ClanBomber: Network Protocol v0.1

November 26, 2018

Johannes Posch (s1810567013)

Contents

1 Basic

1.1 Abstract

This document defines the transfer protocol between the server and the clients for the ClanBomber project. This project will be implemented as part of the Industrial Software Development class by Group 1.

1.2 General

In this protocol, the JSON format is used to transmit the data. The protocol defines that the client always sends the client packet (see chapter 2) to the server. The server responds with a server packet (see chapter 3). Figure 1 shows the sequence for one basic transaction.

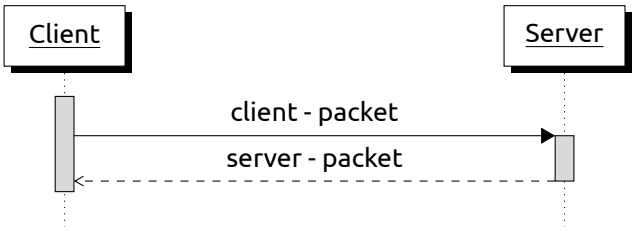

Figure 1: Shows one transaction between client and server

2 Client - Packet

The client - packet whose structure is described in Listing 1 is sent from the client to the server. The packet describes the player's input by a state value. The state also supports an idle state, which informs the server about the presence of the user so that the player continues to receive updates from the server.

```
{
  "state": <input-state>,
  "bomb": <bomb-state>
}
```

Listing 1: client – packet

2.1 Key definition

2.1.1 "state": [integer]

The "state" key describes the player's input. The $< input - state >$ placeholder is replaced by an integer - value with the meaning defined as follows:

- 0: (idle)
- 1: (move) up
- 2: (move) down
- 3: (move) left
- 4: (move) right

2.1.2 "bomb": [integer]

The "bomb" key describes whether a bomb should be placed or not. The $< bomb - state >$ placeholder is replaced by an integer - value with the meaning defined as follows:

- 0: (idle)
- 1: place bomb

2.2 Example

```
{
  "state": 1,
  "bomb": 0,
}
```

Listing 2: client – packet example

Listing 2 shows an example of a client packet that can be sent to the server. This packet transmits the information that the player has pressed the move up key.

3 Server - Packet

The server - packet whose structure is described in Listing 3 is sent from the server to the client. The packet describes the current state of the game.

```
{
  "id": <id>,
  "state": <game-state>,
  "timeout": <timeout-time>,
  "field": <field-definition>,
  "players": <players-definition>,
  "bombs": <bombs-definition>,
  "flames": <flames-definition>
}
```

Listing 3: server – packet

3.1 Key definition

3.1.1 “id”: [integer]

This key specifies the ID of the client alias of the player who receives this package. This identification number is created by the server when a new client connects. The $< id >$ placeholder is replaced by an integer - value.

```
{
  "id": 1
}
```

Listing 4: id example

3.1.2 “state”: [integer]

The “state” key indicates the current status of the game logic. This indicates whether the game logic is ready for a new game, whether a game is currently active, or whether the results are currently being announced. The three modes are described in chapter 4. The $< game - state >$ placeholder is replaced by an integer - value with the meaning defined as follows:

- 0: *pre-game-phase*; waiting for a new game
- 1: *in-game-phase*; actually running a game
- 2: *post-game-phase*; showing the results.

```
{
  "state": 1
}
```

Listing 5: state example

3.1.3 “timeout”: [integer]

Wenn der Timeout während der phase ausläuft und das Spiel noch nicht beendet wurde, dann wird das Spielfeld eingeschränkt bis ein Spieler übrig bleibt.
The “timeout” key outputs the remaining time for the current status. After this time has elapsed, the status of the game logic is changed. If the timeout expires during the *in-game-phase* state and the game has not yet ended, the playing field is restricted until one player remains. The $< timeout - time >$ placeholder is replaced by an integer - value which indicates the timeout in seconds.

```
{
  "timeout": 10
}
```

Listing 6: timeout example

3.1.4 “field”: [array(array(integer))]

The “field” key describes the composition of the playing field. The value of this key is a two-dimensional array, which has the size mxm. In Listing 7 the structure of the array is represented, where the value $m = 3$ is assumed.

```
{
  "field": [
    [<f>, <f>, <f>],
    [<f>, <f>, <f>],
    [<f>, <f>, <f>]
  ]
}
```

Listing 7: field specification

This two-dimensional array contains characters that represent which elements should be displayed on this field in the game. The $< f >$ placeholder is replaced by an integer - value, with the meaning defined as follows:

- 0: empty field
- 1: static; undestroyable block
- 2: dynamic; destroyable block
- 3: item; more bombs

```
{
  "field": [
    [0, 0, 0],
    [0, 1, 2],
    [0, 0, 3]
  ]
}
```

Listing 8: field example

3.1.5 “players”: [array(player)]

The “players” key describes the state and position of the players playing in the game. In the different states of game logic, the meaning of this key can be different. The different meanings can be found in the chapter 4. Listing 9 shows the structure of the array, whith two players.

```
{
  "players": [
    {"id": <id>, "x": <x>, "y": <y>, "live": <live>},
    {"id": <id>, "x": <x>, "y": <y>, "live": <live>}
  ]
}
```

Listing 9: players specification

A player is described by his id, his position and his life. In the following the meaning of a user’s keys is described:

- “id”: This is the identification number of a character. If this value and the value of the main - key “id” match, it is your own figure. The placeholder $< id >$ is replaced by an integer - value.
- “x”: This is the X coordinate of the player on the field. The placeholder $< x >$ is replaced by a float - value.
- “y”: This is the Y coordinate of the player on the field. The placeholder $< y >$ is replaced by a float - value.
- “live”: This is the remaining life of the player. With the value 0 the player is already eliminated. The placeholder $< live >$ is replaced by an integer - value.

```
{
  "players": [
    {"id": 0, "x": 3.2, "y": 36.0, "live": 5},
    {"id": 1, "x": 2.4, "y": 5.5, "live": 3}
  ]
}
```

Listing 10: players example

3.1.6 “bombs”: [array(bomb)]

The “bombs” key describes an array that defines the status and position of the bombs on the field. In Listing 11 the structure of the array with two bombs is shown.

```
{
  "bombs": [
    {"id": <id>, "x": <x>, "y": <y>, "timeout": <timeout>},
    {"id": <id>, "x": <x>, "y": <y>, "timeout": <timeout>}
  ]
}
```

Listing 11: bombs specification

A bomb is described by the id of the player who placed the bomb, its position and the timeout until the bombs is going to burst. In the following the meaning of a bomb’s keys is described:

- “id”: This is the identification number of the player, who placed the bomb. If this value and the value of the main - key “id” match, it is your own figure. The placeholder $< id >$ is replaced by an integer - value.
- “x”: This is the X coordinate of the bomb on the field. The placeholder $< x >$ is replaced by a float - value.
- “y”: This is the Y coordinate of the bomb on the field. The placeholder $< y >$ is replaced by a float - value.
- “timeout”: This is the remaining time until the bomb is going to burst in seconds. With the value 0 the bomb just detonated. The placeholder $< timeout >$ is replaced by an integer - value.

```
{
  "bombs": [
    {"id": 0, "x": 3.0, "y": 4.5, "timeout": 3},
    {"id": 0, "x": 3.9, "y": 2.0, "timeout": 4}
  ]
}
```

Listing 12: bombs example

3.1.7 “flames”: [array(flame)]

The “flames” key describes the position of the flames on the playing field. Listing 13 shows the structure of the array for two flames.

```
{
  "flames": [
    {"id": <id>, "x": <x>, "y": <y>, "timeout": <timeout>},
    {"id": <id>, "x": <x>, "y": <y>, "timeout": <timeout>}
  ]
}
```

Listing 13: flames specification

A flame is described by the id of the player who placed the bomb, its position and the timeout until the flame is gone. In the following the meaning of a flame’s keys is described:

- “id”: This is the identification number of the player, who placed the bomb. If this value and the value of the main - key “id” match, it is your own figure. The placeholder $< id >$ is replaced by an integer - value.
- “x”: This is the X coordinate of the flame on the field. The placeholder $< x >$ is replaced by a float - value.
- “y”: This is the Y coordinate of the flame on the field. The placeholder $< y >$ is replaced by a float - value.
- “timeout”: This is the remaining time until the bomb is going to burst in seconds. The value 0 will not be sent to the clients, because then the bomb is gone. The placeholder $< timeout >$ is replaced by a float - value.

```
{
  "flames": [
    {"id": 0, "x": 44.3, "y": 32.3, "timeout": 0.4},
    {"id": 1, "x": 2.0, "y": 55.3, "timeout": 0.6}
  ]
}
```

Listing 14: flames example

4 Gamemode

The game is played in three different modes.

1. *pre-game-phase*
2. *in-game-phase*
3. *post-game-phase*

The different modes are initiated or terminated by the server. The current mode of the game can be found in the "state" key of the server - packet. In the different modes the fields of the server - packet have different meanings. These are described here.

4.1 pre-game-phase

In the *pre-game-phase* the game mode waits for the players who want to join for the next round. During this phase, the main keys have the following meaning:

- "id": standard meaning
- "state": standard meaning
- "timeout": standard meaning
- "field": standard Meaning
- "players": In this array the players are returned who will play in the next round. The player values x,y,live have no meaning in this mode and will therefore not be sent
- "bombs": This key has no meaning and will therefore not be sent.
- "flames": This key has no meaning and will therefore not be sent.

Listing 15 shows an example package for this mode.

```
{
  "id": 1,
  "state": 1,
  "timeout": 0,
  "field": [
    [0, 0, 0],
    [0, 1, 2],
    [0, 0, 3]
  ],
  "players": [
    {"id": 1},
    {"id": 2}
  ]
}
```

Listing 15: server – packet example for *pre-game-phase*

Gamemode continued on next page…

4.2 in-game-phase

In the *in-game-phase* mode the game is currently active and executed. Clients who connect to the server in this mode will receive information about the game, but will not be able to play. This is also noticed by the fact that the ID of the player is not present in the "users" array of the server - packet. During this phase, the main keys all have the standard meaning. Listing 16 shows an example package for this mode.

```
{
  "id": 1,
  "state": 1,
  "timeout": 85,
  "field": [
    [0, 0, 0],
    [0, 1, 2],
    [0, 0, 3]
  ],
  "players": [
    {"id": 0, "x": 3.2, "y": 36.0, "live": 5},
    {"id": 1, "x": 2.4, "y": 5.5, "live": 3}
  ],
  "bombs": [
    {"id": 0, "x": 3.0, "y": 4.5, "timeout": 3},
    {"id": 0, "x": 3.9, "y": 2.0, "timeout": 4}
  ],
  "flames": [
    {"id": 0, "x": 44.3, "y": 32.3, "timeout": 0.4},
    {"id": 1, "x": 2.0, "y": 55.3, "timeout": 0.6}
  ]
}
```

Listing 16: server – packet example for *in-game-phase*

4.3 post-game-phase

In the *post-game-phase* mode, the winner statistics of the last game are displayed. As in the *in-game-phase*, the following applies here Mode that players can connect but do not show up here yet. During this phase, the main keys have the following meaning:

- "id": standard meaning
- "state": standard meaning
- "timeout": standard meaning
- "field": standard Meaning
- "players": In this array the players are returned sorted by their rank. The player values x,y,live have no meaning in this mode and will therefore not be sent
- "bombs": This key has no meaning and will therefore not be sent.
- "flames": This key has no meaning and will therefore not be sent.

Listing 17 shows an example package for this mode.

Gamemode continued on next page…

```
{
  "id": 1,
  "state": 1,
  "timeout": 85,
  "field": [
    [0, 0, 0],
    [0, 1, 2],
    [0, 0, 3]
  ],
  "players": [
    {"id": 0},
    {"id": 1}
  ]
}
```

Listing 17: server – packet example for *post-game-phase*

5 Coordinate System

The playing field is roughly defined by the “field” array from the server - packet. The first index of the two-dimensional array defines the row in the field and the second defines the column in that row. The origin of the field is in the upper left corner of the playing field. Figure 2 shows the definition of the coordinate system for the playing field. The small numbers in the image define the position in the playing field when the position is passed as x,y coordinate.

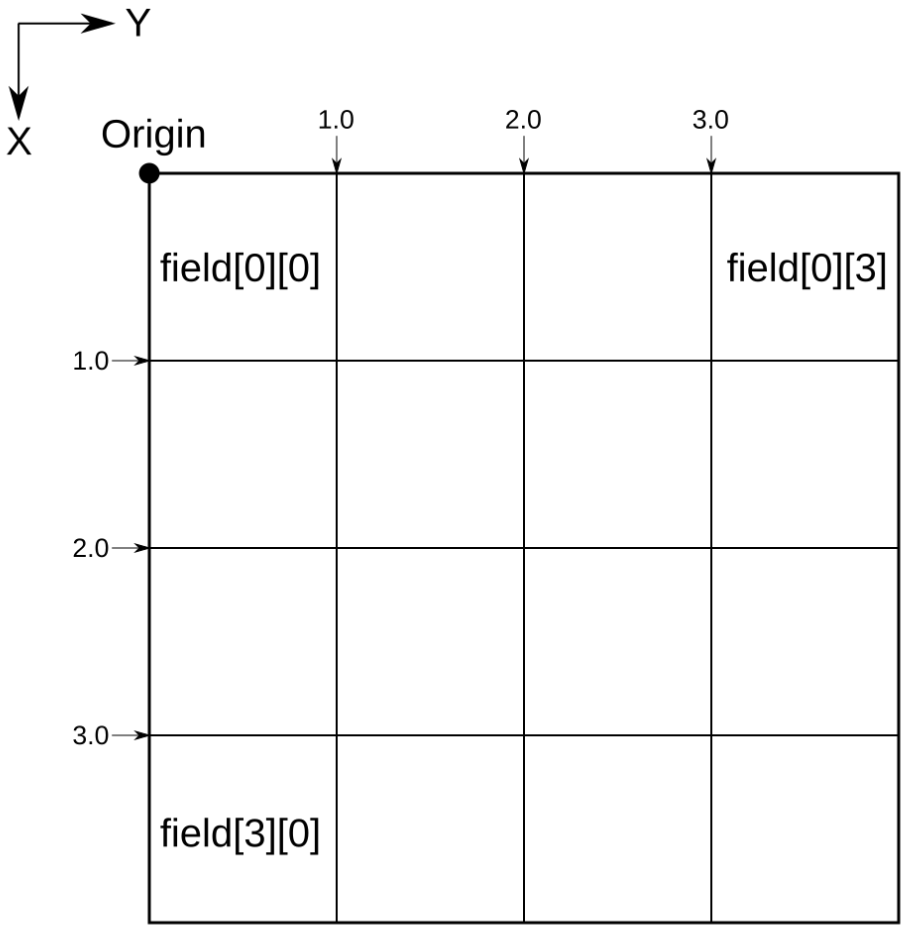

Figure 2: Playingfield coordinate definition