Live Distributed Objects for Service Oriented Collaboration

Ken Birman, Jared Cantwell, Daniel Freedman, Qi Huang, Petko Nikolov, Krzysztof Ostrowski
Department of Computer Science, Cornell University, Ithaca, NY 14853

Extended abstract:

Advanced internet collaboration tools are often promoted as crucial to reducing health-care costs, improving productivity, facilitating disaster response, enabling a more nimble information-aware military, and allowing more immersive professional remote collaboration. We term such tools *service oriented collaboration* (SOC) applications to reflect their ability to facilitate communication at the network edge. SOC systems have garnered increasing appeal to developers and users because of the growing body of rich *service-hosted content*, such as electronic medical health records, geographic information systems, image repositories, weather prediction systems, social networks, and a diversity of other databases. These systems may also interface with sensors, medical devices, video cameras, microphones, and other real-world data sources.

The framework we demonstrate herein definitely addresses many of these applications; however, it also encourages many other uses, oriented more explicitly to entertainment and better aligned with the global theme of the Intetain 2009 Conference: "Playful interaction: with others and with the environment." Our work provides the basis for a modern incarnation of a Second Life environment, one that is constructed atop a more principled and more scalable foundation.

A number of difficulties exist with many current implementations of SOC applications. Media streams generate high, bursty update rates and often require low latencies and tight synchronization between collaborating users. Some applications also require client-to-client security and many, especially medical and military scenarios, cannot extend trust relationships to web service platforms or any third parties. These requirements represent serious impediments to implementation with existing web service standards, which involve the relay of data between clients via a hosted service. Typically, this relay is accomplished through some form of an *enterprise service bus* (ESB), using publish-subscribe models, Really Simple Syndication (RSS) feeds, Representative State Transfer (REST) or Simple Object Access Protocol (SOAP) transfers, message-queuing middleware products like the Java Messaging Service, or other methods. However, relaying data through a central server introduces latencies (based both upon round-trip travel times and processing overhead) and poses obvious scalability issues (explaining, for example, the necessity for such a low density of primitives, 117 per 512 square meters, in Second Life). Moreover, web service security models assure client-to-server security, although this offers no benefit to applications that cannot trust any central server.

Something new is needed: a way to create SOC applications that seamlessly integrates hosted content with the kinds of *peer-to-peer* (P2P) protocols capable of responding to these needs. Here, we demonstrate how Cornell's Live Distributed Objects platform solves this problem, enabling a powerful style of collaboration.

We first must look more closely at the way today's developers build SOC applications. Obviously, web services offer rich options for programmatic interfaces to services. Service platforms usually export some form of *minibrowser* component — namely, an interactive web page with embedded scripts, commonly developed using AJAX, Flex, Silverlight, Caja, or similar technologies and optimized for a specific type of content (for example, interactive maps from Google Earth or Virtual Earth). This embedded script is often tightly integrated with backend services in the data center — services that may not even be directly accessible at a programmatic level. As a result, the only way that new content can be "mashed" into the data available from the service is to have the data center itself compute the mashup. For example, Google's minibrowsers expose composite images that draw on multiple data sources. If the client pans or zooms the minibrowser window, the data associated with the mashup is also zoomed or panned. Google also offers tools to help end-users define new kinds of mashups, but the kinds of data that can be mashed together are limited.

With our Live Distributed Objects platform (as seen in the screenshot below and demonstrated more vividly in the attached video), content from different sources is overlaid in the same user interface window and synchronized so that each source displays data that is geographically and temporally aligned. We designed this demonstration to highlight the contributions of different data sources, but there are no visual or conceptual frame boundaries between the contributed data: elements of this mashup (including maps, 3D terrain features, images of buildings or points of interest, icons representing severe weather reports, vehicles or individuals, etc.) co-exist as layers within which the end user can easily navigate. Data can be drawn from many kinds of data centers. Our example actually overlays weather from Google on terrain maps from Microsoft Virtual Earth, extracts census data from the US Census Bureau and flight information from the US Federal Aviation Administration, and also embeds additional reference points from a local shared data store.

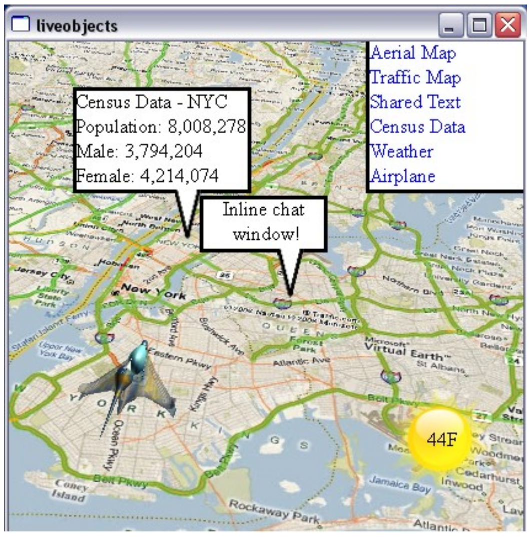
Importantly, Live Objects treats every source or communications channel of content as an object. Thus, our SOC application is not limited to hosted content: it includes components that use direct P2P communication protocols at the edge of the network. We can support any sort of protocol, including client-server, but also overlay multicast, peer-to-peer replication, or even custom protocols designed by the content provider. This makes it possible to achieve extremely high levels of throughput and low latency. It also enhances security as the data center servers need not observe data exchanged directly between peers; we can take advantage of P2P key management protocols that offer provably secure ways to create and share cryptographic keys so that only endpoint hosts can access them.

In conclusion, our work on Live Distributed Objects forms the basis of a new platform for constructing Service Oriented Computing (SOC) applications, while revealing unexpected difficulties in building high-performance SOC systems using current web services standards. These problems are particularly acute with client-side mashups, where scalability and performance problems with enterprise-service bus (ESB) components are central limitations. Live Objects solve these problems, allowing a combination of hosted content with P2P protocols in a single object-oriented framework. This enables both enterprise functionality for health care, finance, and military uses, as well as entertainment capabilities in virtual worlds. For those to experience it first-hand, our platform can be freely downloaded from http://liveobjects.cs.cornell.edu.

Physical Demo Requirements:

We can provide a laptop to power this demonstration but would benefit from some form of larger display so that it can be viewed by a larger audience. Depending upon the environment, this could either be a large LCD screen or an LCD projector. We would need only two standard AC outlets and do not require any related sound capabilities. We can have our demo established within ten minutes.

Sample of the demo:



Census Data - NYC
Population: 8,008,278
Male: 3,794,204
Female: 4,214,074

The screenshot at right shows a specific instance of the Live Distributed Objects framework. Associated with this submission is a more substantial video capture (in AVI format) that shows the manipulation of the framework and better parallels our eventual live conference demonstration. That video can be viewed at:

http://liveobjects.cs.cornell.edu/Intetain2009.avi