# Realtime Strategy Game

## Table of contents

## 1. Overview

In a typical application of this type,

- Users would form groups (lattices) lasting for the life of the game, typically no more than a few hours.
- Several user machines would act as servers whose principal function would be to verify that events had occurred and possibly to agree on scores.
- Terrain maps and other such static information might originate with the servers but would be distributed by player nodes, using hypercube links, in order to fairly distribute the communications load.
- Messages relating to events would be distributed partially using the hypercube, to evenly share the load, and partially using direct node-to-node messages. Typically a kill would be reported simultaneously to at least one server and to the victim.
- The game would need to support the formation of ad-hoc chat groups to allow player alliances.
- Users will probably want to store information about people that they have played with previously, so that they can contact them to arrange future games.

Games of this type require very fast communications at all levels. Because messages would pass through other players' machines, encryption would be very desirable. To avoid intolerable delays in verifying results - this must occur while action is in progress - server authentication protocols would have to be fast and simple.

## 2. Non-standard Elements

| | |
|---|---|
| hi-res window | Although the commandline interface might be used for starting the game, the normal user interface would be a window using high-resolution and possibly 3D graphics. |
| joystick or similar | While mouse support might be acceptable in the short run, it is likely that in the long run joysticks and other similar devices would have to be supported. |
| maps, other static information | Users would expect to be able to share relatively large amounts of static information such as terrain maps and graphics relating to vehicles, events (eg explosions), weapons, and stock characters. These might be downloaded prior to a game, using hypercube "broadcast" to distribute the load fairly. Storage would normally be persistent. |
| buddy list, avatars | The requirements here are the same as for the chat application. |

| | |
|---|---|
| game-specific messages | Player movement and events would be broadcast to other player nodes and to server nodes, which would be responsible for verifying events. Because of the importance of speed, these messages would be short. Probably udp would be used for many. |
| game state | The stream of messages would be passed to a software module responsible for maintaining the game state. |

## 3. Implementation Sketch

The software would consist of two distinct blocks: the game proper and a simplified chat system. The chat system would be used for communications between allies; it might be that such alliances would be required to last for the duration of the game.

This chat subsystem would be usable as the basis for the Chat model application. The difference between the two would be that Chat might not need authentication and encryption but it would definitely need an IRC proxy and/or universal chat interface, which the Strategy Game would not need.

It is unlikely that messages would be routed, because passing through a **broker** would significantly slow the game. On the other hand, because messages would be on occasion passed through opponents' platforms, reasonably robust authentication and encryption would be essential.

Most of the complexity of the game implementation would lie in the front end, in the game window, and then in the internal logic used to synchronize movements and events between nodes.

## 4. Interoperability

We would also need a module running on **server nodes** to authenticate game events. These server nodes would operate on user platforms, but be distinct from the user nodes. Because of the likelihood of attempts to cheat, the servers would need a protocol involving voting among them. The usual Byzantine protocols require unanimity with three servers or less, and more than two-thirds agreement with four or more servers.