Intelligent Thought Engine

William M. Grim

November 17, 2005

1. Introduction

The primary purpose of the Intelligent Thought Engine project is to develop a networked, rule-based data modeling engine that supports logical interaction from thin clients. To support this engine, ITE will develop new, open network protocols and libraries on top of existing network protocols.

2. Background

There is no current system.

3. Proposed Work

3.1. Functional Requirements

3.1.1. Server Configuration

The server will read an XML configuration file containing information about the server itself[1].

1. Bind IP (IP_ADDR_ANY)
2. Bind port (1300)
3. Backlog (10)
4. Administrator email (root)
5. Error limit (50)
6. Error limit clear interval (in seconds: 86400)
7. Use PAM (1)

3.1.2. Plugin Configuration

The server will read an XML configuration file containing information about plugins reachable via local domain sockets.

1. Location

3.1.3. Session Configuration

The system will read an XML configuration file containing default parameters for all data model sessions.

1. Global parameters:
    1. Human time limit per action (in seconds: 0)
    2. Non-human time limit per action (in seconds: 120)

1 Items in parenthesis are default values.

2. As plugins are added, their XML definitions will be placed in separate description files.

3.1.4. Authentication

On the server side, the server will make use of two ways to authenticate clients: flat files or PAM[2]. The use of flat files or PAM is dictated by the "Use PAM" flag in the ITE game server's configuration file. All communication must take place over a secured line.

The quickest way to setup ITE authentication will be flat files, which must contain USER:PASS pairs. A username can be any length and contain any character other than ':', which delimits the username from the password. A password can be any length and can contain any character.

If PAM is being used, the recommended way to setup PAM is to have one of the PAM modules check that the user is legitimate by checking an access[3] flag for that user in something like an OpenLDAP user directory or MySQL/PostgreSQL user database. See also: Plugins (page 2).

3.1.5. Dynamic Joining and Parting

ITE must support the ability for clients to join and part data model sessions at any time. However, it will be up to the rules and actual implementation of plugins as to whether a client may actually join or part a given data model session once a session has begun.

3.1.6. Observation

Clients should have the ability to join any data model session as an observer to witness the action that is taking place with a given data model. This includes the ability to join sessions before they have begun and the ability to join sessions as they are in-progress. As an observer, the observer is denied any communication contact with the interactive clients of the session, including by private message; however, observers may communicate with each other. Also, once observing a session, that client loses the ability to join in-progress sessions as a normal interactive client. Data model plugins can, however, choose to bypass this restrictive rule of denying observers interactive access to the session by passing the server an "Allow Interactive Observers" flag.

3.1.7. Plugins

Plugins should be able to dynamically register and unregister themselves with the ITE game server at any time. Also, once recognized, plugins must provide their name and category to ITE for identification and grouping.

3.1.8. Access Control

Plugins fall into various categories defined by the plugins. Before a client can use a given plugin, the client must first have access rights to that plugin's category.

For example, if five plugins fall into the "games" category, before a client may use plugins of that category, they must have access rights to the "games" group.

3.1.9. Distributed Computation

Servers and clients should be fully distributed, allowing clients to work cooperatively or competitively to come to conclusions on data models generated by plugins. Servers should share various statistics with each other:

---

2 Pluggable authentication modules, which allow for advanced authentication protocols to be setup without modifying user applications.

3 Client's credentials have access to a certain category of plugin(s) and their associated data models.

1. Clients currently connected.
2. Currently running/waiting-to-start sessions.
3. Statistics for each client's identify, such as win/loss/draw/etc.

3.2. Non-Functional Requirements

3.2.1. Diversity

ITE will be suitable for a massive number of distributed data model interactions, including games, mapping systems, and architectural design plans, amongst others.

3.2.2. Documentation

Documentation on all aspects of ITE will be provided to end-users and developers. The documents should always outline all major goals and system improvements to ITE and be continuously available in one form or another. This will help foster improvements to the system, encourage good software design, and give users help manuals full of useful information.

3.2.3. Interface

ITE will not have much of a server interface. In fact, the extent of the server's interface will be XML configuration files, system logging, command-line parameters, and signal handling.

The ITE server will support both command-line parameters for startup control and SIGHUP[4] signal processing for configuration refresh. ITE may also support an interactive shell or network interface that administrators can use to modify the server while it is running, but that is optional.

3.2.4. Hardware

The projected hardware requirements for the server:

1. Any POSIX-compliant OS with OpenSSL and libxml2.
2. Pentium III 500 MHz.
3. 128 MB RAM.
4. Network connection.
5. 1 GB disk space.

3.2.5. Security

Security is of great to concern to ITE, because clients will be connecting and communicating over insecure networks. Clients should not have read/write access to anything other than their own account information, and they should only have read access to other clients' statistics. Please see the Authentication (page 2) and Access Control (page 2)for more details.

3.2.6. Concurrency

ITE must maintain several connections at a time. Therefore, ITE must be able to handle new connection

4 Terminal line hangup signal.

requests quickly and have minimum latency when communicating with its connected hosts.

4. Use Case: Start Server

4.1. Participating Users

1. Initiated by administrator.
2. Communicates with server.

4.2. Flow of Events

1. Administrator starts server.
2. Server scans its XML configuration files for startup information and registered plugins.
3. Server becomes a background daemon, opens its network to remote users and begins servicing requests.

4.3. Entry Conditions

1. Administrator starts server.

4.4. Exit Conditions

1. Server becomes a background daemon.
2. Server produces an error, resulting in a system log with error diagnostics.

5. Use Case: New Connection

5.1. Participating Users

1. Initiated by connecting client.
2. Communicates with server.

5.2. Flow of Events

1. Client connects to the server's listening port.
2. Client is moved somewhere to free up the listening socket.
3. Client's credentials (username/password) are checked:
4. If server is configured to use PAM, server passes Denny's credentials to PAM.
5. Otherwise, server checks a flat-file database of user:pass pairs to see if Denny's credentials match.
6. If client successfully authenticates, it is granted access to the system; otherwise, server sends client a reason stating why authentication failed and terminates connection.
7. Server updates internal list of currently connected clients and their connection information (IP, nickname, etc.).

5.3. Entry Conditions

1. Client connects to the server's listening port.

5.4. Exit Conditions

1. Client successfully connects and is authenticated.
2. Client produces an error, resulting in immediate connection termination.
3. Server produces an error, resulting in a system log with error diagnostics.
4. Credentials don't succeed, a log entry is generated and client is notified before being disconnected.

6. Use Case: Connection Termination

6.1. Participating Users

1. Initiated by connected client.
2. Communicates with server.

6.2. Flow of Events

1. At any time, client requests that it would like to terminate its connection with the server.
2. Server destroys client's connection to any active sessions, and the plugins take care of any cleanup necessary. See also: (page).
3. Server closes connection to client and updates its internal list of currently connected clients. See also: (page).

6.3. Entry Conditions

1. Client sends termination message.

6.4. Exit Conditions

1. Client disconnects according to flow of events.
2. Client abruptly terminates connection, resulting in an error log being generated and stored in the system logs.

7. Use Case: Session State Change (XXX: Needs Updated for "New/Join/Quit" Changes)

7.1. Participating Users

Initiated by server.

Communicates with database back-end.

7.2. Flow of Events

1. Client interacts with an ITE data model using steps outlined in any of the use cases in the "Intelligent Thought Entertainment" requirements analysis document.
2. As the session ends, for various clients, their statistics in the ITE database are updated according to data model rules and plugin category.
3. Each plugin category mandates that a few minimum statistics are recorded that are pertinent to that given category.
4. Game category might record these statistics, for example.
    1. Wins
    2. Losses
    3. Draws
    4. Defaults (Disconnecting/Losing Connection)
    5. Player Type (i.e. Color, number, etc.)
    6. Games Played

7.2.1. Entry Conditions

1. Client sends server a message that indicates a change of session state should take place.

7.2.2. Exit Conditions

1. Server updates database correctly.
2. Database produces an error. The error is logged and the update SQL is appended to a file for later processing by administrators. If the error limit is reached, email the administrators to inform them that the error limit has been reached before the error limit clear interval was reached.

8. Use Case: Connection Termination

8.1. Participating Users

1. Initiated by connected client.
2. Communicates with server.

8.1.1. Flow of Events

1. At any time, client requests that it would like to terminate its connection with the server.
2. Server destroys client's connection to any active sessions, and the plugins take care of any cleanup necessary. See also: (page).
3. Server closes connection to client and updates its internal list of currently connected clients. See also: (page).

8.1.2. Entry Conditions

1. Client sends termination message.

8.1.3. Exit Conditions

1. Client disconnects according to flow of events.
2. Client abruptly terminates connection, resulting in an error log being generated and stored in the system logs.