BattleThreads (offline variant)

Jens Lechtenbörger, Justus Rotermund

May 2020

1 Background

This document describes rules for the game BattleThreads suggested in [Hil+03], aiming for the following learning objectives:

- Explain distinctions between threads and processes
- Explain advantages of a multithreaded organization in structuring applications and in performance

The gameplay of BattleThreads is similar to the one of the famous Battleship, where two opponents each place a number of ships in their own territory and then shoot into the opponent's territory to destroy the opponent's ships.

Students play in teams, where opposing teams follow different rules. One team acts as single process with threads sharing their data structures, while the other team coordinates as set of isolated processes.

2 Playing Field and Goal

Each team's territory is defined by a grid of 8x8 fields. Rows are identified by letters (A-H), columns by numbers (1-8). Each team places 5 ships (two ships occupying 4 fields, three occupying 3 fields), either vertically or horizontally, but not diagonally. A sample territory can be seen in Table 1.

Table 1: Territory with positioned ships

	1	2	3	4	5	6	7	8
A		X	X	X		X		
B	X					X		
C	X					X		X
D	X							X
E								X
F	X	X	X	X				X
G								
H								

Each team aims to destroy all opponent's ships, where a ship is considered to be destroyed when it has been hit at every field. The game is over as soon as all ships of one team are destroyed. The team with no ships left loses, the other team wins.

3 General Remarks

Sheets of paper serve as data structures, which record positions of own ships and knowledge about the opponent's territory.

Whereas a team with multiple threads uses a shared data structure to keep track of joint knowledge, a team with multiple processes must communicate to exchange knowledge. During the game this implies that a team of threads uses one sheet of paper, which all team members can read and write. In contrast, in a team of processes each player has his or her own sheet of paper and explicit communication is necessary to inform team members about relevant events.

While an OS isolates processes from each other, during the game players need to make sure that other processes do not spy on their data structures. Thus, neither players from different teams nor players within a team of processes must be able to look at each other's data structures.

This game exists in two variants, either offline with sheets of paper or online with files. The paper-based variant requires that teams sit next to each other, while the file-based variant requires online access for all team members who can be located anywhere.

These instructions describe the *offline* variant with *sheets of paper*.

3.1 Preparation

Build teams (4 players or more; your instructor may provide more information how to assign students to teams, processes, and threads). Agree on an order in which to shoot later on.

Each team of threads chooses one player as *communicator*. The communicator is that thread to whom processes turn to communicate shots and their outcome.

As processes are isolated from each other during the game, it makes sense that members of process teams agree on a strategy ahead of time. (E.g., does everyone take random shots? What to do in case of hits?)

3.1.1 Paper-Based Variant

Your instructor will provide sheets of paper to play the game in a lecture hall.

3.1.2 Online Variant

You need to create files for data structures before you can begin to play, and different preparations for the two types of teams are necessary.

The following data structures are necessary:

- Team with independent processes
  - Each player receives a private, blank sheet of paper to communicate shots into opponent's territory.
  - Each player receives a private sheet of paper with two grids.
- Team with threads within single process
  - The team receives a shared sheet of paper with two grids.

Gather with all players of both teams in an online space (e.g., group chat or audio/video conference). Decide this ahead of time.

3.2 General gameplay

The game is played in rounds, where each player shoots once per round. The different rules for teams with multiple processes and teams with multiple threads are described in the following sections.

The team with multiple processes starts the game by taking the first shot as explained in the next section. (This choice has no special impact on the game but answers the otherwise open question which team should start.)

4 Rules for Teams with Multiple Processes

4.1 Preparation

1. Each player receives a private, blank sheet of paper to communicate shots into opponent's territory.

2. Each player receives a private sheet of paper with two grids.

   This data structure contains two empty grids, one to record the positions of own ships as well as shots taken by opponents ("own ships"), the other one to record information about opponents' ships ("their territory") during the game.

   This sheet of paper represents data structures that are local to the individual process/player. Therefore, it is important to make sure that no other player can read this sheet of paper.

3. As team, place the 5 ships as described above. Choose seats such that opponents cannot see each other's sheets of paper. Each player marks their positions on his or her own grid "our ships". The opponents must not learn the result.

4.2 Gameplay

4.2.1 First Round (each player shoots once)

1. A player shoots once. For this, the shooter indicates the target field (for example G7) in the sheet of paper that is shared with the communicator thread of the other team.

   As players simulate isolated processes, no other member of the shooter's team may know at what field the player shoots.

2. The communicator thread indicates the outcome of the shot, e.g., "m" (miss), "h" (hit), or "d" (ship destroyed).

   No other member of the shooter's team may know whether the shot was a hit or a miss.

3. The player marks the success of the shot on his or her own grid "their territory".

4. It is now the other team's turn to shoot. Every process records the announced shot in the grid "our ships" (to keep track of what happened and, in particular, to be able to detect whether a ship is destroyed), an arbitrary team member may communicate the outcome.

5. After the other team finished their turn, the next player in the team continues with step 1 until each player has shot once.

4.2.2 Following Rounds (after each player shot once)

1. The shooter can choose whether to shoot or to communicate his or her grid "their territory" with the other team members (processes). This choice is reflected by following the instructions of either (a) or (b):

   (a) The shooter decides to take another shot and follows the instructions for the first round.

   (b) The shooter loudly announces to communicate the status of the grid "their territory" and proceeds to step 2.

2. The shooter announces the outcome of his or her previous shots. The other team members copy this information onto their grids "their territory". The opposing team's members may hear this as well, since only facts already known to them are revealed.

3. It is now the other team's turn.

4. After the other team finished their turn, the next player in the team continues with step 1, again with a decision to shoot or communicate.

5 Rules for Teams with Multiple Threads

5.1 Preparation

1. Choose one player as *communicator*. The communicator is a thread to whom processes turn to communicate about shots and their outcome. The communicator can also take shots just as any other thread.

2. The team receives a shared sheet of paper with two grids.

   This data structure contains two empty grids, one to record the positions of own ships and shots taken by opponents ("own ships"), the other one to record information about opponents' ships ("their territory") during the game.

   This sheet of paper represents data structures that are shared by all threads within one team but are protected from other teams. Therefore, it is important that no player of one team can see the other team's territory.

3. As team, place the 5 ships as described above by marking their positions on the grid "own ships". The opponents must not learn the result.

5.2 Gameplay

1. A player shoots once. For this, the shooter communicates the target field to the other group (for example G7).
2. An arbitrary member of the other team indicates whether the shot was a miss, a hit, or it destroyed a ship. The shooter records this information by writing down "m" (miss), "h" (hit), or "d" (ship destroyed) in the grid "their territory".
3. It is now the other team's turn.
4. A shooter of the opposing team passes a target location to the communicator thread who checks the outcome and replies with "m" (miss), "h" (hit), or "d" (ship destroyed). Also, the communicator marks the location in the grid "our ships" (to keep track of what happened and, in particular, to be able to detect whether a ship is destroyed).
5. After the other team finished their turn, the next player in the team continues with step 1.

6 Sample Data Structures after 15 Shots

6.1 Player of Process Team

Table 2: A process' local data structure

	1	2	3	4	5	6	7	8
A	m		hh	h	m	h	m	m
B				mm				
C								
D				m				
E					m			
F								
G								
H								

The processes communicated their local views on "their territory" three times (only 12 shots were taken). The processes took shots at random and communicated as soon as a hit was landed (except for the two processes hitting on A3, only the first one communicated). The others searched around the hit, however several shots were taken twice. The result is shown in Table 2.

6.2 Thread Team

The threads were able to completely destroy two ships. The general strategy was to target a grid's diagonal and search around one hit for other hits. The result is shown in Table 3.

Table 3: Threads' data structure

	1	2	3	4	5	6	7	8
A						h		m
B						h	m	
C					m	h		
D					m	m		
E				m				
F	h	h	h	h				
G		m	m					
H								

[Hil+03] John M. D. Hill et al. "Puzzles and Games: Addressing Different Learning Styles in Teaching Operating Systems Concepts". In: *SIGCSE Bull.* 35.1 (Jan. 2003), pp. 182–186. ISSN: 0097-8418. DOI: 10.1145/792548.611964. URL: https://dl.acm.org/citation.cfm?doid=792548.611964.

License Information