IEEE Internet Computing

www.computer.org/internet computing

VWRAP for Virtual Worlds Interoperability

Joshua Bell, Morgaine Dinova, and David Levine

Vol. 14, No. 1
January/February, 2010

VWRAP for Virtual Worlds Interoperability

Joshua Bell • *Linden Research*
Morgaine Dinova • *Independent Consultant*
David Levine • *IBM T.J. Watson Research Center*

The IETF has chartered a Virtual World Region Agent Protocol Working Group (VWRAP WG). This article briefly describes the history of virtual worlds, the architecture, protocols, and operation of Second Life (a currently prominent virtual world), and the emergence of standards efforts within the virtual world space. The authors detail the current efforts and timeline of the VWRAP WG.

In September 2009, the IETF chartered a working group on virtual worlds interoperability. Ranging from enterprise workgroup tools for collaboration to massively multiplayer games, the immersive Internet is an emerging companion to the World Wide Web. The Virtual Worlds Region Agent Protocol Working Group (VWRAP WG) represents a focused effort to begin standardizing the services and protocols needed to deliver shared immersive experiences at Internet scales.

When fantasy worlds, simulation, and social networking meet the Internet in an interactive visual form, the result is more than just transient immersive experiences. The new environments are persistent and as diverse as the human imagination, so the term "virtual worlds" is very appropriate. In the same way that the Web encompasses the whole range of human experience, so, too, do these virtual worlds, which are unconstrained by many physical limits. Built on a bedrock of Internet technology, virtual worlds span the globe, forming a new commons for sharing between people and organizations.

The VWRAP WG is addressing interoperability for a family of current and future virtual worlds based on a common data model and similar operational semantics, among which there is already a strong desire to interoperate.

A Brief History

Virtual worlds have a history that's more than four decades old, beginning with the mid-1970s multiplayer games on the PLATO computer-assisted learning system, the first-person Maze War, and textual Multi-User Dungeons. The PC brought text-based games into wider adoption, followed by graphical environments such as Air Warrior and Lucasfilm's Habitat, a 2D social world with persistent spaces.

Based on the High-Level Architecture (HLA) and Distributed Interactive Simulation (DIS) protocols, the US Department of Defense implemented simulations for planning and training over private networks. The Web's emergence and the Internet's widespread adoption in the 1990s led to social 2D worlds such as The Palace. The Virtual Reality Modeling Language (VRML) attempted to follow HTML's path for distributed shared 3D content, but despite efforts such as the Contact client for shared VRML spaces by Blaxxun Interactive, immersive 3D on the desktop primarily consisted of canned spaces such as first-person shooters like id Software's Quake.

The turn of the millennium brought broadband increases and significant improvements in PCs' graphics capabilities. Massively multiplayer online role-playing games (MMORPGs) emerged, such as Ultima Online, EverQuest, and World of Warcraft. These games supported thousands of simultaneous users. Alongside the game worlds, social spaces developed as well, including Active Worlds, There, and Second Life.

Such services effectively define virtual worlds in media and culture: fully immersive 3D worlds rendered by stand-alone client applications accessing a service over the Internet, portraying realistic but not necessarily humanoid *avatars* moving and interacting in shared spaces. These worlds generally include extensive customizations and user-generated content and locations, supporting ad hoc social interaction and organized events such as games and music, as well as micro-economies with user-to-user transactions.

Despite the gradual adoption of standard data formats for content and transport protocols, widely adopted standards for virtual worlds' basic needs — such as placing avatars into shared spaces and mediating their interaction — haven't yet emerged. This has meant that each new service has involved engineering from the ground up, so the opportunity for interoperation between services has been limited.

The Second Life Model

Second Life is a virtual world service from Linden Research. The company began with a vision of building hardware interfaces for accessing virtual environments with robust physics simulations. Early experiments showed that the standard monitor, mouse, and keyboard were sufficient for an immersive experience, and Linden abandoned its hardware effort. However, this early vision influenced Second Life's design.

At a high level, Second Life supports an Earth-like virtual world with conventions such as ground, water, sky, wind, and gravity. The world is partitioned into uniformly sized 256-meter × 256-meter *regions*. Each region is hosted by a *simulator* service, which maintains and updates the state of everything within a region. Multiple regions are laid out as 2D tiles — not necessarily contiguous — and numerous cooperating computers are necessary to support a large virtual world. An instance of such a world is known as a *grid*.

The virtual world's main data components, which together provide the immersive experience, are static *assets*, such as textures, animations, and sound files, and *objects*, which are user-built constructions made from parameterized geometric primitives, as well as other data that defines land geometry and visuals. Assets and objects are the basis for all user-generated content in Second Life, and are also used to create avatars — that is, the visible instantiations of logged-in users. The client-side rendering of these miscellaneous components creates a "mashup" of these visual elements on the screen, and, if done well, the end result is a highly believable and hence immersive representation of the user's avatar within an imagined world.

Second Life's users interact with regions using *clients*, and a standard 3D immersive client is known as a *viewer*. Such viewers authenticate a user and then receive a stream of state information from the region. Viewer controls mediate the user's interaction with the region, navigating the avatar within the virtual space and performing actions within it. Once a user's avatar has entered a region, the user is said to be *in-world*.

Viewers render the virtual space for users. A viewer requests world content from a region and then processes it to form a visual image, which the viewer then presents to the user. Note that this model implies that detailed copies of objects are transmitted to clients, rather than (for example) rendering them to a video stream on the server. In sending user-generated content to clients, this model has fundamental implications for nontechnical issues that often arise, such as ownership and copyright. Similar issues can be found on the Web in the way that Web servers and browsers enable the copying of any content displayed within a Web page. Clients can choose what to request and display, enabling the creation of a broad range of experiences and tools that interact with the virtual world, just as Web crawlers and other tools interact with the Web.

Open Source Virtual Worlds

Second Life's client/server protocol was initially closed. Informed by the gaming industry and anticipating worlds in which most content was dynamic, the protocol aimed to be compact, efficient, and hard to tamper with. It was initially UDP-based — with a fixed format for each distinct message type — and intentionally obfuscated. Users, of course, merely took this as a challenge. First, a proxy application that could inspect and modify packets appeared. Then, John Hurliman and other enterprising developers in the open source community implemented a library initially known as "libsl" (see http://lib.openmetaverse.org/wiki/Main_Page) by reverse engineering the protocol and released the library under the Berkeley Software Distribution (BSD) license, enabling the creation of lightweight clients that could log in to Second Life and perform automatic tasks, such as search indexing.

To encourage such creativity and foster an ecosystem around the Second Life platform, as well as to support its evolution, Linden Lab extended and simplified the protocol, eliminating the obfuscation. Additionally, the lab released the source code to the Second Life viewer application under an open source license (GPLv2), encouraging developers to contribute new features and build their own systems to interact with Second Life.

The Open Simulator Project (OpenSim; http://opensimulator.org) started in early 2007, when Darren Guard and Steffan Andersson combined the possibilities inherent in the open sourcing of the Linden client with the protocol implementation in libsl. From a simple experiment permitting a bare avatar to move in a blank space, OpenSim grew into a complete open source software project focused on building a platform that would enable a range of virtual world implementations. OpenSim currently supports multiple protocol modules running in parallel and a broad range of deployment patterns for its services, including interoperation via HyperGrid (a protocol enabling teleportation links between OpenSim grids) and experimental Open Grid Protocol (OGP) support.

OpenSim is a substantial project programmed in C# and running on both the proprietary Microsoft .NET framework and the open source Mono runtime. It has a fairly traditional open source project community comprising a team of core developers, an extended set of contributors, a lively mailing list, wiki, and IRC channel, and weekly OpenSim-hosted meetings.

For many individuals and organizations, the attraction of running your own virtual world is very strong. The freedom to control your own world without service restrictions creates many new opportunities, and the ability to choose between using hosted services or operating your own servers has led to a steady rise in interest in OpenSim technology. Many independently operated OpenSim regions have federated into open grids such as the OSgrid, and commercial OpenSim-based grids and region hosting services are also appearing. Unsurprisingly, all this activity in the open arena has fueled a desire for interoperation with the commercial world of Second Life as well.

Beyond the Second Life ecosystem, many other projects are creating additional open source worlds. Croquet/Open Cobalt (www.opencroquet.org/index.php/Main_Page; www.duke.edu/~julian/Cobalt/Home.html), Project Darkstar/Wonderland (https://lg3d-wonderland.dev.java.net), and Sirikata (www.sirikata.com) are among these initiatives, and bridging the gap is the realXtend project (www.realxtend.org), which provides OpenSim compatibility but also extends the concept in new directions.

Scaling and Interoperability

In 2007, Linden Lab began to look at the challenges of growing Second Life to "Internet scale." Setting a goal of millions of users and tens of millions of regions, scaling issues became paramount. At the same time, Mark Lentczner, a senior architect at Linden Lab, realized that solving the scaling issues and transitioning to a Web-based protocol would let the company consider standardization and interoperability at the same time.

Linden Lab concluded that scaling challenges occurred along two dimensions: agents (users) and regions (locations). By moving from a two-way client/server model focused on the viewer and simulator to a model with separate agent and region services, the scaling problems could be solved independently of one another.

The viewer application would authenticate against a set of agent-specific services, and these services would then place the agent at a particular location via region-specific services. The viewer would maintain connections to both sets, and the agent and region services would also maintain a communication channel to track the agent's location within the virtual world, update avatar appearance, perform permission checks, and so forth.

This three-way arrangement has many benefits. The simulator would no longer need to handle elements of the protocol that don't involve virtual locations, such as instant messages. And although the design allows for scaling and provisioning of agent and region services independently for a virtual world operated by a single company like Linden Lab, such services could also exist within separate trust domains or be deployed by separate organizations.

A large organization wishing to offer a virtual world to its members would implement an *agent domain* (a set of agent services) on top of its directory service and create a private virtual world for members within a *region domain* (a set of region services). To let members interact with nonmembers, the organization could then establish a trust relationship with the provider of a large public virtual world to allow that virtual world's region-domain services to trust the organization's agent-domain services and vice versa. This would let members take their established identities (agents, embodied as avatars) from their private space to a public space.

This approach enables a wide range of usage and deployment patterns. Using an analogy with tourism, it would allow for widespread travel by agents and exchange of virtual goods between a veritable "galaxy" of virtual worlds, given appropriate trust agreements (potentially including "open-door" or "nil-trust" policies). Large-scale virtual worlds with agent and region services federated across multiple providers would also be possible, with central authorities — perhaps as minimal as the DNS — to manage conventions such as virtual geography, namespaces, or virtual currency exchanges. The analogies with the Web are clear — on the Web, there is no single provider, and users can move between sites easily. The key differentiator is that identity is a first-class concept here: in these virtual spaces, the user is always identified (perhaps pseudonymously), unlike on the Web, where stateless, anonymous browsing is the norm.

Discussion of these concepts took place within Second Life's Architecture Working Group, and the design for interoperability became known as OGP.

During the summer of 2008, IBM, Linden Lab, and OpenSim users collaborated to test initial concepts in Linden's OGP interoperability design.

This work included Linden Lab deploying a test agent domain and working on viewer changes to support the new protocol. IBM created an extension module within the OpenSim project to implement the test protocol. These experiments established basic functionality, and the understanding gained from the work fed into design revisions.

Finally, in 2009, Suzy Deffeyes at IBM used this work as the basis for creating a set of extensions to the Second Life open source viewer, and an open IBM-implemented agent domain is now used to continue testing and refining the design approach.

Open Standards

Linden Lab approached the IETF as a possible venue for developing the protocol, with an eye toward releasing it as an Internet standard. The IETF set up a mailing list, MMOX (Massively Multiparticipant Online Games and Applications), in early 2009, and four months of intense discussion ensued with a broad representation of interests. A key conclusion from this effort was that the huge diversity among virtual worlds was too large for a single group to address in the first iteration of a new protocol.

The MMOX group decided to reduce its scope and focus on the set of virtual worlds similar to Second Life — that is, spatially partitioned into regions hosted by simulation servers, with agents representing users, and user-controlled via client applications. A mailing list for "OGPX" was created within the IETF, with the anticipation of holding a Birds-of-a-Feather (BoF) meeting.

At IETF75 in Stockholm, a charter for an OGPX working group was proposed at a BoF session, with the rationale of focusing on defining protocols for a subset of virtual worlds with common characteristics. The BoF attendees reached broad consensus that although the details needed to be refined, this sort of problem was one that the Applications Area within the IETF should address.

After further discussion, it became clear that terminology was a major hurdle to overcome, even if broad agreement existed on the desired user scenarios. For example, when two distinct world providers offer regions services, and users can transparently move between them, is that one virtual world or two interoperating worlds? Participants realized that although "virtual world" is an evocative term, much like "Web site," it doesn't necessarily have a rigorous technical definition when interoperation is widespread. After overcoming such hurdles, the VWRAP WG was formed, placing the focus on virtual worlds, regions, and agents front and center, and dropping the potentially confusing reference to "grids."

VWRAP models the problem space as sets of services that service providers deploy in a variety of potential patterns to produce virtual worlds.

VWRAP Elements

The VWRAP WG's charter is to define a protocol that supports the creation and interaction of virtual environments in which different organizations can deploy the component services. Where possible, the working group should leverage existing standards for protocols and content formats. The group inherited work initiated under the OGP effort, including

- a type system and associated serialization and validation formats amenable to Representational State Transfer (REST)-style services on top of HTTP;
- authentication between the client application and agent services;
- a protocol for placing a user in a region, including movement between regions;
- a protocol for establishing communication between virtual world users;
- content format descriptions for objects, avatars, and regions; and
- deployment patterns for a wide variety of scenarios.

The emerging protocol uses Web standards such as HTTP extensively to take advantage of existing code libraries and deployed infrastructure. The working group is adopting a REST-style[1] architecture wherever possible — client requests for and updates to server resources are made using standard `GET`, `POST`, or `PUT` actions, possibly over secure connections (using Transport Layer Security).

To support a variety of client abilities, VWRAP performs transfer of structured data using a format known as LLSD that supports common atomic types (booleans, numbers, strings, dates, and so on) and structures (arrays and key/value dictionaries). Through HTTP content negotiation, LLSD data can be serialized equivalently in compact binary form, JavaScript Object Notation (JSON), or XML. To address version skew, LLSD provides rigorous rules for type conversion and default values.

VWRAP uses HTTP-based capabilities (see http://en.wikipedia.org/wiki/Capability-based_security), or unforgeable tokens that grant access to an object or service. This is modeled over HTTP by securely returning a unique, temporary URL derived from a Universally Unique Identifier (UUID) when a client requests access to a service. When the client performs an action (`GET`, `POST`, `PUT`, and so on) on that unique, unguessable URL, the server can implicitly trust the action.

VWRAP employs capabilities extensively: after the agent domain authenticates a user, the client receives a seed capability to the agent domain, which in turn provides a list of capabilities for additional services (agent profiles, appearance, inventory, and so on). The agent domain negotiates with a region domain to place the agent in-world, granting the region access to agent services (such as appearance) via capabilities, and obtaining access to region services in return, including a region seed capability that is then passed back to the client. The region seed capability lets the client gain access to further services, such as region content, event queues for state updates, and local chat.

VWRAP models the problem space as sets of services that service providers deploy in a variety of potential patterns to produce virtual worlds. Mechanisms that implement the services are separated from the policies that manage their use. An extension model is assumed, with additional content types such as meshes (rather than Second Life's current constructive solid geometry) being transported over the same mechanisms.

The working group has set itself a timetable for delivering drafts to the Internet Engineering Steering Group (IESG) as proposed standards throughout 2010 and 2011, and into 2012. We can't state enough that the specifications on which the group is working are not mere rubber-stamped versions of the Second Life legacy protocol. The effort required to design and create multiple implementations that meet the many scenarios anticipated by the working group is substantial, and additional help from interested parties would be appreciated. More information is available at http://tools.ietf.org/wg/vwrap/charters, http://trac.tools.ietf.org/wg/vwrap/trac/wiki, and www.ietf.org/mailman/listinfo/ogpx.

Reference

1. R.T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, doctoral dissertation, Univ. of California, Irvine, 2000; www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Joshua Bell is a director of technology integration at Linden Research. After receiving his BSc in computer science from the University of Calgary, his professional career has included software development and project management at Microsoft. Along with Barry Leiba, Bell cochairs the IETF VWRAP Working Group. Contact him at josh@lindenlab.com.

Morgaine Dinova is an independent consultant in Internet systems and scalable engineering and was previously a university lecturer in electrical engineering and computing in the UK. She has a PhD in multiprocessing research and is currently working on massively scalable architectures for virtual worlds. Dinova is a five-year resident of the Second Life virtual world. Contact her either in-world, or at morgaine.dinova@gmail.com.

David Levine is a researcher in the Social Computing group at the IBM T.J. Watson Research Center. His interests include virtual worlds, social computing, and distributed computing architectures. David has a BA in computer science from Brandeis University. His professional career includes 24 years at IBM Research including research and product development. Contact him at dwl@us.ibm.com.