AgentOS: The Agent-based Distributed Operating System for Mobile Networks

by Larry T. Chen

The Vision: Wide-area Ubiquitous Computing

Many current researchers in the mobile computing arena share the same vision: ubiquitous access to information, data, and applications. **Ubiquitous access** refers to the ability of users to access these computing resources from almost any terminal.

Recent developments relating to the Internet are establishing solid foundations for wide-area ubiquitous computing systems. The recent development of cross-platform languages, the most important of which is Java, enable the deployment of computing systems that span multiple computing platforms and communication networks. Java provides a means of building applications that are accessible from virtually any Internet-connected terminal. This is true because the only requirement for running Java programs (or applets) is a Java-enabled Web browser, which is now a fixture on most computer desktops.

Further evolution of Internet technologies will yield a wide-area network based on component-oriented, dynamic applications, which will support efficient, scalable resource sharing for a large number of mobile and nomadic users. As users gradually grow to rely on the Internet as an indispensible tool, most users will become mobile or nomadic users, or both. While **mobile users** access the Internet from a portable computer, **nomadic users** may move from terminal to terminal. In either case, a user would ideally be able to accomplish the same tasks with equal ease from any location either on his portable computer or at any Internet-connected terminal.

Requirements For This Vision

To deploy a large-scale ubiquitous computing system (a system to manage shared distributed resources over a wide-area) a fault-tolerant network is required. The end-user requirement of this system is network, distribution, and location transparency. To the end-user, the entire network and any terminal attached to it is one large virtual host. The end-user does not care how or where the data and applications are stored.

Several technical requirements accompany this end-user requirement:

- Distributed, network-centric data and application hosting: Distribution transparency over a wide-area, mobile network calls for network-centric data and application hosting; that is, distributed hosts on the fixed network collaborate to provide data and application hosting for individual users.
- Replication and consistency management: With data scattered over a wide-area network, redundancy of data for high availability, fault-tolerance, and security are very important. The system should integrate replication and consistency management of data as a fundamental feature.
- Flexibility, mobility, openness, and dynamicism: A wide-area network such as the Internet is subject to frequent shifts in topology and network conditions. Such volatility in topology is attributed to
    1. frequent changes in the availability of various intermediate network hosts,
    2. mobility of mobile hosts such as laptops, and
    3. general shifts in network usage patterns which may affect bandwidth and host availability.

What Is Wrong with Current Approaches

Although existing frameworks and distributed systems for mobile computing, such as that of Bayou [1] and Rover [2], tackle distributed consistency and replication management, their communication models and programming models are unsatisfactory. The communication paradigms used in these systems are evolved from artifacts of the conventional client/server model, such as RPCs. These mechanisms rely on fixed, well-defined clients and servers, and do not have enough flexibility to deal with the fluidity of wide-area mobile networks.

Agent-based Communication

Agent-based communication paradigms have shown enormous potential for operating in unpredictable, metamorphic environments, such as mobile computing networks.

An example of a high-level distributed task that can be accomplished by agents is the processing of an electronic transaction. Several authorities may need to be contacted before a transaction can be completed. For example, in order to buy a software product electronically, one needs to obtain verification of the user's credit card number, debit the user's credit card account, credit the merchant's account, and deliver the software product back to the user. Such tasks require a series of read/write operations at various nodes, where strict causal dependencies must be enforced (e.g., funds are received by the merchant before the product is delivered).

Why build an agent-based OS?

Agents exhibit the following distinguishing characteristics that are suitable for a wide-area mobile networking environment:

1. Mobility: Mobile computing exhibits an inherently transient nature, i.e., host locations constantly change, and transactions span many nodes and are short-lived. The ability of agents to carry code and data across network nodes underlies their suitability for a transient environment.
2. Flexibility: Agent algorithms may be dynamically modified according to pertinent environmental conditions, i.e., network congestion, node failures, and user movement.
3. Concurrent problem solving: Agents provide a clear, natural paradigm for performing tasks which may have several concurrent aspects.
4. Autonomous security model: Each agent is assigned a privilege level and the agents run in a ``sandbox'', governed by the least-privilege principle, i.e., grant the agent only as much privilege as it needs. Agents may also delegate or transfer user or system privileges.
5. Proxy handling: Agents can behave simultaneously as server proxies or client proxies at any node. Server proxies emulate the functionality or presence of a server while client proxies emulate the functionality or presence of a client (typically a mobile or nomadic client).

Potential Contributions to Mobile Computing Research

We have identified the following issues as the most pervasive and widely-studied problems in mobile computing research. The use of a mobile agent paradigm to address these issues will not only contribute new solutions to these problems, but will also present a uniform platform and methodology for building mobile systems.

1. Distributed, shared data consistency: Maintaining consistency of replicated, interdependent data across distributed sites. Conventional solutions use explicit data synchronization messages for consistency and replica management. An agent-based solution allows the responsibility of consistency management to be assigned to mobile agents.
2. Dynamic proxy server and proxy client management: Emulation of servers and clients to accommodate server failures and client disconnections, and to decrease latency and improve performance. Agents may behave as proxy clients, proxy servers, or both.
3. Disconnected operation: Enabling the operation of applications on mobile hosts while they are disconnected from the backbone network. Agents allow the batch transfer of data and code in preparation for disconnected operation or upon reconnection.
4. Intelligent query processing: Performing a complex query task from a mobile host, with the query possibly being executed across multiple nodes and with multiple strategies. Complex query logic may be embedded in agents, which are naturally tailored for moving among various nodes.
5. Secure, distributed transactions: Guaranteeing security and integrity of database transactions in a mobile environment.

Performance and Overhead Considerations

Bytecode Interpretation

First generation Java has been known to run slowly because of the interpretation overhead associated with Java interpreters. However, recent research and development of just-in-time compilers (JITs) [3] promise to deliver compiled-code performance while maintaining cross-platform portability. **Just-in-time compilers** dynamically compile bytecodes into native machine instructions during execution. Code that is repeatly executed (e.g., kernel code) will approach the speed of compiled code.

Latency Associated With Code Transport

Transmitting the agent code across the network eliminates conventional client/server communication. In mobile and wide-area network environments, aggregating multiple messages into one transmission is more efficient than disjoint multiple transmissions, due to the high cost of connection setup and tear-down. Agents may also carry only code that is essential for their operation. Pieces of code could be dropped during an agent's trip if they are no longer needed.

AgentOS: ``Of Agents, For Agents, and By Agents''

So how does AgentOS fill the vacuum? System functions, such as resource management, are carried out by agents, hence an operating system *of* agents. With a system full of agents, including system, application, and user agents, we need an operating environment to manage these agents; hence, this is an operating system *for* agents. When a user, application, or system agent needs to accomplish a task, it generates and sends out mobile agents to contact other agents. Underlying protocols are partially generated *by* agents.

The research goals of the AgentOS project are:

- To show that an efficient OS for mobile applications can be built using agents.
- To demonstrate that concurrent, mobile agents represent the most suitable paradigm for developing and deploying mobile computing applications.
- To show that Agent-based protocols are ideal for consistency management, network fault detection, and intelligent/adaptive routing.
- To show that Agent-based systems are applicable to real-life applications, such as factory equipment monitoring, and electronic transactions.

Related work

Related projects fall into three categories: Operating systems for the Internet, mobile computing support systems, and mobile agent systems. The AgentOS project takes a unique approach by uniformly using the mobile agent paradigm to implement an operating system for the Internet.

The use of operating systems for the Internet is a relatively new concept. Inspired by the proliferation of Java Virtual Machines in Web browsers, the ``Internet OS" attempts to define a set of basic services essential to a migratory, cross-platform Internet application. The JavaOS, by Sun Microsystems, is slated to be used in network computers, and this embraces the concept of the network as an OS. However, network computers adopt the client/server model. AgentOS, on the other hand, embraces a mobile agent model. Apple has recently described the concept of an operating system for the Web, which in its view, is merely the aggregation of browser-embedded Java VMs viewed as a virtual OS. AgentOS extends the VM concept beyond the browser to intermediate nodes on the network and mobile nodes such as portable computers and wireless devices.

System support issues in mobile computing, especially replicated data management and disconnected operation, have been addressed by numerous projects. The notable projects, Bayou by Xerox PARC and Rover by MIT, have approached the issues in the same general fashion: by extending the client/server model to accommodate the peculiarities of mobile computing, such as extending RPCs to support queueing and callbacks, and using dynamically migratible objects to facilitate caching for disconnected operation. AgentOS takes an innovative approach by abandoning the client/server paradigm and uniformly adopting the mobile agent paradigm to implement solutions for mobile computing data and resource management.

The mobile agent paradigm was introduced several years ago. The pioneer projects in the mobile agent paradigm include Telescript and Messengers. Since the introduction of this paradigm several other projects have surfaced, including CyberAgent (by FTP Software) and Aglets (by IBM). CyberAgent is a framework also written in Java, but its primary focus is on intranet management. The CyberAgents execution model assumes one agent, rather than a cooperating group of agents. IBM's Aglets defines a general-purpose mobile agent framework, but lacks essential functionality such as merging and splitting, dynamic agent generation and spawning, and inter-agent communication.

Related Project	**Focus**	**Advances Made by AgentOS**
JavaOS/HotJava	Small footprint OS; dynamic browser extensibility	Mobile/nomadic computing features
Java Beans	Flexible component architecture	Agent-oriented
``Operating system for the Web" (Apple Computer)	Safe environment for component-oriented, Internet based applications	Mobile computing features, agent-oriented framework

Telescript	Agent language, platform, communication, and management	Focus on system agent management
Mobile Service Agents	Proxy client for mobile hosts to interface with existing servers	Lightweight, multi-faceted agents
Bayou/Rover	Optimistically replicated data sharing	Agent-implemented, adjustable consistency level
CyberAgent	Agent framework for network management	Multi-agent management
Aglets/Mobile Agent Framework	Agent framework	Merging/splitting, dynamic agent generation, inter-agent communication
Messengers	Agent framework for distributed simulation	Use of Java as underlying platform for wider compatibility

References

1

Demers, Alan, Karin Petersen, Mike Spreitzer, Douglas Terry, Marvin Theimer, and Brent Welch. ``The Bayou Architecture: Support Data Sharing among Mobile Users." In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.

2

Joseph, Anthony D., Alan F. DeLespinasse, Joshua A. Tauber, David K. Gifford, and M. Frans Kaashoek. ``Rover: A Toolkit for Mobile Information Access." In *Proceedings of the Fifteenth Symposium on Operating System Principles*, December 1995.

3

Reynolds, Franklin D. ``Evolving an Operating System for the Web". *IEEE Computer*, Vol. 29, No. 9 (September 1996) pp. 90-92.

Larry Chen can be reached at http://www.ics.uci.edu/~larryc or larryc@ics.uci.edu. More information

about the AgentOS project can be found online at http://bolero.ics.uci.edu/agentos/. Larry T. Chen is currently a graduate student at the University of California, Irvine. His research interests include distributed object systems, mobile agent systems, and ubiquitous computing using mobile and ad-hoc networks. He is also a co-founder of Avanteer, Inc., a company dedicated to building collaborative software in Java.