Manycraft: Massively Distributed Minecraft

Mathieu Valero, Raluca Diaconu, Joaquín Keller

Manycraft: Massively Distributed Minecraft

Mathieu Valero
Manycraft
Paris, France
m47h13U@manycraft.net

Raluca Diaconu
Laboratoire d'Informatique de Paris 6
Université P&M Curie, Paris, France
raluca.diaconu@lip6.fr

Joaquín Keller
Orange Labs
Issy-les-Moulineaux, France
joaquin.keller@orange.com

***Abstract*—Minecraft is a popular game with more than 20 million paying users and many more playing the free version. However, in today's multiplayer mode, only a few thousand users can play together. For a common setting, when players cannot modify the map, we have designed a distributed system to scale the number of users. The Manycraft software runs on users' machines which communicate through Kiwano infrastructure. This paper describes its architecture and implementation.**

Manycraft node is available for download at http://manycraft.net, so we expect players to participate and join the demonstration at the given time slot.

I. Introduction

Minecraft is a game where users build their own world. Players do not have specific goals to accomplish; they choose how to play the game. Minecraft has forgone realism for creativity and simplicity. This is probably the recipe of its huge popularity: estimations attribute Minecraft more than 20 million paying users and many more using the free editions.

In multiplayer mode, users connect to a server in order to interact and communicate with each other within the same world. However, servers are strongly limited in the maximum number of simultaneously connected users: the most powerful ones reach only a few thousands [1].

In creative mode, where the purpose is to build things, players add and remove blocks. But currently, in order to present their creations, players either host a server on their own computer and allow others to connect, or record a video and share it on the internet. The first solution is cumbersome and costly: the user needs to maintain a server for a few friends that might visit, while the second one does not offer those interested the possibility to visit the world.

Moreover many popular gaming experiences such as racing, shooting, role playing, emphasize interactions between players and the map is not meant to be modifiable.

Using Kiwano [2], an infrastructure for scaling virtual worlds, we have designed and implemented *Manycraft*, a solution to allow an unlimited number of users to simultaneously play in a map without modifying it.

II. Minecraft

A Minecraft *map* is made of simple cubic *blocks*. All blocks have the same size and and only vary in their types (wood, coal ore, stone,...). Each player is represented by an *entity*. Around its position, each has a square-shaped *awareness area* providing the visible map and *neighbors* to be transmitted. There are also game entities called *mobs* affected by the environment the same way as players.

To play together in multiplayer mode all clients connect to the same server. The server hosts the world: it notifies about entities and delivers the map to the connected players.

After connection, the player is spawned at a position chosen by the server. During the game, the Minecraft client sends to the server player actions, namely position updates, block destruction, etc., and is notified back about events occurring in the awareness area.

Minecraft protocol [3] messages can be divided into the following categories: map, entity and control.

Map Map related messages –list of blocks with descriptions– are generated when a player arrives at a new position or when blocks in the awareness area are modified. The server sends only the differences with respect to the last configuration.

Entities When a player or a mob moves, the new position is send by the server to every player that should be aware of it. Entity related messages also bear chat, avatar animation or other entity actions.

Control Periodically, the server sends keep alive messages to clients. There are also cyclic events, such as day-night alternation, that generate messages at regular time intervals.

III. Manycraft

Manycraft is meant to allow an unlimited number of players to interact in a read-only Minecraft map. To join the system the player runs a Manycraft node on her own computer.

The Manycraft node is a Minecraft server "modded" in order to divert the entity messages to Kiwano which, in return, notifies about the events occurring in the player's neighborhood. The local Minecraft server has the map preloaded and its main duty is to deliver it to a unique player. The load generated by the avatars has been shifted entirely to Kiwano.

A. Kiwano

Kiwano is a scalable distributed infrastructure for virtual worlds, designed to support an unlimited number of moving objects updating their position at arbitrary high frequencies. In Kiwano the set of moving entities is distributed onto many servers, each taking care of a group of entities based on their geographical proximity.

To free the application layer from the distributed internals, Kiwano provides an API [4] to developers. When connecting, a client is assigned a proxy, the entry point to Kiwano for the entire session.

Kiwano API receives three types of commands from clients: *update* entity state, *message to all* and *private message*. On the other side, it sends four types of notifications: the full

list of *neighbors*, neighborhood *updates*, *message* from another client, and *remove* neighboring entities. All Kiwano messages are encoded in json, common now in most programming languages. Messages have an *appdata* field to carry data specific to the virtual world to scale.

```
{  method:"update",
   urlid:"http://manycraft.net/u6nMnT",
   iid:"john.smith",
   lng:52.5, lat:15.3, angle:40, alt:40,
   appdata:{ manycraft: {
                sneaking:false,
                pitch:0.147 }} }
```

Fig. 1: Update command

```
{  method:"updates",
   urlid:"http://manycraft.net/u6nMnT",
   iid:"john.smith",
   new:[{ urlid:"http://manycraft.net/w28QPu",
          iid:"alice.smith",
          lng:52.5, lat:15.3, angle:40, alt:40,
          appdata:{ manycraft: {
                   sneaking:true,
                   pitch:3.2 }} } ]
   old: [ ("http://manycraft.net/w28QPu",
       "charlie.doe"), ] }
```

Fig. 2: Neighborhood update notification

```
{  method:"msg",
   urlid:"http://manycraft.net/u6nMnT",
   iid:"john.smith",
   recipients:
       [("http://manycraft.net/8Qw2RJ",
       "alice.smith"), ],
   msg:"",
   appdata:{
       manycraft:{animation:"swing_arm"} } }
```

Fig. 3: Message notification

B. Manycraft Node

To join a Manycraft world, the player downloads the node preloaded with a Minecraft map. The node is composed of a regular Minecraft server and a proxy. Once the node is running on her machine, she connects through the proxy her regular Minecraft client to the local Minecraft server embedded in the node. The local server receives the packets from the client and serves the blocks of the map as the player moves. The proxy also sends, after translation, player movements and actions to Kiwano.

Kiwano notifies back about the movements and actions occurring at nodes of the neighboring players. These notifications are translated and processed by the internal Minecraft server which generates the packets for the client. Remote players movements and actions are then rendered by the client.

As players are in the same map, see each other and interact: they are together in the world.

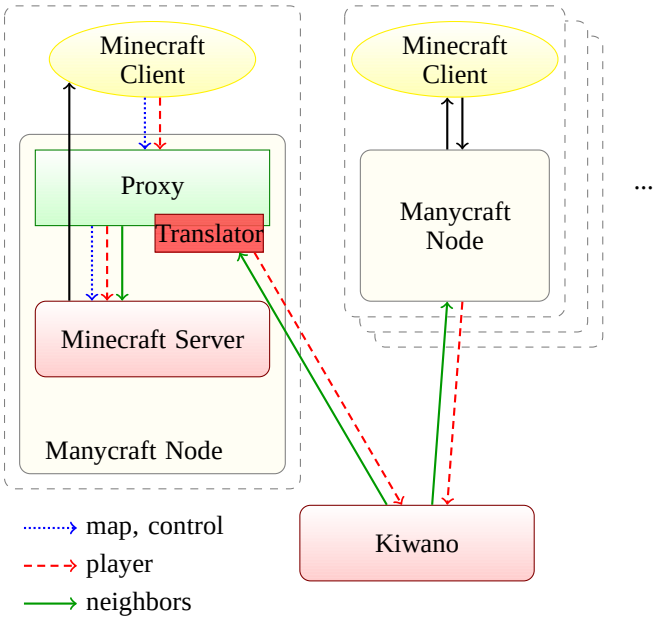

Fig. 4: Manycraft Architecture

C. Bridging Minecraft over Kiwano

The essential task of a Manycraft Node is to route the relevant Minecraft traffic from and to Kiwano. The proxy inspects the messages issued by the client. All the messages, including player movements, are forwarded without modification to the Minecraft server. On position updates the Minecraft server sends the surrounding blocks. Messages concerning player actions are translated and sent to Kiwano which sends back notifications about the neighbors. These messages are translated into Minecraft messages so that the neighbors entities can be placed on the map as mobs. This way, the server attached to a player has all the information corresponding to the awareness area.

Minecraft actions to Kiwano commands: Each entity related message issued by the client is intercepted by the proxy and translated to a Kiwano command. The Minecraft account name is translated into a Kiwano object id, see Section III-A. Player's state modification generates a Kiwano update command, player's coordinates and yaw are respectively mapped on lat, alt, lng and angle fields. Minecraft specific description of the avatar is sent in the appdata field with appid 'manycraft'. Chat and player actions are translated into Kiwano message commands.

Kiwano notifications handling: Kiwano notifications inform a node about actions and state changes of remote players in the neighborhood. Remote players are represented in the Minecraft server as mobs. Kiwano notifications trigger movements and actions of these mobs. Upon notification of entity arrivals/departures in player's awareness area the mobs are created/removed accordingly in the Minecraft server.

D. Implementation

CraftBukkit [5] is a Minecraft server modded to accept plugins. Minecraft clients connect and communicate with a CraftBukkit server in the same way as with a regular Minecraft server; CraftBukkit is thus transparent to the clients.

Plugins can listen to events such as players joining/leaving, avatar animations, map state changes etc. and add custom handlers. In particular, this enables intercepting messages between client and server and altering the server behavior.

The Manycraft node is a CraftBukkit server with a specific plugin. In order to intercept messages to the server, the Manycraft plugin listens to events related to players and generates the corresponding Kiwano commands, as described in Table I. For example, when a player joins the server, the plugin opens a Kiwano stream. On player movement or stance modification, the plugin sends Kiwano commands to notify her neighbors.

Minecraft player events	**Kiwano command**
join	connect to Kiwano
quit	disconnect from Kiwano
kicked	disconnect from Kiwano
move	update lng, lat, alt
look	update angle
toggle sneaking	update appdata
toggle flying	update appdata
toggle sprinting	update appdata
animation	message to all appdata
chat/say	message to all
chat/tell	private message

TABLE I: Minecraft actions translation

On the other side, the plugin converts Kiwano notifications into local Minecraft events (see table II). For instance, on reception of entity updates the plugin updates, or creates, the local representation of remote players.

Kiwano notification	**Minecraft handling**
update	create or update entities
remove	remove corresponding entities
message	issue corresponding action: animate entity, display message, etc.

TABLE II: Kiwano notifications translation

According to the nature of the map and different world physics, there are other types of events that can be handled to be synchronized across Manycraft nodes. A common example are the doors opening and closing.

E. Manycraft Scalability

The Minecraft server embedded in the node supports only the load of one client. The traffic with Kiwano is a subset of the normal Minecraft traffic and remains low. Finally, the translation and routing processes of the proxy do not generate any significant load.

Compared to a regular Minecraft client, the CPU load of the user's computer is only increased by the low activity of the embedded server. The memory depends on the size of the preloaded map but remains constant and low in our tests. Because in average the number of neighbors provided by Kiwano is constant, resources used by the proxy remain constant. The external communication of a Manycraft node happens only with Kiwano and the throughput is similar to a Minecraft client, minus the incoming map description messages. All in all, compared to classical setting —a Minecraft client connected to a remote multiplayer server,— the load of the user's machine is lower regarding bandwidth and slightly higher regarding CPU and memory usage.

IV. Demonstration

Since its early versions, in order to test the system, we have invited users to connect and comment about their experience. On a regular basis we organize testing parties where we try to gather a maximum number of users.

The demonstration will be organized similarly: using social media, fora and mailing lists, we will advertise as widely as possible the Manycraft demonstration at Netgames'2013 inviting people to connect to Manycraft at the scheduled time slots. Using the Minecraft embedded chat, distant connected users will be asked during the demonstration to give their impressions, report problems and propose improvements.

To run the node, the requirements are minimal –be a Minecraft registered user– and the installation is straightforward –download and run. Therefore having a number of participants to the demonstration in the thousands is not inaccessible and will be our goal.

V. Conclusion

This demonstration shows a solution to distribute the load of a multiplayer Minecraft server among several machines. This opens the possibility of a Minecraft world accepting simultaneously an unlimited number of players.

The scalability of the multiplayer mode allows new perspectives in massive social interactions such as concerts, fairs, public demonstrations, battles, etc. However, this first version of Manycraft only implements a limited set of Minecraft features. A full-fledged version should include the whole set of avatar animations and actions as well as more complex map dynamics.

Following the recipe of Minecraft success, Manycraft should allow users to mod and adapt the system to their needs. To do so, we plan to provide an API for third party developers to implement their own features on top of Kiwano.

In today's version, Manycraft provides a unique map. An interesting feature we plan to implement is to allow users to extend the world with their own maps. This would give birth to a unified Manycraft world as an extensible patchwork of many Minecraft maps.

References

[1] "A directory of minecraft servers," http://minecraftservers.org/. Retrieved August 10, 2013.

[2] R. Diaconu and J. Keller, "Kiwano: A scalable distributed infrastructure for virtual worlds," in *High Performance Computing & Simulation (HPCS)*, 2013.

[3] "Minecraft current stable protocol," http://mc.kev009.com/Protocol. Retrieved August 20, 2013.

[4] "Kiwano application programming interface," http://kiwano.li/.

[5] "Bukkit, a free, opensource, extension of minecraft multiplayer server," http://bukkit.org.