A Study of Mobile Agent Platforms for Distributed Web Crawling

Niraj Singhal $^1$, Ashutosh Dixit $^2$, R. P. Agarwal $^3$, A. K. Sharma $^4$

$^1,^3$Faculty of Electronics, Informatics and Computer Engineering
Shobhit University, Meerut, India
$^2,^4$Department of Computer Engineering
YMCA University of Science and Technology, Faridabad, India
sonia_niraj@yahoo.com

Abstract - In the traditional centralized crawling techniques the pages from all over the web are brought to the search engine side which results a lot of unnecessary Internet traffic. In the distributed crawling with migrating agents, the mobile agent is send to the server side that brings only required pages to the search engine side which reduces unnecessary overhead. The mobile (migrating) agents are processes capable of roaming a wide area networks, interacting with foreign hosts, gathering information on behalf of its owner and coming back having performed the duties set by its user. The mobile agents acquire the information from the local environment, filter locally and either stored with the agent or forwarded to some receiving destination. In this paper we focus on use of mobile agents in web crawling, various available options of mobile agent technologies and enhancements needed.

Keywords - Web search engine, migrating agents, agent platforms

1. Introduction

The centralized crawling techniques are unable to cope up with constantly growing web and results a lot of network traffic. Here, the pages from all over the web are brought to the search engine side and then processed. When a page is brought to the search engine side for processing, many a times it is found that it was not needed. In such cases the efforts made to send request to the web server and bringing the page to the search engine side become useless. In the distributed crawling based on mobile crawling agents (i.e. migrating agents or migrants) [13,15,16,17], the process of selection and filtration of web documents can be done at web servers rather than search engine side which can reduce network load caused by the web crawlers [16,17].

Mobile agent [10] is a type of software agent with the feature of autonomy, social ability, learning, and most importantly, mobility. It is a composition of computer software and data, which is able to migrate (move) from one computer to another autonomously and continue its execution on the destination computer. In migrating crawling agents, the mobile code generated by search engine side transfers and executes on web servers, an environment controlled by another party. Because of mobility of mobile agent, the security problems [9] have also become a bottleneck for development and maintenance of mobile agent technology. In this paper we focus on use of mobile agents in web information retrieval. We also discuss various available options of mobile agent technologies and enhancements needed.

2. Related Work

An agent is an autonomous entity that acts on behalf of others in an autonomous fashion, performs its actions in some level of proactivity and reactivity, and exhibits some levels of the key attributes of learning, cooperation and mobility [2]. Nwana [10] identifies seven type of agents i.e. collaborative agents, interface agents, migrating agents, information agents, reactive agents, hybrid agents and smart agents. Migrating agents (migrants) are computational software processes capable of roaming wide area networks such as the web,
interacting with foreign hosts, gathering information on behalf of its owner and coming back having performed the duties set by its user.

Mobility allows an agent to move, or hop, among agent platforms. The agent platform provides the computational environment in which an agent operates. The platform from which an agent originates is referred to as the home platform, and normally is the most trusted environment for an agent. One or more hosts may comprise an agent platform, and an agent platform may support multiple computational environments, or meeting places, where agents can interact. They may cooperate or communicate with other agents making the location of some of its internal objects and methods known to other agents without necessarily giving all its information away (figure 1).

![Figure 1. Crawling system based on migrating agents](image)

Some key characteristics of mobile agents are [10,13] migration, data acquisition, route determination and communication. The mobile agents interrogate their local environment to acquire the information necessary to achieve their goals. This information needs to be filtered locally by the agent before it is either stored with the agent or forwarded to some receiving destination. Once an agent has finished with a network node, it must make a decision of where to move to next. The ability of agents to communicate is fundamental to mobile systems. Two methods for agent communication to take place are Network-oriented agents and Node-oriented agents.

Some of the advantages of mobile agents are [10,13,17] bandwidth, latency, asynchronous task execution, fault tolerance and peer-to-peer communication. When very large volumes of data are stored at remote hosts, these data should be processed in the locality of the data rather than transferred over the network. The main concern is, move the computations to the data rather than the data to the computations. By migrating to the location of the resource, a mobile agent can interact with the resource much faster than from across the network. While the agent acts on behalf of the client on a remote site, the client may perform other tasks. Instead of being online for a longer period, a mobile user may develop an agent request while being disconnected, launch the agent during a brief connection session, and receive back the agent with the result at some later time.

### 2.1 Search Engine based on Mobile Agents

A search engine (figure 2) is a coordinated set of programs that is able to read every searchable page on the web, create an index of the information it finds, compare that information to a user's search request (i.e. query), and finally return the results back to the user. Search engines operate as a link between web users and web documents. Without search engines, this vast source of information in web pages remain veiled for us. It is a searchable database which collects information from web pages on the Internet, indexes the information and then stores the result in a huge database where from it can be searched quickly. A general web search engine has three parts i.e. Crawler, Indexer and Query engine.
The web crawler is a module that searches the web pages from the web world. These are small programs that peruse the web on the search engine's behalf, and follow links to reach different pages. Starting with a set of seed URLs, crawlers extract URLs appearing in the retrieved pages, and store pages in a repository database. The indexer extracts all the uncommon words from each page and records the URL where each word has occurred. The result is stored in a large table containing URLs; pointing to pages in the repository where a given word occurs. The query engine is responsible for receiving and filling search requests from users. It relies on the indexes and on the repository. Because of the web's size, and the fact that users typically only enter one or two keywords, result sets are usually very large.

In the distributed crawling with migrating agents approach [4,13,15,16], agents allow packaging a conversation and dispatching it to a destination host where the interactions can take place locally. Migrating agents are also useful when it comes to reducing the flow of raw data in the network. When very large volumes of data are stored at remote hosts, these data should be processed in the locality of the data rather than transferred over the network. The main concern is, move the computations to the data rather than the data to the computations. By migrating to the location of the resource, a mobile agent can interact with the resource much faster than from across the network that reduces network traffic also.

The distributed crawling with migrating agents approach (figure 3) uses a crawler manager at the search engine site, that deputes migrating crawlers to the web servers with a list of URLs of respective web servers. The migrating crawler, on reaching a server crawls the pages, select the best of the pages for its collection and comes back to the search engine with the collection. It reduces unnecessary overhead of being the unnecessary pages to the search engine site. The size of the collection can further be reduced by filtering the required specialized web pages and even compressing them.
3. MOBILE AGENT SYSTEMS

Various mobile agent systems available are Agent Building and Learning Environment, Agent TCL system, Aglets, Anchor Toolkit system, ARA, Concordia, Grasshopper, HTTP-Based Infrastructure for Mobile Agents, JADE, MAF, Mole, PIAX, SensorWare, SMARD, SPRINGS, TACOMA, TAgent, Tracy, Trinity, Tryllian and Voyager. These systems differ in their goals, motivations, purpose, security, and implementations but they all by and large provide common functionalities that support migration of agents, programming languages, communication between agents and various forms of security. Now, let us have a look all these mobile agent development environments.

3.1.1 Agent Building and Learning Environment

The Agent Building and Learning Environment (ABLE) [39] project started in early 1999 at the IBM T.J. Watson research laboratory. The aim was to produce a fast, re-usable and scalable toolkit for creating intelligent software applications. This software technology has been applied to areas such as autonomic computing, agent-based modeling and simulation, complex workload generation, business rules and policy, and adding intelligence to pervasive computing devices.

ABLE is a Java-based framework, component library, and productivity toolkit for building intelligent agents that can use machine learning and reasoning. It is designed to be used by applications involved in autonomic computing, data mining, forecasting, planning, retail and resource balancing etc. It provides a distributed platform allowing agents to be configured, run, and managed across different physical systems. Its framework provides a set of Java interfaces and base classes used to build a library of JavaBeans called AbleBeans. The library includes AbleBeans for data transformation and scaling, for rule-based inferenceing, and for machine learning techniques.

3.1.2 Agent TCL system

The Agent TCL system [18,20,23,35] is a model for supporting mobile agents developed at Dartmouth College. The architecture of Agent TCL is based upon the server model. Mobile agents provide all the services that are available within the system. The execution of agents is handled by an interpreter that is appropriate to the source language of the mobile agent. The interpreter is extended to support three extra modules i.e. security, state capture and server. Security module prevents an agent from performing malicious actions, State capture
module packages and restores the internal state of an agent, and Server API module allows interaction with the server to provide migration and agent communication. Server handles the management of local agents and incoming agents and also provides mechanism for enforcing security, providing a hierarchical name space in which agents can be referenced and allowing agents to address each other locally.

Agent TCL appears to be the most flexible architecture, since it supports state oriented migration, multiple languages and networking protocol. It also supports security mechanisms but it lacks in constraining agent execution because of which resources are directly accessible via TCL and freely available for use by agents. It is inefficient than other interpreted languages. It lacks object-oriented features, thus making it difficult to write and debug large scripts.

TCL is inefficient as compared to other interpreted languages and is ten thousand times slower than optimized C. Its non-object-oriented nature provided no code modularization aside from procedures, thus making it difficult to write and debug large scripts. It lacked facilities for capturing the internal state of an executing script, which are essential for providing transparent migration at arbitrary points and lack of constraining agent execution. Resources, directly accessible via TCL are freely available for use by agents.

Even with these drawbacks, Agent TCL appears to be a decent system.

3.1.3 Aglets
Aglets [18,19,24] developed at the IBM Tokio Research Laboratory is completely made in Java, granting a high portability of the agents and the platform both. An aglet is a Java agent, able to autonomously and spontaneously move from one host to another. Aglets follows an applet like developing paradigm and user only needs to define a few methods to implement the behaviour of agents. Aglets is secure because it is developed in Java and is compliant to the Java2Security Manager. Aglets is flexible, since it allows user to extend the platform in order to implement new functionalities. Aglets is an agent development kit that is quite simple to learn and to use.

Aglets has contributed significantly to the field of mobile agents. Its disadvantage is that the proxies it provides are not dynamic proxies i.e. they cannot be used after the agent they point to moves to another place, therefore the programmer must obtain himself an updated proxy, if needed, before using it. As every agent is assigned a single thread, the programmer must avoid the execution of long-running tasks: otherwise, that would prevent agent events such as incoming messages from being considered. The platform does not support remote calls to agents or assigning them user-friendly identifiers.

3.1.4 Anchor Toolkit system
The Anchor toolkit [3,22] handles the transmission and secure management of mobile agents in a heterogeneous distributed computing environment. The toolkit protects the agents being dispatched between hosts through encrypted channels. A mobile agent’s host platform is required to sign the agent’s persistent state before dispatching the agent to the next platform. The signed persistent state can be used later to detect potential problems with the agent’s state. The architecture of the Anchor toolkit consists of an Agent Viewer, Agent APIs, Anchor Server, Anchor Security Manager (ASM), Anchor Class Loader (ACL), Secure Agent Transfer Protocol (satp) handler, Anchor Java Naming and Directory Interface (AJNDI) and Anchor Java Native Interface (AJNI).

3.1.5 ARA
Ara [11,12,18,23] is a platform for the portable and secured execution of mobile agents in heterogeneous networks. In Ara a mobile agent is a program able to move at its own choice and without interfering with its execution, utilizing various established programming languages. The platform provides facilities for access to
system resources and agent communication under the characteristic security and portability requirements for mobile agents in heterogeneous networks. In Ara, mobile agents are programmed in some interpreted language and executed within an interpreter for this language, using a special run-time system for agents, called the Core (the central part of an Ara system). The core mediates any access from an application agent to the host system or to another agent for security and portability.

Ara is concerned with designing of secure and portable execution of mobile agents. It is less concerned with certain features such as agent cooperation patterns, intelligent behavior, user modeling etc. Ara programming model consists of agents autonomously moving between and staying at places, where they use certain services, provided by the host or other agents, to do their job. Ara’s security model is flexible as domains of protected resources can be dynamically created in the form of places, and that the admission of agents to such a domain, as well as their actual rights at that place, can be controlled in a fine grained manner down to individual agents and resources.

3.1.6 Concordia

Concordia [8,14,18,23] is a full-featured framework developed at Mitsubishi Electric Information Technology Center America. It provides platform for the development and management of network-efficient mobile agent applications for accessing information anytime, anywhere. At the highest level, a Concordia system consists of a Java Virtual Machine (JVM), a Concordia Server running on a machine in a network, and a mobile agent running in the system. Concordia consists of a set of components that provides services such as communication, administration, persistent storage, security etc.

In Concordia agent communication is held either through asynchronous distributed events or collaboration. Asynchronous distributed events are events that agents receive via the Event Manager component. The agent determines the type of events an agent receives when it first registers with the Event Manager. The Event Manager can forward events to an agent even after it migrates to another system. Collaboration extends agent communication by enabling multiple agents to perform complex distributed computations more effectively by correlating their results and altering their behavior based on the combined results.

Its security model provides support for two types of protection i.e. protection of agents from being tampered with, and protection of server resources from unauthorized access. Concordia uses encryption to protect agents during transfer and, to protect resources on each server, Concordia relies on its Security Manager component to manage resource protection. The Security Manager authenticates each agent by verifying its identity. If the identity matches, then the agent is able to access the resource.

3.1.7 Grasshopper

Grasshopper [6,19] is a mobile agent platform that is an integration of the traditional client/server paradigm and mobile agent technology. It is developed compliant to the first mobile agent standard of the Object Management Group (OMG) [38]. A Grasshopper system can be composed of different regions. It provides agent developers with interesting features, including a graphical user interface to manage agents, agencies, and regions. By defining regions, the developer can benefit from dynamic proxies. In Grasshopper there are two types of agents i.e. mobile agents and stationary agents. The actual runtime environment for both is an agency and on each host at least one agency has to run to support the execution of agents. A Grasshopper agency consists of two parts: the core agency and one or more places. Core Agencies represent the minimal functionality required by an agency in order to support the execution of agents.
Various services provided by a Grasshopper core agency are Communication Service, Management Service, Persistence Service, Registration Service, Security Service and Transport service. Communication service is responsible for all remote interactions that take place between the distributed components of Grasshopper. The management services allow the monitoring and control of agents and places of an agency by users. The Grasshopper persistence service enables the storage of agents and places on a persistent medium. The registration service of each agency is connected to the region registry which maintains information of agents, agencies and places in the scope of a whole region. Grasshopper supports two security mechanisms i.e. external and internal security, where External security protects remote interactions between the distributed Grasshopper components and Internal security protects agency resources from unauthorised access by agents. It is also used to protect agents from each other, and this is achieved by authenticating and authorising the user on whose behalf an agent is executed.

3.1.8 HTTP based Infrastructure for Mobile Agents

The HTTP based infrastructure for mobile agents [1] is a project being developed at the Goethe University in Germany. It is designed to provide a low-level infrastructure to support agent mobility and communication through the use of HTTP.

An agent server is a process that executes on every host that can be accessed by agents. Its tasks include accepting agents, creating an appropriate runtime environment for agents to execute within, supervising the execution of agents and terminating agents if required. In addition to this, the agent server must also organise the transfer of mobile agents to other hosts, manage communication between agents and their users, and perform authentication and access validation. Agents are transferred as encapsulated Multipurpose Internet Mail Extension (MIME) documents in a stateless manner, by posting them to a special URL that is managed by an agent server. Upon receiving this agent, the agent server parses the code of the mobile agent and determines whether it is acceptable.

3.1.9 JADE

Java Agent DEvelopment Framework (JADE) [2,3,5,18,19,26] is a java software framework that allows implementation of multi-agent systems. The agent platform can be distributed across machines where the configuration is performed via a remote GUI. It is a middleware that aims at supporting the development of applications that address this evolution as its fundamental is the peer-to-peer intelligent agent approach. It allows the development of systems of peers able to work in a proactive way, according to usage rules given by owners, to communicate and negotiate with others, directly and regardless to their role and position and to coordinate in order to solve complex problems in a distributed way.

An agent is composed of different concurrent and non preemptive behaviors which can be added dynamically. Among the benefits, we could indicate that there is a wide variety of tools provided, for e.g. for remote management and monitoring of agents and to track interchanged messages and it can be integrated with different software. The main disadvantage is that mobility is not a key element in JADE. So, it focuses on other functionalities relevant to the development of multiagent systems. The JADE built-in Agent Mobility Service supports mobility among containers within the same JADE platform.

3.1.10 MAF

Mobile Agent Framework (MAF) [2,18,27] is a Python research prototype that provides a set of primitives to facilitate the development of distributed mobile agent. It is a research prototype which carries the several goals in mind, it aims to provide a set of primitives to facilitate the development of distributed mobile agent; second, it strives to meet the application requirement from distributed sensor field which is to provide a light-weight,
self-organized and secure agent platform, although this framework itself is primarily developed using Python language, it should provide a mechanism to be able to incorporate and integrate effortlessly with a variety of "foreign agents" written in other languages such as C and C++.

3.1.11 Mole

Mole [18,23] is the first mobile agent system that has been developed in the Java language and provides a stable environment for the development and usage of mobile agents in the area of distributed applications. There is strong migration and weak migration. Mole supports asynchronous communication using an event driven model. In the event driven model, depending on these events, internal rules, state information and timeout intervals, output events are generated, that in turn may be the input for other synchronization objects. Mole allows synchronous and asynchronous messages among agents along with RPC type communication.

It uses a Sendbox security model in which service agents have access to system resource, providing controlled, secure abstractions of these resources inside the agent system. Moreover, service agents may offer access to legacy software, using the native code interface offered by Java. This does not cause any security problems, because the service agents are immobile and may be started only by the administrator of the location. User agents may only communicate with other agents and have no direct access to system resources.

3.1.12 PIAX

P2P Interactive Agent eXtensions (PIAX) [2,18,29] is an open source framework that integrates mobile agents paradigm and P2P structured overlay network. It allows to build a scalable and efficient federated system in a large-scale distributed environment where various kinds of data and processes are located in each device. Overlay network enables pervasive devices to communicate with each other efficiently, while agent platform on the overlay network encourages the devices to cooperate with other devices.

PIAX is a Java class library that integrates mobile agent platform and P2P structured overlay network. The features of PIAX are powerful P2P agent mechanism with simple APIs, built-in versatile P2P overlay networks, multi-overlay mechanism for plug-in independent P2P overlay networks, unified APIs for discovering peers and messaging between agents.

3.1.13 SensorWare

SensorWare [2,18,30] is an implementation of mobile agent environment for wireless sensor network. Its scripting language is based on Tcl, and scripts can move their code and data from node to node, autonomously. The distributed algorithms are realized as control scripts that are autonomously replicated or migrated in the proper sensor.

SensorWare abstracts the run-time environment of a sensor node using a set of services, and a scripting language to form scripts out of these services. These scripts perform certain tasks when executing in a node. Scripts can also move their code and data from node to node, autonomously. So, distributed algorithms are realized as control scripts that are autonomously populated in the proper sensor nodes after a triggering user injection.

3.1.14 SMARD

Secure Mobile Agent Rapid Development (SMARD) [2,18,32] is a development environment oriented programming of mobile agents based applications. Using a graphical interface, programmers can design, build and launch mobile agents which are intended to run on JADE platforms.

3.1.15 SPRINGS
SPRINGS [18,19,21,31] developed by the Distributed Information Systems Group at the University of Zaragoza in Spain, focuses on scalability and reliability in scenarios with a moderate and high number of mobile agents. Its development has been inspired by the features of other popular platforms, such as Voyager and Grasshopper. It proposes a hierarchical architecture based on regions.

It provides full location transparency for movements so that the programmer does not need to specify network addresses but just the name of the destination, and for calls, through the use of dynamic proxies. The main disadvantage of SPRINGS is perhaps that it does not support agent communication and does not provide sophisticated security mechanisms. Despite it is easy to use, it does not offer any graphical tool to the user.

3.1.16 TACOMA
Tromoso And Cornell Moving Agents (TACOMA) [2,17,18,23,33] is being developed by the University of Tromoso and Cornell. It is primarily concerned with providing operating system support for agents. TACOMA considers agents, either stationary or mobile, to be computational unit of the system. Each agent has three storage mechanisms Folder, the essential unit of data that is accessible by an agent, Filling cabinets, in which folders can be stored, which are stationary data repositories and, Briefcases, which are containers that agents carry them.

An agent in TACOMA is a piece of code that can be installed and executed on a remote computer. Such an agent may explicitly migrate to other hosts in the network during execution. TACOMA agents store data in folders. A subset of the folders are identified with individual hosts and collected in the file cabinets managed by the hosts, the remaining folders comprise a briefcase that is moved from host to host along with the computation. A TACOMA agent executing on one host moves to another host by using TCP to communicate with TACOMA software at the destination host. Agents are migrated using a simple primitive called meet. Agent can cause another agent to be executed by invoking the meet operation and naming a target agent and a briefcase. The effect of the operation is to terminate the agent invoking the meet and then start executing the target agent with the specified briefcase.

3.1.17 TAgent
Travel Agent (TAgent) [2,34] is a Java based platform which allows to develop mobile agent. These mobile agents can act on behalf of their owner without the requirement for the user to interact. It provides an easy service extending an easy agent development platform and a secure design of the agents. TAgents project aims to build an open source mobile agent system. It’s main focus is on extendibility and flexibility. The TAgents platform was designed to be easy to extend. The basic platform can be extended with all kinds of service from database access up to the free choice of the communication protocol. Easy development of an agent or system agent. The basic agent class comes with a rich feature set and any other service can easily be accessed through the service providing mechanism.

3.1.18 Tracy
Tracy [18,19,28] was developed at the University of Jena in Germany and has a plugin-oriented architecture. Plugins are the software components that can be added dynamically to a running agency, if required, in order to provide high-level services such as inter-agent communication, migration, security etc. Tracy agencies are lightweight and extensible. The platform also offers several migration strategies for agents. A key disadvantage of Tracy is that it does not support remote communications between agents, an agent must travel to the agency where another agent is running in order to communicate with it.

3.1.19 Trinity mobile agent framework
A Study of Mobile Agent Platforms for Distributed Web Crawling

Trinity [2,36] is mobile agent based framework that allows to create for wide range of mobile agent types. It allows rapid development of non-complex agents. It is mobile agent framework for creation of wide range of mobile agent types. The goal is to provide a tool for rapid development of non-complex agents. Trinity mobile agent framework is an open source software that perform Intelligent Agents tasks. It’s free for both personal and commercial use, thus the perfect choice for those that want an alternative for intelligent agents programs.

3.1.20 Tryllian

Tryllian [18,19,37] developed by the homonym company in 2001 is based on a sensing-reasoning action mechanism. It allows programmers to define a reactive and proactive behavior of agents. It proposes a taskbased programming model and communication among agents is achieved through message passing and in accordance with the FIPA [25] standard. It also provides a persistency service. The main disadvantage of Tryllian is that it does not offer location transparency. In addition, its task based and asynchronous model could be difficult to use, due to its differences with the classical procedural programming. The use of a single thread per agent could be inefficient and a limitation for the programmer. Tryllian provides a large set of configuration options, which could be overwhelming. Finally, it does not offer facilities for synchronous communication or conventional method invocation.

3.1.21 Voyager

Voyager [2,3,18,19,23] is a java agent-enhanced Object Request Broker (ORB) created by ObjectSpace Company. Its goals are to provide programmer to create state of the art distributed programs quickly, and easily while providing a lot of flexibility and extensibility for the products that are being created with the voyager system. Voyager supports RMI, DCOM, and CORBA architecture to provide stationary client server applications, which makes this system very flexible. Voyager uses regular java syntax to create remote objects and move them between applications. It transparently locates the agents and sends them message as they work, even if the agents are moving, all is done for programmer.

Voyager provides location transparency through forwarding chains of proxies. Voyager is an interesting platform, with several functionalities, which eases the development of distributed systems. A key disadvantage of Voyager is that it is a commercial product not available for free, which could prevent many researchers from using it in favor of other alternatives available. The forwarding chain mechanism used to track mobile agents could also be inefficient as the whole chain must be traversed in order to locate an agent, and weak as a single broken link makes the agent unreachable.

4. CONCLUSION

In this paper, we present traditional centralized crawling techniques, and distributed crawling with migrating agents. We also presented how it reduces unnecessary overhead. It also presented various mobile agent platform technologies. There is no single mobile agent framework that can be alternative to all of the functionality that mobile agents are expected to support. It needs to take more advantage of the characteristics of mobile agents. It is also evident that the development of mobile agents must lie in solutions to the Maintain integrity, security and protection of mobile agents and the platforms, deciding route of mobile agent movement, inter-mobile agent-system communication and collaborations, monitoring the flow of information, execution of a running mobile agent.

REFERENCES


ISSN: 2319 – 1120 /IJAEST/V1N2:111-121 ©IJAEST


A Study of Mobile Agent Platforms for Distributed Web Crawling


[32]. http://smard.sourceforge.net/ (accessed on October 1, 2012)
[33]. http://www.tacoma.cs.uit.no/ (accessed on September 6, 2012)
[34]. http://www.tagents.org/ (accessed on October 3, 2012)