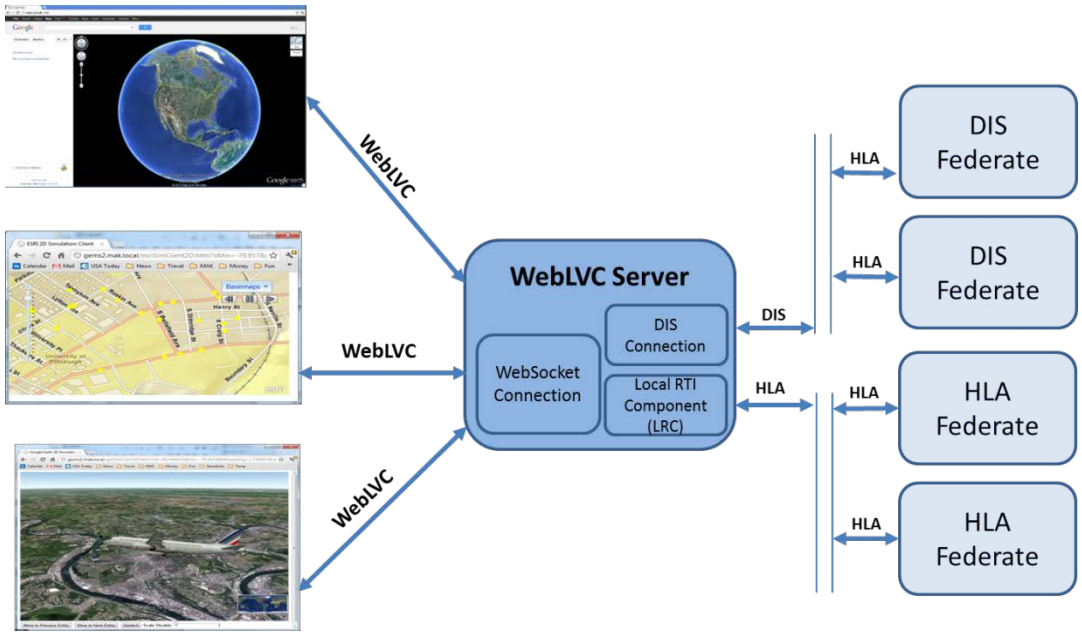
# WebLVC

## Linking Web-based Applications with Traditional M&S Federations




WebLVC Study Group - Spring SIW 2013

# Agenda

- Introduction and Administrative
- What problem is this group trying to solve?
- WebLVC Overview
- Current Status and Demonstration
- Experiences with web technologies in M&S
- Discussion…
- **Do we have consensus on moving forward?**

- Start to think about the details (time permitting)

# Our Goals

- Develop common understanding of the problem
- Evaluate the proposed WebLVC concept, and decide whether to recommend a PDG to pursue a WebLVC SISO Standard
- Help refine and develop the WebLVC idea (if there is consensus that it is the right starting point), to build maturity before PDG kickoff

# Schedule

- Face-to-face meetings:
  - Fall SIW 2012
  - Spring SIW 2013
- Final Report one month after Spring SIW, 2013
  - So that a WebLVC PDG can be kicked off at Fall SIW 2013, **if** this SG recommends such a PDG

# Study Group Officers

- Chair – Len Granowetter, VT MAK
- TAD – Kevin Gupton – ARL University of Texas
- Vice Chair – Don MacGregor – NPS (Moves)
- Secretary - Vacant

# The Problem

- How do we link web-based applications with each other, and with traditional M&S federations in a way that is:
  - High-performance enough for the needs of M&S applications
  - Natural to use in a web environment
  - Flexible enough to support interoperability regardless of protocol used in the target federation

# WebLVC Introduction

# Web Technologies

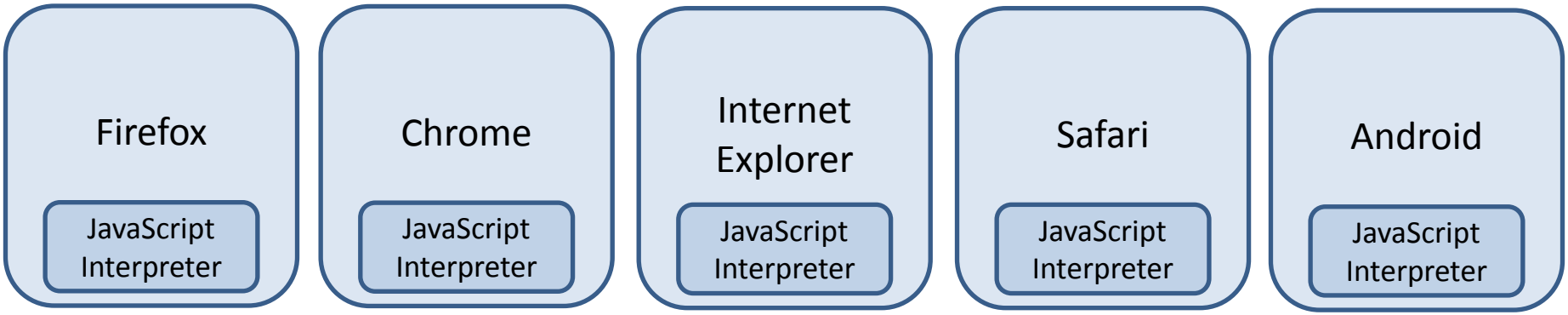
- HTML5
  - Native support for canvas, video, etc.
- WebGL
  - 3D graphics in a browser without plug-ins
- WebSockets
  - Flexible, real-time, bi-directional networking
- Explosion of JavaScript libraries
  - Including game engines
- JSON – JavaScript Object Notation
  - Format for data representation in JavaScript
- Mobile technologies

# JavaScript

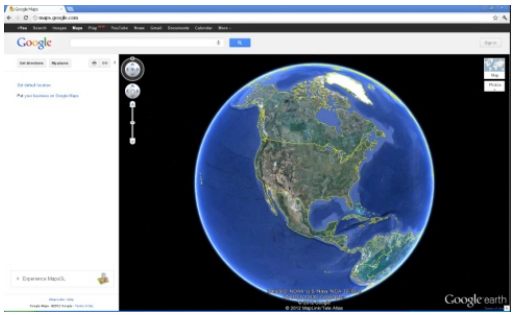

Lots of languages are used on the server side

- PHP, Python, Ruby, C++, Java, JavaScript, etc.

But JavaScript is ***the standard*** language for client-side web application development




When we talk about needing a protocol for “web-based applications”, we mean client-side JavaScript applications running in a browser



HLA, or DIS, etc.

Browser

JavaScript Application

WebLVC

WebLVC

CGF

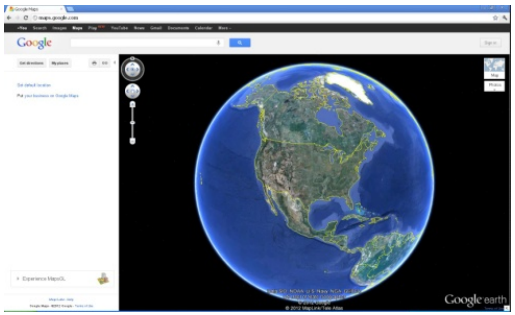
Logger

Simulator

Fast enough for M&S applications

Natural to use in JavaScript

Flexible enough to support various target federation protocols



Participates in federation on behalf of web-based federates – regardless of protocol used in federation

HLA, or DIS, etc.

Browser

JavaScript Application

**WebLVC.js**

**WebLVC**

**WebLVC Server**

WebLVC Connection

HLA or DIS Connection

CGF

Logger

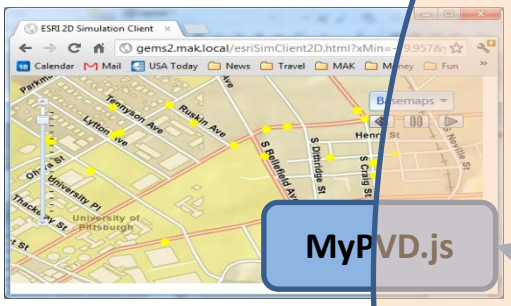
Simulator

Implements client side of protocol in JavaScript

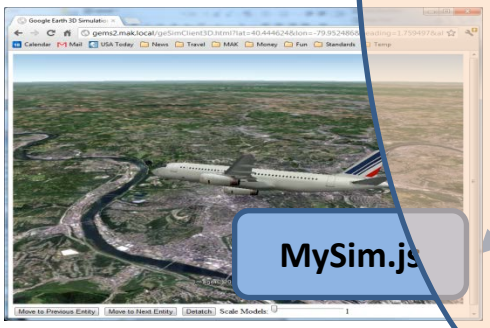
Efficient protocol for communication between web-based clients and server

A WebLVC Server that can bridge between WebLVC applications and both DIS and HLA applications

MyStealth.js



MyPVD.js



MySim.js

WebLVC

WebLVC

WebLVC

**WebLVC Server**

WebLVC Connection

DIS Connection

Local RTI Component (LRC)

DIS

HLA

DIS

DIS

HLA

HLA

DIS Federate

DIS Federate

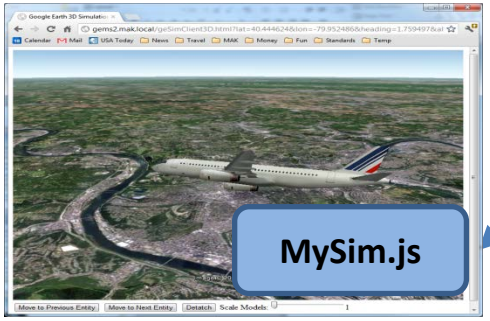
HLA Federate

HLA Federate

What we need to standardize

Special case of no traditional
DIS or HLA federates. The Server is passing data
only among the various WebLVC clients






WebLVC

WebLVC

WebLVC

**WebLVC Server**

WebLVC
Connection

# Designing the WebLVC Protocol

# Transport Mechanism


- A developing W3C Standard
- Essentially a full-fledged TCP Socket, available to JavaScript applications in a browser
  - Persistent
  - Bi-directional (supports push)
  - High-performance
  - Support in all major browser vendors
  - Built for interactive, low-latency web applications

# Message Meta-format: JSON

JSON = JavaScript Object Notation

- A *simple*, text format for encoding structured data

Uses JavaScript syntax

- JSON messages *directly* represent JavaScript objects
- Support built into JavaScript language
- No parsing necessary on client side! No marshaling!
- *Becoming* the leading choice for web applications…

Simple JSON Example:

```
{
  “ContainerType” : “Can”,
  “SizeInOunces” : 12,
  “Brand” : “Coca-Cola”,
  “Flavor” : Cherry
}
```

# WebLVC Protocol: Encodes Semantics of Simulation in the JSON Format

```
{
   MessageKind : 1,
   ObjectName : “F-16 Alpha”,
   ObjectType : “WebLVC:PhysicalEntity”,
   EntityIdentifier : [1,2,1],
   EntityType : [1,2,225,1,3,0,0],
   WorldLocation : [4437182.0232, -395338.0731, 873923.4663],
   VelocityVector : [57.04, 32.77, 89.263],
   Orientation : [-1.65, 2.234, -0.771],
   Marking : “F-16”,
   DamageState : 0
}
```

# Defining the Object Model

- Could auto-generate protocol by parsing DIS standard, and mapping to JSON representation
- Could auto-generate protocol by reading HLA FOM, and mapping to JSON representation
- Could auto-generate protocol by reading ANDEM and mapping to JSON representation

- OR could design WebLVC messages by hand
  - Apply human decision-making to perform “intelligent translation”
  - Make sure that data representations, etc. make sense in the client/server, JavaScript environment

# Current Direction: Support Both

- Built-in “Standard” Object Model (based on semantics of DIS/RPR)
  - Built manually through “intelligent translation”
  - Much as RPR was originally built from DIS
- Flexibility to support arbitrary object models, including simple extensions to existing messages
  - Either manually (more intelligent encoding)
  - OR auto-generation from an HLA FOM (or eventually ANDEM) by following standard rules
- Allows users to trade off ease of extension with quality of the result
  - Maybe manually generate for only a *subset* of DIS/RPR

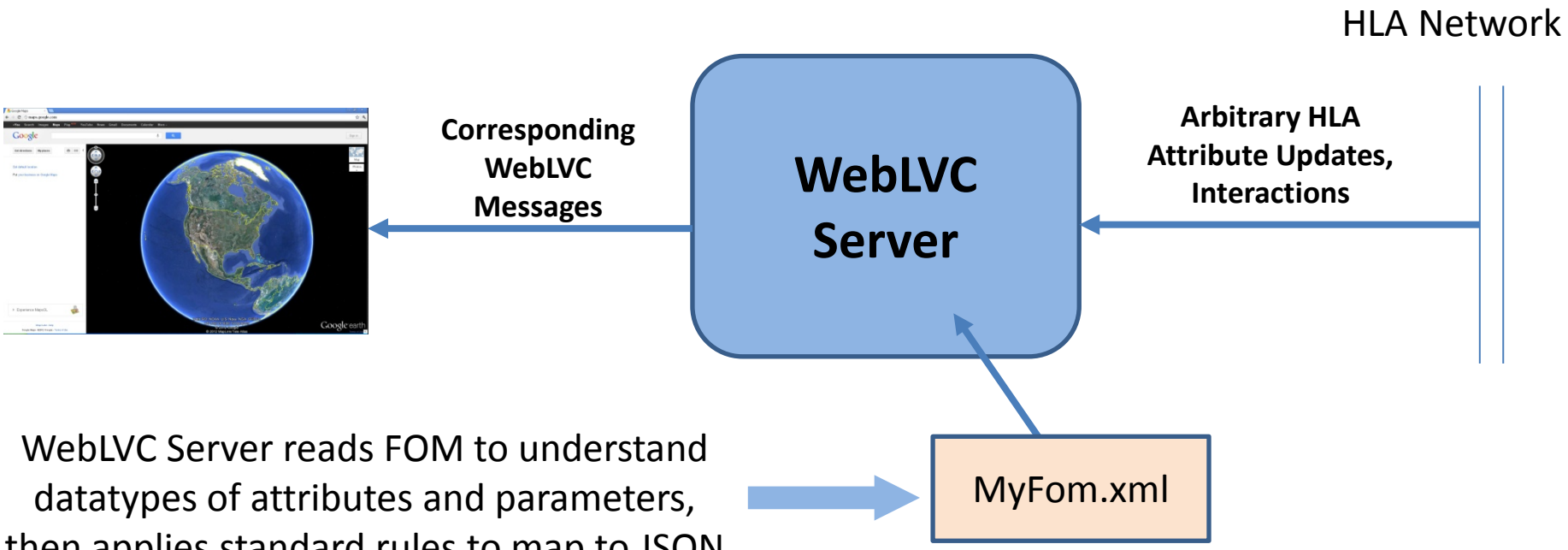
# Current Status

# Current Status - Technical

- Draft 0.2 includes messages for:
  - Physical Entity, Aggregate Entity
  - Weapon Fire, Munition Detonation
- Prototyped, but not yet in published draft:
  - Environmental Entity, Stop/Freeze, Start/Resume, CreateEntity, RemoteEntity, SetData, Acknowledge
- ASTi has volunteered to author WebLVC messages for Transmitter, Receiver, Signal…
- MAK has implemented application-specific extensions (to control VR-Forces, VR-Vantage)
- Performance proven: Over 1,000 entities!

# Current Status - Technical

- Concept of auto-generating WebLVC extensions from HLA FOM has been implemented!
  - Developed standard mapping rules, but haven't yet incorporated them into the Specification draft



# Current Status - Adoption

- US Air Force AFMSTT program has adopted WebLVC to support thin-clients for AWSIM, etc.!
- AGI worked with MAK to integrate WebLVC with “Cesium” WebGL-based rendering SDK
- MAK has integrated WebLVC with GoogleEarth, ArcGIS JavaScript API, OpenLayers, etc.
  - I/ITSEC demo: WebLVC PVD driven by OneSAF.
- MAK hosting free WebLVC Testbed online
- Pelican Mapping demonstrated native iOS PVD using WebLVC at I/ITSEC
- Nearly 70+ members of the Study Group reflector
- 20+ companies experimenting with WebLVC

# Demonstrations

# Experiences with Web Technology in M&S

So…do we have a consensus to recommend forming a PDG to develop the WebLVC Standard?

# Things to Start Thinking About

# Interest Management

- Client-server nature and unicast nature of the web architecture allows new possibilities
  - Server sends separately to each client anyway
  - So might as well ask for exactly what you want!
  - Not just filtering/subscription, but coordinate system, data rate, dead-reckoning strategy, etc.

*"I want updates for aircraft only, in Web Mercator coordinates, within this geographic box only, and I don't want to do client-side dead-reckoning, so please do server side dead-reckoning, and send me updates at 20 Hz."*

# Tons of small but important decisions

- Use strings or numbers for enumerated values?
- First-class concept of Object Type?
- Required versus optional attributes?
- Global object identification
- Default values for unsent attributes
- Trading off efficiency for ease of use

```
"EntityID" : [1,2,3],
        OR
"EntityID" :
{
   “Site” : 1,
   “App” : 2,
   “Entity” : 3
}
```