SOCOM: Bringing a Console Game Online

Presented By:

Seth Luisi
Sr. Producer
Sony Computer Entertainment America

Glen Van Datta
Director of Online Technology
Sony Computer Entertainment America

Dr. Bob Gutmann
Software Engineer
Zipper Interactive, Inc.

Why should we listen to you?

- SOCOM has sold "thru" over 1 million units in North America alone since its release on August 28th, 2002.
- There are over 300,000 active SOCOM players that have played online in the last 30 days.
- 50,000 to 60,000 people play SOCOM everyday.
- On the weekends, players log over 170,000 hours of game time for a very high average of 3 hours per person.

No really, why should we listen to you?

- In the last 7 days, consumers have logged 1,038,134 hours on SOCOM.
- During peak hours (3-7pm PST), there are 11,000 to 12,000 simultaneous people playing SOCOM.
- On an average day, SOCOM has more simultaneous players than all but one of the competing online "PC Games"; including Battlefield 1942, UT2003, Medal of Honor, America's Army or Quake3.
- The only reason why we are here is because we want to see many more successful, online console games.

Design Goals for SOCOM Online

1. **Create 2 separate, full games**
   - **A full online game and a full single player game**
     - **This equals double the work/resources**

2. **Provide the player with a "full" online game experience**
   - **16 players**
     - **Required client/server, which is $$$**
   - **Voice Chat**
   - **Built in community support**
     - **Clans and ladder rankings**

Design Goals for SOCOM Online (continued)

3. Make it easy for console players
   - UI Screens need to be very well designed
     - Quick and easy to navigate
     - Use familiar interface conventions
     - Persistent appearance
     - Psychological grouping

4. Focus the online gameplay
   - Don't try to do everything; try to do one thing really, really well and build upon that
     - You have to ship before the console hardware changes

5. Enable the community to police itself
   - Password protected games
   - Easy to vote out players

Major Production Hurdles

- **Everything was brand new, everything…**
  - **A dependency nightmare**
- **Testing, testing, testing…**
  - **Be sure to perform a "real" Public Beta test**
- **NAT and Firewall Devices**
  - **AKA – A peer to peer nightmare**
- **Post game release server deployment**
  - **You mean we're not done?**
- **Localization**
  - **Japanese and Korean text input… ☹**
    - **We need to display 5,000+ Kanji and 2350 Hangul???**

Advice for Creating an Online Console Game

- Look at online PC games for guidance
- Don’t look at online PC games for guidance
- Enable player to player communication
  - No communication = no community
- Add community features
  - Ladders, clans, stat tracking, etc…
- Bring console production values to online gaming

Advice for Creating an Online Console Game (continued)

- Try to anticipate the worst
  - Because the reality will be much worse than what you can anticipate
    - Cheating
    - Exploits
    - Rude behavior
    - Spamming

SCE-RT Networking SDK - Architecture

Glen Van Datta
Director of Online Technology
Sony Computer Entertainment America

SCE-RT SDK/Infrastructure

- SCE-RT PS2 Online Titles
- SCE-RT Medius SDK
- SCE-RT DME SDK
- SCE-RT Client Simulations
- SCE-RT Server Monitoring tools

SCE-RT PS2 Online Titles

- Currently integrated into 28 1st party titles world wide
- Growing by 2-3 per month
- Good PS2 Title sells millions
  - Expect > 20,000 simultaneous online
- Easily > 200,000 simultaneous World Wide soon
- Over 500,000 Network Adapter units sold in U.S (Projected for 3/2003)
  - Network Adapter sales give accurate online user base
  - XBox Live kits similar information
  - Unlike PC
- Internet Game Soon Required for competitive edge

SCE-RT Medius SDK

- **The SCE-RT Medius SDK is the player matching service for the online experience.**
- **Infrastructure**
  - **Medius Universe Manager**
  - **Medius Lobby Server**
  - **Medius Authentication Server**
  - **Medius Proxy Server**
  - **Medius Database Caching Server**
  - **Medius Universe Information Server**
- **Client Side Coding Requirement**
  - **Medius Client**
  - **Medius Game Communication Library (MGCL)**

SCE-RT Medius SDK (Continued)

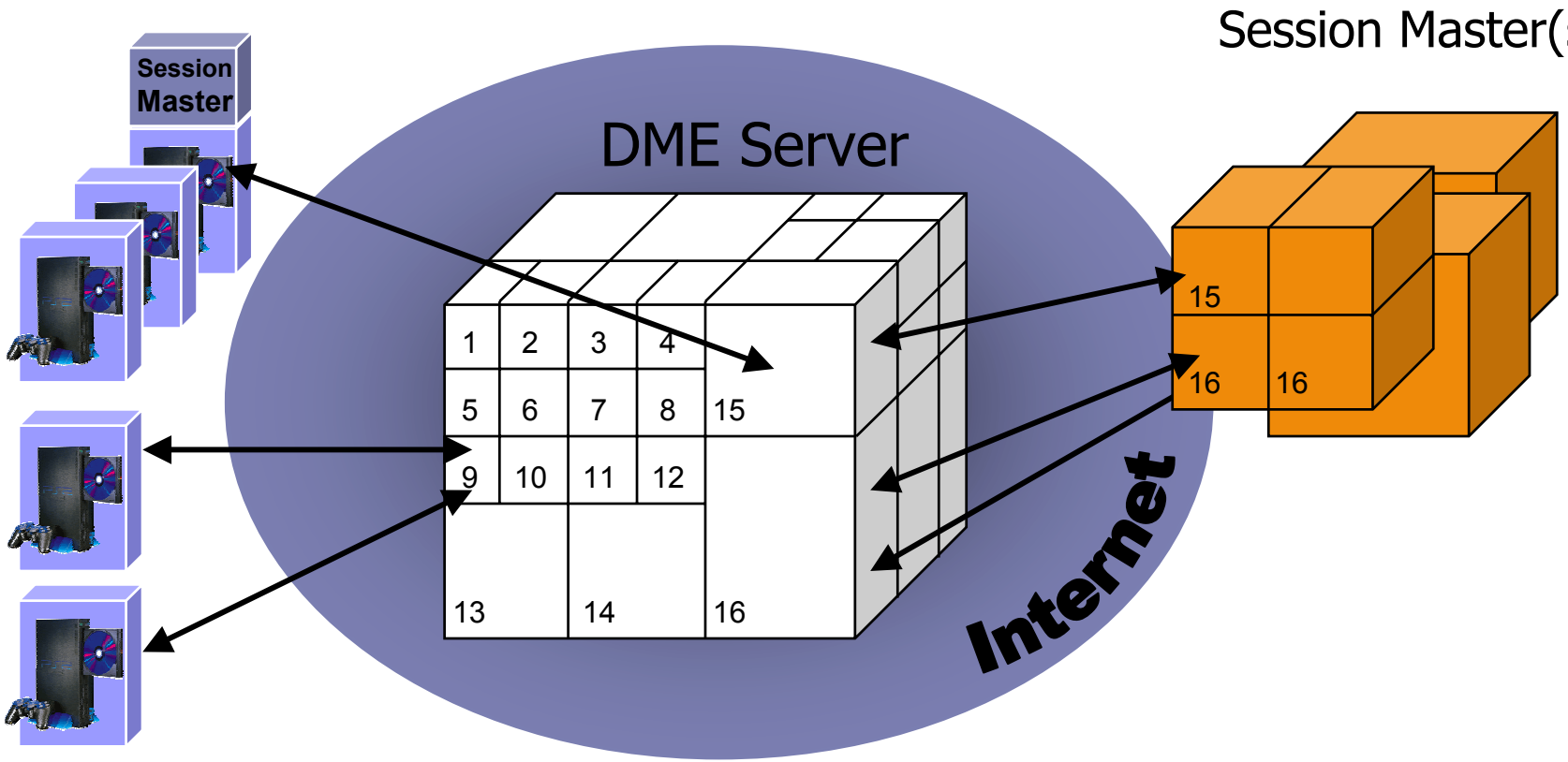
- **Client/Server architecture**
  - **Composes all backend services for online gaming**
  - **User authentication and security engine**
  - **Player matching and game list management**
  - **Create a game, join a game, play the game**
  - **Persistent player statistic tracking**
  - **Lobby and in-game chat**
  - **Buddy Lists and Instant Messaging**
  - **Clan and Ladder support**
  - **Client platform: PlayStation 2, Linux, Windows**
  - **Server platforms: PlayStation 2, Linux, Windows**

SCE-RT DME SDK

- **The RTIME Distributed Memory Engine (DME) SDK is a central part of the game application. The DME SDK is packaged into 3 components, DME client, DME Session Master and DME Server**

- **Native object interface**
- **Broadcast scheduling**
- **Client-side & server-side filtering**
- **Global timebase**
- **Reliable & latency critical communications**
- **Security**
- **Peer to peer client connections**
- **Integrated Server**
- **Client platform: PlayStation 2, Linux, Windows**
- **Server platforms: PlayStation 2, Linux, Windows**

DME Client/Session Master/Server



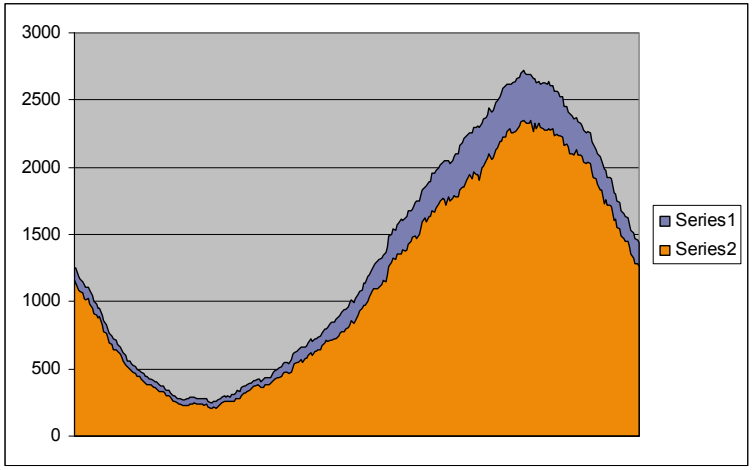
Client Simulations

- Linux Based
- Runs on Production or Development machines
- 1000 clients on production, 500 client on development
- Represents identical Medius capability
- Mimics Game Traffic rates
- Loosely based on User behavior
  - 85-92 percent in game
    - Creating games
    - Joining games
    - Playing games
  - 4% in lobby
  - 5% in lobby/game transition
  - 1-6%% chatting

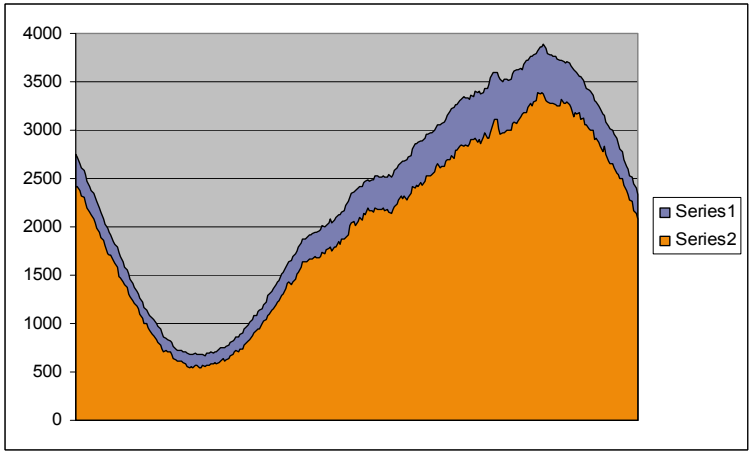
Server Monitoring tools

- Backend Monitoring (Server Executables)
  - System Health Monitoring
  - CPU, Memory, IO, Frame times, Connections
  - User Monitoring
    - Application Counts
    - Chat
    - User IP addresses
    - Hours played
    - Logins
- User data very valuable
  - What features/games users like most
  - Where users are coming from and what times
  - Sine wave usage graphs

Server Monitoring tools (Continued)



- **2 weeks after Launch**



- **10 weeks after Launch**

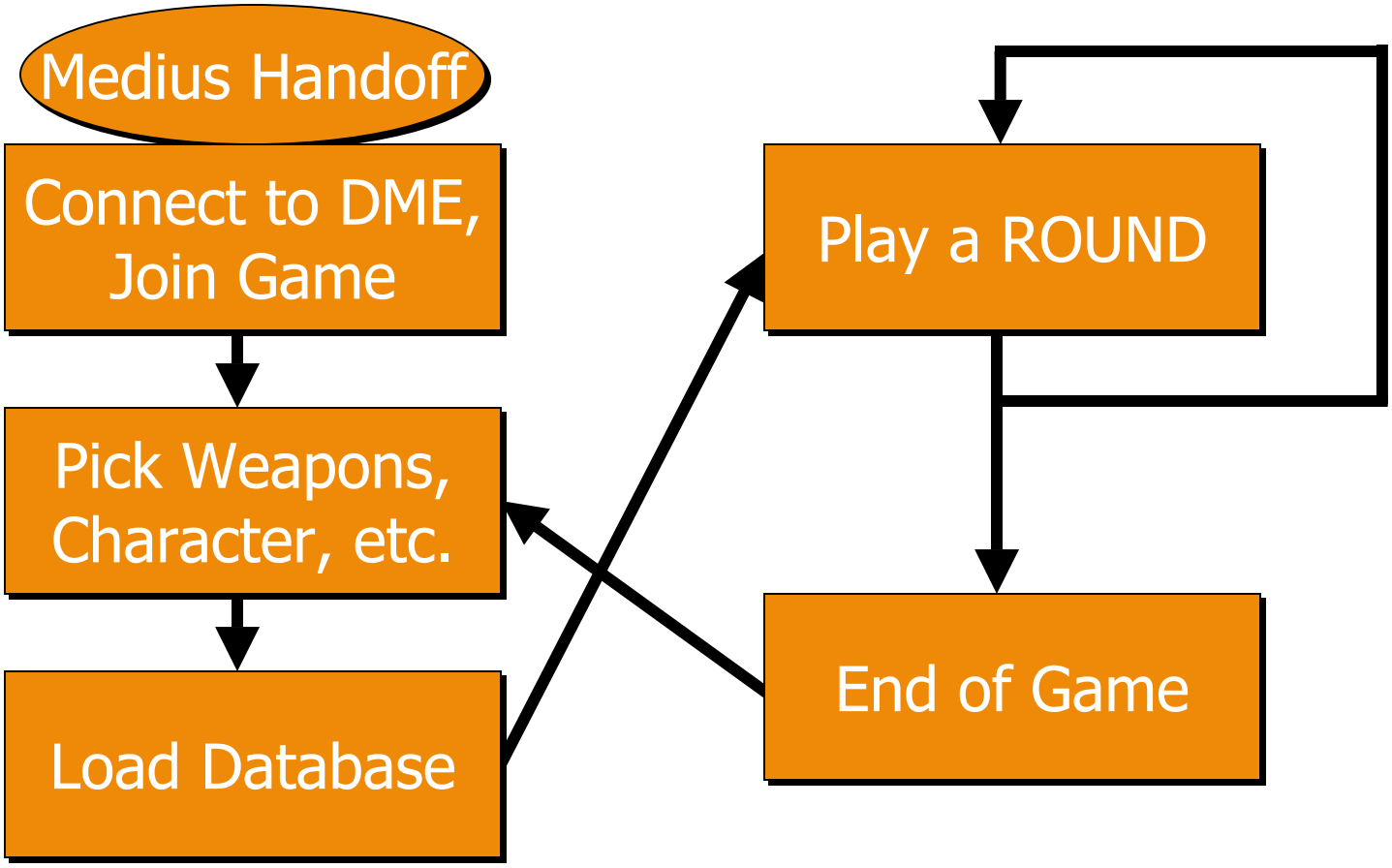
SOCOM In-Game Multiplayer

Bob Gutmann
Software Engineer
Zipper Interactive, Inc.
bobg@zipperint.com

Game Based on SCE-RT DME Architecture

- Client-Server network
- Two modes of data transmission
  - Objects – Sent on a "schedule"
  - Messages – Sent on demand
- DME takes care of network interface
- One player is the "Session Master"
- Voice chat not based on SCE-RT code

SOCOM Online Game Flow



DME Network Objects

- Basically a C-struct
- Each field is registered with the DME
  - Data type (size)
  - Associate with a schedule
- Define callbacks
  - Create – When a new object is born
  - Update – When any field is updated
  - Destroy – When the object is removed

SOCOM Network Objects

- Persona – Created as you join
- Game – Created by the SM
- SEAL – Created at end of DB load
- Grenade – Created on demand
- FootBomb – Created at end of DB load for demolition games

SEAL Object Example

- Contains avatar information (position, velocity, health, and so on)
- Position data (floats) @ 2.0 units, 100 ms
- Velocity data (8 bits) @ 0.0 units, 100 ms
- Horizontal velocity applied to position between updates
- Smooth position changes when update comes in

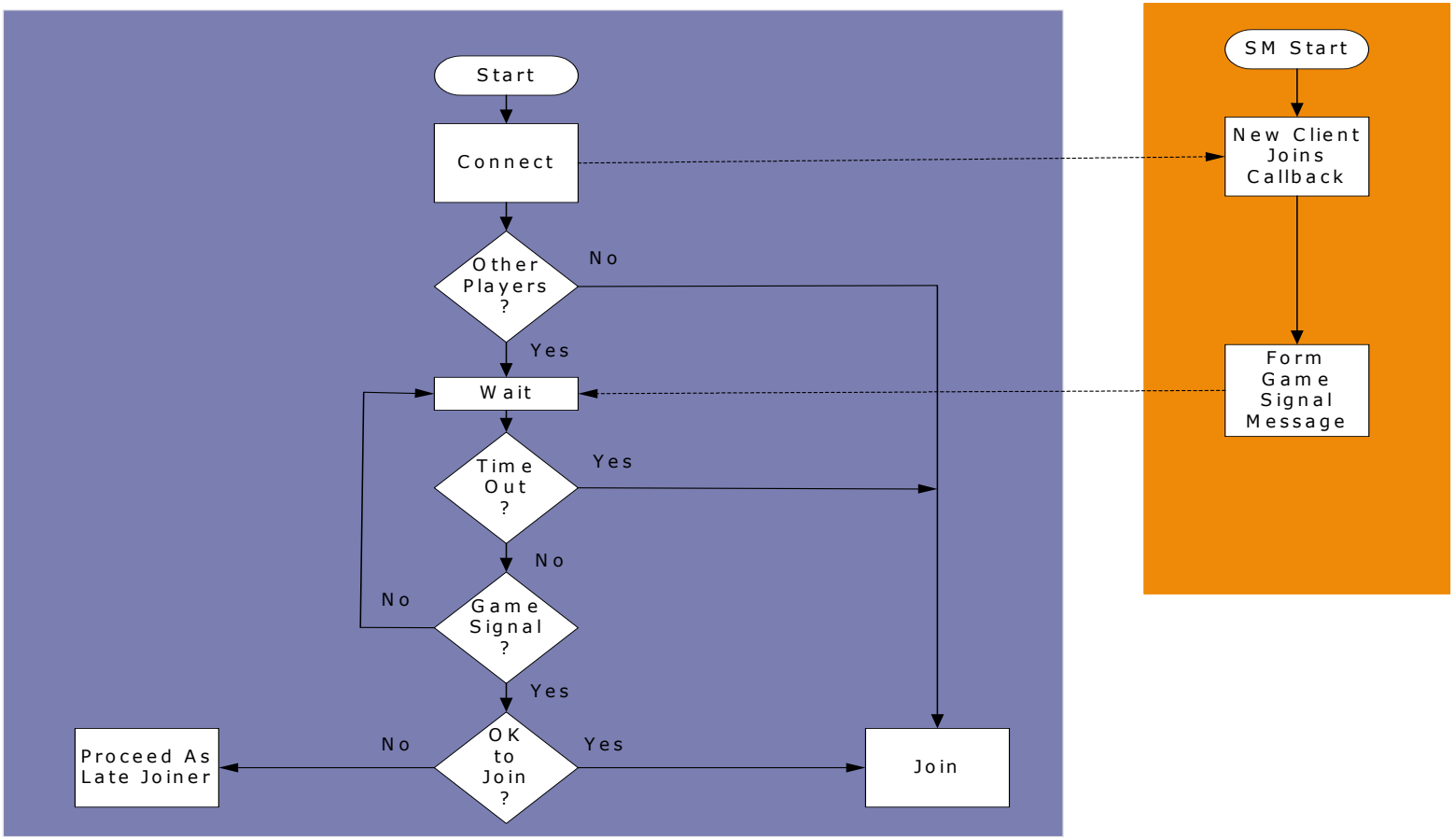
DME Network Messages

- For relatively rare updates
- Also C-struct based
- Register message & callback (handler)
- Register each field
- Can be variable length

Housekeeping Messages

- New client connects
- Client disconnects
- Client joins
- Client leaves – possibly a new SM assigned

SOCOM Join Logic



Voice Chat Overview

- Developed by Secret Level Inc.
- Public domain G.723 CODEC (3KB/sec)
- UDP protocol (point-to-point)
- Walkie-talkie type with one talker per channel at a time (256 channels)
- Channel arbitration required
- UDP/NAT problem

How to Stay Out of MP Trouble

- Get out of single player mindset (player actions)
- Limit animations that move (important) stuff
- Establish interfaces with other code segments
- Understand what causes data transmissions
- Test as many ways as possible (“Squiggle”, etc.)

What Went Wrong

- SCE-RT Issues
  - Developing code as we were
  - Changing the API as we go
  - Documentation
  - Server scalability
- Too Much Logic in UI Scripts
- Test Tools - ?
- Voice chat integration

What Went Right

- SCE-RT Issues
  - Very responsive to reported bugs
  - DME Code finally stabilized
- Reasonably Good I/F With Other Code
- Good I/F Between BEI-Medius and Zipper

Thank you!

Q&A Time