

**Distributed Computing**

Introduction by Guest Editors

*Eric Hughes*



*Kevin Kelly*



For almost as long as there have been computers, we have sought new and ever-better ways to build software so that programs running on multiple computers can work together. As in human endeavors, we find that coordinated efforts can produce better results than the separate efforts of the isolated. The distribution of computing across faster processors, using faster networks, brings new challenges and opportunities for systems that can scale, are secure, have good performance, and can tolerate faults in the software and hardware.

This issue will show how distributed computing technologies compare, and how they are used by government and military organizations. We kick things off with Ed Shrum's article detailing military requirements for distributed computing technology

*Continued on page two*

**IN THIS ISSUE**

Tracking the Hawk Page 4

Standards for Real-Time Distributed Object Computing Page 5

Software Agent Systems Meet Web-Centric Component Architecture Page 6

Eliminating the Middleman: Peer-to-Peer Technology for Command and Control Page 8

How Peer-to-Peer Systems Work Page 9

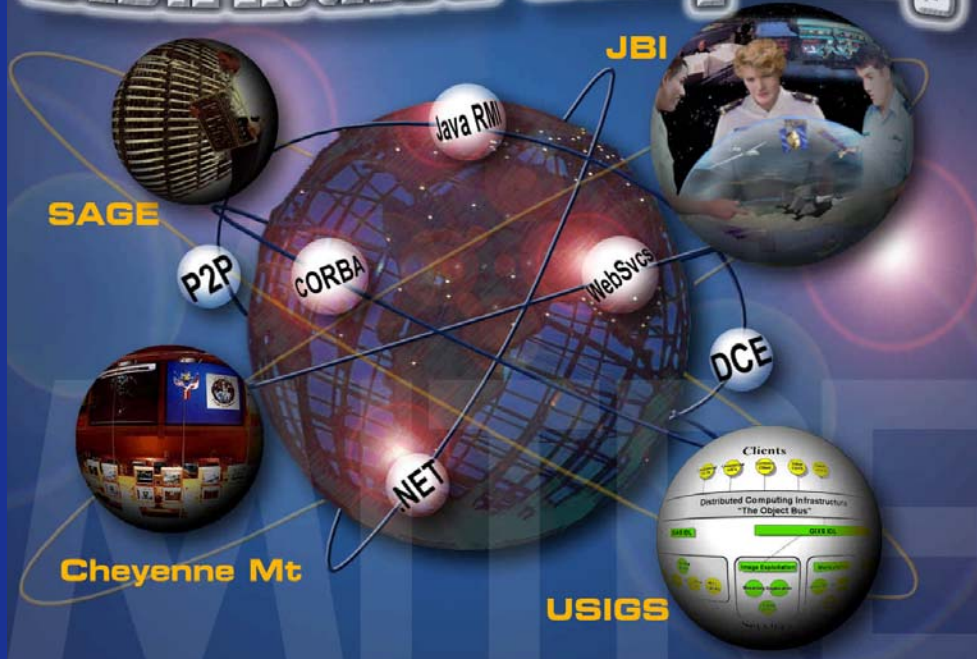
Authentication: Are You Who You Say You Are? Page 10

The Roots of Distributed Computing Page 12

Also *The Edge* Online Page 12



**Distributed Computing**



**Distributed Computing Aids Battlefield Control** *By Ed Shrum*

We constantly hear about distributed computing technologies at conferences and read about them in trade newspapers and in new titles on local bookstore shelves. Many of the systems we take for granted in our daily lives such as cellular telephones, automated teller machines, and stock trading systems rely on distributed computing. Who in the military uses it? Which techniques does the military use?

Not surprisingly, the Army relies heavily on distributed computing technologies. Why? Ground operations need rapid recovery and redeployment from attacks, quick reconfiguration for new missions, easy scalability from platoon- to division-level activities, and immunity from single points of failure. These requirements imply smaller, lightweight hardware instead of large servers, and software functionality that is distributed across multiple machines.

Press coverage of military systems generally focuses on how warfighters use these systems, so we seldom hear much about the software technologies that underlie them. This article gives an overview of how the Army is using distributed computing and provides lessons learned from one specific project, including advantages and limitations encountered. It also highlights MITRE's role in the transition to newer distributed technologies.

*Continued on page two*

## Introduction

*Continued from page one*

and the Army's current directions. Next is John Kane's article on the Intelligence, Surveillance, and Reconnaissance (ISR) Information Service (ISRIS), which describes how basic Web technologies and novel subscription services are combined to provide real-time services and data from today's operational and developing ISR platforms. ISRIS shows what can be done with distributed computing today, where available resources and technologies joined in new ways can produce powerful capabilities.

Dock Allen's sidebar on real-time distributed computing illustrates the natural evolution of the technology to address concerns of highly reliable, embedded, and performance-constrained applications. Margaret Lyell's article then explores the world of software agent architectures. In addition to comparing agents to other technologies, the article shows how to connect distributed agents to an existing system.

Stan Manoski reports on the U.S. Army's plans to use peer-to-peer (P2P) computing in the era of the "TOC-less" Army; Paul Silvey's sidebar on P2P computing augments the arti-

cle from a technology perspective. David Slattery then examines the issue of authentication in distributed systems, an issue of critical concern to deployed troops. The issue concludes with Ed Shrum's brief history of distributed computing, which places the preceding articles in the context of an evolving discipline.

For more information, contact Eric Hughes at 781-271-7486, [hughes@mitre.org](mailto:hughes@mitre.org) or Kevin Kelly at 732-389-6725, [kkelly@mitre.org](mailto:kkelly@mitre.org)

## Distributed Computing Aids

*Continued from page one*

The Army's Maneuver Control System (MCS) and the larger "system of systems" known as the Army Battle Command System (ABCS) have been using distributed computing since the early 1990s to connect with clients, internal processes, and external systems. Army commanders rely on ABCS to plan and execute battles, while MCS is the central system that interfaces with the other systems to gather tank and other units' positions, field artillery information, intelligence data, air and missile defense inputs, combat support statistics, weather reports, and other data to

display on a map background for the commander as the Common Tactical Picture (CTP).

Early versions of the MCS code simply used UNIX remote procedure calls, but the project soon migrated to the Open Software Foundation's Distributed Computing Environment (DCE). In time, problems developed as the move to an all-digital Army placed increasing demands on distributed systems.

DCE was efficient when passing small, simple data parameters, but as software became increasingly complex, it was too slow and unreliable for the complex data types required for today's more

sophisticated calculations. Memory leaks and other errors in the DCE libraries caused severe performance degradation when systems were heavily used over long periods of time. Random nonrecoverable failures meant that DCE had to be reinitialized every time the system was restarted, which caused system initialization times to become unacceptably long. The need to configure each computer as a single DCE "cell" blocked reconfiguration of the different functions resident on any given host and defeated some of the important Security and Naming service features. Because newer MCS software features an object-oriented design, calls to the more traditional DCE routines required special handling. As developers began to use the Java language their graphical user interface conversions necessary, and this made n more of a bottleneck.

MCS redesigned its CTP to improve performance. The redesign was particularly important because the CTP is reused by the other ABCS systems. MITRE played a key role in incorporating new technologies into the upgraded CTP. The Common Object Request Broker Architecture (CORBA) was used to replace the no-longer-supported DCE, and Java was used to speed up



design of simpler, easy-to-use graphical user screens. MITRE headed a team that selected a CORBA vendor; and several MITRE employees who had expertise in distributed computing, CORBA, and Java were then assigned to work with the developers to design the new architecture, supply jump-start technical advice, and debug design problems. MITRE wrote development guidelines, evaluated security risks, and even provided a DCE-CORBA bridge to ensure interoperability with legacy systems.

MITRE's design ideas enabled the screen developers to use Java's Remote Method Invocation as a direct interface to CORBA without any conversions, and will also allow future development to interface directly to Sun's Jini technology for mobile code architectures and to Java 2 Enterprise Edition (J2EE™) application servers. Java screen prototypes in a MITRE services-based simulation of MCS had paved the way for these technologies, which made it straightforward to convince the sponsor of the need to incorporate these approaches.

Detailed performance measurements of the final product showed that replacing DCE with CORBA yielded a *tenfold* performance improvement. This speedup was due to faster manipulation of complex data types and characterization of every battlefield object as a true object-oriented abstraction, allowing all objects to be cached for rapid retrieval and displayed on the map. Many user screens have been simplified and recoded in Java. The Army has successfully deployed MCS in several field exercises, and other systems within ABCS are

now making similar transitions to CORBA.

Another example of just how far the distributed computing technologies are taking current military applications is the Joint Tactical Radio System (widely known as software radios). In this system all of the communications functions except the radio frequency hardware are built within a CORBA software framework. Also, the operational flight software for unmanned aerial vehicles uses CORBA to communicate with distributed components and controls of the aircraft.



In addition to supporting the object-oriented interfacing techniques that CORBA now provides to MCS, MITRE recently prototyped methods for interoperability between legacy systems and small hardware devices such as personal data assistants. MITRE has also constructed an exploratory prototype of a mobile code architecture using Jini, in

which software is downloaded to the user's device only when it is needed.

Currently, MITRE is constructing two prototypes that provide ABCS and MCS functions to users via a Web Services architecture; one is based on a J2EE Application Server design, the other on Microsoft's .NET technology. These prototypes will undergo in-depth performance studies to evaluate the different technologies and various underlying protocols, as well as the effects of network bandwidth, transmission delays, and errors.

Distributed computing has played a critical role in Army systems and will continue to do so. Recently emerging real-time standards, written with input and guidance from MITRE staff, create new opportunities to apply CORBA and Java, open up the use of distributed computing in time-critical control systems, and provide important benefits to a wide range of users (see sidebar by Dock Allen on page 5). As J2EE, .NET, and Jini evolve, MITRE will prototype their architectural benefits and measure their performance. MITRE staff will continue to be involved with the emerging distributed computing technologies and can provide timely technical advice and guidance not only to the Army, but also to sponsors such as the Air Force, Navy, Internal Revenue Service and other civilian government agencies.



For more information, contact Ed Shrum at 732-935-5597 or [eshrum@mitre.org](mailto:eshrum@mitre.org)

## What Is Distributed Computing? Why Is the US Army Interested?

- A distributed system consists of multiple executables interacting with each other over a network
- Why use distributed computing?
  - Geographically diverse locations
  - Replication of functionality in mission-critical applications
  - Several small machines are cheaper than one large one
  - As capacity is exceeded, more machines can be added

## The Challenges Facing Distributed Computing

- Different hardware platforms
  - Workstations, personal computers, notebooks, personal data assistants
- Different software operating systems (each has its own internal data representation)
  - UNIX, Windows, Linux, Windows CE, etc.
- Interprocess communication (IPC)
  - Each process must establish a communications channel with every process it talks to
  - Data formats must be converted.
  - Need to support multiple protocols: Sockets, IPC/RPC, TCP/IP, VME, others

TCP/IP: Transmission Control Protocol/Internet Protocol  
VME: VersaModule Eurocard

# Tracking the Hawk *By John Kane*

**I**magine being able to follow—in real time—the progress of a live Global Hawk or Predator unmanned aerial vehicle (UAV), or any other manned or unmanned tactical intelligence, surveillance, and reconnaissance (ISR) platform, all from within your Web browser. Link that up with user subscription services like those being developed by the Air Force Joint Battlespace Infosphere (JBI) and the Department of Defense (DOD) Web portal development efforts and what do you get? The next generation of DOD Web services.

MITRE's Intelligence, Surveillance, and Reconnaissance Information Service (ISRIS) research project is experimenting with the latest commercial Web technology to make the real-time services and unprocessed sensor data from today's operational and developing ISR platforms available to the DOD community with no client-side requirements other than a commercial Web browser. We chose Global Hawk as the proof-of-concept platform because of its open ground station architecture, standard data formats, and multiple sensor data types: electro-optic, infrared, synthetic aperture radar, and ground moving target indicator.

MITRE has developed and demonstrated an ISRIS prototype server for the Global Hawk. With it, users can access past, future, and current mission, navigation, and collection plans in a dynamically generated report or in an interactive map display. They can also review sensor data for quality and utility before using time and bandwidth to retrieve the raw data. Collected primary imagery is converted into a browser-compatible format that displays an image thumbnail and a list of National Imagery Transmission Format Standard header information. Real-time situational awareness information allows users to monitor an ongoing mission. The map display provides



situational awareness information, including current UAV position, flight path, waypoints, and collection targets. Each data point on the map can be queried for detailed textual information. A geospatial search capability is also provided that enables users to search for collection targets and completed imagery within an area of interest. The prototype uses the XML (eXtensible Markup Language) capabilities of the ISRIS commercial off-the-shelf software to enable data export to external systems and applications such as common operational picture displays and collection/sensor/mission management tools.

MITRE is currently developing a proof-of-concept ISRIS Web service for the Predator. The Predator platform introduces the challenge of providing Web

browser access to real-time video clips and ephemeris data. The interactive map display provides situational awareness information, including current UAV and sensor target position, flight track, and sensor track. We are also investigating low-bandwidth Web-based video stream/clip display solutions.

Who would be interested in such a capability? How about a cross-service Time Critical Targeting cell collaborating to strike mobile targets using the real-time sensor data and dynamic sensor retasking capabilities of both Global Hawk and Predator UAVs? How about an exploitation center that identifies a suspicious object on a wide-area-search image and needs the current UAV locations in order to determine the quickest means to initiate a high-resolution

image retasking request? Or how about a Joint Task Force Commander, who wants to check on the progress of the current theater UAV missions without interrupting the operational staff?

All levels of the Department of Defense, from the Assistant Secretary of Defense for Command, Control, Communications, and Intelligence to the individual armed services, have issued calls for Web enablement and levied requirements and mandates to bring more content and capability to the battlespace internets. The ISRIS project is pushing leading-edge Internet technology to the ISR ground station and, in doing so, is opening up a new level of information accessibility and Web services to the command, control, communications, computers, intelligence, surveillance, and reconnaissance community. The introduction of these raw data sources, some

**All levels of the DOD have issued calls for Web enablement... which requires a distributed Web service computing architecture.**

with real-time feeds, will change the static product paradigm of today's battlespace internets and require a distributed Web service computing architecture in which to operate effectively.

While this architecture does not exist today on our operational networks, the MITRE Technology Program, in concert with the DOD, is working on the new technology, processes, and models necessary to make it a reality. MITRE's Air Force-sponsored Next Generation JBI

Core Services research project is developing the distributed architecture for publishing, subscribing, transforming, and personalizing information objects. This publish-and-subscribe model will define the broker service between the data producers, such as ISRIS, and the consumers, who could be end users, applications, or systems. MITRE's Distributed Metadata Services, or DiMeS, project, is developing the profiling language for the JBI that will be used by both service providers and consumers to match users with data, and data with users. The ISRIS project is collaborating with both efforts to produce the next generation of distributed Web services for the DOD.



For more information, contact John Kane at 757-673-5706 or [jkane@mitre.org](mailto:jkane@mitre.org)

## Standards for Real-Time Distributed Object Computing

By Dock Allen

Over the last 12 years, middleware and programming standards have evolved for distributed object computing, including the Common Object Request Broker Architecture (CORBA), Unified Modeling Language (UML), and the Java language. As these standards achieved commercial acceptance, MITRE initiated, coordinated, and led activities to extend them for real-time systems.

MITRE chairs the Real-Time Special Interest Group in the Object Management Group, which owns the CORBA and UML standards, leads the Java Community Process Distributed Real-Time Specification for Java Expert Group, and has participated in the Real-Time CORBA specifications, Java Community Process Expert Group for the Real-Time Specification for Java, and J-Consortium Real-Time Core Extensions. MITRE's participation has leveraged MITRE standards research and ensured that the standards share a compatible programming model.

As a result of MITRE's efforts, CORBA now supports both real-time fixed priority distributed computing and dynamic scheduling, as well as multicast and time synchronization, and a set of UML profiles for real-time systems has been specified. A set of related CORBA extensions supports embedded systems, fault tolerance, and parallel processing. Ongoing work focuses on "pluggable" transports, real-time data distribution, load balancing, online updates, and a real-time event/notification service. Real-time CORBA forms part of the real-time extensions to the Department of Defense's (DOD) Common Operating Environment, and is being used in systems such as the Air Force's Theater Battle Management Command and Control System.

Recently, the Java platform has been extended independently by the Java Community process and the J-Consortium to support real-time systems on single Java platforms. A third specification under development will extend the Java Community Process specification to include distributed real-time Java applications. This technology is being considered for future DOD applications.

Some challenges for MITRE are to extend the real-time support in CORBA to other paradigms such as transactions, continue the real-time and related extensions of UML, expand the areas of Java that support real-time systems, bring Quality of Service-based real-time support to Web services and peer-to-peer computing, and devise theory and methodologies for dynamic asynchronous real-time systems.



For more information, contact Dock Allen at 781-271-8216 or [dock@mitre.org](mailto:dock@mitre.org)

# Software Agent Systems Meet Web-Centric Component Architecture

By Margaret Lyell, Gary Vecellio, Lowell Rosen, Holly Xiao, Michelle Casagni, John Warren

**M**any MITRE sponsors are migrating various legacy applications to a Web-centric architecture, a design in widespread commercial use today. This type of architecture involves a multi-tier server system. Software that encapsulates the business logic resides on a server tier. Clients can be Web browsers or specialty applications, such as a map client. Enterprise Application Integration products, application servers, and portal products typically have this type of architecture.

An alternative architecture, the software agent framework, has emerged from laboratories and into the commercial domain. The software agent architecture arose out of advances in

accepted, with different software agent types having additional attributes.

Agent types include the personal assistant agent, the human avatar agent, and the mobile agent. Due to emphasis on attributes of asynchronous communications, cooperation, and coordination, agents bring flexibility to system design and offer loose coupling alternatives for system interoperation. In the future, we expect applications to be developed to work with agent systems or to be developed in an agent framework.

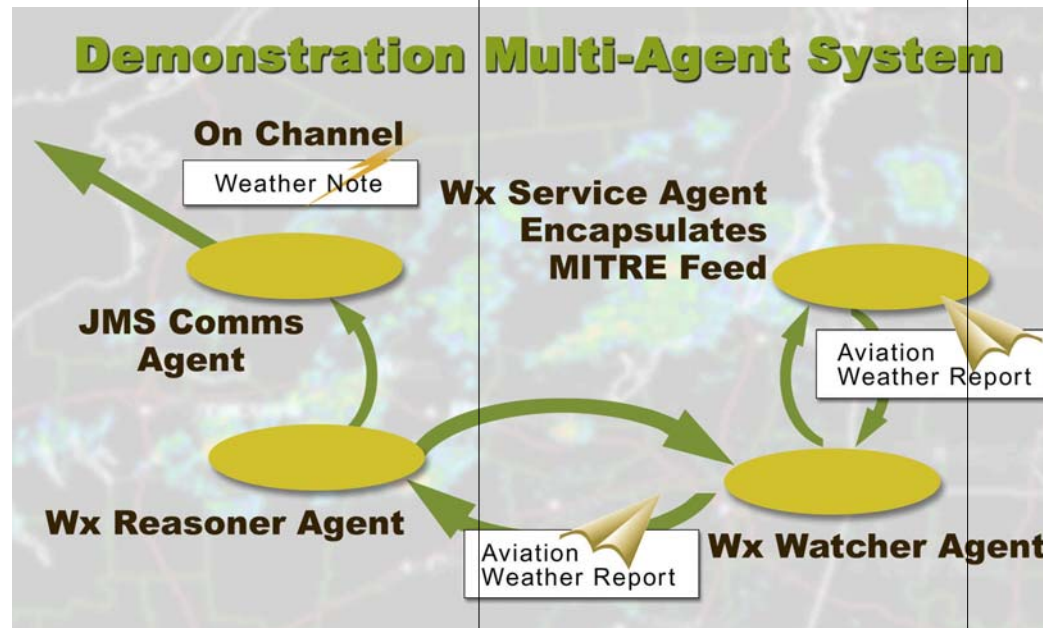
If this occurs, most organizations that have already ported selected applications to the Web-centric architecture face another round of decisions about migration, this time to software agent frame-

erabilities, synergies, and conflicts between applications in the Web-centric architecture and software agent systems. Initially, we focused on the role that messaging—a key element of distributed computing—plays in promoting interoperability between the two frameworks. We did so by developing a multi-agent system to address one aspect of the interoperability problem: that of a Web client receiving information produced in the software agent framework. In our demonstration system, the product is a Weather Note; it is offered to clients of the Web-based architecture via enterprise-level asynchronous messaging.

We are using the Java 2 Enterprise Edition (J2EE™) platform as the Web-centric architecture exemplar and the Foundation for Intelligent Physical Agents Open Source (FIPA-OS) agent development environment as the representative software agent framework. Open source versions of both platforms are available. The J2EE is based on open specifications, whereas the FIPA-OS platform is based on standards set by FIPA. Open standards enabled us to compare relevant specifications across frameworks and make initial assessments prior to starting development. They also allow users to avoid vendor lock-in. Open source code permits code inspection for purposes of assessment and modification.

## J2EE Platform

In the J2EE architecture the bulk of computations is done on the server rather than on the user's computer. The multiple layers, or tiers, of the architecture promote flexibility and reusability because developers need



several technology areas, including distributed artificial intelligence, software component technologies, and human-computer interaction. With all of these contributing areas, it is understandable that multiple definitions for "software agent" arose. A core set of attributes that compose a software agent is now

works. Will they find that the latest commercial software they would like to incorporate into their Web-architecture system is agent-based? How will they use this new software efficiently?

To answer these questions, a MITRE project is examining potential interop-

only modify or add a particular layer, rather than rewrite an entire application if the underlying technology changes. The J2EE platform comprises several technologies as well as the application client and applets in the client tier. It includes container technology—essentially middleware that provides system-level services to the components in a container. The J2EE platform also includes application programmer interfaces.

One of these, the Java Message Service (JMS), supports asynchronous messaging. Our project made use of JMS to facilitate interoperability.

### Software Agent Systems

As the illustration on page 6 shows, our demonstration system incorporates four software agents. There is no fixed client-server arrangement, nor does one agent direct the actions of all other agents. The Weather Service Agent “wraps” the weather feed, which contains two reports important to aviation: meteorological aviation reports and terminal aerodrome forecasts. From this raw weather information it creates an intermediate product, the Aviation Weather Report, which it then stores internally. The Weather Watcher Agent organizes Aviation Weather Reports along a geographical and time-ordered scheme and provides quick retrieval of these reports for agents that ask for them. The Weather Reasoner Agent must decide if a Weather Note—such as “Fog at Airport XYZ”—is warranted. It uses the Aviation Weather Reports as input to its decision-making process. Once the Weather Reasoner Agent has issued a Weather Note, it needs to communicate it to clients. However, the clients are outside of the multi-agent system; they are actually clients

## FIPA

FIPA Message

FIPA Envelope

FIPA Envelope {name, value} pairs

FIPA Transport-Message

FIPA Transport-Message Payload

## JMS (J2EE)

Aggregate JMS  
Message Headers

JMS (optional)  
Message Properties

JMS Message

JMS Message Payload

of the J2EE system, residing in its client tier, who would like to receive the Weather Note product. The Weather Reasoner Agent uses the JMS Comms Agent to send the Weather Notes to them.

Communication among agents does not simply involve message passing, but is instead a conversation, made up of a series of messages. For example, if the Weather Watcher Agent is to organize the Aviation Weather Reports, it must first request them from the Weather Service Agent. This request conversation involves a minimum of three messages.

### Message Mapping

If the demonstration system were a robust commercial product, one goal would be to provide the Weather Note to as many clients as possible. However, many J2EE clients would encounter a semantic mismatch with the software agents, and would be unable to communicate. Moreover, the client base would be limited if all clients were required to speak “agent-language” and participate in conversations. If the JMS Comms Agent could take the conversational messages from the Weather Reasoner Agent, extract the Weather Note information, and

then construct and forward a message that could be used in an enterprise messaging system, clients of the enterprise messaging system could receive the product.

Our prototype system demonstrated how to achieve this. We created mappings between FIPA messages and JMS messages (as shown above), and then instantiated the mappings in software as the JMS Comms Agent. This agent serves as the bridge between the FIPA-compliant multi-agent system and clients of the J2EE/JMS platform.

Our particular client, a map display client, subscribes to a “fipa-weather” channel using JMS. The Weather Notes appear whenever a user drills down on an airport location. Currently, the notes can be read in text form, easily displayed as map pop-ups. It would be straightforward, however, to put them in XML format, which would facilitate having software programs process the information. We are now investigating how software agents work in the Web services realm.



For more information, contact Margaret Lyell at 703-883-6509 or [mlyell@mitre.org](mailto:mlyell@mitre.org)

# Eliminating the Middleman: Peer-to-Peer Technology for Command and Control

By Stanley Manoski



Each day millions of people around the world use tools with such strange names as Morpheus, Bearshare/Gnutella, Freenet, and the notorious Napster. The tools, collectively known as peer-to-peer (P2P) technologies, have become the leading-edge way to exchange multimedia files over the Internet but are largely unexplored for the exchange of structured information or for government use. In a P2P system, each computer, called a node, has the ability to function as both a client

and a server. In practice, the majority of users of P2P systems function as distributed clients only. The sidebar by Paul Silvey (see page 9) describes the general characteristics of P2P systems.

The ability to deliver and use timely, accurate information in a secure manner is key to the success of any battlefield activity. Information flowing through battlefield systems is used to determine the courses of action in the battle; therefore, deployed forces need highly redundant distributed systems that are hard to kill. Could P2P technologies offer new capabilities for command and control systems in the battlefield environment? Using filesharing technologies as an illustration, this article suggests how they might help to address the needs of military users. The online version of the article ([www.mitre.org/pubs/edgesummer\\_02/manoski2.html](http://www.mitre.org/pubs/edgesummer_02/manoski2.html)) summarizes the characteristics of a few well-known P2P systems.

MITRE has been involved in a variety of projects using P2P technology for defense needs. These efforts range from analyzing security aspects of the emerging P2P technologies for the North Atlantic Treaty Organization (NATO), to Defense Advanced Research Project Agency (DARPA) investigations on Collaboration Technology (e.g., Groove) and Network Centric Command, Control, and Intelligence, to potential use of P2P in Army Battle Command Systems. MITRE is helping to realize the potential of P2P technology to enhance future battlefield systems.

## The Battlefield System Environment

How might battlefield systems benefit from applying P2P tools? Battlefield systems generally have highly redundant, dynamic (ad hoc) network infrastructures, and communication usually takes

place over low-bandwidth “pipes.” Both systems and networks need to be reliable and secure, and systems should be quickly deployable and self-configuring or adapting. Because the average soldier is not an engineer, systems should also be easy for people with little technical training to support and understand.

Addressing conventions pose a continuing problem in battlefield systems. Do you reference individual soldiers, units, machines, or platforms? How do you reference them: by IP (Internet Protocol) address, domain name, URN (uniform resource name), or URL (uniform resource locator)? What happens when they move around the battlefield, or when a unit or individual reattaches to an existing unit or “split-jumps” from the original unit?

Battlefield systems must ensure the integrity and authenticity of information and control access to information. They must ensure that users are who they say they are (authentication), and ensure that authenticated users are allowed to retrieve the information (authorization).

Battlefield systems are increasingly automated. Information is not just retrieved or “pulled,” rather some information is automatically distributed, or “pushed,” among battlefield systems on the basis of domain-specific criteria.

Finally, information, like people, generally has a sphere of influence. Some information is used, as is, by pockets of users. Other information is refactored into other forms usable in other locales. For example, situation awareness (e.g., location of friendly and enemy troops) has different spheres of influence. Detailed information about individual soldiers or platforms has critical value to others in the vicinity, but higher echelons in the battlefield require such information to be aggregated or “rolled up.” Battlefield sys-



tems must avoid overloading users with information.

### Matching P2P Capabilities with Battlefield Needs

No existing P2P system can provide the single solution to problems faced in battlefield system environments. However, some of the key concepts and aspects of available implementations may help to resolve certain difficulties. For example, all of the P2P systems mentioned are easily maintainable and self-configuring, and all allow any node simply to attach to the network at either a central point or from anywhere. These networks evolve without user intervention other than usage, so the maintenance and robustness exhibited by P2P networks appear to correlate well with battlefield requirements.

With regard to addressing, all P2P systems bypass traditional domain name server usage. Each uses its own scheme for addressing: generally mapping a user-defined ID to the current IP location of the user. If a user relocates, simply connecting to the system creates a new association, so the user can “float” in the net-

work to other locations with full functionality. This capability would be valuable for mobile users, such as deployed Army units.

Node connections in some commercial P2P networks change over time. More capable (i.e., higher bandwidth) nodes migrate to the center of the networks because they better support traffic needs; thus, the network adapts to the capabilities of the nodes composing it. Such networks are also highly resilient: the lack of central control makes them very difficult to disrupt. The software of certain other P2P networks permits “intelligent downloading,” in which a selected file can be simultaneously downloaded, or retrieved piecemeal, from multiple nodes and rebuilt at the final destination. The system automatically cancels the connections of nodes that perform poorly during transfer and uses other nodes with better performance to retrieve the missing file fragment. If a download is terminated for any reason, it can be resumed later to complete the transfer. These types of features could potentially make P2P networks especially valuable for MITRE’s military sponsors.

Of all the commercial systems currently available, few can ensure information integrity and authenticity. Freenet, which uses encrypted keys, is one of the most promising P2P technologies for these requirements. The very nature of the network protects the key requests, data files, and even user anonymity. However, managing keys remains a major issue, as does discovering or sharing the keys used for making requests. Moreover, unlike the other P2P technologies discussed, Freenet has no “search” mechanisms that allow users to request metadata about files.

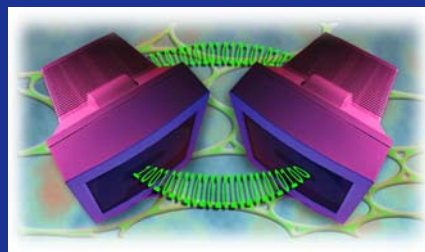
One aspect of Freenet with special relevance to the battlefield environment is the natural migration of information to where the information is most often requested. However, all current P2P systems are “pull” technologies, whereas battlefield systems require automation, and “push” services often are one aspect of automation. A “push-based” Freenet would be an interesting prospect.



For more information, contact Stanley Manoski at 732-389-6772 or [manoski@mitre.org](mailto:manoski@mitre.org)

## How Peer-to-Peer Systems Work *By Paul Silvey*

Peer-to-peer (P2P) information management and computing architectures use decentralized or distributed control mechanisms to share resources—typically data or processing cycles—in a scalable, flexible, and fault-tolerant manner. P2P is often seen as an evolution from two-tiered client-server and three-tiered middleware-based architectures, which assign particular functions to distributed computers, toward a more homogeneous environment in which each information processing and storage node (peer) in a network can simultaneously act as a producer, consumer, and facilitating intermediary. In practice, many systems based on P2P concepts exhibit at least some functional partitioning, such as having one or more specially designated peers act as rendezvous sites or matchmakers. Current systems that incorporate P2P concepts include Gnutella, Freenet, SETI@Home, and Groove Networks. An open source effort known as



JXTA provides an interesting general-purpose architecture and protocol standard on which P2P systems can be built.

Generally speaking, all resources shared in a P2P network may be considered services. In this view, offering information to be shared or offering computations to be performed are both instances of providing services and delivering results. This service-centric view allows us to think of peers as agents with knowledge that other peers can access remotely through con-

versational interactions. One peer might then ask another for an answer to a question, or request that some work be done on its behalf. This is one reason why P2P architectures, “Web service” architectures, and Multi-Agent Systems are often seen as similar. In their pure form, P2P architectures leverage the power of the network through inter-peer communication at runtime, giving each peer equal access to the current state and collective knowledge of the online membership (the nodes currently connected). The inherent ability of a P2P network to distribute workloads across multiple machines enables it to scale to handle large information management or computationally complex problems. These kinds of large-scale problems are common throughout the federal government, intelligence community, and military services.

*Continued on page eleven*

# Authentication: Are You Who You Say You Are?

By David Slattery

**T**he multiplicity and unpredictability of today's missions require the U.S. armed forces to be increasingly mobile and adaptable, while leveraging resources effectively across wide geographical and organizational spaces. This has led to an increased reliance on distributed computing systems. However, setting up and maintaining distributed systems under the strain of battlefield conditions and with limited manpower presents serious administrative challenges. Emerging approaches in distributed computing address many of these challenges by allowing software services to form dynamic groups in which they can discover each other's services, agree on communication protocols, and interact with each other without requiring prior manual configuration. However, these new approaches also give rise to a number of security concerns. For example, how does one verify the authenticity of the services that form such self-configuring groups? Distributed computing systems need to address these concerns to protect the integrity, confidentiality, and availability of data and resources on the network. MITRE and the National Security Agency are currently researching and developing solutions to address these security concerns.

Our goals are to develop security solutions for distributed systems that can be readily tailored to meet the security requirements of individual environments, are easy to maintain, and are robust. We are currently prototyping these solutions with Sun Microsystems' Jini Network Technology. Jini provides a distributed computing infrastructure that supports dynamic interaction between users and services, and a programming model for the construction of reliable services. Users and services use this programming model and



infrastructure to call each other, discover each other, and announce their presence to other services and users.

One of the most critical security services is authentication, because it constitutes the cornerstone for additional security mechanisms such as authorization and auditing. Participants in a distributed system must be able to satisfy each other about their respective claims of identity. We therefore selected authentication as the first security service to implement in our prototype—a service greatly complicated by the distributed nature of the system.

We have developed a software prototype that uses X.509 digital certificates to authenticate the hardware and software services in Jini. The prototype allows Jini services and clients to authenticate each other and establish encrypted Secure Socket Layer or Transaction Layer Security connections for use in secure communica-

tions. It also enables the client to authenticate the mobile code based on a digital signature before the code is executed. Having a trusted agency digitally sign the mobile code allows us to provide assurances that the mobile code comes from a trusted source and has not been altered or corrupted since it was signed.

One of the key tasks in designing distributed systems is to ensure the system is sufficiently flexible and extendable to adapt to diverse and evolving requirements. To address this challenge, our prototype leverages the Java Authentication and Authorization Service package, a Java version of the standard Pluggable Authentication Module (PAM) framework. The PAM framework allows applications to remain independent of authentication schemes, because it can integrate multiple authenti-

cation mechanisms by plugging them into an application at runtime. Application developers who use the PAM interface with a single high-level application programming interface are decoupled from changes in security policies and authentication mechanisms. System administrators have the flexibility of selecting one or more authentication technologies on the basis of their local security policy and are not required to modify each application. We are investigating possible incorporation of authentication technologies in addition to X.509 digital certificates, such as Kerberos and smartcards, via the PAM framework.

Although this prototype offers a flexible authentication scheme, it requires that each client or application manage and configure its own individual security mechanisms. This leads to increased system complexity. As a result, issues as basic as ensuring that all clients have the same

security policy can become serious administrative burdens.

MITRE has created a second prototype that reduces the complexity inherent in such approaches by moving the bulk of the authentication processing to a centralized service. This allows users and applications to authenticate to a known and trusted entity. It also reduces the management complexity for system administrators, who would have only one service to maintain and monitor.

In addition, the second prototype has added support for authorization, a security service closely tied to authentication. When an entity successfully authenticates to our prototype, the service compares its identity to an authorization policy to determine the entity's privileges and then issues an authorization token to the authenticated entity. This token is a digitally signed statement that contains information about the entity's name, privileges, and any constraints on the token. The token can then be presented to a Jini service, which can use the token to determine what access privileges should be given to the owner.

These prototypes provide a powerful framework that can be extended to include other authentication and authorization technologies beyond those we

have built and to support additional security services. And we see room for further improvement. When security services are centralized, they can be leveraged to many diverse applications and reduce administrative complexity, but they can also become desirable targets and a single point of failure. On the other hand, moving toward a fully distributed approach carries with it the additional risks and complexities associated with maintaining coordination, management, confidentiality, and integrity among multiple security services. Therefore, our goals for this year's research strike a balance between these two approaches. Our primary emphasis will be on ensuring the fault tolerance of decentralized security services by distributing the services among a set of servers and using replication algorithms to mask faulty servers.

A primary question for the next prototype is how a group of distributed processes can agree on the authenticity of a user when a malicious adversary may corrupt some of the processes and disrupt the network. We believe that we can improve the system's overall integrity by not completely trusting any single process, deriving results from the majority of correct processes, and increasing the quantity and diversity of authentication services that work in collaboration, and in this way decrease the risk of system-wide failure.

We are currently researching ways in which a group of distributed and diverse authentication processes can reliably reach agreement on the authenticity of a principal through such techniques as distributed consensus algorithms and threshold cryptography.

The importance and complexity of distributed computing systems continues to increase, leading to greater strains, being placed on system developers and administrators. To help relieve these strains, distributed systems are becoming more automated and dynamic. It is essential that we have security solutions that can rapidly adapt to these environments, are easy to maintain, and are robust. Our first prototype allows extensive customization right down to each individual client and application. Our second prototype still allows for rapid and extensive customization but moves the security services to a centralized location for easier management. This year we are developing a next-generation prototype that can take centralized components, such as our authentication service, distribute them among a set of servers, and tie them together to form highly robust and resilient services.



For more information, contact David Slattery at 781-271-5543 or [djslatte@mitre.org](mailto:djslatte@mitre.org)

## How Peer-to-Peer Systems Work

*Continued from page nine*

Traditionally, servers need to have a fixed, known address on the network, and ideally they are "always on." Clients, however, prefer the flexibility of transience, connecting only when they want to and using different network addresses in different usage modes. When a peer is both a client and a server, these goals are at odds with one another. The solution requires P2P systems to support "peer discovery protocols" that allow peers to find each other by indirect means, and to expect results to differ over time as peers leave and join the network. Peer transience is a problem that instant messaging or "chat" systems have to address. Planning for transience also tends to

make P2P systems more robust with respect to failures, a feature that makes them potentially applicable to military command and control applications.

Run-time directory services such as peer discovery can also be used to handle the problem of indexing near-real-time or perishable information, which is another form of transience. Web crawlers, for example, cannot keep up with the continuously and rapidly changing state of Internet-accessible content. P2P approaches to information management allow queries to be propagated to peers that might be able to answer particular questions without requiring any prior crawling or central indexing. This mechanism is generally known as distributed search.

In the Distributed Metadata Services (DiMeS) MITRE project, we are exploring ways to advertise what peers have and what they want using a general-purpose profiling language based on structured metadata from community-of-interest eXtensible Markup Language (XML) schemas. Our profiling language is designed to allow intermediate peers to remember and aggregate service descriptions selectively from profiles they have been told about or have discovered, enabling the knowledge network to route requests, results, and peer-referrals more efficiently.

For more information, contact Paul Silvey at 781-271-7502 or [psilvey@mitre.org](mailto:psilvey@mitre.org)



**For newsletter feedback or  
submission information, contact:**

Regina Hansen

Email: [hansen@mitre.org](mailto:hansen@mitre.org)

Mail Stop: N310

Telephone: (703) 883-7301

© 2002 The MITRE Corporation

All rights reserved.

*This document has been approved  
for public release.*

**The MITRE Corporation**

202 Burlington Road  
Bedford, MA 01730-1420

7515 Colshire Drive  
McLean, VA 22102-7508

[www.mitre.org](http://www.mitre.org)



# THE ROOTS OF DISTRIBUTED COMPUTING

by Ed Shrum

The MITRE Corporation has its roots in distributed computing and has been a principal player in this field from its very beginning. In the 1950s, MIT, IBM, AT&T, Burroughs, and others designed the Semi-Automatic Ground Environment (SAGE) air defense system to protect the United States from long-range bombers and missiles. SAGE was actually a distributed computing system, with pairs of main and fail-over computers in multiple control centers across the country. MITRE was founded in 1958 to conduct all research, development, and systems engineering for the SAGE project, and MITRE's early years were totally dedicated to SAGE. During the 1960s, MITRE applied the same distributed computing techniques to its first contract with the Federal Aviation Administration: the SAGE Air Traffic Integration (SATIN) project. In the same time period, the Air Force directed MITRE to design the now-famous North American Aerospace Defense Command (NORAD) Combat Operations Center for Cheyenne Mountain along the same lines as SAGE.

Distributed computing became available to a more general development community in the 1970s, with the UNIX operating system providing a way to execute a software routine on another computer using remote procedure calls (RPCs). RPCs were a simple concept in an operating system, but their use in real applications was complex because they forced programmers to concern themselves with registering process numbers, host names, input and output data compatibility, process management, error handling, and time synchronization.

Over time, a family of support library routines developed and was included in UNIX. They were eventually standardized in the 1980s by the Open Software Foundation as the Distributed Computing Environment (DCE). Several vendors built DCE products and ported them to various operating systems in addition to UNIX. DCE represented a huge step forward. It was vastly easier to use than the basic operating system calls, and the vendors provided documentation and support beyond the UNIX operating system online documentation. DCE also included rudimentary services for naming (a Yellow Pages-like lookup service), time, and security, all essential to any distributed computing application.

The Common Object Request Broker Architecture (CORBA) standard was written in the early 1990s. In the same time frame, other vendors developed Message-Oriented Middleware (MOM), which placed inter-machine messages in a queue so that if a machine were busy or off line, its messages would be saved until it was back in service. More recently, Sun Microsystems introduced its Java language Remote Method Invocation and Jini, a method of building systems in which code is downloaded to the user's machine only when it is needed. CORBA has recently been extended to real-time environments. Most implementations of these technologies use the RPC mechanism.

Today, the technical literature abounds in articles about Web Services, a loosely coupled architecture in which most functionality resides in services accessible on the network rather than installed on the user's computer. Sun's Java 2 Enterprise Edition standard for application servers is available today, and Microsoft .NET tools were officially released in 2002. There is also much current interest in eXtensible Markup Language (XML), used for data interoperability, which supports distributed computing through its XML-RPC function. The Simple Object Access Protocol (SOAP), originally part of the .NET concept, uses XML for messaging.

## **The EDGE Online**

The Web version of this issue [http://www.mitre.org/pubs/edge/summer\\_02](http://www.mitre.org/pubs/edge/summer_02) includes an additional article that complements those in the print version. David Koester gives insight into networking for high-performance distributed computing, illustrating MITRE support to commercial technology experiments.