Multiagent Social Computing

Ben Choi, Louisiana Tech University, USA

ABSTRACT

This article provides a framework for extending social networks to social computing. When people join social networks, such as Facebook and discussion groups, their personal computers can also join the social networks. This framework facilitates sharing of computing resources among friends and groups. Computers of friends and groups act autonomously to help each other perform various tasks. The framework combines many key technologies, including intelligent agents, multi-agent system, object space, and parallel and distributed computing, into a new computing platform, which has been successfully implemented and tested. With this framework, any person will have access to not only the computing power of his or her own personal computer but also the vast computing power of a community of computers. The collective capabilities of humans and computers working in communities will create complementary capabilities of computing to achieve behaviors that transcend those of people and computers in isolation. The future of computing is moving from personal computers to societies of computers.

Keywords: *Cloud Computing, Intelligent Agents, Multi-Agent System, Parallel Distributed Processing, Social Computing, Social Network*

INTRODUCTION

This article extends the concept of socially intelligent computing to provide a framework that facilitates sharing of computing resources among people in communities. When people joins online communities, such as social network sites (e.g., Facebook, Myspace, and Orkut), discussion groups, Wikipedia, or cloud computing sites, their personal computers can also join the communities. This framework provides system design processes, methods, and tools to harness the collective capabilities of humans and computers. Large number of computers working together and helping either other in the communities creates new collectively intelligent systems. Any people on the Internet can join computing communities and so does any networked computers, creating social computing systems ranging from few persons to an Internet-scale cloud of machines and people. The collective capabilities of humans and computers working in communities will create complementary capabilities of computing to achieve behaviors that transcend those of people and computers in isolation (National Science Foundation, 2010).

Current researches on parallel and distributed computing and grid computing attempt to employ a very large number of computers to solve very large computing problems. These researches focus solely on computing speed. They partition a very large computing problem into small pieces, send each pierce to be computed by a computer, and then wait for all the results. This centralized control method of computing

DOI: 10.4018/jwp.2011100105

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

simply ignores the problem of collaboration between computers. On the other hand, current researches on distributed file sharing based on peer-to-peer networks attempt to allow every person to share his/her files and storage spaces through a decentralized network. This distributed file sharing method facilitates sharing of storage spaces but ignores the needs to share computing power.

Our projects attempt to create a platform for computers themselves to collaborate with each other to share computing power. In this platform, computers can help each other both in term of running applications and providing computing power. If a person needs to complete some tasks that are not capable on his own personal computer, his computer will ask other computers for help. His computer makes requests to other helping computers, which complete the required computations and return the results back to his computer. If a person working on certain job needs more computing power, her computer will ask other idle computers for help. Any person using a computer will have access to not just the computing power of his/her own computer but also the vast computing power of a community of computers.

Our projects attempt to create a platform for computers themselves to collaborate with each other to share computing power. In this platform, computers can help each other both in term of running applications and providing computing power. If a person needs to complete some tasks that are not capable on his own personal computer, his computer will ask other computers for help. His computer makes requests to other helping computers, which complete the required computations and return the results back to his computer. If a person working on certain job needs more computing power, her computer will ask other idle computers for help. Any person using a computer will have access to not just the computing power of his/her own computer but also the vast computing power of a community of computers.

Our projects combine many key technologies, including parallel and distributed computing, intelligent agents, multi-agent system, object space, and multicast protocol, to form a unified computing platform. The platform should require minimal user involvement and system administration. To achieve this, our projects extend the notions of intelligent agents (Plekhanova, 2002) and multi-agent system (Shamma, 2008; Dignum, 2009) to conceive of a computer as a whole including its software and hardware as an active agent. A computer acts autonomously like a person in a community. Computers, having various abilities and workloads, join together to form workgroups where they can help each other both in terms of the abilities and the workloads. This in turn requires a share place for the computers to communicate with each other. To achieve this, our projects extend the concept of Object Space to become an Active Space, which can function as a rendezvous, a repository, a cache, a responder, a notifier, and a manager of its own resources. This further requires a computer to be able to broadcast its requests to some or all computers in the workgroup. To achieve this, our projects use multicast network protocols for the communication.

The remaining of this paper is organized as follows. We outline the related researches and discuss the problems associated with current personal computers. We define the framework of Multiagent Social Computing by building societies of computers. Based on the framework, we describe an implementation of a platform for general computing, and then describe another implementation of a platform for high performance. Finally, we give the conclusion and outline the future research.

RELATED RESEARCHES

Current systems for socially intelligent computing provide mediums to facilitate humans to share knowledge while current researches on collaboration focus on allowing people to work together. For instance, Microsoft NetMeeting provides a complete Internet conferencing solution. These researches do not intend to

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

address the problem for computers themselves to collaborate.

Although currently most computers are networked and can communicate with each other, they cannot yet fully work together and help each other. Our society is currently facing three major problems on computing as outline below:

(P1) Personal computer works alone
(P2) Wasting computing resources on personal computers
(P3) Cloud computing overloads the servers

(P1) A personal computer is working alone for one person. The ability of a personal computer depends on the installed software and the processing power of its CPU. If a person needs some new applications and more computing power, the person needs to buy new software and new computer.

Current researches on parallel and distributed computing and grid computing attempt to employ a very large number of computers to solve very large computing problems (Berman, 2003; Foster, 2003; Joseph & Fellenstein, 2004). For instance, Folding@home (Pande, 2008) uses a very large number of personal computers and PlayStations to tackle previously intractable problems in computational biology. SETI@home (Anderson et al., 2002) uses millions of personal computers (Volunteer computing) (Miller et al., 2009) worldwide to search for extraterrestrial intelligence.

These researches focus solely on computing speed and solving very large problems. They partition a very large computing problem into small pieces, send each pierce to be computed by a computer, and then wait for all the results. This centralized control method of computing simply ignores the problem of collaboration between computers.

On the other hand, current researches on distributed file sharing based on peer-to-peer networks (Subramanian & Goodman, 2005; Androutsellis-Theotokis & Spinellis, 2004; Steinmetz & Wehrle, 2005) attempt to allow every person to share his/her files and storage spaces through a decentralized network. This distributed file sharing method facilitates sharing of storage spaces but ignores the needs to share computing power.

(P2) Our society is currently wasting a lot of computing resources on the unused personal computers. Most personal computers are idle most of the time. Home computers are not being used when people are working in their offices and their office computers are not being used when they return home. Personal computers are getting more powerful and yet most of the computing power in our society is being wasted.

(P3) Current trends to cloud computing will result on overloading the servers. Current researches on Cloud computing (Rittinghouse & Ransome, 2009; Velte et al., 2009) focus on delivering Web services. Their proposed method is to move computation away from personal computers into servers, which perform all the needed computation and send the results to personal computers. Powerful personal computers only serve as input/output devices and as displays. Most of the computing power is wasted even when the computer is being used.

The computing platform described in the following sections uses a share space for intelligent agents to communicate with each other. This share space is built upon an extended notion of Object Space (Freeman et al., 1999). An Object Space is a shared medium that simply acts as a rendezvous for agents to meet there either to serve or be served without the knowledge of each other identity, location, or specialization. Other variations of Object Space are JavaSpace (Freeman et al., 1999), IBM's TSpaces (IBM Almaden Research Center, 2010), TONIC (2008), JINI (2010), and TupleSpace (Carriero, 2001). Object Space has also been used for other applications. One of the proposed applications (Engelhardtsen & Gagnes, 2002) utilizes an Object Space as a repository of various roles where agents adapt to changing demands placed on the system by dynamically requesting their behavior from the space. A framework for cluster computing using JavaSpace has been described in (Batheja & Parashar, 2001), which uses a network management module for monitoring the

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

state of the agents and uses the state information to schedule tasks to the agents. JavaSpace has also been used for scientific computation (Noble & Zlateva, 2001). These support the perspective that JavaSpace can be used for high performance computing.

BUILDING SOCIETIES OF COMPUTERS

The future of computing is moving from personal computers to societies of computers. This article describes a framework for networked computers to work in groups where computers can help each other perform various tasks. In this framework, a computer acts autonomously like a person in a community. Computers, having various abilities and workloads, join together to form workgroups and to benefit from belonging to communities. The framework address the three problems outlined in the last section in the following ways:

(S1) Creating a framework for personal computers to work in groups
(S2) Allowing idle computers to help others performing computation
(S3) Allowing cloud computing to use personal computers to provide services

(S1) The framework allows personal computers to work in groups. The computing framework is formulated such that any computer on the Internet can join workgroups. A workgroup can also link to other workgroups forming a whole community of computers. The organization of the computers can mimic the organization of a community. A computer can belongs to several workgroups and benefit from them. Figure 1 depicts a simple organization of twelve computers to form two workgroups. The center of a workgroup is the workgroup manager that is depicted by a ring. The workgroup manager is a computer serving to provide and maintain a share space for communication. Each of the computers is depicted by a circle. Two of the computers join both workgroups and one of the computers joins another workgroup not shown in the figure. The two workgroups are linked together and one of the workgroups is linked to another workgroup not shown in the figure. A computer has the freedom to join or leave any workgroup at any time. It is a free community and the computing community evolves over time like human community.

The ability of a workgroup is more than the sum of ability of its individuals. The function of a workgroup can be considered similar to the function of a discussion group. A computer needs a task to be done and posts the request on the workgroup. Another member in the workgroup reads the request, finishes the task, and posts the result on the workgroup. In this analogy, the workgroup serves two purposes:

(1) it is a shared place for communication and
(2) it is a depository of shared knowledge.

Also similar to hosting a discussion group on a server, a computer can be used as a workgroup manager. The workgroup manager helps maintain the shared place for communication and organize the shared knowledge base.

The workgroup provides a shared knowledge base for the computers. This function of the workgroup is also similar to the function of a discussion group. If we have a question, we may find the answer on the prior postings of our discussion group. In this case, we do not need to ask anyone to help. Similarly, if a computer needs a task to be done and can just find the results on the shared knowledge base, and then there is no need to repeat the computation.

The workgroup provides a mechanism for parallel and distributed computing. If a computer needs two tasks to be done, it can post both of them on a workgroup. Two other computers can concurrently work on the two independent tasks. This analogy can be extended to multiple tasks and multiple computers working in parallel.

(S2) The framework allows idle computers to help others performing computation. When people joins an online communities, such as social network sites, discussion group, Wikipedia, or cloud computing site, their personal

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

computers can also join the community. There will have an option for a person to choose to decide whether one would allow the personal computer to help others performing computation. There will also be statements to assure the person that this will not infringe the personal privacy and computer security (more details later). Once the person agrees, an intelligent software agent will be downloaded and run on the computer, which will then act autonomously like a person in a community to help other members in the community.

(S3) The framework allows cloud computing systems to use person computers to provide services. The computers of cloud computing systems are also belonging to computing communities. They can request other members of the communities for help on performing computations. This extends the concept of cloud computing to form parallel and distributed cloud computing. People use the services provided by the cloud computing systems and on the other hand they contribute to the communities by allowing their personal computers to help others. This forms an Internet-scale of machines and people working together in communities.

In order for the framework to be successful, we need to address social requirements as well as technical requirements. Social requirements, such as protecting personal privacy and computer security, in most cases, are more important than technical requirements. Advanced in technology serves people and their societies. A list of requirements and their solutions are provided in the following:

(R1) Benefits to people
- Providing free services to allow persons to use massive computing recourses

(R2) Persons do not need to do administrations
- Employing intelligent agents and self-organization

(R3) Protecting personal privacy
- Employing share space for communication, receivers not knowing senders' IP address

(R4) Cross platform working on PC, Mac, or Linux
- Using Java technologies

(R5) Protecting the security of personal computer
- Using Java technologies, not allowing access to local files

(R6) Not slowing down personal computing
- Only helping others when persons are not busy, employing mobile-agent technologies

(R7) Fault tolerance
- Employing techniques used in multi-agent systems
-

(R1) The main purpose of the framework is to provide benefits to people. One solution to the requirement is to provide free services to allow people to use massive computing resources. The free services can be in the form of Web-based applications (such as those used in cloud computing), free share-ware software that people can download and use, or a set of API (application programming interface) that allow people to write their own software to use the massive computing resources.

(R2) The framework should not require persons to do administrations. We choose to employ intelligent software agents, which interact autonomously with other intelligent agents in other computers forming a self-organizing community. When persons are joining an online community, they can allow their computers to join the community as well. Once agreed, intelligent software agents will be downloaded and run on the computers. From then on, the persons do not need to do any administrations. When they decided to leave the community, the software will be removed.

(R3) To meet the requirements with high confidence and reliability, it is very important to assure to people that taking part in the computing community will not infringe their personal privacy. To insure anonymity, we employ share space for communication. Any request for help is put on the share space. Any member in the workgroup can pick up any tasks from the share

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

space and help complete the tasks. The completed results are put back on the share space. There is not direct communications between members and the members do not know the IP addresses of other members.

(R4) Since today's computers run three major operating systems, the system should work on PC, Mac, and Linux. We choose to use Java technologies to support the cross platform.

(R5) To meet the requirements with high confidence and reliability, it is also very important to assure to people that taking part in the computing community will not infringe the security of their personal computers. We employ the security features provided by Java technologies and do not allow the installed software to access any local files.

(R6) Taking part in the computing community should not slow down personal computing. When people want to use their own computer, they should have the highest priority. One solution to this requirement is to only allow their computers to help others when persons are not busy working on their computers, which is when their computers are in idle state. Even when their computers are helping others, they can interpret the helping process and use their computer freely. There are two solutions to resume the helping process. One is to resume the process when the local computer is no longer busy. However, this may not be a good solution since we don't know the wait time. The other solution is to send the current state and partial results back to the share space, where other computer will pick up the partially competed task and resume the computations. To be able to achieve the later solution, we employ mobile-agent technologies, which allow computational states be saved and resume computation later in some other computers.

(R7) To meet the requirements with high confidence and reliability, fault tolerance needed to be addressed. The community of computers, each of which runs an intelligent agent (Plekhanova, 2002), working together forms a multi-agent system (Shamma, 2008; Dignum, 2009). When a computer tries to help doing a task but cannot finish it for any reason, any other computer can pick up the task and finished it. When more computers than tasks are available, several computers can perform one single task. A computer can join more than one community and requires help from the communities. This also provides a form of fault tolerance for the workgroup manager. However, the down time of a workgroup manager will affect the community is a larger extend than the down time of a single computer.

Multiagent Social Computing provides a general framework for multiple computers to work together in groups to share computing power and knowledgebase. To show the capabilities, this computing framework has been implemented and tested by two research projects, which are described in more detail in the following sections.

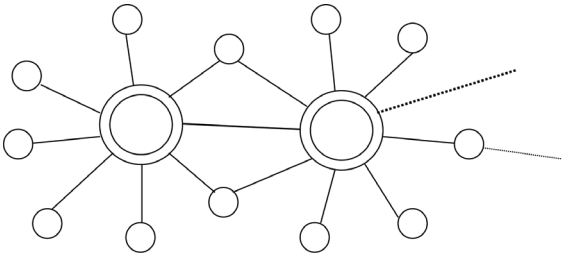
IMPLEMENTING A PLATFORM FOR GENERAL COMPUTING

Based on the framework for Multiagent Social Computing, a computing platform for general computing has been developed, implemented, and tested (Choi, 2010). For the implementation of this platform, several key technologies have been used, including multi-agents, Javaspace, code mobility, caching, and multicast network protocol (Williamson, 1999; Wen, 2001). Figure 1 shows an overview of varies agents joining Space to form workgroups. In general, a workgroup may consist of a large number of agents and the agents may join several spaces (only five agents and two spaces are show in the figure). Agents and spaces are implemented using JINI and Javaspace API's developed by Sun Microsystems (JINI, 2010).

A computer can run many agents and assume many roles. In this platform, we defined three types of agents as shown in Figure 2. A computer requests other computers for help by using Requesting Agents. A computer serves other computers by using Special Function Agents and General Function Agents. A Special Function Agent can only perform a specific predefined task. A General Function Agent can

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 1. Computers joining to form two workgroups (A ring depicts a workgroup manager and a circle depicts a computer)



perform any task that is specified by a Requesting Agent.

Code mobility technique is used in this project to enable the platform for general computing. When a computer needs more processing power, it can send both the program code and the data through a Requesting Agent to a Space. The request, in this case, consists of both the program code and the data is stored on the Space. Another computer running a General Function Agent will monitor the Space and retrieve the pending request. The General Function Agent retrieves both the program code and the data. It uses the program code to process the data and generates the required results, which is then send back to the Space. The results are stored in the Space. The requesting computer, through the Requesting Agent, will then retrieve the results from the Space. In general, a large number of requests can be send to a Space and a large number of computers will concurrently work on the requests, creating a general purpose, parallel and distributed computing platform.

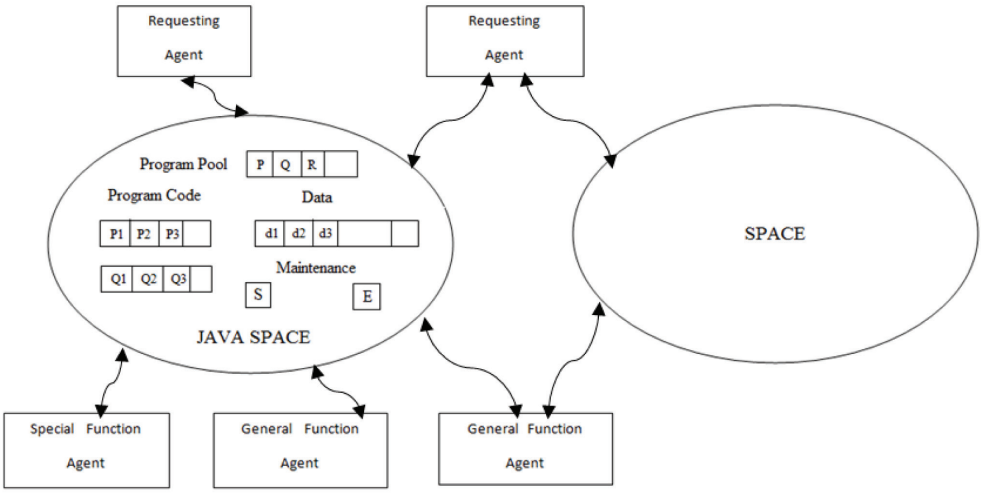
A Special Function Agent, unlike a General Function Agent, can only perform a specific predefined task. In this case, the Requesting Agent does not need to send the program code, but only the name of the specific function and the data. A Special Function Agent retrieves and processes the request that matches its specialty. This processing method is similar to remote execution. However, the communications, in this case, are all through Space. A Requesting Agent does not need to know the destination IP address of a Special Function Agent.

A Space is used in this project not only as a share place for communications but also as a repository of knowledgebase. A sever computer can run the service of a Space. However, unlike other servers, this server does not process requests. Its main purpose is to serve as a share place for Agents to meet. In this project, we use multicast network protocol for an Agent to discover a Space to join. Thus, an Agent does not need to know the IP address of a Space. Through the multicast network protocol, it broadcasts its wish to join a Space. A Space responds to the request, and then establishes direct communication with the Agent. An Agent communicates with a Space, by placing requests on the Space and by retrieving results from the Space. Both the requests and the results are cached on the Space, which now serves as a repository of knowledgebase. The requests program codes are cashed on the Space, thus that the Requesting Agent does not need to resend the same program codes for used with different sets of data. The computed results are also cashed on the Space, thus that when another Requesting Agent needs the same request, it simply retrieves the results from the Space.

This computing platform has been implemented in our lab using several computers each of which has a network card that supports multicast protocol. Many test cases have been successfully executed to verify the various functionalities of this platform (Bingi, 2010). Some test cases, for example, test the fault

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 2. Agents joining space to form workgroups



tolerance of this computing platform, in which a General Agent died (maybe due to computer malfunction) before completing a task. In this case, another General Agent was able to pick up and finish the task. Some test cases test the ability for a Requesting Agent to send both the data and the program code as a request and then a General Function Agent picked up the request, completed the task, and returned the results. The tests results showed that the platform is applicable for general purpose computing.

IMPLEMENTING A PLATFORM FOR HIGH PERFORMANCE

Based on the framework for Multiagent Social Computing, a computing platform for high performance computation has also been developed, implemented, and tested (Choi & Dhawan, 2003, 2004). Although the framework is for general purpose, parallel and distributed computing, a search engine application that serves millions of users was chosen as a test case for implementing and testing the platform. Figure 3 shows the agent and space architecture designed for search engine (Choi, 2001, 2006, 2010). The search engine architecture consists of agents to handle requests, a space for searching, agents to handle search words, a space for searching words, and agents to retrieve search results.

High performance of the architecture is achieved by simply adding more agents, spaces, or networks. Another feature of the architecture for high performance is the result of using space as cache. When an agent needs to perform certain task and finds that the result of the task is already stored in the space, there is no need to repeat the computations. The agent simply reads the results from the cache. This not only reduces repeated computations when several agents need the same result but also reduces the response time, which is practically beneficial for search engine applications.

The architecture is highly scalable. Any number of agents, spaces, or networks can be added. Adding an Agent is as simple as connecting the agent to a network. The agent will then discover a space in the network and become part of the workgroup. It is a plug and play process. No manual configuration is needed. Similarly, adding a space is simply connecting the space to the network and the space will broadcast its present through the multicast network. Adding a network is as easy as connecting agents and spaces into the network. This is made possible by the fact that

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

agents and spaces can be connected to multiple networks through multiple ports.

High availability and fault tolerance is achieved through multiple agents, spaces, and networks. For instance, having multiple agents performing the same role, the failure of an agent only downgrades the performance and will not affect the overall functionality of the system. Replacing an agent can be as simple as disconnecting the agent from the network and connecting another one. Similarly, having multiple spaces, the failure of one space again will only downgrade the performance. An agent pending for a request to be completed will discover that the space is not available and will then send the request to another space. Similarly, having multiple networks, the failure of one network will only downgrade the performance in a larger extent. Agents and spaces can continue to communicate through their ports that are connected to a live network.

Using the Computing Platform for Search Engine Servers

This subsection provides implementation details for using the computing platform for search engine servers. The Agent Space Architecture for Search Engines (Figure 3) has been implemented and tested. To reduce development time we implemented the architecture by utilizing JINI API (JINI, 2010) and by using Apache and Tomcat for Web application server.

Implementing Agents

An agent automatically discovers a space to join as soon as it is started. After joining a space, an agent becomes a member of the workgroup specified by the name of the space. For instance, as shown in Figure 3, the agents in the bottom row, Agent A, Agent B, and so on, are members of workgroup Space Word. An agent can be a member of multiple workgroups. For instance, each of the Agent Word in Figure 3 is a member of workgroup Space Word and a member of workgroup Space Search. We implemented this automatically discovery process for the agent by utilizing the Discovery and Lookup protocols provided by JINI. In particular, as shown in Figure 4, we use a SpaceListener interface that extends the DiscoverListener interface defined in JINI. During the discovery process, an agent broadcasts a request through a multicast network. Active Spaces, such as Space Search or Space Word in Figure 3, actively monitoring the multicast network will respond to the request. As soon as the agent receives a response from a space, it becomes part of the workgroup.

All agents will automatically monitor events to be handled within their workgroups. The events include new tasks need to be completed or new results returned. An event is implemented as a type of net.jini.core.event. RemoteEvent provided in JINI API. To reduce communication overhead, we further divide events into two types: group events and individual events. Group events, such as new tasks needed to be handled, are broadcast to all members of the workgroup. Individual events, such as new results that an agent is waiting for have returned, are sent to the particular agent waiting for the events.

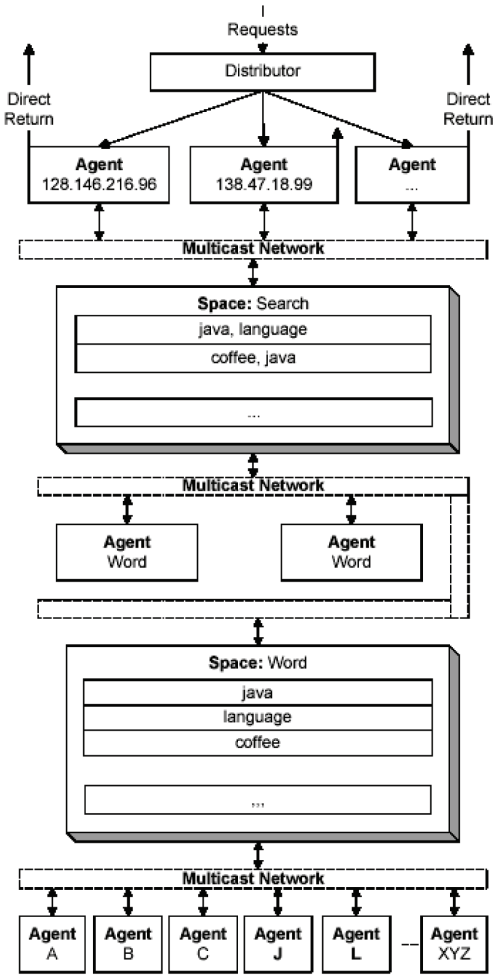
Different types of agents are able to handle different types of tasks. For example, as shown in Figure 3, those agents directly below the distributor are able to handle search requests and able to send the search results directly to the clients, while those agents in the bottom row of Figure 3 are able to search the database for web pages matching the requested keywords. To capture the common characteristics of agents and the individual abilities of different types of agents, we implemented each of the agents as defined in Figure 4. The individual abilities of different types of agents are specified by using the attribute called config that has the type Configuration defined in net.jini.config. Configuration of the JINI API.

Implementing Spaces

To implement and test the Active Spaces, the Space Search and the Space Word for our Search Engine Architecture (Figure 3), we utilized the Lookup service and the JavaSpace service (JINI, 2010). The Active Spaces will automatically respond to agents during the process for

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 3. Agent space architecture for search engines



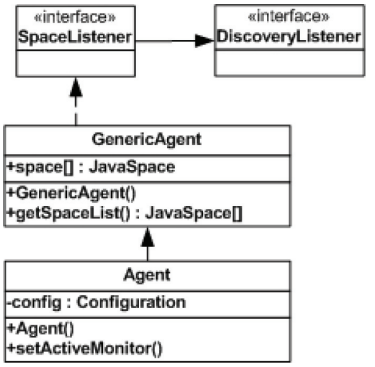
the agents to discover spaces to join. This is made possible through the Lookup service and the multicast networks (to be described). The Lookup service monitors multicast requests and helps establish direct communication between an agent and a space.

Our Active Space functions like a repository for agents to write, read, or take (that is read and remove) tasks, for which we used the corresponding functions provided by JavaSpace service. We also make our Active Space functions like a cache, in which the results of a task are stored for a specific duration. When an agent needs to complete certain task and finds that the results of the task are already available, it will simply retrieve the results. We simulated the first-in-first-out cache replacement policy by using the Lease function provided in JINI API. After a lease has expired, the results of a task will be removed from the space.

When a new task is written into an Active Space, the space will broadcast this event to all worker agents who are listening to the space. This function is implemented by using the notify method available in JavaSpace service. In addition, we also use the notify method for a

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 4. Implementing an agent



space to notify a particle agent when the results, that the agent is waiting, have been written by other agent.

Implementing Multicast Networks

We implemented our multicast networks by using three protocols, multicast request protocol, multicast announcement protocol, and unicast protocol, which are provided in JINI API. Multicast request protocol is used by agents to discover Active Spaces via Lookup service. Multicast announcement protocol is used by Active Spaces to announce, via Lookup Service, their presence in the network. Unicast protocol is used to establish direct communication between agents and spaces after the discovery process has been completed.

We simulated the three different multicast networks as depicted in Figure 3 by using one Ethernet hub. This simulation does not allow partitioning of network traffics, but is sufficient for our testing purpose. An agent is preset to discovery certain spaces. For example, the agents in the bottom row of Figure 3 are preset to discover Space Word, while an Agent Word is present to discover both Space Word and Space Search.

Implementing Distributor

The distributor receives requests from clients and distributes the requests to be processed by agents as depicted in Figure 3. For our testing purpose, we used Apache web server to receive requests from clients and to pass the requests to Tomcat application server. We implemented the distributor by using a Java Server Page running within Tomcat and redirecting the requests to the underlying agents. For load balance, the distributor redirects a request to an agent that is least recently used.

Our experiences and testing results of the architecture (Figure 3) indicate that such a platform is easily configurable, extensible and hence mitigates the management issues confronted by existing search engine architectures.

CONCLUSION AND FUTURE RESEARCH

This article describes a framework for Multiagent Social Computing, where networked computers work in groups and help each other perform various tasks. A computer can run many agents and assume many roles. Agents join a workgroup through a share place of communication called Space. A sever computer can run the service of a Space. A Space is used in the framework not only as a share place for communications but also as a repository of knowledgebase. Based on the framework, two computing platforms have been implemented and tested. One platform is designed for general purpose computing, which uses code mobility to

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

allow a computer to specify a request by sending both the program code and the data. Another platform is designed for high performance and especially for use on search engine applications. Our experiences with these projects indicate that the framework of Multiagent Social Computing has many advantages, including high performance, scalability, and availability, requiring less human intervention, and providing natural fault tolerance.

The future of computing is moving from personal computers to societies of computers. When people join social networks, their personal computers can also join the social networks. The framework allows sharing of computing resources among friends and groups. The framework is also applicable for extending the notion of cloud computing. It allows computation to be highly parallel and distributed all over the networked computers, and allows personal computers to join to form large computing communities. The framework harnesses the collective capabilities of humans and computers. It has the potential to change the future of computing. Future personal computer will no longer be working alone for one person but will work with a large number of other computers helping other people. Any person using a computer will have access to not only the computing power of his/her own personal computer but also the vast computing power of a society of computers.

REFERENCES

Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M., & Werthimer, D. (2002). SETI@home: An experiment in public-resource computing. *Communications of the ACM*, *45*(11), 56–61. doi:10.1145/581571.581573

Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, *36*(4), 335–371. doi:10.1145/1041680.1041681

Batheja, J., & Parashar, M. (2001). *A framework for opportunistic cluster computing using JavaSpaces.* Retrieved from http://www.caip.rutgers.edu/

Berman, F., Fox, G., & Hey, A. J. G. (Eds.). (2003). *Grid computing: Making the global infrastructure a reality*. New York, NY: John Wiley & Sons.

Bingi, S. C. (2010). *Code mobility with cache in distributed systems using Javaspaces*. Ruston, LA: Louisiana Tech University.

Carriero, N., & Gelernter, D. (2001). A computational model of everything. *Communications of the ACM*, *44*(11), 77–81. doi:10.1145/384150.384165

Choi, B. (2001). Making sense of search results by automatic web-page classifications. In *Proceedings of the WebNet World Conference on the World Wide Web and Internet* (pp. 184-186).

Choi, B. (2006). *U. S. Patent No. 7,134,082: Invention: Method and apparatus for individualizing and updating a directory of computer files.* Washington, DC: United States Patent and Trademark Office.

Choi, B. (2010, October). My Internet. In *Proceedings of the International Conference on Web Information Systems and Mining* (pp. 171-175).

Choi, B. (2010, November). Multiagent workgroup computing. In *Proceedings of the IADIS International Conference on Internet Technologies & Society* (pp. 75-82).

Choi, B., & Dhawan, R. (2003). Distributed object space cluster architecture for search engines. In *Proceedings of the High Availability and Performance Computing Workshop*.

Choi, B., & Dhawan, R. (2004). Agent space architecture for search engines. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (pp. 521-525).

Dignum, V. (Ed.). (2009). *Handbook of research on multi-agent systems: Semantics and dynamics of organizational models*. Hershey, PA: IGI Global. doi:10.4018/978-1-60566-256-5

Engelhardtsen, F. B., & Gagnes, T. (2002). *Using JavaSpaces to create adaptive distributed systems.* Retrieved from http://www.nik.no/2002/Engelhardtsen.pdf

Foster, I., & Kesselman, C. (Eds.). (2003). *The Grid 2: Blueprint for a new computing infrastructure*. San Francisco, CA: Morgan Kaufmann.

Freeman, E., Hupfer, S., & Arnold, K. (1999). *JavaSpaces: Principles, patterns, and practice*. Reading, MA: Addison-Wesley.

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

IBM Almaden Research Center. (2010). *TSpaces.* Retrieved from http://www.almaden.ibm.com/cs/TSpaces/

JINI. (2010). *Jini specifications and API archive.* Retrieved from http://www.javawhat.com/viewWebsite.do;jsessionid=E64F3BA42260139575124 6AB5160AF03?id=520683

Joseph, J., & Fellenstein, C. (2004). *Grid computing: On demand series*. Upper Saddle River, NJ: Prentice Hall.

Kilduff, M., & Tsai, W. (2003). *Social networks and organizations*. Thousand Oaks, CA: Sage.

Miller, F. P., Vandome, A. F., & McBrewster, J. (2009). *Climateprediction.net: Personal computer, Parametrization (climate), Volunteer computing, Berkeley Open Infrastructure for Network Computing, University, BOINC Credit System, FLOPS, Climate model*. Mauritius, Africa: Alphascript Publishing.

National Science Foundation. (2010). *Social-computational systems: Program solicitation (Tech. Rep. No. NSF 10-600)*. Arlington, VA: National Science Foundation.

Noble, M. S., & Zlateva, S. (2001, June). Scientific computation with javaspaces. In *Proceedings of the 9th International Conference on High Performance Computing and Networking*.

Pande, V. (2008). *Folding@home distributed computing home page.* Retrieved from http://folding.stanford.edu/

Plekhanova, V. (2002). *Intelligent agent software engineering*. Hershey, PA: IGI Global. doi:10.4018/978-1-59140-046-2

Rittinghouse, J., & Ransome, J. (2009). *Cloud computing: Implementation, management, and security*. Boca Raton, FL: CRC Press.

Shamma, J. (Ed.). (2008). *Cooperative control of distributed multi-agent systems*. New York, NY: Wiley-Interscience.

Steinmetz, R., & Wehrle, K. (2005). *Peer-to-peer systems and applications*. New York, NY: Springer.

Subramanian, R., & Goodman, B. D. (Eds.). (2005). *Peer to peer computing: The evolution of a disruptive technology*. Hershey, PA: IGI Global.

TONIC. (2008). *Scientific computing with JAVA TupleSpaces.* Retrieved from http://hea-www.harvard.edu/~mnoble/tonic/doc/

Velte, T., Velte, A., & Elsenpeter, R. (2009). *Cloud computing, a practical approach*. New York, NY: McGraw-Hill.

Wen, S., Griffioen, J., & Calvert, K. (2001). Building multicast services from unicast forwarding and ephemeral state. *Computer Networks*, *38*(3), 327–345. doi:10.1016/S1389-1286(01)00292-4

Williamson, B. (1999). *Developing IP multicast networks* (*Vol. 1*). Indianapolis, IN: Cisco Press.

Copyright © 2011, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.