Xadrez Chess Server Development Report #1

Felipe Bergo
bergo@seul.org

July 7th, 2002

1 A Brief Introduction to Xadrez

Xadrez is a new effort to develop a free chess server software. By free we mean free in the GNU sense of freedom, people who has access to the software should be have as much freedom as the people who developed it. This is accomplished by developing under the Affero General Public License [1].

Xadrez attempts to be somewhat compatible with FICS [2], which provides free access, but no source code to the server software, for many reasons. This means Xadrez will use a text-based protocol suitable for telnet connections. We intend to make Xadrez compatible with the current interfaces available for FICS (winboard/xboard [4], eboard [3], Thief [5] and others).

Xadrez is being developed on a GNU/Linux operating system, in the C++ programming language, using the PostgreSQL [6] as database backend. Xadrez should be compatible with any POSIX-compliant environment where the GCC C++ compiler is available, so most free BSDs, Mac OS X, GNU Hurd and Unix-like architectures should be able to run Xadrez. Please note that language and database choices are not negotiable: the core server will be written in C++ using PostgreSQL, if you don't like it, don't even bother whining. In the future, additional modules may be needed (automated lectures, administration support bots, tourney bots), and new languages may be chosen for each, based on common sense.

Xadrez is the portuguese word for *chess*, in case you are wondering the origin of the name.

Xadrez started as a solo project, but due to time constraints, I have not been able to work as much as I'd like on it. This report aims to document the project structure, known issues and sketch a short-term plan to get the project moving. Volunteer developer work is wanted.

Xadrez is still quite simple and understandable for the kind of work it does, and that's good. More comments about team practices will be made throughout this document.

2 What Xadrez Should Be Able To Do

This is a very quick list of what the server software should be capable of doing:

- Allow users to connect and authenticate with either guest accounts or login+password pairs.
- Allow users to chat either directly or using channels (broadcast channels like shouts included).
- Allow users to match other users for games.
- Allow users to post ads (seeks) for games.
- Detect move correctness, game end, as well as draw possibilities.
- Implement game clocks and allow flagging.
- Keep a history of the last games a user played.
- Allow observation of live games and examination of stored games.
- Keep various statistics about user activity, including a rating.
- Allow easy inclusion of new chess variants.
- Lots of other things. Anyone willing to actually enumerate the features feel free to do it. FICS is huge, I never said this project would be easy. But getting the basics done is, the current code base is already a nice starting point.

Users connect to the server using TCP stream connections (see `man 7 tcp` in a Linux/BSD system).

3 Server Architecture

The server state is *completely* stored in the database backend. The basic architecture of a Xadrez Server is depicted in Figure 1.

Most developers and small servers will run Xadrez and PostgreSQL in the same computer, so the High Bandwidth, Low Latency channel will be a Unix socket, which is a very fast IPC[1] channel. But Fast Ethernet (100 Mbps) should allow running database and server in different computers, as well as having several load-balanced servers sharing a database context (and thus looking as a single server to the user).

The Xadrez process runs several inner tasks and its own scheduler, so one process can manage all users at once, if so is desirable. Since all inner tasks are independent, and server state is always kept on the database backend, it should be possible to scale up the server by adding more Xadrez processes, either in the same computer or not, distributing the work load. Figure 2 shows one possible arrangement for load balancing.

[1] Inter-Process Communication

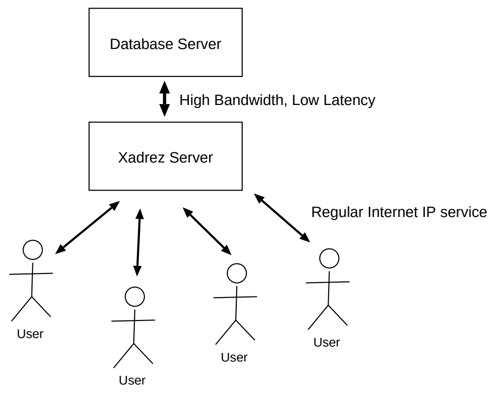

Figure 1: Basic Server Architecture

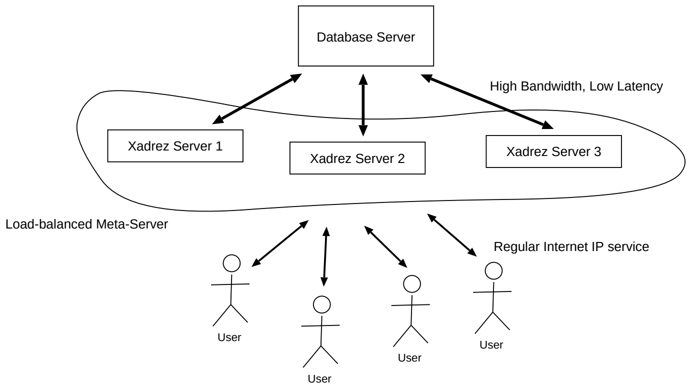

Figure 2: Scaled-Up Server Architecture

4 Implementation Details

Xadrez is meant to be object-oriented, so the running software can be treated as a collection of objects. Whenever a class name is cited, the word in square brackets after it, like **TcpServer** [tcpserver.h], is the file where the class declaration is. For the sake of sanity, all non-abstract classes have their methods in a file of same name but .cc extension, tcpserver.cc in the above example.

4.1 main.cc

Execution starts in `main.cc`, where basic configuration is read from the `xadrez.conf` file, and channel configuration is read from `channel.conf`. Configuration-related code is in `rc.cc`.

Current implementation is not prepared for multiple server processes, so it always sweeps the database into a clean (no users connected, no games going on) state.

After initialization, a scheduler is created and the 2 basic tasks added to it:

```
  Scheduler s;

  s.appendTask( new TcpServer(Port) );
  s.appendTask( new HouseKeeper() );
  s.run();
```

4.2 scheduler.cc

Scheduler [scheduler.h] is a simple round robin scheduler. In the future it may be useful to implement priorities. The scheduler is a container for **Task** [abstract.h] objects. The comments in abstract.h should be quite self-explanatory about what a task is expected to implement.

4.3 tcpserver.cc

TcpServer [tcpserver.h] is a task that waits for connections in a given TCP Port. When a new connection is received, a **Shell** [shell.h] task is attached to it and added to the scheduler.

4.4 housekeeper.cc

the **HouseKeeper** [housekeeper.h] task implements all checks that the server must do regardless of receiving user input. Currently it cleans up zombie clients (whose **Shell** object was removed from the scheduler but whose online status was not cleared from the database) and applies autoflag. Current run() body is:

```
void HouseKeeper::run() {
  tick++;
  if (tick>100) tick=0;

  if (! (tick%20) )
    checkDisconnections();

  if (! (tick%10) )
    autoFlag();
}
```

Adding priorities to the scheduler could simplify this task's behavior.

4.5 shell.cc

Shell [shell.h] is a task that serves as interface between the user and the server backend. Its purpose is to authenticate the user and do two-way translation: user-typed commands to programmatic server method calls and sometimes server method output to user-readable output. Shell performs command expansion (e.g. "t" to "tell") and parses command parameters. The Shell connects the user (represented by the NetIo *net object) to the server backend (represented by the Server *svr object).

4.6 netio.cc

netio.cc [netio.h] implements buffered, non-blocking I/O. Its purpose is to prevent us from seeing explicit select() calls elsewhere in the code.

4.7 dbclient.cc

DBClient [dbclient.h] implements an interface to a PostgreSQL database, providing quick proxies to SQL-execution. Should Xadrez ever be converted to another database backend, it would be a matter of converting this class's body and (unfortunately) some of the SQL statements throughout the **Server** parent classes. Each instance of DBClient creates a new connection to the database (which usually means a new process is forked by the main database backend), but all Shell and Server instances share the same primeval instance of DBClient, keeping only one connection to the backend.

4.8 chess.cc

Position [chess.h] implements the game of chess. Currently chess, crazyhouse and suicide rules are implemented, but suicide game end situations are not there, there's (I think, must check) a little bug in crazyhouse checkmate verification, and several draw verifications must be written (50-move count, no material situations).

4.9 icsboard.cc

IcsBoard [icsboard.h] represents an ICS position. Currently implements FICS style 12, we should implement style 1 ASAP, since it makes debugging much easier (connecting with telnet is much faster when debugging, and several xterm+telnet sets are usually easier to handle than several GUI clients).

4.10 tstring.cc

tstring [tstring.h] was taken from eboard (and before that a primitive version was used in gPS [7]), it implements safe string tokenizing.

4.11 server.cc and op*.cc

server.cc and all op*.cc were once one single 7000-line server.cc. Being that unmanageable, I broke it into a class hierarchy. **Server** [server.h] implements a programatic vision of the chess server. Its functionality was split by "theme" into the following hierarchy shown in Figure 3.

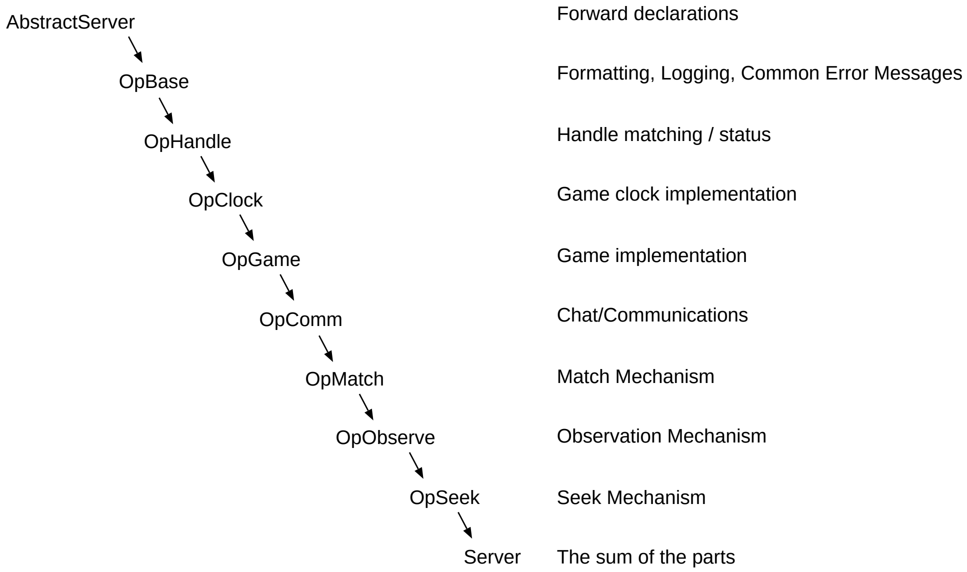

Figure 3: Server Class Hierarchy

The order of inheritance was chosen to minimize forward declarations. Where it couldn't be avoided, methods were declared abstract in **AbstractServer** [abstract.h]. More op*.cc will come for sure, they should be inserted somewhere above **Server** [server.h] as appropriate.

5 Database Schema

The database declaration can be found in the `db.sql` file. A brief summary of each table is given below.

who	online users
msgqueue	inter-user communication buffer
conncheck	used to detect disconnections
saytarget	current target of the say command for each user
channels	channel list
inchannel	who is in which channels
chanprefs	channel list for offline users

intvarset	list of integer variables and default values
textvarset	list of string variables and default values
intvars	current integer variable values per user
textvars	current string variable values per user
accounts	registered users
admins	admin list
matches	pending match requests
games	ongoing games
gamehistory	global server game history
playerhistory	per-player game history
pstat	meant to keep pstat-like data
gamepos	positions of ongoing games
gametypes	list of variants
observers	who is observing what
typealias	aliases for variants
requests	pending requests (abort / draw / adjourn)
gameclocks	clocks for ongoing games
log	server log
seeks	active seeks
outcomes	game-end codes/messages

6 How to get Xadrez Source Code

Xadrez is currently available in CVS-only. For anonymous access, see `http://savannah.gnu.org/cvs/?group=xadrez`.

If you want to help develop Xadrez, don't walk, RUN to `http://savannah.gnu.org` and create an account. Be sure to subscribe to the xadrez-devel mailing list: `http://savannah.gnu.org/mail/?group=xadrez` and announce who you are and what you want to do there. In the future, I expect you guys to be writing these reports with me.

7 Compiling and Running

You'll need a working PostgreSQL 7 installation before anything else. It's recommended that you get used to it, so reading its docs is a good idea. I've been developing Xadrez with PostgreSQL 7.0.3 installed from source in the `/usr/local/pgsql` prefix, if you use something else you may need to change Xadrez's Makefile. I will make Xadrez configure itself with GNU autoconf/automake sometime soon.

Create the database in PostgreSQL with whichever name you like most (and edit xadrez.conf to reflect it), then run `db.sql` from psql to create the tables. If you change something in the database, run `drop.sql` before rerunning `db.sql`.

After that, just `make` should compile xadrez. To run on port 5000, `./xadrez`.

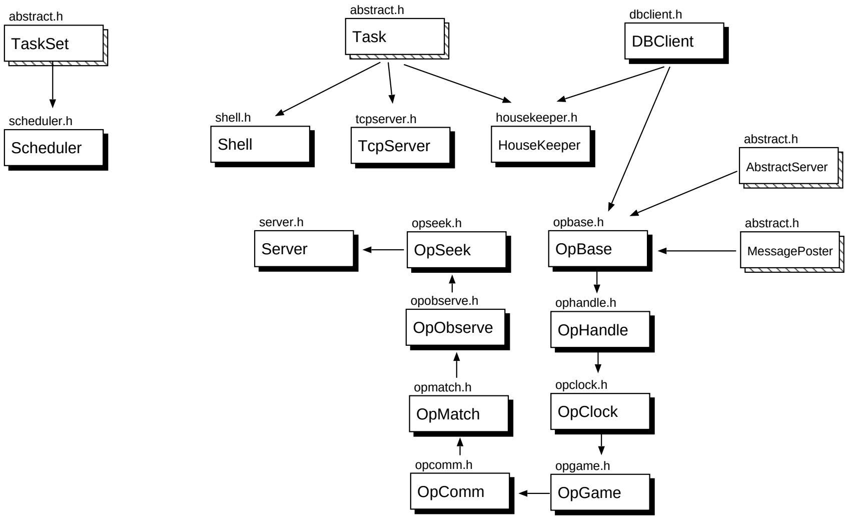

Figure 4: Class Hierarchy (lone classes not shown)

Xadrez is currently known to work with eboard [3] and xboard [4]. And telnet, of course.

8 How to Help

I'm currently overwhelmed by my graduate course's work, the current state of things isn't very pretty. The following is a short list of things that can be done right now, so if you get more or less how the code works, especially the **Server** hierarchy, you probably can do something:

- Implement history and pstat commands.
- Implement examination.
- Think of a rating system and discuss/explain it in xadrez-devel.
- Implement draw-detection in chess.cc
- Write help files (in the help directory) for the commands.
- Provide a way to generate class inheritance charts automatically from the code. I haven't investigated it yet, I used doc++ once and didn't like it, maybe something better exists and I don't know ?
- If you think the op*.cc thing was ugly, suggest something better. (Note: a monolithic 7000-line .cc file is not better).

- Implement Style 1 Boards.
- Implement Aliases.
- Make a detailed list of commands to implement (from FICS) so people can pick up tasks more easily.
- Write a decent test suite for the Position class.
- Write a decent configuration app. xacfg.cc (makes xacfg) was a first try based on the `dialog` program, but let's be honest, it sucks. We need a console-based (no GUI) application to configure database parameters and add/remove accounts.
- Make **Shell** send the correct telnet/tty escape code to hide the password, whichever it may be.
- Write a class to allow sending email easily.

  ```
  Mail *m;

  m=new Mail();
  m->from    = "xadrez@xadrez.net";
  m->to      = "john@star.com";
  m->subject = "moves for game John vs. Archibald";
  m->body    = game.toPgn();
  m->send();
  ```

  Is a reasonable sample usage.
- My Internet connection is a slow, expensive, third-world dial up, I don't have where to run a test server. Does anyone have a 24/7 online box, preferrably with low load, where we could keep a test server running ?
- Figure out an improved version of this list.

9 Ramblings About Random Topics

I have the intention to separate lightning/blitz/standard ratings for variants. The idea is to keep both a per-speed rating and a general rating, so each player would have 4 ratings per variant. This conflicts with the FICS way of representing games because it uses just one letter to represent the kind of game.

One user tried to connect to Xadrez from a Macintosh telnet client and got no line breaks. I'm sending only LF (\n) to break lines. We should fix it to CR+LF, LF+CR or whatever is popular these days.

References

[1] Affero General Public License, http://www.affero.org/oagpl.html

[2] Free Internet Chess Server, http://www.freechess.org

[3] eboard chess interface, http://eboard.sf.net

[4] winboard / xboard, http://www.tim-mann.org/xboard.html

[5] Thief, http://www.thebugboard.net/tbb_downloads.htm

[6] PostgreSQL database, http://www.postgresql.org

[7] Graphical Process Statistics, http://gps.seul.org