An Overview of NECI's Generic Game Server

Michael Buro & Igor Đurđanović
(mic|igord)@research.nj.nec.com

NEC Research Institute
4 Independence Way
Princeton NJ 08540, USA

Abstract. We give a high level description of our Generic Game Server (GGS) and its accompanying client software. The system is based on a flexible communication architecture and provides a rich set of features unseen in traditional game servers — such as a wide variety of board games and game setups and fair game modes for both standard and random starting positions. Moreover, the task of adding new game services and GUIs is eased by reusable server and client software components.

Keywords: Internet Game Server, Client Software, GUI.

1 Introduction

In this paper we present NECI's Generic Game Server (GGS) and its underlying communication architecture. GGS's game features are unique: in addition to usual game server functionalities like chat and rating systems, for each game it supports all common game clock modes, a wide variety of game setups, and fair game modes for standard and random starting positions. Furthermore, all parts of GGS are licensed under the GNU Public License [9] -- giving programmers free access to the source code, which speeds up the development of new services. Currently, GGS offers Amazons [6], Checkers, Chess, Hex, Go, Othello, and Phutball [7] services. Additional services for turn-based games and graphical user interfaces for them can be implemented in a matter of hours due to reusable software components for communicating moves, saving/restoring adjourned games, rating updates, displaying boards on GUIs etc.

GGS is a loosely coupled network of services and clients built around a central server. In the context of game servers its main advantages compared to the traditional "one server, one game" approach are

- A central connection point for everybody which allows people with the same interests (e.g. machine learning in games) to exchange ideas even though they might try their ideas on very different games.
- Messaging, news, tournament director and other services that each game server implements in one way or the other can be extracted into a separate services which serve all game services, thus simplifying client software even further.
- The central server does not specify any service-client or client-client protocol. Rather it just provides a communication medium, leaving each service to implement and enforce its own client protocols. However, services should cooperate on designing a protocol for intra-service communication (e.g. tournament director).

In what follows, we provide a bottom up view of GGS: first the underlying communication architecture is described. Then we give an overview of the game service features and the available client software. An outlook on future improvements regarding scalability and additional services concludes the paper.

2 GGS's Communication Architecture

GGS is a loosely coupled network of multiplexors, services, and clients which are all connected to a central server (Fig. 1) that accepts/sends and relays text messages using the TCP protocol. The central server serves the following purposes:

- it provides a central connection point for all clients and services.
- it performs authentication of clients and services.
- it allows clients and services to communicate with each other.

Multiplexors were introduced to get around operating system limitations, which restrict the number of simultaneously open file descriptors, and to provide a TCP-tunnel in case the central server is located behind a firewall. Theoretically, the central server augmented with one thousand multiplexors could accommodate over one million clients and services simultaneously. However, at one point, this will lead to a communication bottleneck. The upcoming release will address this problem.

The communication between the central server, clients, and services can either be raw or compressed by the standard gzip or bzip2 protocols. Secure connections to the central server can be established by using a SSL proxy server. To ease the communication between clients and services, the central server provides alias and variable mechanisms allowing each user to tailor its front-end to his/her taste. In addition, the central server supports the concepts of groups and channels. The group mechanism is used to group clients by their functionality and administration level (client, service, sysadmin). The creation of groups

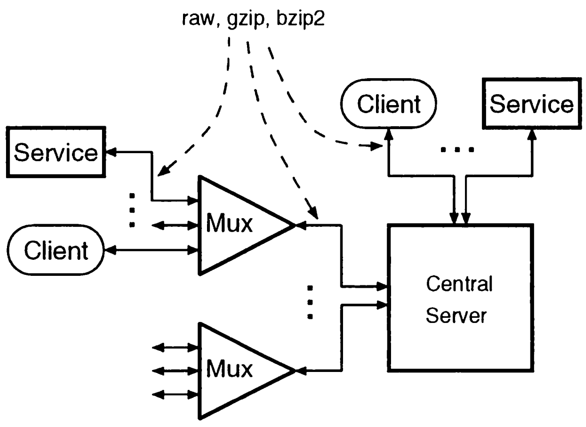


Fig. 1. Client–Service–Server Architecture

and assignment of clients to them is restricted to system administrators. The channel mechanism is a convenient way for several clients to talk to each other simultaneously resembling chat room functionality. Users can freely create and destroy, join and leave channels, which can be either public or private. Finally, the central server gives clients access to user statistics, news and help files.

Services connect to the central server like any other client and they behave like servers in every other aspect, i.e. they serve clients. Except for a thin protocol between the central server and services that incrementally keep services updated about clients connecting to or leaving the network, the central server does not enforce any particular protocol between services and clients. Every service can implement its own set of commands, groups, channels, aliases, and variable mechanisms.

3 Game Services

Currently, GGS provides service and client software for the following turn–based board games: Amazons, Checkers, Chess, Hex, Go, Othello, and Phutball. On the service side, all software is written in C++. The common service functions — such as communication with the central server, game handling, storing and retrieving of game transcripts, storing of player variables, and rating updates — are implemented as a library. Adding a new board game service is as simple as implementing a small abstract C++ interface which defines hooks to the game specific service parts. The same is true for the graphical user interface GGSA. Written in Java for allowing cross–platform operation, it follows the same object oriented philosophy: the game specific details are encapsulated on top of the base functionality that each GGS graphical client provides. As a result, new (game) services can easily be added as long as they do not require substantially different functionality compared to the services we already have implemented.

3.1 Unique Features

When designing the game services, we focused on ensuring a fair and competitive playing environment and on providing a rich variety of game modes. Furthermore, we strived for keeping the communication protocol and game transcript format simple and generic. These considerations led to the following service features, which mostly are unique to GGS. Each game service supports

- the Glicko rating system [4] which also models rating variability,
- a universal clock that can emulate all common game clocks including the Fisher, Bronstein, and byoyomi mode,
- a common format for archiving games,
- different board sizes/shapes (Fig. 3), and
- fair game handling.

We feel that using random starting positions has advantages over the traditional fixed position approach. For instance, players do not have to worry about opening preparation. Moreover, starting in a single position makes perfect information games unfair if the game value is different from zero. A prominent example is Hex, where it can be shown that the first player wins, or the second player if the swap rule is used. A forthcoming paper will be devoted to this topic. In it we will describe the *komi* and *synchro* game modes we have implemented in GGS, which ensure fair games for both fixed and random starting positions.

3.2 Client Software & Resources

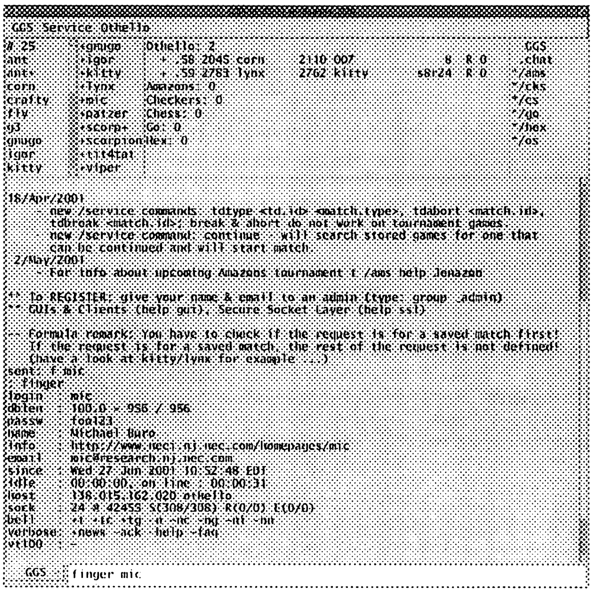
The following client software is included in the GGS source tree:

- A simple C++ Othello client for connecting Othello programs to GGS.
- A generic C++ client currently supporting Amazons, Checkers, and Othello.
- GGSA – the GGS Applet for all game services (Fig. 2,3). Runs in every browser.
- A wrapper interface for connecting standard–i/o driven game programs.

Chris Welty has written following Windows software for GGS:

- LION a Windows Othello GUI for GGS (binary only)
- ODK (Othello Development Kit) A collection of C++ classes for connecting Othello programs to GGS.
- TD a global tournament director for all game services.

Utilizing these clients and wrappers we have connected our strong Othello programs Logistello [2] and Kitty, our Amazons program AmsBot, Robert Hyatt's



```
GGS Service Othello
# 25      <gmigo    Othello: 2                                    GGS
ant       <igor       + .58 2045 corn     2110 007        8  R 0  .chat
ant+      <kitty      + .59 2783 lynx     2762 kitty   s8r24 R 0  */ams
corn      <lynx     Amazons: 0                                    */cks
crafty    +mic      Checkers: 0                                   */cs
fly       <patzer   Chess: 0                                      */go
g3        <scorp+   Go: 0                                         */hex
gnugo     >scorpion Hex: 0                                        */os
igor      <tit4tat
kitty     <viper

18/Apr/2001
    - new /service commands: tdtype <td.id> <match.type>, tdabort <match.id>,
      tdbreak <match.id>; break & abort do not work on tournament games.
      new /service command: continue - will search stored games for one that
      can be continued and will start match.
 2/May/2001
    - For info about upcoming Amazons tournament t /ams help Jonazon

** To REGISTER: give your name & email to an admin (type: group .admin)
** GUIs & Clients (help gui), Secure Socket Layer (help ssl)

-- Formula remark: You have to check if the request is for a saved match first!
   If the request is for a saved match, the rest of the request is not defined!
   (have a look at kitty/lynx for example ...)
sent: f mic
: finger
login   : mic
obten   : 100.0 > 956 / 956
passw   : foo123
name    : Michael Buro
info    : http://www.neci.nj.nec.com/homepages/mic
email   : mic@research.nj.nec.com
since   : Wed 27 Jun 2001 10:52:48 EDT
idle    : 00:00:00, on line : 00:00:31
host    : 138.015.162.020 othello
sock    : 24 # 42453 S(308/308) R(0/0) E(0/0)
bell    : +t +tc +tg -n -nc -ng -nt -nn
verbose : +news -ack -help -faq
vt100   : -

GGS   finger mic
```

Fig. 2. GGSA's main window showing user and game lists and GGS's response to "help" and "finger" commands.

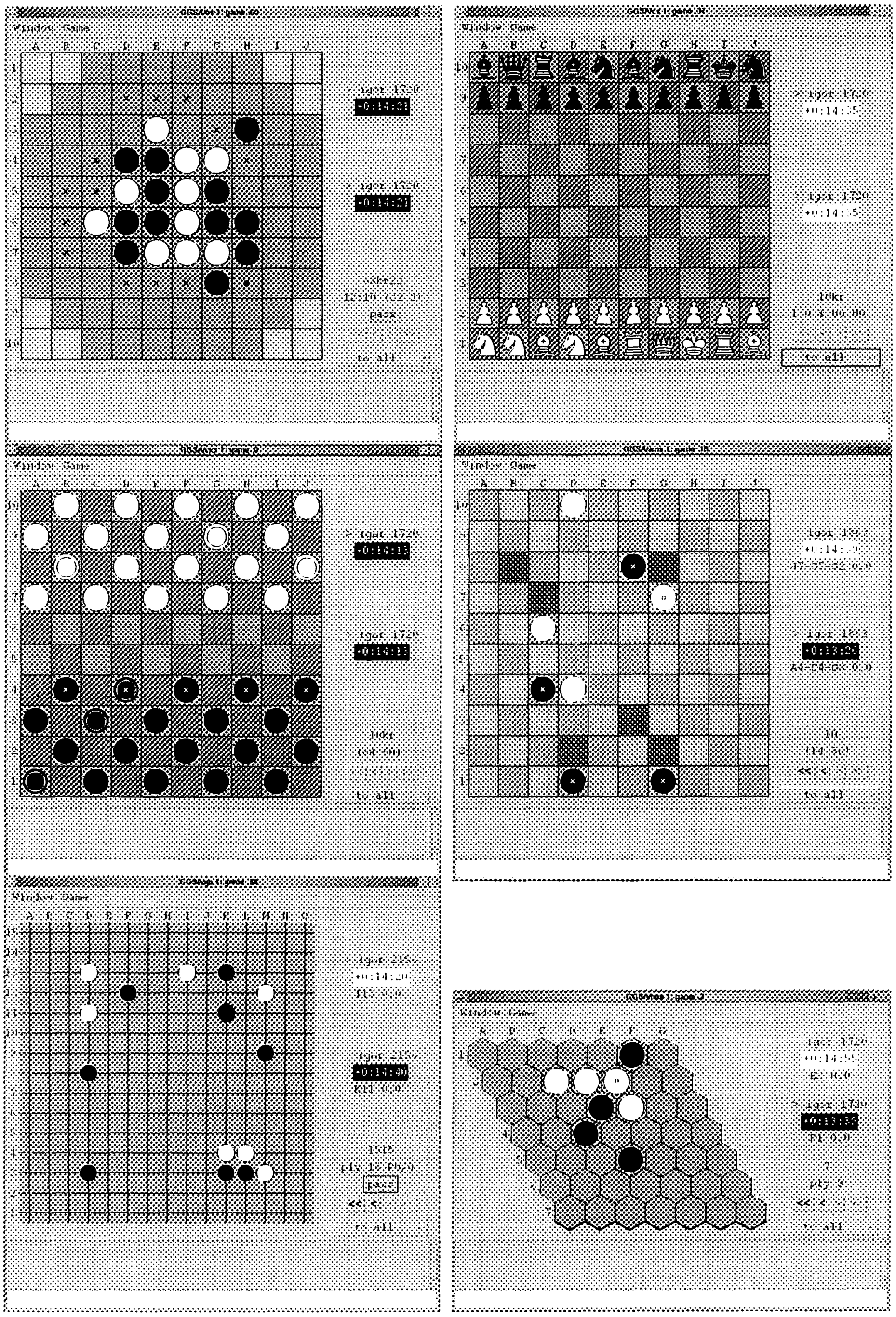


Fig. 3. GGSA screen snapshots of non-standard starting positions (Othello, Chess, and Checkers) and positions from the opening (Amazons, Go, and Hex).

freeware Chess program Crafty [5], and GnuGo [8] to GGS. The GGS software and documentation is available at [1,3]. GGS itself can be reached by telnet or the GUIs at multiplexor ports 4000 and 5000 of external.nj.nec.com.

4 Outlook

In case GGS really takes off, we realize that the central server and services -- depending on their popularity -- would eventually become communication bottlenecks. Therefore, we plan a more distributed and scalable architecture for the next release of GGS, in which a central server *gate* performs load–balancing when redirecting connecting clients to a network of central servers. The concept of distributed servers may not be crucial for low–bandwidth applications such as the turn–based game services or instant messaging and news tickers that we intent to implement. However, it is essential for file sharing and for multiplayer real-time game services, which we plan to support as well.

Besides providing a wide range of services, the acceptance of a service network largely depends on the user interface. In this regard the current situation is less than optimal as the applet GGSA currently only supports the English language, still requires expert GGS knowledge, and suffers from buggy and incomplete browser Java implementations. To alleviate this situation we intend to port GGSA to the cross–platform C++ graphics toolkit Qt [10], which provides a uniform, Unicode capable interface under Microsoft Windows, the X-Window system, and soon Mac OS-X, while allowing to run code natively.

References

1. Michael Buro. GGSA homepage.
http://www.neci.nj.nec.com/homepages/mic/ggsa. Web site.
2. Michael Buro. Logistello homepage.
http://www.neci.nj.nec.com/homepages/mic/log.html. Web site.
3. Igor Đurđanović. GGS homepage.
http://www.neci.nj.nec.com/homepages/igord/gsa-ggs.htm. Web site.
4. Mark E. Glickman. The Glicko system.
http://math.bu.edu/people/mg/papers/gdescrip.ps. Paper.
5. Robert Hyatt. Crafty download.
ftp://ftp.cis.uab.edu/pub/hyatt/. FTP site.
6. Play By Email Server. Amazons rules.
http://www.gamerz.net/pbmserv/amazons.html. Web site.
7. Play By Email Server. Phutball rules.
http://www.gamerz.net/pbmserv/phutball.html. Web site.
8. GNU Software. GnuGo homepage.
http://www.gnu.org/software/gnugo/gnugo.html. Web site.
9. GNU Software. GPL FAQ.
http://www.gnu.org/copyleft/gpl-faq.html. Web site.
10. Troll Technologies. Qt homepage.
http://www.trolltech.com. Web site.