PROJECT DARKSTAR

Coding Project Darkstar Games: Practical Concepts and Techniques

Jeff Kesselman
Chief Instigator, Project Darkstar
Sun Microsystems Laboratories

What is Project Darkstar?

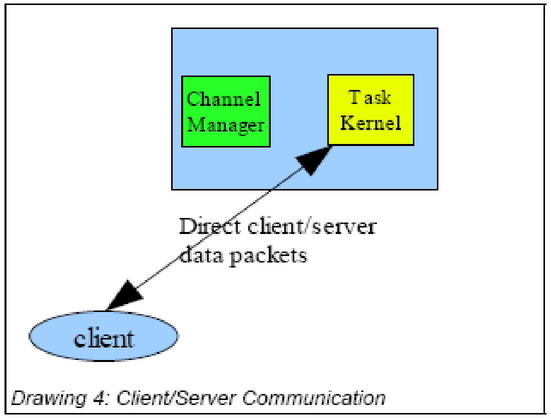
- Hopefully you heard this already
  - Massively Scalable SW Server Technology for Online Games
    - Dynamic Load Balancing Across Many Machines
    - Enterprise Class Performance & Reliability
    - Simple Programming Model
    - Shardless Architecture
    - Open Source

What will be covered

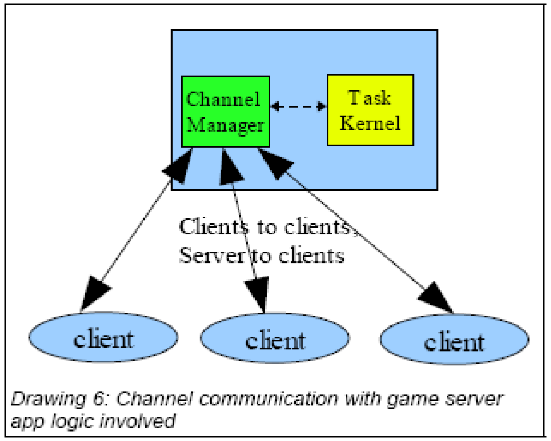
- Project Darkstar coding environment
- Basic Project Darkstar server app patterns
- Examples from DarkMUD

Project Darkstar Application Environment

- Two kinds of communication
  - Direct client/server
  - Channels



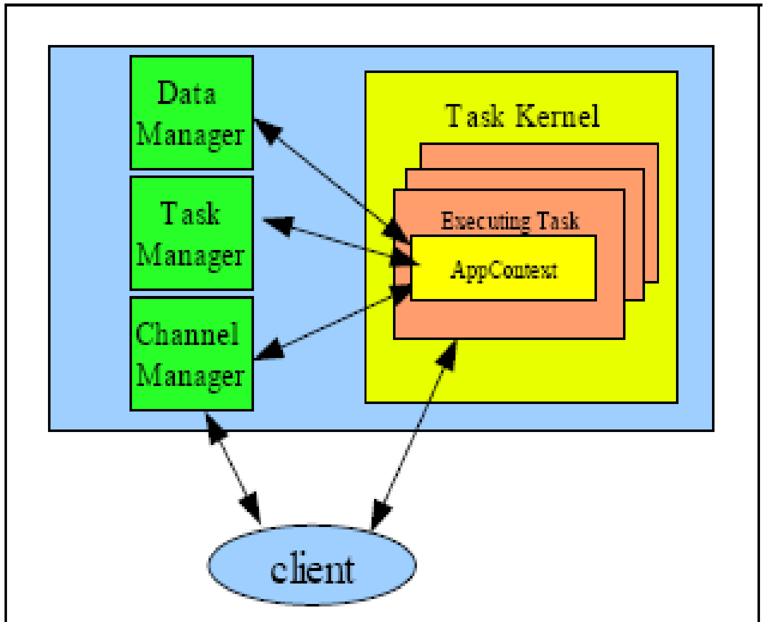
Drawing 4: Client/Server Communication



Drawing 6: Channel communication with game server app logic involved

Project Darkstar Application Environment

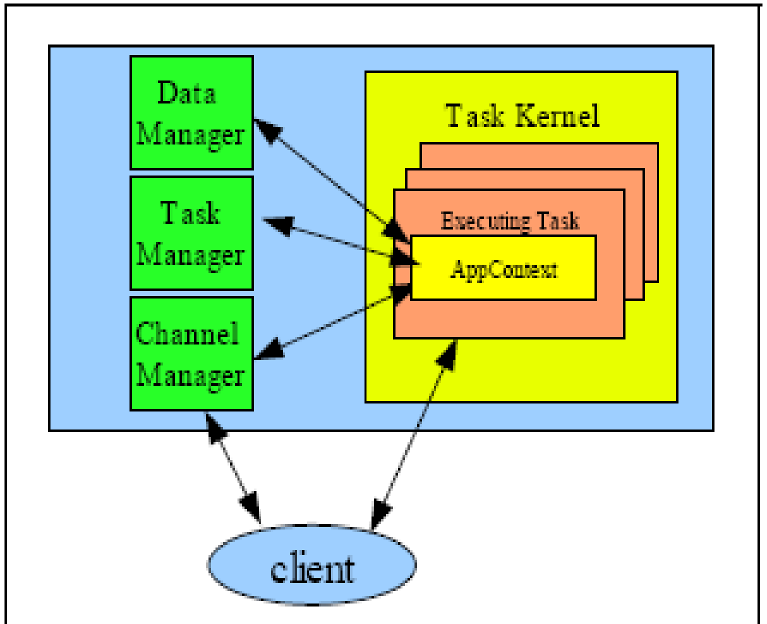
- Task Kernel
  - Executes tasks in response to events
  - Appears mono-threaded to app coder
    - ACID Transactional
    - Persistent
      - Managed Objects
- AppContext
  - Gateway to the system



Drawing 7: The Task Execution Environment

Project Darkstar Application Environment

- Services
  - Called by task code
  - Can generate events
  - Three standard
    - Channel Manager
    - Data Manager
      - ManagedObjects
      - ManagedReferences
    - Task Manager
      - Start new tasks
  - Extensible on a per game basis



Drawing 7: The Task Execution Environment

Basic Project Darkstar App Patterns: AppListener

- All starts with an AppListener
  - AppListener is an interface that defines the most basic system events
    - initialize()
    - loggedIn()
  - AppListener Managed Object is created for you.
    - initialize() called when created
  - AppListener has two duties
    - initialize application
    - handle users who just logged in

Basic Project Darkstar App Patterns: AppListener

- AppListener initialize() from DarkMUD

```
/**
         * This is where we initialize the SGS application.
           *  This gets run only once, the
           *  first time the application is brought up, or the next start-up
           * after the object store gets cleared.
           */
          public void initialize(Properties arg0) {
              // The DataManager is used to store persistent objects
              DataManager dmgr = AppContext.getDataManager();

              // This is the room all newly created players will be placed in.
              GenericRoom startingRoom = new GenericRoom("starting room");
              startingRoom.setDescription("a big empty room.");

              // This saves a ManagedReference to the starting room
              // so that we can put new players in it when they log in
              startRoomRef = dmgr.createReference(startingRoom);

             etc...
```

Basic Project Darkstar App Patterns: AppListener

- AppListener initialize() from DarkMUD

```
public ClientSessionListener loggedIn(ClientSession arg0) {
                // This tries to get a pre-existing MudUser object
            // for the user who just logged in, using a naming
            // convention base on his or her login name

       String name = USERPREFIX + arg0.getName()
         MudUser user = null;
         String welcome = null;
         try {
              user = AppContext.getDataManager().getBinding(
                       name, MudUser.class);
               if (user.isLoggedIn()) {//but it says were already logged in
                    return null; // reject the login
                }

          welcome = "Welcome back to the JavaOne MUD!\n";

       } catch (NameNotBoundException e)
```

What happens if the NameNotBoundException is thrown?

Basic Project Darkstar App Patterns: AppListener

- If name was not bound....

```
    } catch (NameNotBoundException e) {
                try {
                    // create a new MudUser ManagedObject
                    user = new MudUser(name);
                    AppContext.getDataManager().setBinding(name, user);

                    // get back the starting room from the ManagedReference
                    // we saved off.
                    // We use getForUpdate because we know we are about
                    //to change its state by adding another user to its
                    // inventory
                    GenericRoom startingRoom =
                        startRoomRef.getForUpdate(GenericRoom.class);
                    startingRoom.addToInventory(user);
                    welcome = "Welcome to the JavaOne MUD!\n";
                    users.add(AppContext.getDataManager().createReference(
                              user));
                } catch (Exception e2) {
                    e2.printStackTrace();
                    return null;
                }
             }
            // make user the ClientSessionListener for this user session
            return user;
        }
```

Basic Project Darkstar App Patterns: ClientSessionListener

- Returned to system from loggedIn callback.
  - In above, was the user ManagedObject
    - MudUser implements ClientSessionListener
  - Common and handy pattern
- ClientSessionListener handles two events
  - ReceivedMessage()
    - End point for direct client to server communication
  - disconnected()
    - Notification of end of client session

Basic Project Darkstar App Patterns: ClientSessionListener

- ClientSessionListener receivedMessage() from DarkMUD

```
    public void receivedMessage(byte[] arg0) {
            String command = new String(arg0).toLowerCase().trim();
            StringBuffer output = new StringBuffer();

            parse(command, output);

            if (output.length() == 0){
              sendToUser("Nothing happens.\n");
            } else {
              sendToUser(output.toString());
            }
        }
```

- MUDs use string commands
  - Converts back to String and passes to parser
  - More commonly binary packet protocol handler

PROJECT DARKSTAR

Basic Project Darkstar App Patterns: ClientSessionListener

- ClientSessionListener disconnected() from DarkMUD

  -
    ```
    public void disconnected(boolean arg0) {
            session = null; // session is no longer valid
            setLoggedIn(false);
     }
    ```
- Session Cleanup
  - ClientSession object is no longer valid so we null the reference to it
    - Created and managed by system
  - Set a boolean so others know we are logged out
  - In more complex app, might delete ManagedObjects no longer needed

PROJECT DARKSTAR

Coding Project Darkstar Games: Practical Concepts and Techniques

Jeff Kesselman

jeffrey.kesselman@sun.com

http://www.projectdarkstar.com

Austin Game Conference - 2007