System Programming
Programming HW 2

TAs

ntucsiesp@gmail.com

IRLAB

Outline

- Goal
- Problem Description
- Execution Flow
- Tasks
- Grading
- Submission
- Punishment
- Reminder

Goal

- In this assignment, you are required to…
  - Understand how to use fork() and exec() to create and execute new process.
  - Understand how to use pipe and FIFO to communicate between processes.
    - mkfifo(), pipe()

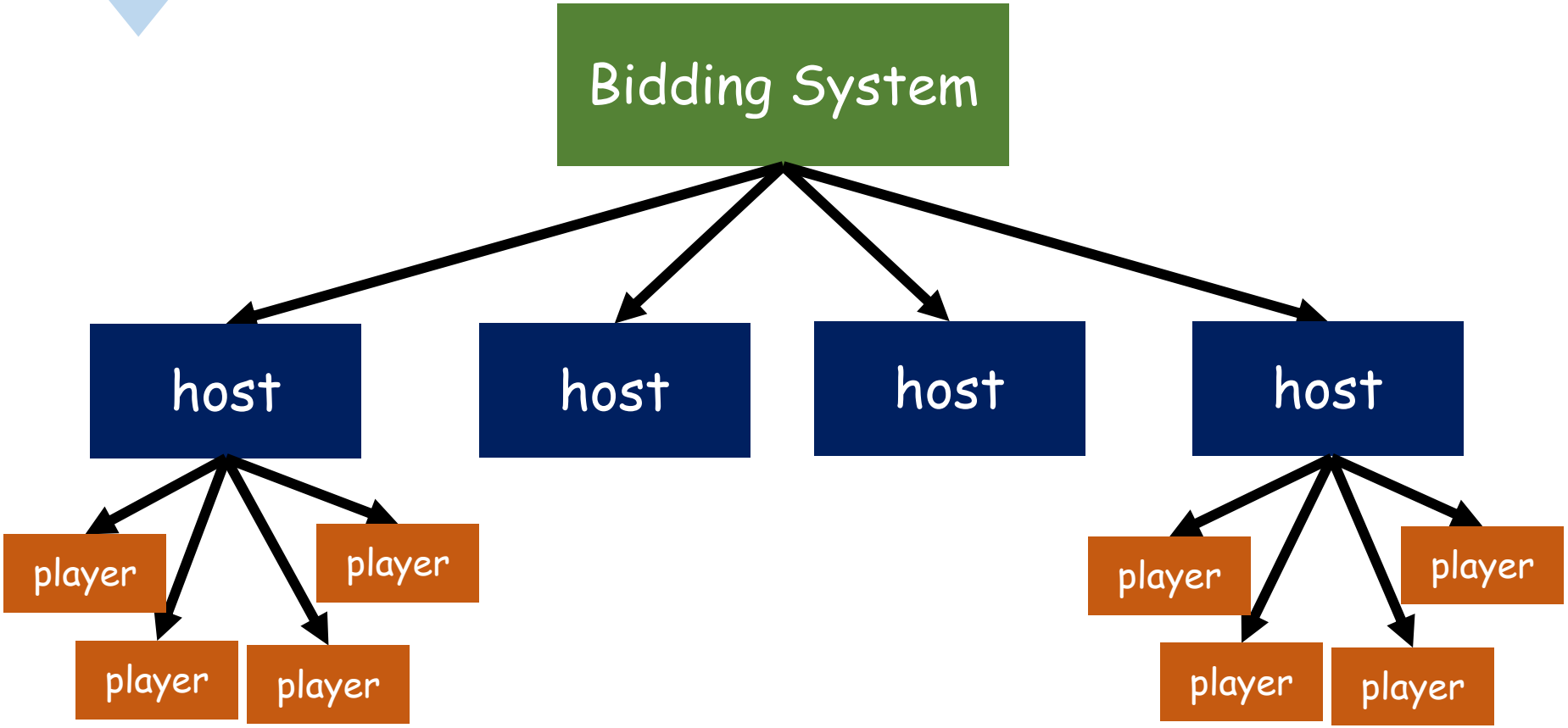
Problem Description

- Implement a bidding system which will handle a sequence of competitions.
- A Bidding system to handle the competitions
- Several hosts to hold these competition.
- Many players join in a competition.

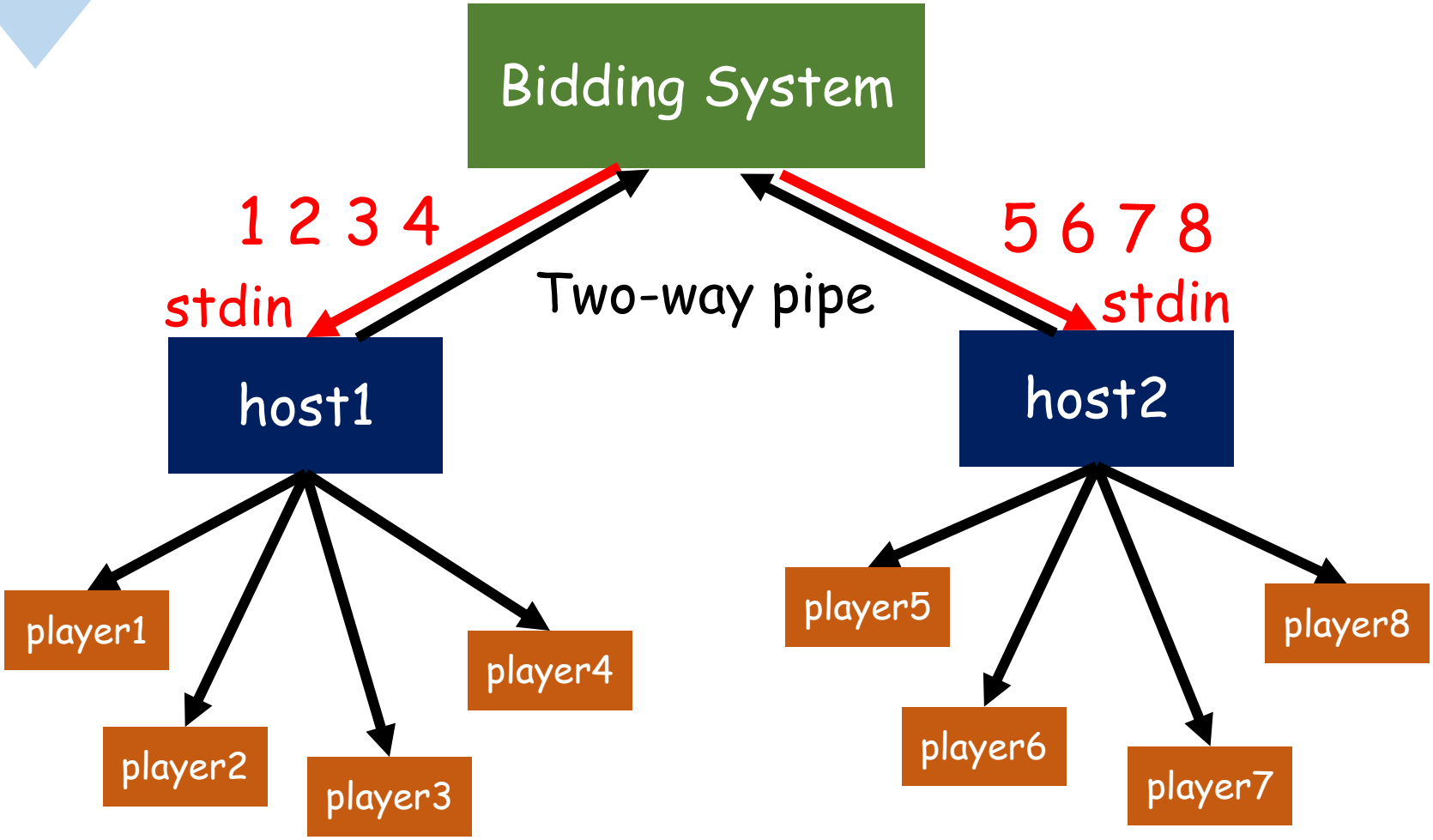
Problem Description

- We can assign host number and player number to the Bidding System.
- There are C(player_num, 4) competitions.
- A competition will hold 10 rounds.
- At the beginning of each round, host will tell how much do they have(as well as others).
- Each round every player will get 1000 dollars.
- The largest and unique announcement will win the item.

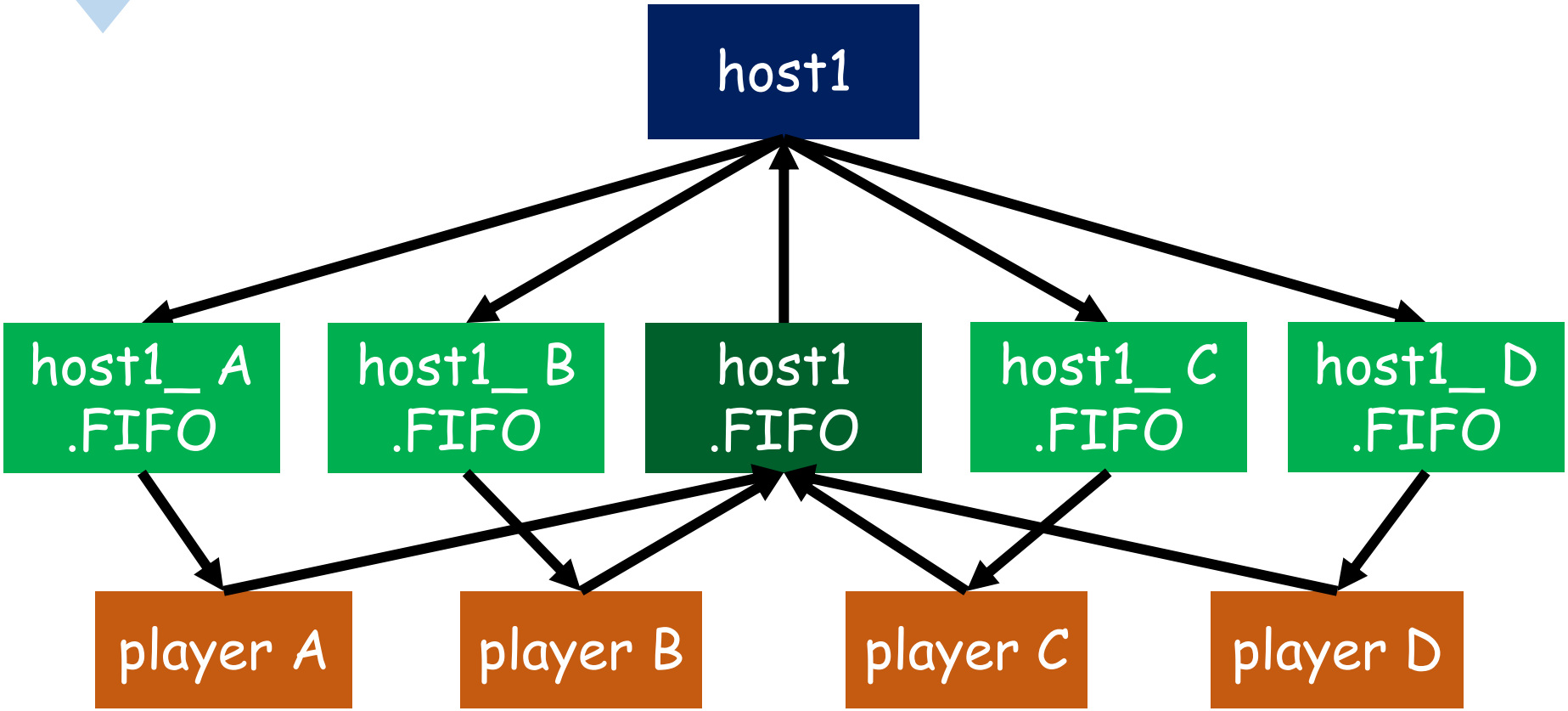
Bidding System Structure



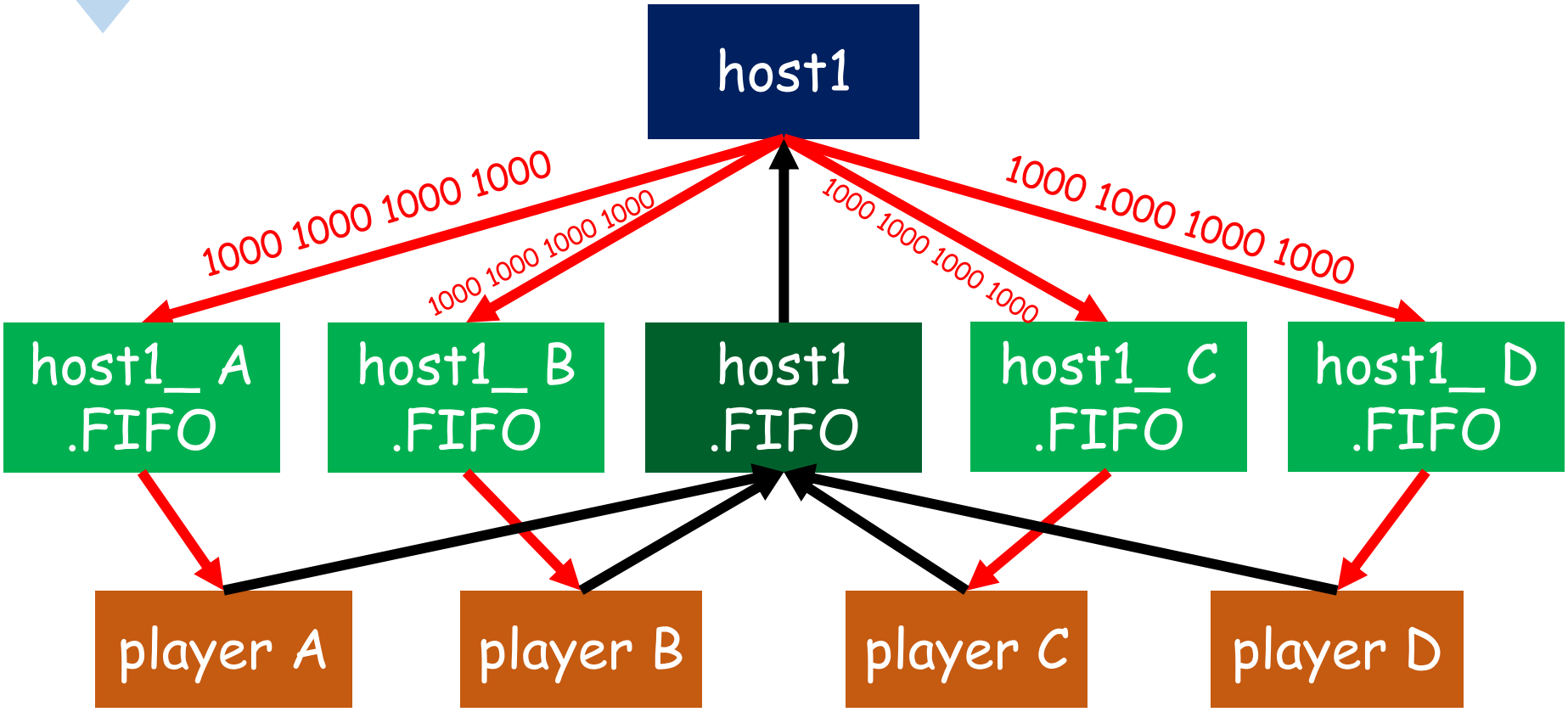
Execution Flow



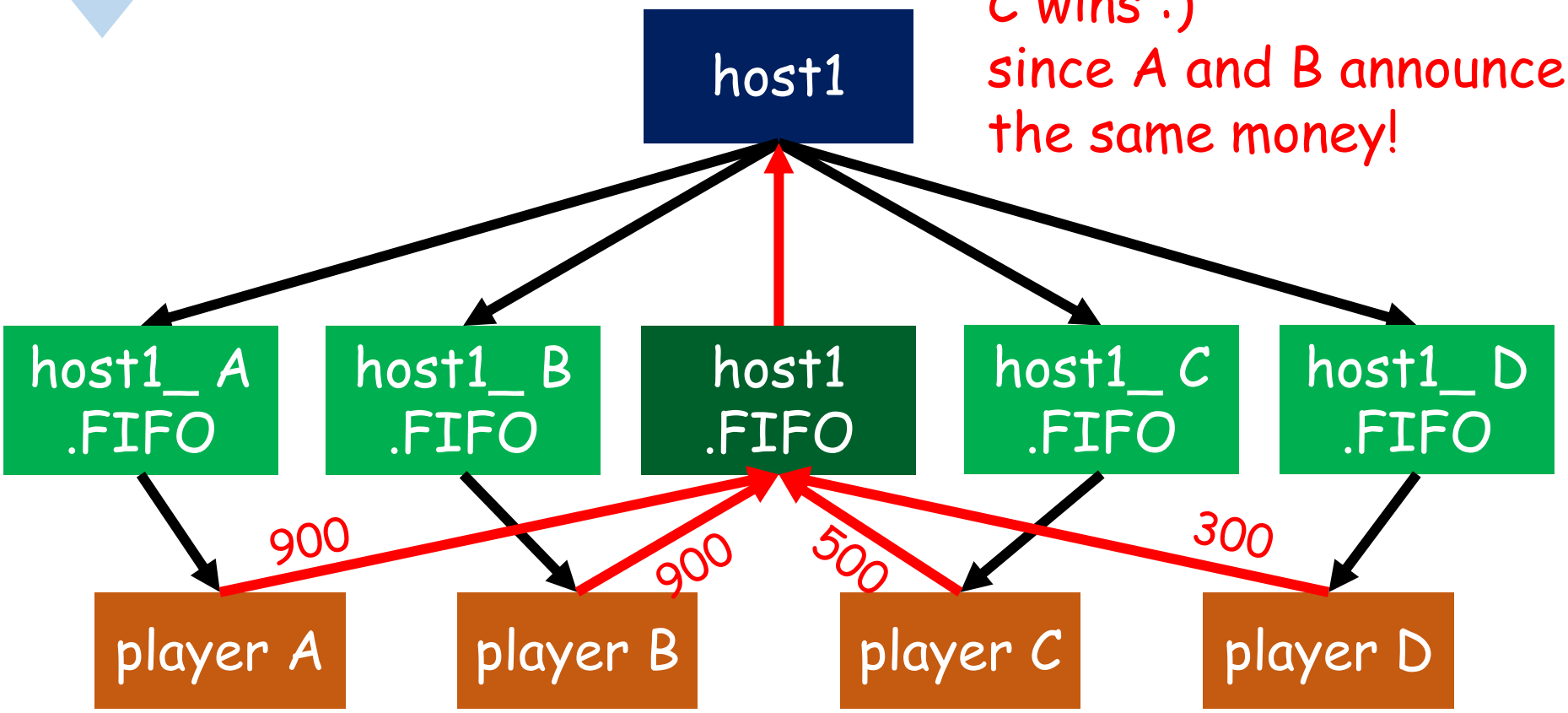
Execution Flow



Execution Flow



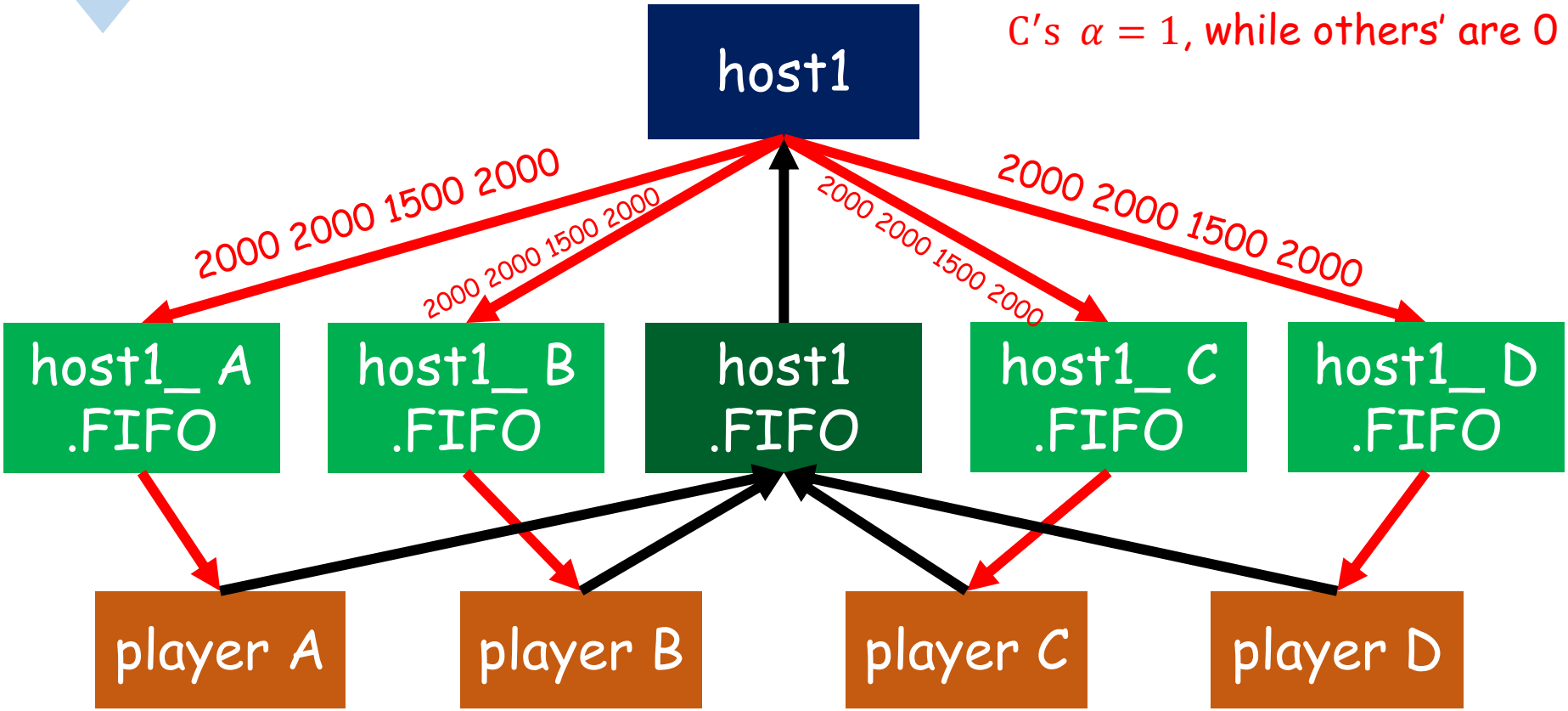
Execution Flow



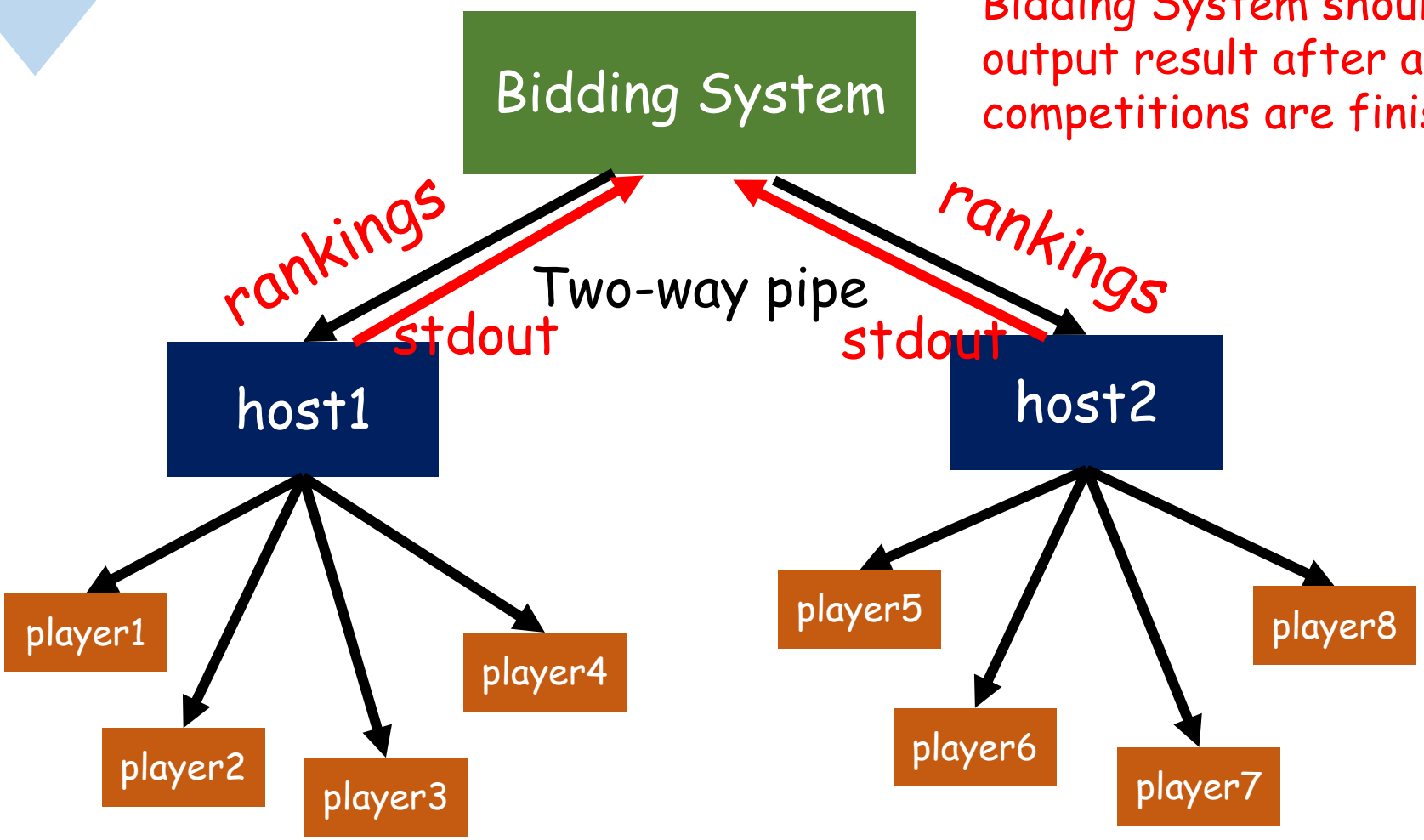
Execution Flow

$$money_{next} = 1000 + money_{prev} - \alpha \times pay$$

C's $\alpha = 1$, while others' are 0



Execution Flow



Tasks

- ./bidding_system [host_num] [player_num]
  - Create host_num hosts with id = 1 ~ host_num.
  - Assign every 4 players(id =1~player_num) to host.
    - Separate with space, id in ascending order
    - [player_id] [player_id] [player_id] [player_id]\n
  - Communicate with host through pipes.
    - Remember do redirection before you execute host

Tasks

- ./bidding_system [host_num] [player_num]
  - Once all competitions are finished, send

    "-1 -1 -1 -1\n" to all hosts.
  - Output result
    - Separate with space and put \n in the end.
    - 1 [ranking]\n

      2 [ranking]\n

      ⋮

      player_num [ranking]\n

If they got 7, 10, 3, 7, 3, then
1 2\n
2 1\n
3 4\n
4 2\n
5 4\n

Hint

```
//Build two-way pipe

pid = fork();

if(pid == 0){
    // child process
    // Redirection
    // exec()
}
else{
    //parent process
    //record pipe_fd
}
```

Tasks

- ./host [host_id]
  - Get 4 player id from standard input.
  - Create 5 FIFOs
    - host[host_id]_A.FIFO, host[host_id]_B.FIFO, host[host_id]_C.FIFO, host[host_id]_D.FIFO, host[host_id].FIFO
  - Stop when receiving "-1 -1 -1 -1\n".

Tasks

- ./host [host_id]
  - Send how much do they have at the beginning of each round.
  - After 10 rounds, output rankings to standard output
    - [playerA_id] [ranking]\n
      [playerB_id] [ranking]\n
      [playerC_id] [ranking]\n
      [playerD_id] [ranking]\n

Tasks

- ./player [host_id] [player_index] [random_key]
  - player_index = {A, B, C, D}
  - Random_key = [0, 65535]
  - Announce should be of the format
    - Separate with space
    - [player_index] [random_key] [money]\n

Tasks

- ./player [host_id] [player_index] [random_key]

  pay all their money in turn

  - A->B->C->D->A…
  - Round1 A pays 1000, round2 B pays 2000, …

Grading

- 1. Your bidding_system works fine.(1pt)
- 2. Your bidding_system schedules host effectively.(0.5pt)
- 3. Your bidding_system executes host correctly.(0.5pt)
- 4. Your host works fine.(2pt)
- 5. Your player works fine.(1pt)
- 6. Completeness.(1.5pt)
- 7. Produce executable files successfully.(0.5pt)

Grading

- TA's bidding_system, host, player
- Student's bidding_system, host, player
- We will judge your code in the following way
  - bidding_system + host + player
  - bidding_system + host + player
  - bidding_system + host + player
- So please follow the SPEC carefully

Submission

- Submit SP_HW2_{student_id}.tar.gz to CEIBA
- All the following files should be in the folder named {student_id}
  - 1. bidding_system.c
  - 2. host.c
  - 3. player.c
  - 4. Makefiles(as well as other *.c)
  - 5. readme.txt

Punishment

- You will get NO credits if plagiarism
- Late submission
  - 5% for each day
- Error format
  - wrong file name/format
  - wrong output format

Reminder

- Please read SPEC files carefully.
  - We will strictly follow the SPEC
  - including input/output format
- Test your code on the CSIE workstation
- Start your work ASAP and do not leave it until the last day

Reminder

- Use ssh to access from linux?.csie.ntu.edu.tw
- Use scp to copy files from linux?.csie.ntu.edu.tw

Q&A