, MA: Addison Wesley.

28. Turney, P.: 1999, 'Learning to Extract Keyphrases from Text'. Technical Report ERB-1057, National Research Council Canada, Institute for Information Technology, Ottawa, Canada. http://extractor.iit.nrc.ca/.
29. Watts, D. J. and S. Strogatz: 1998, 'Collective dynamics of 'small-world' networks'. *Nature* **393**(6684), 440–442.
30. Weibel, S., J. Godby, E. Miller, and R. Daniel: 1995, *OCLC/NCSA Metadata Workshop Report*. http://www.oclc.org:5046/oclc/research/conferences/metadata/dublin_core_report.html.
31. Wellman, B. (ed.): 1999, *Networks in the Global Village*. Boulder, CO.: Westview.
32. Yu, H.: 2000, 'Keyphrase-Based Information Sharing in Multi-Agent Systems'. Master's thesis, Faculty of Computer Science, University of New Brunswick, Fredericton, NB. Canada.
33. Yu, H., A. A. Ghorbani, V. Bhavsar, and S. Marsh: 2000, 'Keyphrase-Based Information Sharing in ACORN Multi-Agent Architecture'. In: *Proceedings of MATA 2000, Mobile Agents for Telecommunication Applications, Springer LNCS 1931, Springer-Verlag, Berlin*. pp. 245–258.

Address for Offprints: