## NAME

**go-kgp** — A versatile KGP server

## SYNOPSIS

**go-kgp** [ **-conf** *file* ] [ **-dump-config** ] [ **-debug** ] [ **-about** *file* ] [ **-db** *file* ] [ **-host** *hostname* ] [ **-port** *port number* ] [ **-isolate** *type* ] [ **-sched** *spec* ] [ **-timeout** *seconds* ] [ **-webport** *port number* ] [ **-websocket** ] [ **-help** ]

## DESCRIPTION

**go-kgp** implements the KGP (Kalah Game Protocol), for both TCP and WebSocket connections. It can be used both to host public, random matches as well as local, organised tournaments.

The server provides a number of game schedulers

Local tournaments can make use of *Docker*: **https://www.docker.com/**, to isolate clients from one another, as described under **TOURNAMENTS**.

## OPTIONS

**-conf** *file*
Path to a configuration file. If empty, check if `server.toml` is defined, otherwise the default configuration is used.

See **CONFIGURATION**, for details on the syntax and options.

If this options is used, the configuration can be edited and reloaded during execution by sending the process a SIGUSR1 signal. Note that this cannot affect all options instantaneously.

**-dump-config**
Dump the default configuration on to the standard output stream.

**-debug**
Enable debugging output, and increase the detail or regular logging.

Equivalent to the `debug` option.

**-about** *file*
A *Go template*: **https://pkg.go.dev/text/template** used to generate an "About" page. If not used, no about page will be generated.

Equivalent to the `web.about` option.

**-db** *file*
Path to a *SQLite 3*: **https://www.sqlite.org/index.html** database. The database is used to log agents, games and tournament results.

Equivalent to the `database.file` option.

**-host** *hostname*
Interface to bind the server when hosting public games. Will default to `0.0.0.0` if not specified.

Equivalent to the `tcp.host` option.

**-port** *port number*
Port to bind the server when hosting public games. Will default to 2671.

Equivalent to the `tcp.port` option.

**-isolate** *type*
Isolation mechanism to use for local tournaments (See **TOURNAMENTS**, for more details).

Equivalent to the `tournament.isolate` option.

**`−sched`** *`spec`*
A scheduler specification, used to match clients. See **Scheduler Specification** below for more details.

Equivalent to the `sched` option.

**`−timeout`** *`seconds`*
Number of seconds granted to a client to make a move.

Equivalent to the `game.timeout` option.

**`−webport`** *`port number`*
Port to bind the web interface to.

Equivalent to the `web.port` option.

**`−websocket`**
Enable websocket connections to the public server.

Equivalent to the `websocket.enabled` option.

## CONFIGURATION

The configuration file can be used to avoid specifying the same configuration options when restarting the program. The configuration is written in the*T OML*: **https://toml.io/en/**, and can be stored in any file. To tell **`go-kgp`** what file to load the configuration from, use the **`−conf`** flag. This can be further simplified, if the configuration is stored in the file `server.toml` in the current working directory. If neither are found, the default configuration is used without any further comments.

When writing a configuration, it is recommended to use the **`−dump-config`** flag, that writes out the default configuration out onto the standard output stream:

```
$ go-kgp -dump-config > server.toml
$ vi server.toml
$ go-kgp # loads the new configuration
```

See **OPTIONS** above for a list of configurable options.

### Scheduler specification

A scheduler decides on how to pair clients into games. A scheduler specification can sequence a number of schedulers and “pseudo-schedulers” to be executed in sequence. When each scheduler indicates that it is finished, the next one is started. When the last scheduler finishes, a server-shutdown is initiated.

The specification itself is a string, where each scheduler is delimited by a string. Some schedulers might take arguments, which are in turn delimited using periods.

The default scheduler uses a FIFO queue to match clients. If you were to specify this scheduler by default, you would pass the string

```
"fifo"
```

via the **`−sched`** flag ( or the `sched` option in the configuration file ) .

Here follows a complete list of schedulers:

fifo A “First-In First-Out” scheduler, reduces the waiting time by matching incoming and waiting clients in order they are noticed by the server. This scheduler is adequate for public tournaments and will never indicate that it is finished. Clients are immediately added back to the waiting queue, as soon as their games are over.

rand, random
: An instantaneous scheduler that let's each incoming clients play a game against a built-in random agent. The scheduler never terminates, and does re-enqueue a client after the game is over.

  The main use for this scheduler is as a base-line test to verify if a client has a minimal viable strategy.

rr, round-robin
: A "round robin" tournament scheduler, that will match up each client with each other client exactly once. The scheduler takes one argument, denoting the board size. That is to say that `round-robin.6` is interpreted as a "round robin" tournament, where each game is played on a Kalah (6,6) board (six pits on each site, with six initial stones).

  This scheduler terminates after all games have finished.

bound
: A pseudo scheduler that filters out all clients with a score below the first argument of the scheduler (`bound.1000` would filter out each client with a score *less than* 1000). It will immediately terminate.

skim
: A pseudo scheduler that takes an argument *n*, and only keep the *n* best clients. In case it the *nth* best client cannot be determined, the scheduler will take all clients with the same score.

reset
: A pseudo scheduler that will reset the score of each agent to the argument value.

(Pseudo schedulers) can be used to modify the list of participants between rounds of a tournament. For example, the scheduler

```
"rr.4 skim.10 !.0 rr.8 skim.5 rr.12"
```

would be interpreted as "start a round robin tournament on board sizes (4,4), then take the ten best clients, reset the score, proceeded to the next round on (8,8), take the 5 best clients, then start another round robin round on (12,12), and finally terminate".

## Configuration Options

`sched` A scheduler specification. See **Scheduler Specification**.

`debug` Enable debug logging.

`endless`
: Enqueue clients back into the queue after finishing a game. This option is only used by the "fifo" scheduler.

`database.file`
: Path to the SQLite database.

`database.threads`
: Number of database managers (workers in the thread pool synchronising operations on the database).

`database.timeout`
: Timeout in nanoseconds for a database query to complete, before it is aborted.

`database.optimise`
: Enable periodic database optimisations.

`tournament.list`
: Path to a file of tournament participants. Each line should list the name of one client. The interpretation of this name depends on the isolation mechanism.

`tournament.isolation`
: How to start a client in a local tournament. The currently legal values are:

  “process”
  : The name is interpreted as a directory name in the working directory, and each client is executed by calling the `run.sh` script in the said directory.

  “docker”
  : The name is interpreted as a docker image, and the client is executed in a new container.

`tournament.warmup`
: Number of seconds the server will wait for a all clients to connect. If a client takes longer, it will be disqualified.

`tournament.docker.memory`
: Memory in bytes each Docker container is granted.

`tournament.docker.swap`
: Swap in bytes each Docker container is granted.

`tournament.docker.cpus`
: Number of CPUs each Docker container is granted.

`tournament.docker.network`
: Name of the network the docker container will be connected to.

`game.sizes`
: List of board sizes to choose from for “fifo” games.

`game.stones`
: List of initial stones to choose from for “fifo” games.

`game.timeout`
: Number of seconds

`game.earlywin`
: Enable the detection of games that cannot be won by one side.

`game.slots`
: Limit the number of concurrent games. If 0, then there is no limit.

`game.skiptriv`
: Enable the automatic execution of trivial moves ( when an agent has only one choice ) .

`web.enabled`
: Enable the web interface, used to view agents, the scoreboard and game logs.

`web.port`
: Port to bind the HTTP server to, for the web interface.

`web.limit`
: Limit to the number of entries a table may list.

`web.about`
: Path to a Go template that will be used to generate a “About” page.

`websocket.enabled`
: Start a public WebSocket server. Depends on`web.enabled`.

`tcp.enabled`
: Start a public TCP server.

`tcp.host`
: Interface to bind

`tcp.port`
: Port to bind the public server TCP to.

`tcp.ping`
: Enable the usage of `ping` commands to check if the connection is alive.

`tcp.timeout`
: Number of seconds after which the server will terminate a connection, after not receiving a `pong` for a `ping` command.

`tcp.retries`
: Number of attempts the server will attempt to send out a (partial) message.

## DEVELOPMENT

The development takes place on GitHub in the **https://github.com/KWARC/kalah-game** repository, under the `server/go-kgp` directory. Please report issues, bugs and feature requests on there.

## AUTHORS

**`go-kgp`** was initially designed, developed and documented by Philip Kaludercic <`philip.kaludercic@fau.de`>, for the KWARC Group.