

ARCHITECTURAL COMPONENTS OF INFORMATION SHARING SOCIETIES

JONATHAN CARTER AND ALI A. GHORBANI

*Faculty of Computer Science,
University of New Brunswick,
Fredericton, NB, E3B 5A3, Canada*

STEPHEN MARSH

*Institute for Information Technology,
National Research Council,
Ottawa, ON, K1A 0R6, Canada*

Two similar multiagent systems have been designed to address the issue of information sharing within a multiagent system. This paper examines architectural components that have been added to our information sharing societies, ACORN and MP3. Through this exploration, we conclude that these components and their underlying concepts can be added to other information retrieval societies.

ACORN consists of a set of information sharing locations referred to as Cafés. Cafés are defined as meeting locations for like-minded agents. Like-minded agents are defined as agents that share a common set of interests. As an example, a café may contain agents that are interested in information relating to cars. A dynamic café clustering method is developed. The performance evaluation of the proposed structure for the café is presented. The concept of a *fat/thin* agent architecture is introduced. This agent architecture allows for minimizing network traffic as agents traverse the network in search of or distribution of knowledge. The directory server component is presented along with its relation to the fat/thin agent architecture. Lastly, anonymity service provider which allows anonymity for users is introduced.

The MP3 society exists with the sole purpose of finding MP3's throughout a given network. Through this society, the core design issues of agent verification, and agent validation are addressed and solutions are presented through respective interface components.

Key words: Agent, fat/thin agents, multiagent systems, clustering, MP3, information sharing.

1. INTRODUCTION

An *information sharing society* is a society of agents that attempt to exchange relevant information with each other in the hopes of satisfying a user's request. Each user can generate query or information agents. A query agent is an agent that traverses a network in search of finding 'useful' information for the agent's owner. Useful information has two definitions: (a) information that directly pertains to an immediate query issued by the user and (b) information that is relevant to the user's personal interests. An information agent is one that represents information relevant to the owner's personal interests and can freely navigate networks without any primary goals. Both types of agents contain the social network of known users connected to the agent's owner along with respective interests of those users.

There are two primary types of goals that must be achieved in any information sharing society. Users have many short term goals of information retrieval. Short term goals are constituted as finding information in as little time as possible based on a specific query. Technologically, these goals are reflected through the use of databases that use index retrieval techniques to collect information from many pre-specified networks. Currently, Google is a popular database that satisfies these goals. Long term goals of information retrieval are constituted as finding information that reflects the general interests of a given user without a specific query or constraint on time imposed by that user. These types of goals have been realized through the use of multi-agent systems. Examples of such systems are the Kasbah framework for e-commerce (Chavez and Maes, 1996), and matchmaking systems such as

Yenta (Foner and Crabtree, 1996; Foner, 1996, 1997) and ACORN (Agent-based Community Oriented Routing Network) (Marsh and Masrour, 1997).

ACORN is a multi-agent architecture designed for information distribution and retrieval within an established network. It embodies the concept of autonomous mobile information in a peer-to-peer community of users and provides an agent-based architecture using community based approaches for information retrieval and provision across networks. ACORN is written completely from scratch in Java.

ACORN can facilitate the exchange of information pertaining to MP3's. Alternatively, a separate society of agents devoted to MP3 information distribution has also been established to address the same issues. In both societies, the primary objective is to solve the problem of information retrieval approach using mobile agents (Jennings, 2000). This solution will be expressed through a design strategy that addresses many of the common issues inherent in mobile agent development.

ACORN's main drawbacks were primarily to do with efficiency and lack of support for socially required capabilities such as anonymity. This paper presents significant architectural concepts that have been integrated into ACORN and/or the MP3 societies. These architectural concepts include:

- the capability of users to remain anonymous (or private),
- the ability of the system to support a *fat / thin* agent structure (see Section 3.4),
- a Directory Server architecture and
- a dynamic café clustering method (just-in-time information sharing scheme).

A *fat* agent is defined as an agent containing all of the knowledge (visited, recommended, and known lists) of the agent. A *thin* agent contains some subset of this knowledge. This architectural extension allows for traversal of thin agents to reduce communication complexity from quadratic to linear order. Lastly, a *café* is defined as a virtual meeting place for like-minded agents (see Section 3.4).

The second society, referred to as the MP3 Society, is implemented using the IBM Aglet Software Development Kit (Lange and Oshima, 1998) written in the Java programming language. Its original purpose is related to the inherent legal controversy surrounding MPEG's encoding technology (Daily, 2000). Many system administrators of commercially owned Internet sites are removing found MP3's from their FTP and WWW sites for a myriad of reasons.

Most commonly, these files are removed to minimize unwanted network traffic and to keep resources allocated to business-related tasks. Many other sites devoted to MP3 distribution are continuously shutdown and restarted due to economic constraints or technical issues.

In both cases, the physical location of MP3's are changing on a daily basis. As such, users seeking MP3's often find broken web links, shutdown FTP sites, and obsolete MP3 file listings. Users can commit several hours searching for a favorite MP3 that inevitably has moved or simply been deleted.

This paper is organized as follows. Section 2 briefly describes the architectures of both information sharing societies. Section 3 examines the proposed architectural components that should belong to an information society including: a directory server, an Anonymity Service Provider (ASP), a dynamic Café, a clustering method as a way of controlling the café population on a given server, the fat / thin agent, a graphical user interface, and agent validation/verification procedures. The experimental results of extensive testing and performance evaluation of the proposed dynamic café clustering method are presented in Section 4. Finally, conclusions of the present study of both systems are summarized.

2. INFORMATION SHARING SOCIETIES

ACORN is a multi-agent based system that uses the concept of ‘information as agent’ to route information around networks (or communities) of people. In ACORN, every piece of information (a document, in whole or in part, an image, a sound, in fact, anything that can be distributed) is represented by an agent. In multiagent systems, agents have widely different definitions.

For our work, an agent is defined as a software entity that has independence from its environment. It performs specific actions based on a set of tendencies but cannot be directly controlled by another agent. Its goals may be explicitly assigned by another agent or a user. It is mobile in the sense that it can move freely from system to system with respect to data contained within. Each agent has some knowledge of its creator, its content, and its community of other known agents and human users. The knowledge of others is represented by a unique identifier along with a corresponding set of keywords reflecting interests that belong to that individual. Lastly, agents can have a planner that allows for various ways of achieving goals.

Figure 1 shows an overview of the architecture of ACORN at a site before recent additions. The primary role of the Client is to provide a user interface to the ACORN system. It provides the creation, browsing, control, tracking and deletion of incoming and outgoing ACORN agents. The ACORN server resides at the point of entry from a network to a site. All mobile agents (information agents or search agents) must enter the site through the Server. The Server’s primary task is to protect the site and decide what kind of mobile agent is allowed in. It also controls mobile agent migration carried out by sending agent cores from server to server. The agent core contains all the information the agent needs to accomplish its task. A café is defined as a virtual meeting place for ACORN agents to share relevant community data.

The power of a multiagent system heavily depends on its social structure, its ability to monitor agents’ activities and its support for group formation. The ACORN architecture as shown in Figure 1 does not support one of the central ingredients of a complex social structure, anonymity. It has no structure in place to provide real-time communication with agents which left the site. The current café structure (static café) provided in ACORN is a simple yet effective information sharing scheme; however, a café with the support for *just-in-time* information sharing (Carter et al., 2002) (i.e., dynamic clustering of agents within the café) will extensively improve the social ability of ACORN.

While ACORN’s goals are to effectively distribute general information of any kind, the MP3 society focuses on the distribution of information related to MP3’s such as their corresponding URL. A primary user objective is to make this task as easy as possible for both the consumer and the advertisers associated with MP3’s. The consumer is defined as the user that is seeking a given MP3. The advertiser is defined as the user that seeks to inform the user of their specialized site that brings together agents that are interested in the particular music genres. Their roles and responsibilities dictate that they must use and interact differently with the sites.

In both societies, the need of a café is exemplified. As well, both societies need a way of tracking agents through the use of a directory server. Lastly, both societies are composed of different classes of individuals using the system in different ways. As such, interfaces must be constructed to accompany different users. The next section addresses the common features needed by both systems in more detail.

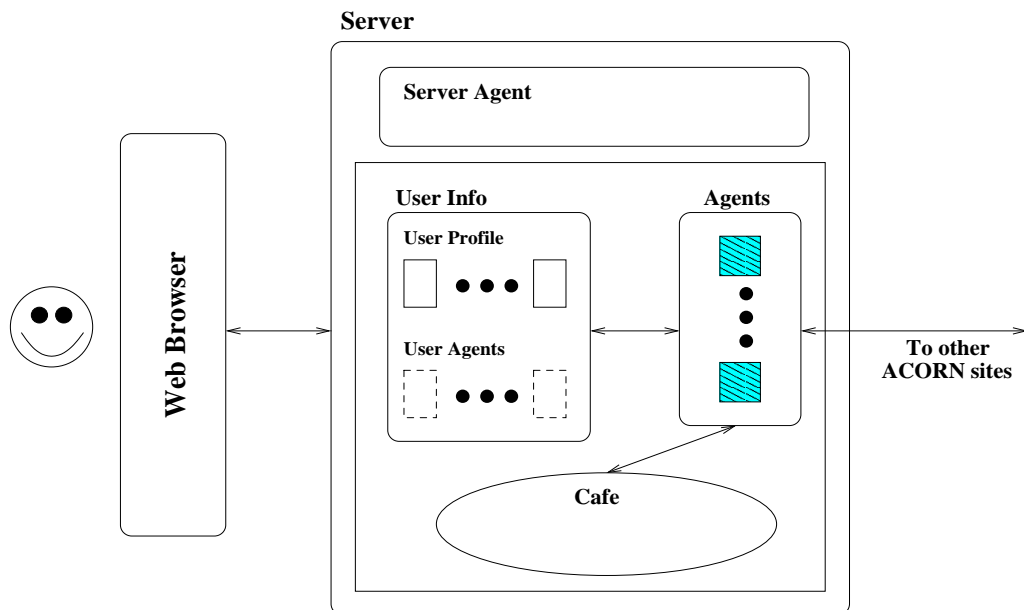


FIGURE 1. Overview of a single ACORN site.

3. PROPOSED ARCHITECTURAL COMPONENTS

The suggested components of an information sharing society are made to improve its functionality. The idea of adding an anonymity server which allows anonymity for agents is to improve social capabilities. The following concepts will be extended in more detail:

- The directory server improves social performance. The directory server acts as a central repository of objects that describe the physical location of agents. In order for users and agents to communicate with each other in real time, they must know of each other's physical location on the network. As agents move from system to system, they must register their change in location with a predefined directory server.
- Through the use of a directory server, clients will be able to communicate with their agents in real time. This instant communication allows the updating of an agent's itinerary by the client or a redefinition of the agent's agenda as it traverses the network. Hence, there is an increase in the performance of the agent and the society in accomplishing goals.
- A fat/thin agent model allows for an increased performance through a reduction in network traffic as agents traverse the network. (see Section 3.4)
- Furthermore, cafés are introduced with the intent of increasing performance through a reduction in total mingling of agents on any given server. This reduction is accomplished through a systematic grouping of 'like-minded' agents into dynamic cafés upon entering the server.

Figure 2 shows the latest architecture of ACORN. In the new architecture, Clients (the user's interface to the ACORN) present their user interface via JSP pages that are displayed to the user on any standard web browser. As always, users interact with the main server to

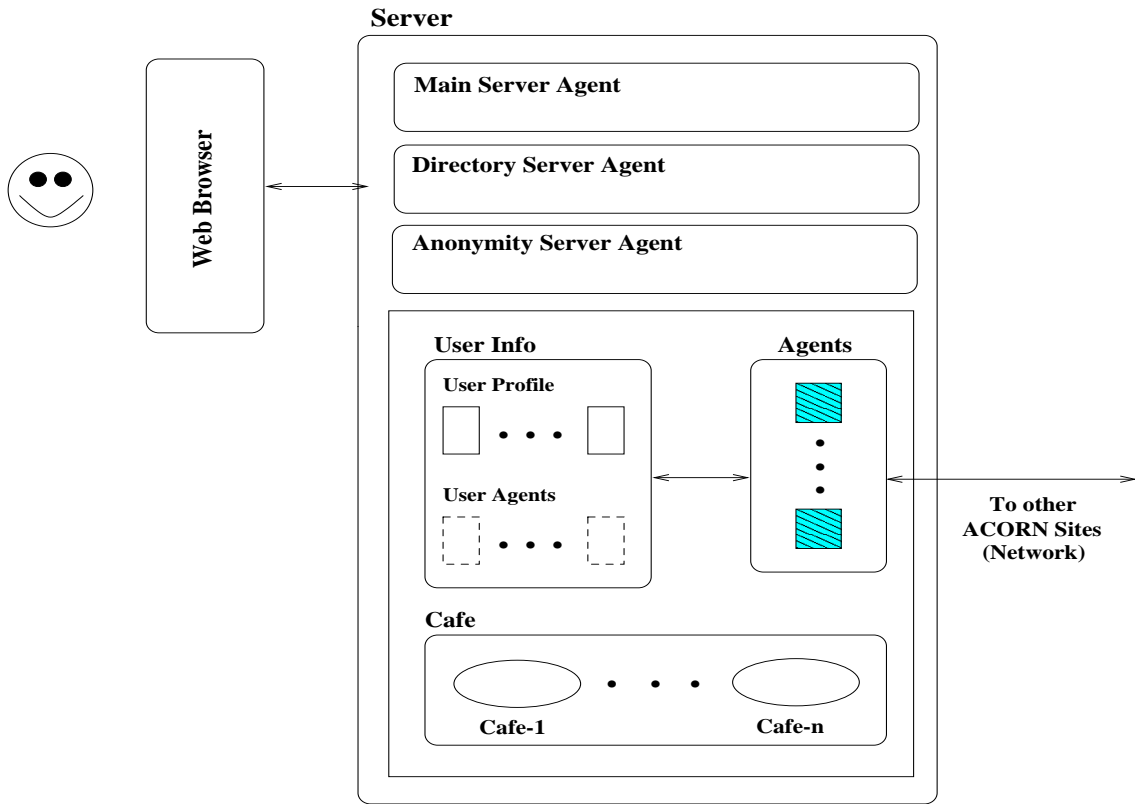


FIGURE 2. ACORN new architecture overview.

create and distribute their agents.

The Directory Server provides the ability to track and control agents which are off site. Clients can choose to directly interact with the directory server through an interface via JSP pages. This interface will act as a focal point for the real-time communication possible between client and agent.

The Anonymity Server includes routines that are used to anonymise information agents before they are sent to the network, and to re-instantiate them as necessary on their return. This is done through the removal of the agent's owner profile information. This information links the agent to an owner's e-mail address, geographical location, personal set of interests, and much more. In order to return to the original owner, the anonymous agent returns to the anonymity server where the removal process is reversed.

Within the overall server, cafés can now be static or dynamic and created on-the-fly. The collective agents' interests change with time; hence, there is a need for dynamic cafés. Static cafés are used as a place for agents with fixed interests known ahead of time. Dynamic cafés host agents with varying interests. The challenge in establishing a dynamic café is to optimize the similarity of interests amongst the currently available agents.

3.1. Agent Configuration

Within the MP3 society, an authorized account must be established for a user. The agent must be individually tailored to suit the needs of the consumer. In Figure 3, the user is presented with a dialog box for inputting various configuration parameters. Primarily, the consumer customizes the agent through the choice of travelling patterns. A travelling pattern dictates how an agent plans on traversing sites within the confines of an established and possibly growing itinerary.

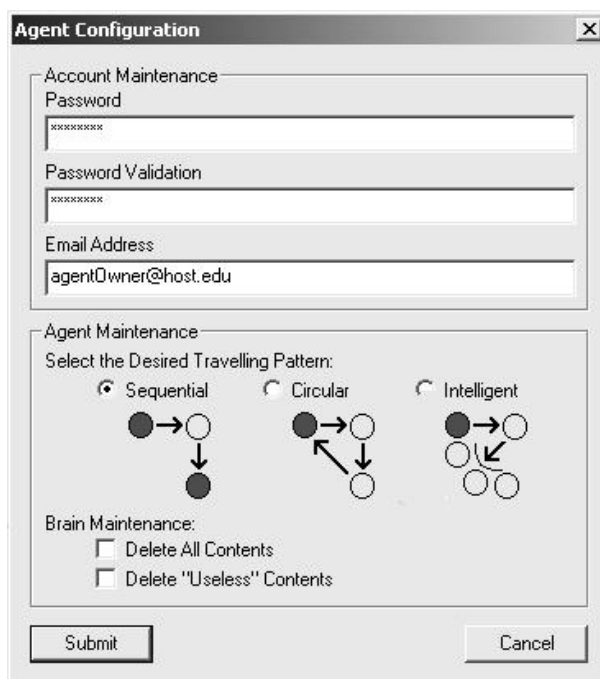


FIGURE 3. Consumer Agent Configuration Snapshot.

The first and simplest method of travel involves the sequential pattern. As the name suggest, a sequential pattern involves a priority queue of site to be visited. This queue grows through communication with fellow consumer and advertiser agents. After all sites have been visited, the consumer agent is directed back to its point-of-origin where it waits for the original user. Fortunately, this origin is manually set by the user. As such, the agent can be directed to return to any online site while the user remains offline. This frees users from having to perform manual searches. Instead, they can turn their computer off while their agent performs an exhaustive search of a society of agents.

The second travelling pattern involves the circular pattern. This pattern can be considered a descendant of the sequential travel pattern. In this case, the inherited behavior of travelling down a queue of sites is still prevalent. However, at the end of the queue, the consumer agent immediately returns to the first member of the queue and repeats the process again. This cyclic activity continues until time has expired as specified by the user. At such time, the rest of the current cycle is completed and the agent returns to the point of origin.

Normally, one would suspect that activity should halt upon the agent discovering that

no more MP3's will be found within the given loop. Instead, the behavior has been defined such that the activity continues in the interest of maximizing the possible number of URL's corresponding to the given MP3. Due to the nature of the problem, we believe that this approach is better due to the unreliable nature of possibly expired URL addresses. Eventually, the agent returns to the predefined point-of-origin.

Primarily, this type of travelling pattern is ideal for advertising agents. An advertiser is going to be interested in marketing their site to specific demographics of agent society. As such, it is assumed that agents of a particular demographic will visit agent sites contained within familiar circuits for that demographic.

Advertisers may not want to promote their sites on other sites based on political or social factors. From this logic, an advertiser may want a highly controlled path for agent travel along with a repeated advertising presence within these familiar circuits. Thus, the circular plan of travel is ideal for advertisers that want control over content and distribution.

The last travelling pattern is a direct descendant of a circular travelling pattern. This plan reflects some level of intelligence that the other plans failed to reflect. Like the circular plan, the intelligent pattern moves in a circular motion of travel until time has expired for the agent. In order to move from site to site, this pattern ranks all unvisited sites based on a probability of success model. This probability of success dictates the likelihood of connecting with the site combined with a previous history of successful searches. If a site connection fails, the probability of success in connecting with the site is reduced by an unfixd percentage and is flagged as unvisited.

As an example, consider an itinerary involving sites A, B, and C that have a 90% chance of establishing a connection. If site A has the highest success rate for finding addresses of previous searches, site A should be picked first. If the connection fails, the probability of establishing a connection decrements by an experimental amount (fixed or variable) and the itinerary is reexamined from scratch using the new values. Eventually, site A will receive the lowest priority if it continues to be offline. To avoid the possibility of starvation, it is flagged as unvisited to allow for a subsequent attempt at visitation.

After the agent travel pattern has been established, an additional 'brain' parameter to the agent must be set. By definition, the brain of an agent acts as the knowledge base for the agent. It stores all associations between MP3's and URL addresses. The brain resides as a secure agent on the host that is defined as the point-of-origin. The brain is used by agents to assist one agent in helping another in searching for MP3 URL's. Naturally, a high level of security must be maintained to verify that an agent's 'brain' is communicating with its owner's agent.

The user can perform various utilities on the agent's knowledge base. These actions are usually performed after the agent has been established and the agent has already learned some associations between MP3 requests and the respective sites. The user may choose to start over fresh and delete all current associations. This is useful for situations of known obsolescence of information. Naturally, obsolescence will occur with time as the sites corresponding to a given MP3 continue to change. If the agent's knowledge base is several weeks old, it is very likely that all associations are invalid and the agent may contribute to globally false associations. As such, it makes sense to perform routine purges.

3.2. Agent Validation

After the account and agent maintenance have been performed for a fledgling agent within the MP3 society, a new agent or an existing agent must enter the agent validation stage. Within the agent validation phase, the user is given the opportunity to communicate their personal desires to the agent. These desires provide the necessary distinctions between

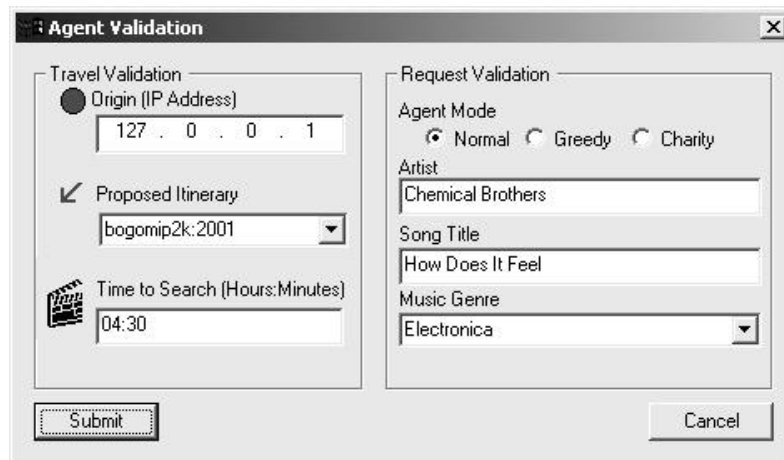


FIGURE 4. Consumer Agent Validation Snapshot

each consumer agent with respect to goals and overt behaviors. The screen snapshot is depicted in Figure 4.

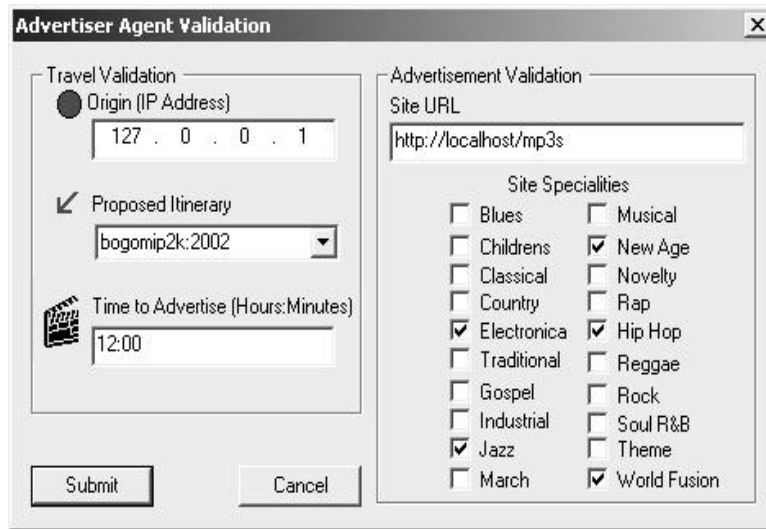
The advertisers are responsible for letting consumers know about their site's existence and the types of music their site specializes in. Agents that are seeking a particular style of music are designed to go to sites that specialize in that style of music so that agents can quickly contact other agents that share a similar interest and possibly a similar knowledge base of that style of music. Some of the agents on the advertised site may be exclusively put in 'charity mode' by the advertisers in order to promote their style of music. An agent in charity mode acts like a benevolent agent. It will not receive any immediate rewards or compensating benefits for its actions.

Like the consumer agent, an advertiser agent must be fully configured before the advertiser can reflect their own personal desires in the agent. After the agent configuration section, the advertiser can now interact with the advertising agent to set the advertising message they wish to broadcast. This is depicted in Figure 5.

As seen in this figure, the advertiser must establish the point-of-origin, as required by all agents. Furthermore, the advertiser must manually establish the itinerary of sites to visit. It is best to allow the advertiser complete control over which sites are going to be visited as the itinerary may be based on marketing, political, or other business decisions that are of far greater importance than allowing for a free roaming advertiser agent. The equivalence of the expiration time of the consumer agent is the time to advertise within the advertising agent. After an advertiser's time has expired, it returns to the point-of-origin. Lastly, the actual message of the advertisement must be defined. This message includes the address of the site along with the genres of music it is trying to attract.

3.3. Anonymous Agents

In our information societies, inherently open architectures exist. Agents can interact and be examined by anyone at anytime. It is quite easy for an agent to acquire a significant amount of information about a user by interacting with the agents they release onto a given network. Currently, agents reflect the unique address of the user along with geographical,



The image shows a dialog box titled "Advertiser Agent Validation" with a close button (X) in the top right corner. It is divided into two main sections: "Travel Validation" and "Advertisement Validation".

Travel Validation:

- Origin (IP Address): A radio button is selected, and the text box contains "127 . 0 . 0 . 1".
- Proposed Itinerary: A dropdown menu shows "bogomip2k:2002".
- Time to Advertise (Hours:Minutes): A text box contains "12:00".

Advertisement Validation:

- Site URL: A text box contains "http://localhost/mp3s".
- Site Specialities: A list of checkboxes with the following checked items:
 - Electronica
 - Hip Hop
 - Jazz
 - World Fusion

At the bottom of the dialog are "Submit" and "Cancel" buttons.

FIGURE 5. Advertising Agent Validation Snapshot

social, and intellectual knowledge of that user. A querying agent will contain the actual query generated by that particular user. Over time, a profile of the user can be built by examining this data with the user's address as a key. Such profiling could lead to data mining and flagrant abuse of that knowledge.

Anonymous agents offer a way of disassociating a user with their particular agents. As well, anonymous agents can be selectively stripped of the geographical, social, or intellectual knowledge of the user. Without the option of anonymous agents, the users' lack of confidence in the system will lead to underutilization. This will undermine the effectiveness of the system distributing information.

Within our information sharing society, a user must initially send an agent to a designated ASP. This server saves the original incoming agent that reveals the true identity of the agent and generates an appropriately anonymous agent without the sender's e-mail address. The server proceeds to send it out to the society with the home defined as the ASP instead of the original sender's address. Upon receiving the updated agent, the ASP merges the incoming anonymous agent with the original agent and sends it back to the sender.

Naturally, there are trust issues that must be resolved concerning the exchange of information between anonymous and non-anonymous agents within the system. There are two primary concerns of trust that include the following: agent-to-agent communication, and agent-to-client communication. In the former, two agents meet in a café. In the latter, an incoming agent wishes to distribute its information to a client's database. For establishing trust, two agents within a café must have a high degree of interests in common with each other along with a list of recommended agents in common. In the case of agent-to-client interaction, both must have an even higher degree in common for trust to be established. We realize that the proposed model does not address all the issues surrounding trust. Subsequent works are needed to extend this model for full utilization of trust.

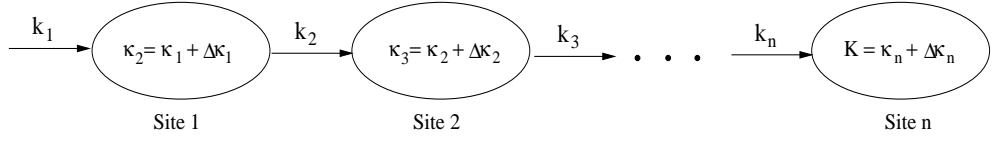


FIGURE 6. Conventional network transmission of agent knowledge.

3.4. Fat / Thin Agent

Agents traverse the network and store their acquired knowledge. Note that ACORN's query agent and information agent are analogous to MP3's consumer agent and advertiser agent, respectively. Over time, one can expect an increase in attained knowledge as query agents socialize with information agents and other query agents in cafés. Naturally, this leads to an increase in the size of an agent and overall network congestion.

In Figure 6, there is an illustration of knowledge contained at each site for a given agent as it traverses the network. Let k_x denote the knowledge of an agent at site x . In the special case of k_1 , each agent has an initial set of knowledge issued by the agent's owner. Let Δk_x denote the new knowledge acquired at site x . Lastly, we define \mathcal{K} as the sum of knowledge transmitted throughout the network. The equation outlined below derives \mathcal{K} as a function of the total number of sites, n , visited by an agent.

$$\begin{aligned} \mathcal{K} &= \sum_{i=1}^n k_i \\ &= k_1 + (k_1 + \Delta k_1) + ((k_1 + \Delta k_1) + \Delta k_2) + \dots + (k_1 + (\Delta k_1 + \Delta k_2 + \Delta k_3 + \dots + \Delta k_{n-1})) \\ &= nk_1 + (1 + 2 + 3 + \dots + n)\Delta k \end{aligned}$$

Assuming that $\Delta k_1 = \Delta k_2 = \Delta k_3 = \dots = \Delta k_n$, we have

$$\mathcal{K} = nk_1 + \frac{n(n-1)}{2}\Delta k.$$

The increase in total transmission of knowledge, \mathcal{K} , appears to be quadratic in nature as a function of number of sites visited. Clearly, this is unacceptable in terms of scalability.

An addition has been made to the architecture to allow for a *fat / thin* agent implementation. A *thin* agent simply contains the globally unique identifier (GUID) of the agent along with the address of a directory server. A *fat* agent contains all of the knowledge encompassed by a given agent. As an agent traverses a given network and collects more knowledge, its knowledge base grows in unpredictably increasing ways. As the knowledge portion grows, the overall size of the agent gets large. Hence, the network traffic load will increase substantially.

The goal of using fat / thin agents is to minimize the network traffic load as agents learn about their environment and move. At the same time, the use of the previously established directory server allows intelligent distribution of knowledge from the server in a context-sensitive way. This results in further reduction of extraneous knowledge across a network.

In Figure 7, we propose the alternate architecture for the transmission of knowledge that reduces the total transmission of knowledge to linear behavior instead of the original quadratic. The original knowledge, k_1 , continues to move with the thin agent as it traverses the network. As the thin agent acquires new knowledge, Δk_n , it is uploaded to a directory server where it is saved. The equation outlined below demonstrates the linear growth behavior of \mathcal{K} with respect to total number of sites visited. Assuming $\Delta k_1 = \Delta k_2 = \dots = \Delta k_n$, we have

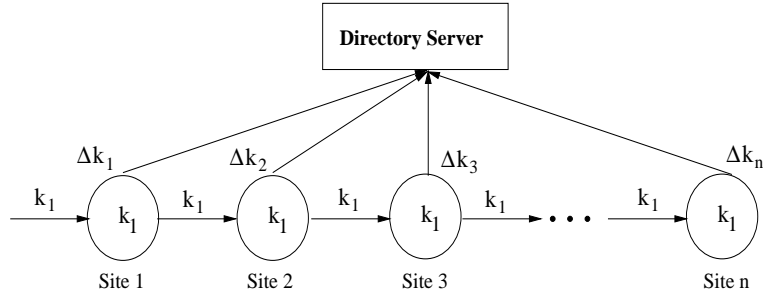


FIGURE 7. Transmission of knowledge using fat/thin agent architecture

$$\mathcal{K} \simeq nk_1 + n\Delta k.$$

For a large number of sites, this architecture offers significant savings over the previous one in terms of transmission reduction of new knowledge.

Savings in the new architecture occur in many different ways. If the thin agent decides to return to its origin, the directory server transmits the new knowledge all at once. This returns to the thin agent and the agent becomes fat. Before it returns home, the new fat agent could filter out any new ‘junk’ knowledge that it deems invalid. This would further reduce the transmission of wasteful information.

The directory server can also act as a planner for a thin agent in the event that the agent has used all of its original knowledge. In such a case, the server could provide recommendations based on the new knowledge set. At such a time when the thin agent uses all of its knowledge, the transmission of k_1 is no longer required. This further reduces the transmission of knowledge and subsequent size of the thin agent.

3.5. Directory Server

Information societies should allow users real-time communication with their agents. Real-time communication is highly desirable for many reasons. Users may wish to receive a progress or status report as to the current location of their query agent. They may wish to know where the agent has been travelling, where it plans to go to next, or how many sites are left to visit before returning to the user. A user may suddenly decide that the agent is irrelevant and must be terminated immediately. The user may want to make manual changes to the agent’s projected travel plans or what the agent is interested in finding. Lastly, the user may want to make manual changes to the agent’s knowledge. From a technical developer’s perspective, real-time communication with agents is desirable because it allows the developer to acquire various debugging statistics that allow for performance enhancement and general design modifications.

There are several different architectures that have been proposed to offer real-time communication with an agent (Carter et al., 2001). We have decided to incorporate a directory registration strategy. Under this strategy, an agent must register itself with a specified server upon entering and exiting a given site it is visiting. Upon entering the site, the agent uploads a notification to the server that contains pre-defined information such as the following: current location, next visiting location, number of sites left to visit, and many more, depending on the user preferences. The most critical piece of information is the current location be-

cause it allows the user to track the location of the agent, hence the possibility of real-time communication between user and agent.

The agent directory comes complete with a directory manager and a separate interface for system administrators that wish to perform maintenance on the directory. The directory manager automatically performs periodic maintenance routines that are predefined by the system administrators.

Aside from the directory manager, the system administrator has the option of logging into a JSP-enabled web interface that allows the system administrator to interact directly with the agent status core database. The system administrator can remove individual entries, clear the entire database, or simply get a refreshed listing of the current contents of the database.

3.6. Clustering

Initially, agents gather together in cafés on a given server in order to mingle and exchange useful information. Cafés can be statically generated based on a predefined set of keywords that are related to a given set of interests expressed by the agents. Upon receiving an incoming agent, the café manager is responsible for assigning the agent to the appropriate cafés. The decision-making process for café membership is based on keyword matching between the agent's interests and the café's collective interests.

One café could be generated on each server that represents all possible interests of the agents. This leads to the socialization of agents that have nothing in common. Query agents that have nothing in common with information agents will not gain any new knowledge through interaction. Both parties have knowledge that does not reflect the others' client interests.

As an alternative, static cafés could be constructed that attract agents with specific interests reflected in the cafés' keywords. For each given set of topics, a separate café could be generated and the like-minded agents could more effectively socialize together. As a trade off, cafés would have to be manually constructed and deconstructed to reflect the changing interests of agents over time.

Such a system of managing cafés is inadequate because it neglects the dynamic nature of cafés. Café members and their interests change over time. As such, the entire café's collective interests should change over time to reflect the changing interests of the current members. Hence, simple keyword matching of static interests may not effectively match a given agent with the right set of agents with the closest set of similar interests. Amongst many similar cafés, keyword matching cannot reflect the degree of interests of the café in a particular interest shared by the agent and the group. We propose the use of partitioning (clustering) algorithms such as *k-means* clustering algorithm as a better way of interest matching. A clustering algorithm such as the k-means algorithm can be used to create groups of agents that are close to each other based on a measure of distance or similarity.

In k-means clustering algorithm the goal is to divide the n input samples into p clusters, minimizing the final variance. It uses the *mean* value of the samples in a cluster as the cluster center. It also makes the assumption that the number of cluster centers that will be required to adequately represent the sample space is known *a priori*. The objective criterion used in the algorithm is typically the squared-error function defined as

$$J = \sum_{i=1}^p \sum_{x \in c_i} |x - m_i|^2,$$

where x is the point in the space (vector) representing a sample and m_i is the mean of cluster

c_i . The data center m_i is defined as,

$$m_i = \frac{1}{|c_i|} \sum_{x \in c_i} x$$

Such a clustering will result in a set $\mathcal{C} = \{c_1, c_2, \dots, c_p\}$ of clusters, where each c_i is a subset of the total number of samples, n . The first step in k-means algorithm is to arbitrarily choose p centers (it is traditional to let k samples randomly chosen from the data set to serve as initial centers). It then repeats the following two steps until the criterion function J does not change after an iteration.

1. Assign each sample to its nearest data center m_i ($i = 1, 2, \dots, p$), forming a new set of clusters.
2. Recompute the new cluster centers.

A k-means based clustering algorithm was implemented to replace the café manager's current decision making process for allocating cafés and assigning agents to those cafés. We have implemented an on-line version of the k-mean algorithm where we find which cluster (café) is the closest to the current agent a . The café whose center is closest wins the competition. This clustering algorithm (referred to as the *dynamic k-means* algorithm) dynamically creates cafés that reflect (more accurately) the interests of their members. This accurate reflection allows agents to quickly and efficiently associate with a more restricted set of agents. Hence, the total mingling time of an agent is significantly reduced along with the load on the server.

The dynamic k-means algorithm requires that each agent's set of interests be represented as a weighted keyword vector, that is, agent a is represented as

$$a = \{(k_1, w_1), (k_2, w_2), \dots, (k_q, w_q)\}$$

where k_i and w_i represent keyword i and its corresponding weight, respectively, and q denotes the number of keywords (agent's interests). Through the use of this vector, incoming agents are assigned to existing clusters based on their Euclidean distance to the given set of cluster centers. If the agent's distance to the closest cluster center is above a certain threshold value, the café manager will generate a new café with a corresponding cluster and assigns that agent's membership to that café. A cluster center represents the statistical mean of all of the interests of the current members. When agents enter or leave the cluster, the center must be adjusted to reflect the change in interests and corresponding weights. In our work, subsequent experimentation is required to determine the correct threshold value. The next section discusses our experimental results concerning clustering.

4. EXPERIMENTAL RESULTS

Previously, ACORN had been tested only in a limited arena, with few users and only a few machines for servers. Obviously, this was an inadequate situation for an architecture which is aimed at multiple users over a large network. In order to carry out adequate performance evaluation and testing, we developed a novel concept of multiple autonomous virtual users (Bhavsar et al., 2000). A virtual user in this testbed is a process which creates agents, using data from individual client cores. In this work, we have used multiple autonomous virtual users within a testbed to carry out extensive testing and evaluation of the proposed clustering

scheme. See (Carter et al., 2002) for detailed results on travelling patterns and the anonymity server.

Experiments must be conducted when we are using the dynamic k-means clustering algorithm for café generation because we wish to determine an optimal value for the minimal distance between two cluster centers. Optimality is defined as the minimum time required to assign an incoming stream of agents to their appropriate clusters. This time includes the actual time it takes to generate new clusters. In other words, we wish to maximize the performance of cluster assignment.

Within this experiment, we were able to represent each agent as a vector of keywords with corresponding weights. These weights were normalized and the keywords were generated using random alphabetic words. Each agent was assigned a fixed number of keywords randomly picked from a list of 100 potential keywords. In total, there was an agent population of 1500 agents. It is expected that this is a good approximation to a highly specialized community of users with a limited set of interests.

Within the experiments, many variables were adjusted in order to measure the total time for cluster assignment. These variables include the following: total agent population, minimal cluster distance limit, and the number of keywords per agent. Agent population proceeded in increments of 50 from 1 to 1501. The minimal cluster distance limit proceeded from 0.20 to 0.80 with increments of 0.02. Lastly, the number of keywords per agent starts from 1 to 20 with increments of 1.

4.1. Cluster Centers

In order to interpret the results of our experiments, it is necessary to understand the underlying process of calculating the center of a cluster. The calculation of the center of a cluster is done as follows. As an agent enters a cluster, each dimension of a cluster center is calculated by simply summing all known weights of the particular keyword found amongst all members of the group and dividing by the total population of that group. The process is repeated for each dimension of the cluster to arrive at a final center that reflects an average normalized vector of keyword interests expressed by the group. As members exit this group, the weights of the collective interests will inevitably change. Along with that change must come a redistribution of weights of interests. As such, the center must be adjusted to reflect the change in weights.

Initially, the number of keywords per agent was fixed at 5 while the other variables were allowed to vary. As depicted in Figure 8, the total time for cluster assignment increases quadratically as the agent population increases. At the same time, the cluster assignment time increases as the cluster distance limit increases beyond 0.40. It appears the value 0.40 is optimal for the case of 5 keywords per agent on average. A side profile of the surface graph is depicted in Figure 9.

The data presented suggests that as the minimal cluster distance limit increases, the process of readjusting the cluster center dominates the total time for cluster assignment. In order to minimize the total time for cluster assignment, one must focus on simplifying the process of adjusting the cluster center. In our case, subsequent research should be aimed at reducing the complexity of algorithms used for calculating the center. It is anticipated that this reduction in complexity will lead to a reduction in time at the expense of accuracy.

After the total time for cluster assignment is analyzed, a curious valley in total assignment time is noted that becomes more pronounced as agent populations increase. This valley suggests that it is possible to further minimize the time required for cluster assignment by altering the total number of clusters established. The data indicates that the total number of clusters created is directly related to the minimal cluster distance limit between any two

clusters. Therefore, it makes sense that this minimized assignment time is a subsequent function of the minimal cluster distance limit.

In order to verify that an optimal value did consistently exist, the number of keywords per agent was changed as the agent population was fixed at its maximum of 1500. Figure 10 denotes the results. An optimal assignment time appears to exist so long as the number of keywords per agent lies between 2 and 18 as depicted in Figure 10. This optimal assignment time can be seen through the valley. More specifically, the value of 0.40 is reaffirmed so long as the average number of keywords per agent remains approximate to 5 as depicted in Figure 9.

Overall, this data indicates that the optimal value of cluster assignment is a function of the average number of keywords per agent and minimal cluster distance. If an average number of keywords per agent can be established, it is possible to discover the optimal value of the cluster assignment time.

In our case, it is concluded that a value of 0.40 for minimal cluster distance is optimal under the assumption that the average number of keywords per agent is 5. In a commercial implementation, statistical gathering of keywords per agent could allow for intelligent adjustment of this parameter based on an individual basis. Subsequent research is needed to develop a learning mechanism for automatic adjust of this parameter over time with the goal of minimizing cluster assignment time.

4.2. Dynamic Clustering

We were interested to know if dynamic clustering ever results in the worst-case scenario of each agent being assigned to their own cluster in a realistic environment. This would result in no social contact amongst agents as all agents would have nothing in common by definition. We define participation within a cluster as membership within that cluster. In Figure 11, average participation of agents per cluster is calculated by dividing the agent population by the cluster population. It is observed that the number of agents per cluster is always greater than 1 as agent populations increase in size. This suggests that clustering will always result in at least some common grouping amongst a set of users in a realistic setting.

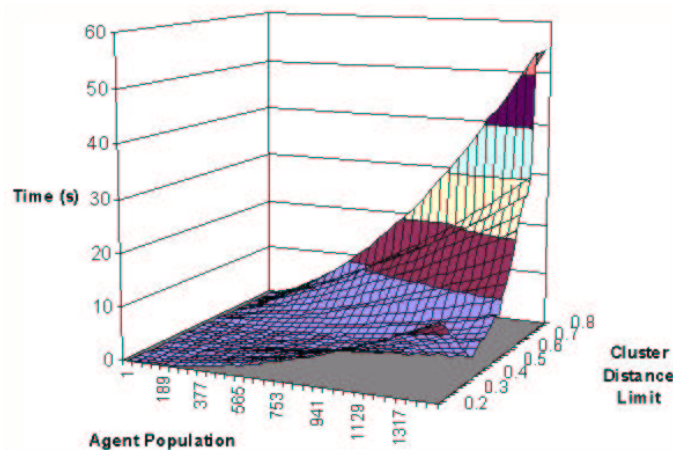


FIGURE 8. Cluster Assignment Time

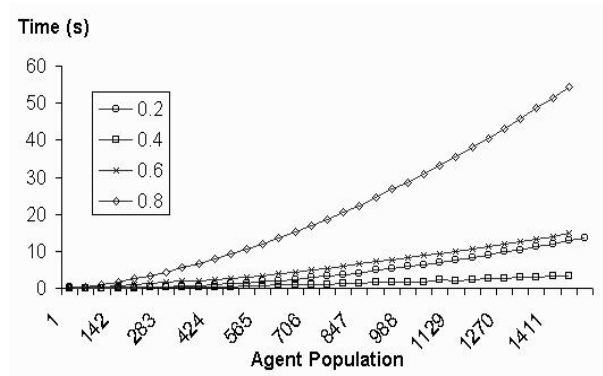


FIGURE 9. K-Means Completion Time

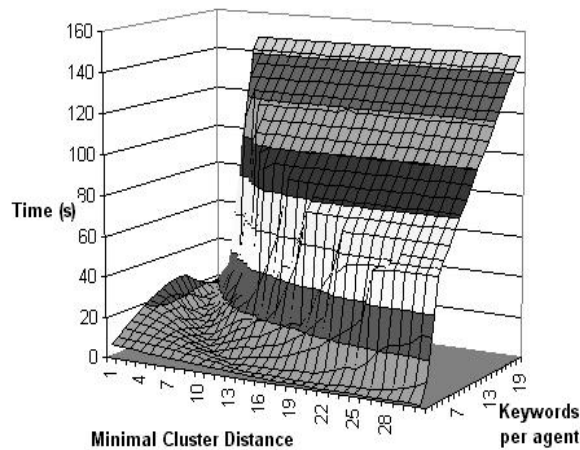


FIGURE 10. Dynamic K-Means Completion Time

However, it should be noted that these results might be sensitive to the algorithm used in random assignment of keywords to each agent.

On a related note, we wanted to gain insight into the relationship between the dimension of the cluster center and the minimal cluster distance limit. Through this understanding, it is possible to further understand the impact of altering the minimal cluster distance limit on the overall requirements of system resources.

This examination is achieved by looking at the magnitude of the cluster dimension of all clusters as they grow in size. By definition, cluster dimension reflects the total number of interests of the agents contained within a given cluster. From a pragmatic perspective, it is important to place an upper bounds on the total number of considered keywords when calculating and adjusting a cluster's center. An upper limit on the total number of keywords was arbitrarily set at 50.

As one sees through Figure 12, the average cluster dimension increases as the minimal

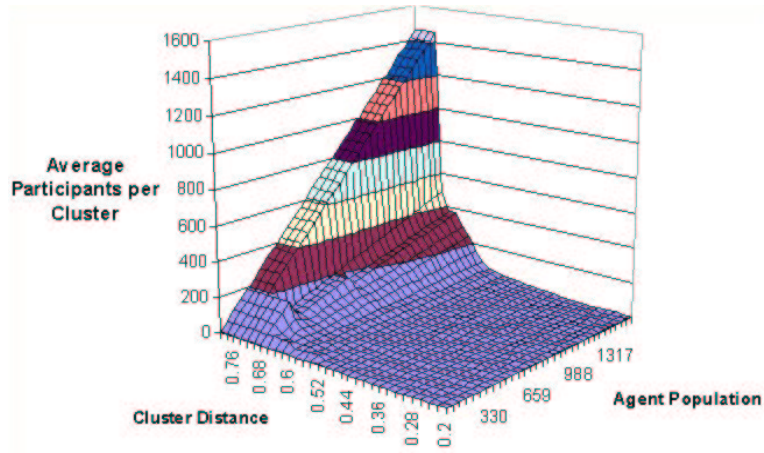


FIGURE 11. Average Participation of Agents per Cluster

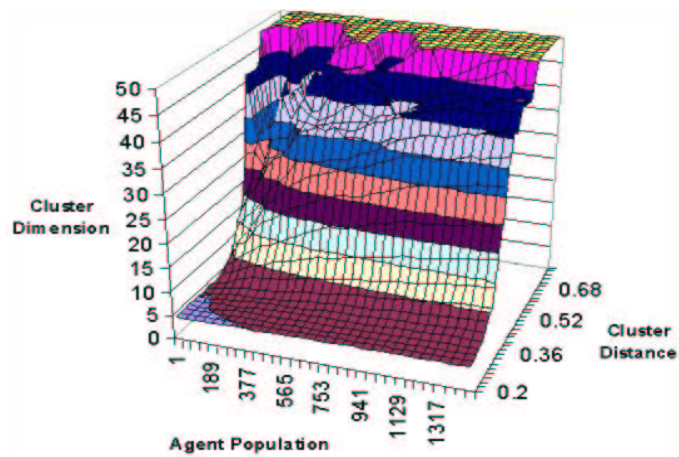


FIGURE 12. Average Cluster Dimension

cluster distance limit increases. This makes sense because as the minimal cluster distance limit increases, the cluster population decreases while the agent population remains the same. Hence, the average participation per cluster increases and the dimension must reflect that. As the minimal cluster distance limit remains constant, the average cluster dimension appears to plateau at some unknown point as agent populations increase with a fixed set of available keywords. From this data, one can conclude that as the minimal cluster distance limit approaches a limit of 0.76, the dimension of any given cluster center has a natural limit. In reality, we should observe the same phenomena provided the number of keywords is limited.

5. CONCLUSIONS

In this paper, two separate but related information sharing societies were analyzed with respect to component design.

First, we introduced two interfaces that address the issues surrounding agent configuration and validation within the MP3 society. The interfaces can be used as an example for other information sharing societies that will need agent configuration and validation stages with the involvement of users.

Then, we presented the concept and implementation of an anonymous agent and its role within the ACORN society. Within any information sharing society, an ASP should allow for disconnection between agent identity and user identity. ASP's bring up the complex issues of trust. All information sharing societies including an ASP must address these security aspects.

The fat/thin agent architecture is presented as a means of reducing the amount of redundant knowledge that is transmitted between servers as agents traverse a network. Information sharing societies must address the issues surrounding the representation of knowledge within the agent and how that knowledge is accessed.

The fat/thin agent architecture requires the use of a directory server. The directory server allows for real-time communication between users and agents. Furthermore, it can act as a planner for the agent with the application of the fat/thin agent. There are cases where it is desirable to allow real-time interaction between users and their agents. Through the use of a directory server, this is possible.

Lastly, the concept of clustering (static and dynamic) is introduced as it relates to cafés within ACORN. Agents must group to allow for efficient mingling and learning from each other. Experiments were conducted to examine the properties of our dynamic k-means clustering algorithm. Through the application of this algorithm, any information sharing society can improve this mingling process considerably.

6. ACKNOWLEDGMENTS

This work was partially funded by the Natural Science and Engineering Research Council of Canada (NSERC) through grant RGPIN 227441-00 to Dr. Ali Ghorbani.

References

- V.C. Bhavsar, A.A. Ghorbani, and S. Marsh. A performance evaluation of the ACORN architecture. *High Performance Computing Systems and Applications*, Nikitas J. Dimopoulos and Kin F. Li (Editors), Chapter 2, pages 9–19, Kluwer Academic Publishers, 2002. www.cs.unb.ca/profs/ghorbani/ali/papers/hpcs00.ps.
- Jonathan Carter, Ali A. Ghorbani, and Bruce Spencer. 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, pages 23–28, June 2001.
- Jonathan Carter, Ali. A. Ghorbani and Stephen Marsh. Just-in-time information sharing architectures in multiagent systems. *To appear in Proc. of the First International Joint*

Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002), Palazzo Re Enzo, Bologna, Italy, July 15-19, 2002.

- A. Chavez and P. Maes. 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, 1996. <http://agents.www.media.mit.edu/groups/agents>.
- Advertising Age Interactive Daily. Napster fans still buy cd's. Uniform Resource Locators (URL), September 2000.
- L. Foner and I. B. Crabtree. Multi-agent matchmaking. *BT Technology Journal*, 14(4): 115–123, 1996.
- L. N. Foner. A multi-agent referral system for matchmaking. In *The First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, London, UK*, 1996. <http://foner.www.media.mit.edu/people/foner/yenta-brief.html>.
- L. N. Foner. Yenta: A multi-agent, referral based matchmaking system. In *First Conference on Autonomous Agents (Agents '97), Marina del Rey, California*, pages 301–307. ACM Press, 1997. <http://foner.www.media.mit.edu/people/foner/yenta-brief.html>.
- Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117: 277–296, 2000.
- Danny B. Lange and Mitsuru Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, Reading, Massachusetts, 1998. chapter 10 pp. 171–177.
- S.P. Marsh and Y. Masrour. Agent augmented community information — the acorn architecture. In Howard Johnson, editor, *Proceedings CASCON 97: Meeting of Minds, Toronto*, pages 72–81. IBM Centre for Advanced Studies, Toronto, 1997.