Snarl Protocol

This document collects the message sequence diagrams and JSON data definitions for the messages of the SNARL remote protocol.

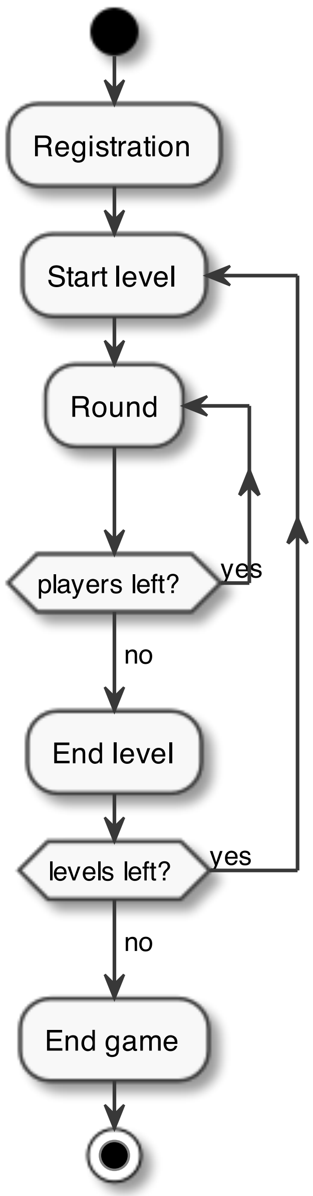
Message sequences

Phases

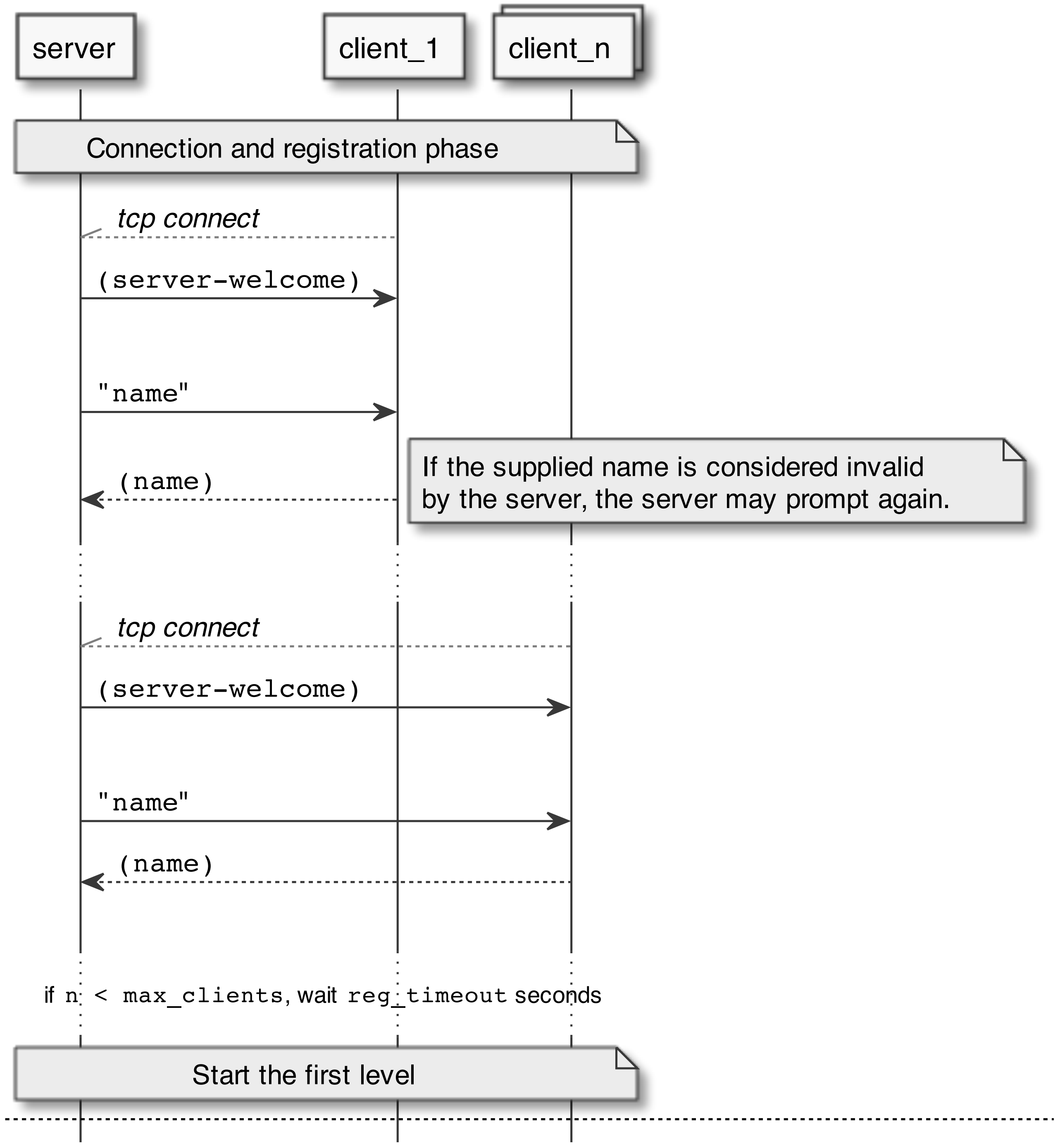
The protocol specifies 5 phases:

1. Start-up, registration of players
2. Start of a level
3. Perform rounds
4. End of a level
5. End of the game

Figure 1: Protocol phases

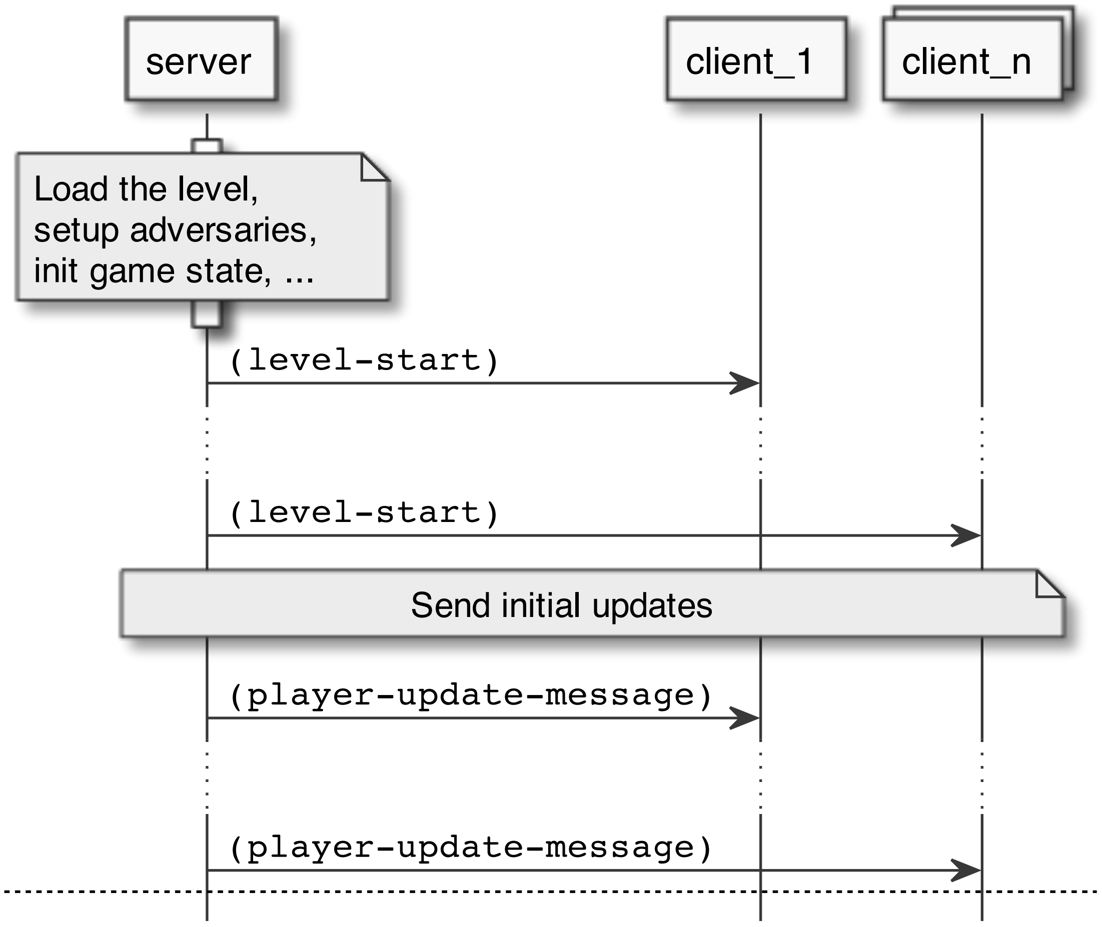


Registration of players


server
client_1
client_n
Connection and registration phase
tcp connect
(server-welcome)
"name"
(name)
If the supplied name is considered invalid by the server, the server may prompt again.
tcp connect
(server-welcome)
"name"
(name)
if n < max_clients, wait reg_timeout seconds
Start the first level

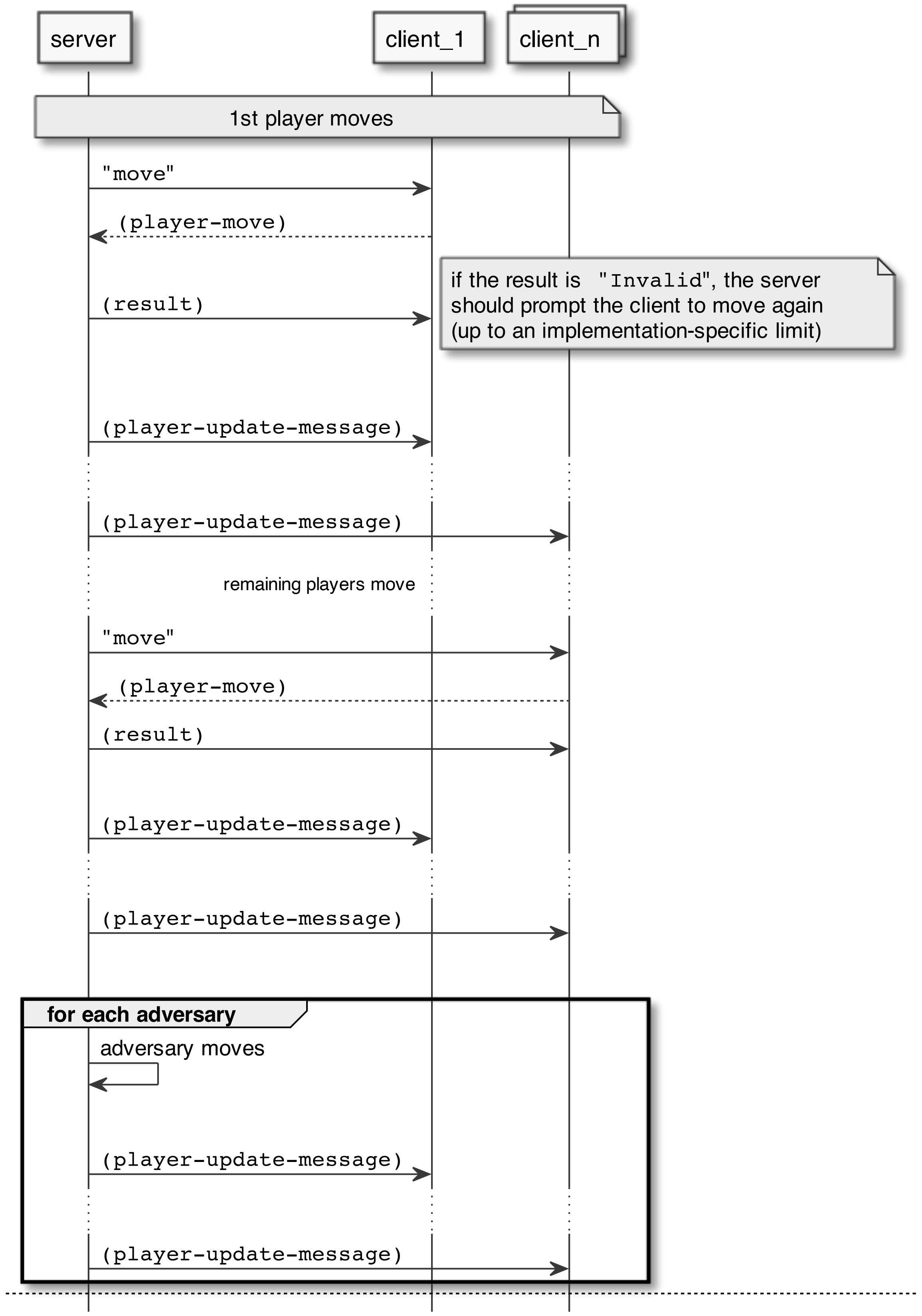
After a client connects, the server sends a "welcome message" and requests the player's name, which is supplied as a JSON string (in double quotes). `max_clients` and `reg_timeout` are implementation-specific parameters, which may be part of the server configuration.

Starting a level



To start a level, the server sends a message notifying the clients that the level is about to start. This is followed by initial updates about the players' whereabouts.

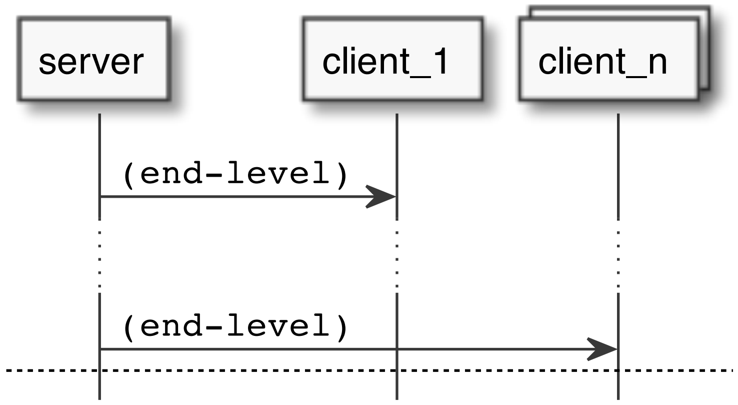
Playing a round



A round consists of the server asking a player for their move, then updating each player about the state, then repeating the same for the remaining players. A player's move might be invalid and the server may ask the player to supply another move. After all players supply their moves, adversaries move and players get updated after each adversary's move.

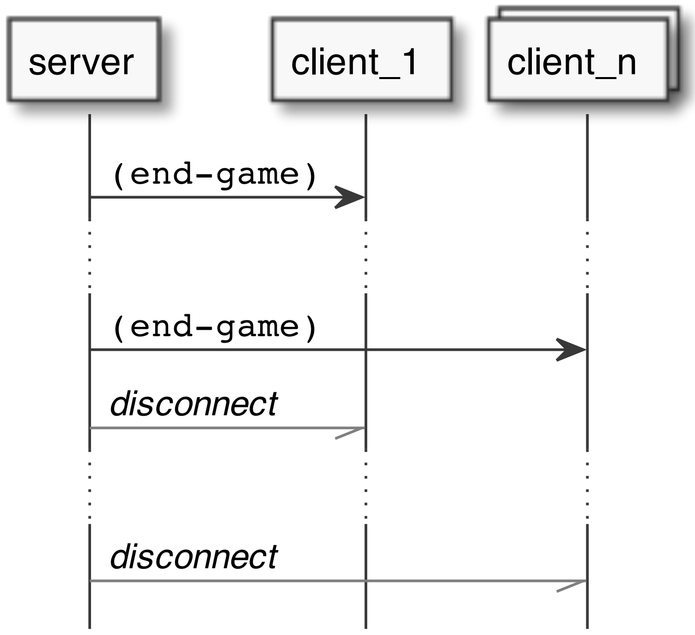
Ending a level

Once all players have left the level (by exiting or being expelled), the server updates the players about this fact.



Ending a Snarl game

When the last level is completed, the server updates the players about the end of the game and closes all connections.



JSON Message Definitions

server-welcome

A (server-welcome) is the following object

```
{ "type": "welcome",
  "info": (server-info)
}
```

- (server-info) is an implementation specific JSON string containing the server version information.

name

A (name) is a JSON string containing alpha-numeric characters, representing the player's registration name.

start-level

A (start-level) is a JSON object signalling the start of a level. It includes the number of the level as a (natural) and a list of active players.

```
{ "type": "start-level",
  "level": (natural),
  "players": (name-list)
}
```

player-update-message

A (player-update-message) is a JSON object containing a player update and an optional message from the server.

```
{ "type": "player-update",
  "layout": (tile-layout),
  "position": (point),
  "objects": (object-list),
  "actors": (actor-position-list),
  "message": (maybe-string)
}
```

A message can be a relevant piece of information the server wants to relay to the player. A good example would be messages of the form `"Player <name> <event>."`, where <event> is one of

- `"moved"`

- `"found the key"`
- `"was expelled"`
- `"exited"`
- `"disconnected"`

The client might want to display these messages to the human player.

Alternatively, a message can be `null`. Other fields are as defined in Milestone 7. In particular:

- `(tile-layout)` is a 5x5 2D JSON array of tiles within the player's view (see Milestone 3),
- `(object-list)` is an unordered list of `(object)`, which is the JSON object

  ```
  { "type": (object-type), "position": (point) }
  ```

  with `(object-type)` one of `"key"` or `"exit"` (originally defined implicitly in Milestone 4, but not named), and
- `(actor-position-list)` was defined in Milestone 5.

`player-move`

A `(player-move)` is `(actor-move)` from Milestone 7, that is, the following JSON object:

```
{ "type": "move",
  "to": (maybe-point)
}
```

A `(maybe-point)` is one of the following:

- `null` representing a skipped move
- `(point)` as defined in Milestone 3, representing an *absolute* position within the level.

`result`

A `(result)` is one of:

- `"OK"`, meaning "the move was valid, nothing happened"
- `"Key"`, meaning "the move was valid, player collected the key"
- `"Exit"`, meaning "the move was valid, player exited"
- `"Eject"`, meaning "the move was valid, player was ejected"
- `"Invalid"`, meaning "the move was invalid"

`end-level`

An `(end-level)` is the following JSON object:

```
{ "type": "end-level",
  "key": (name),
  "exits": (name-list),
  "ejects": (name-list)
}
```

The fields `"key"`, `"exits"` and `"ejects"` summarize who found the key, who exited and who was ejected by an adversary.

`end-game`

An **`(end-game)`** is the following JSON object:

```
{ "type": "end-game",
  "scores": (player-score-list)
}
```

A **`(player-score-list)`** is a JSON array of **`(player-score)`**, which is the following JSON object.

```
{ "type": "player-score",
  "name": (name),
  "exits": (natural),
  "ejects": (natural),
  "keys": (natural)
}
```

Each player registered in the game should have a **`(player-score)`** entry, listing the number of times they exited, were ejected and the number of times they found a key.