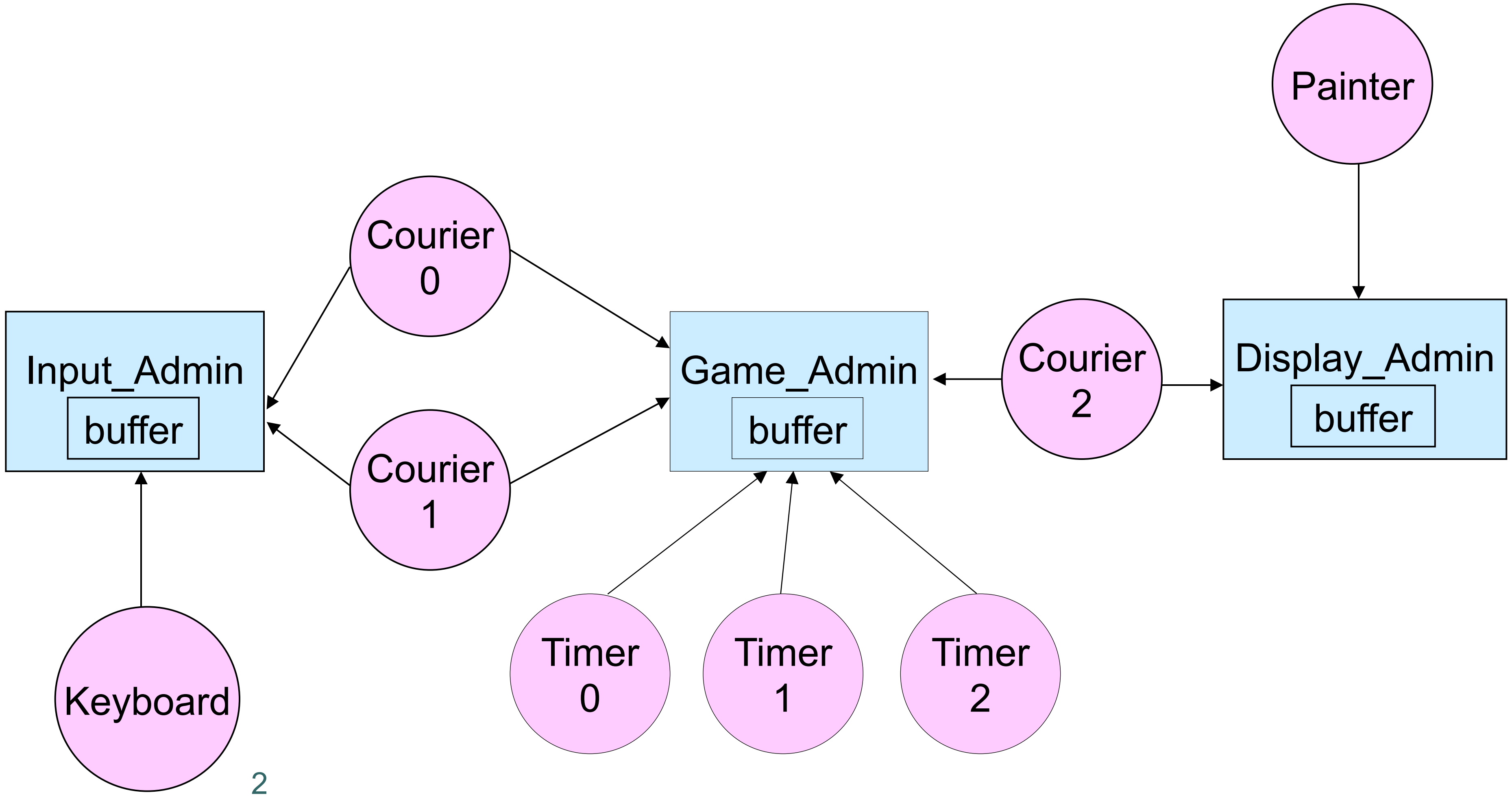
ESTR3106 Principles of Programming Languages

Administrator and Worker
More on Project

Tutorial 3

Processes Design



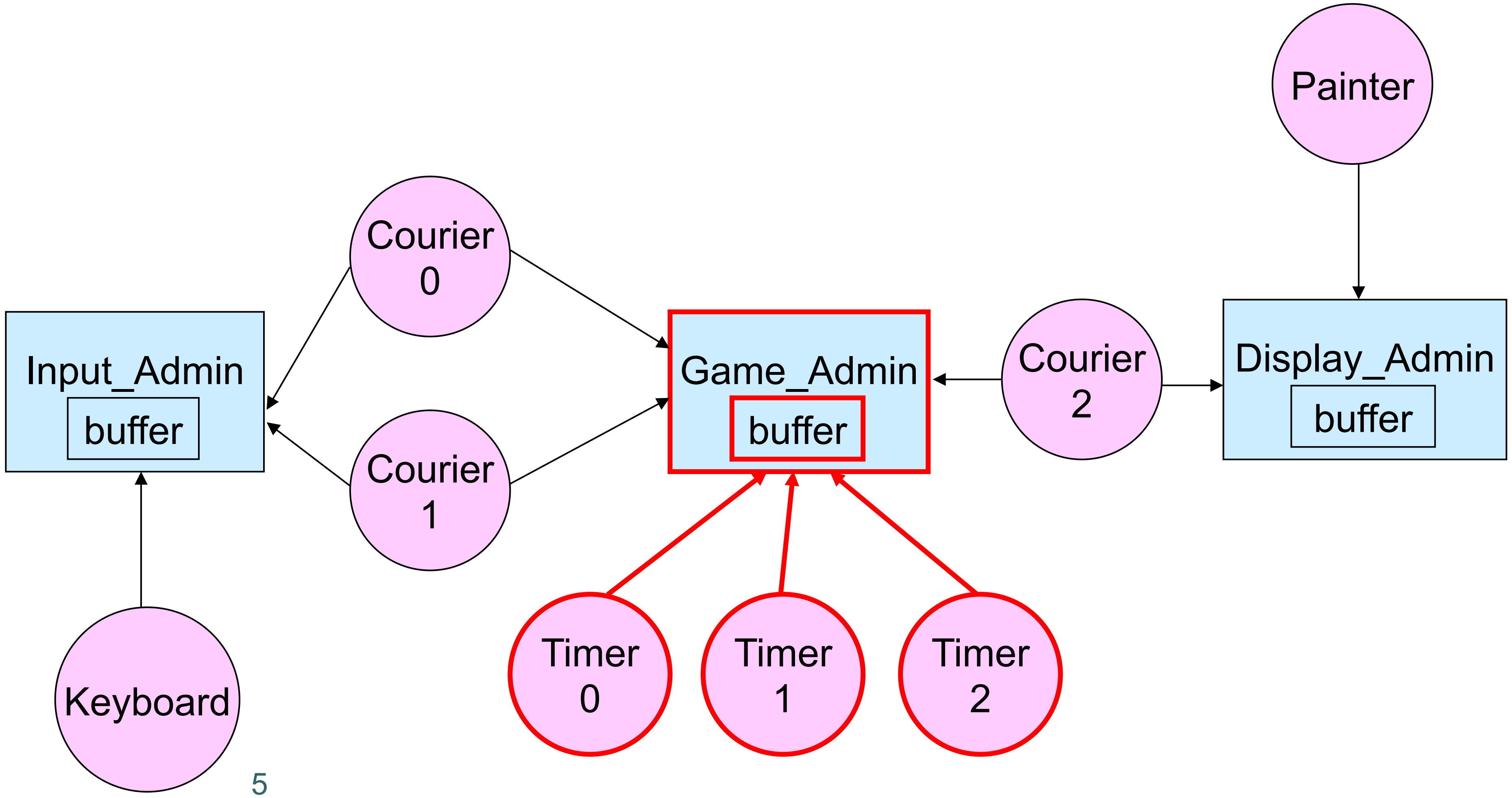
What should be done?

- Administrators
  - Game_Admin – maintains the rules of the game, the state of the arena.
  - Display_Admin – maintains output screen
  - Input_Admin – maintains human player’s control
- Workers
  - Timer – sleeps for a time interval
  - Courier – relays messages
  - Painter – paints the output to screen
  - Keyboard – gets human inputs from keyboard

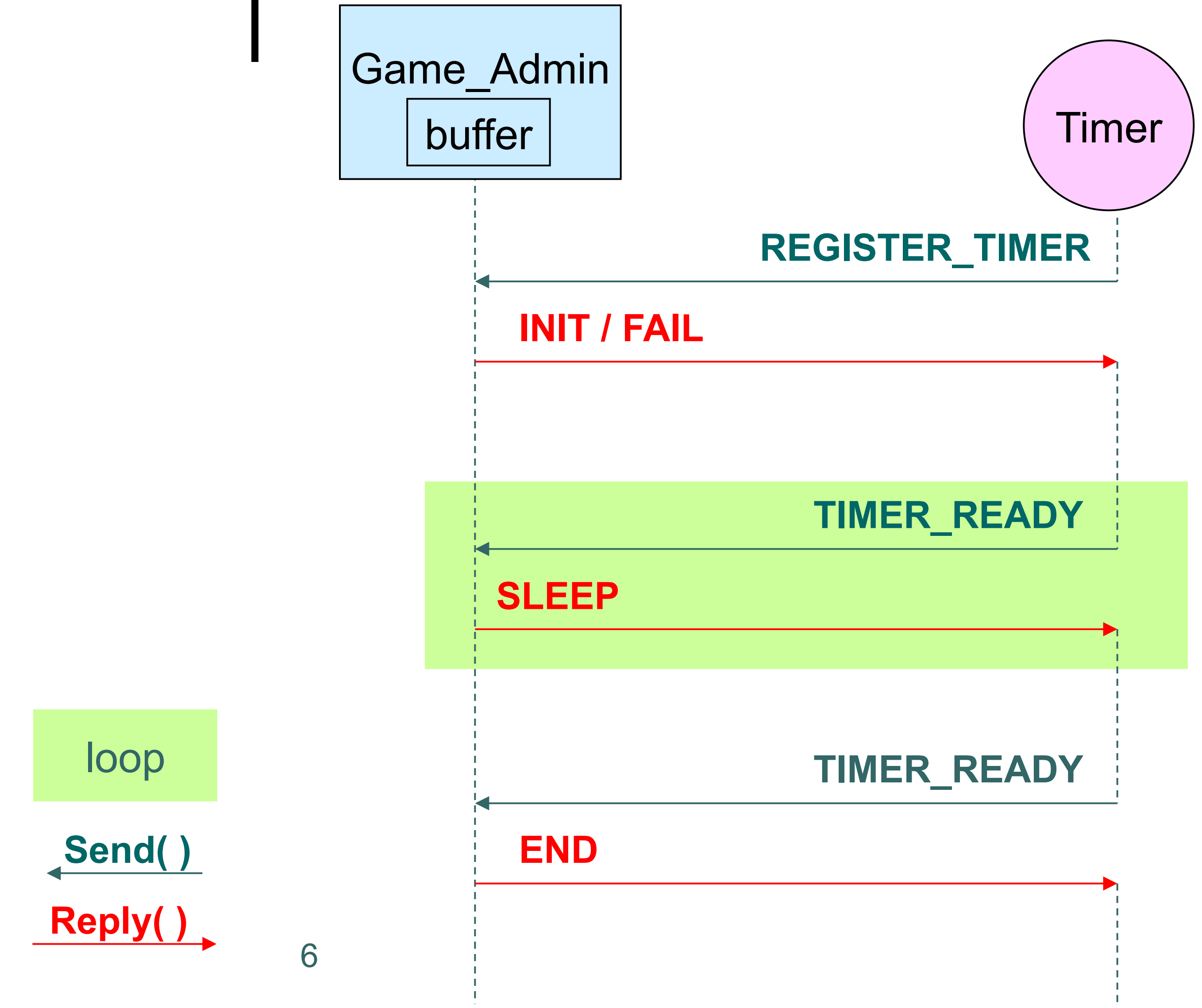
Three Stages

- Registration:
  - workers send message to administrators
  - couriers register to both administrator
- Working (Main Loop):
  - workers and administrators cooperate with each other
- Finish
  - game admin propagate ending messages to other processes

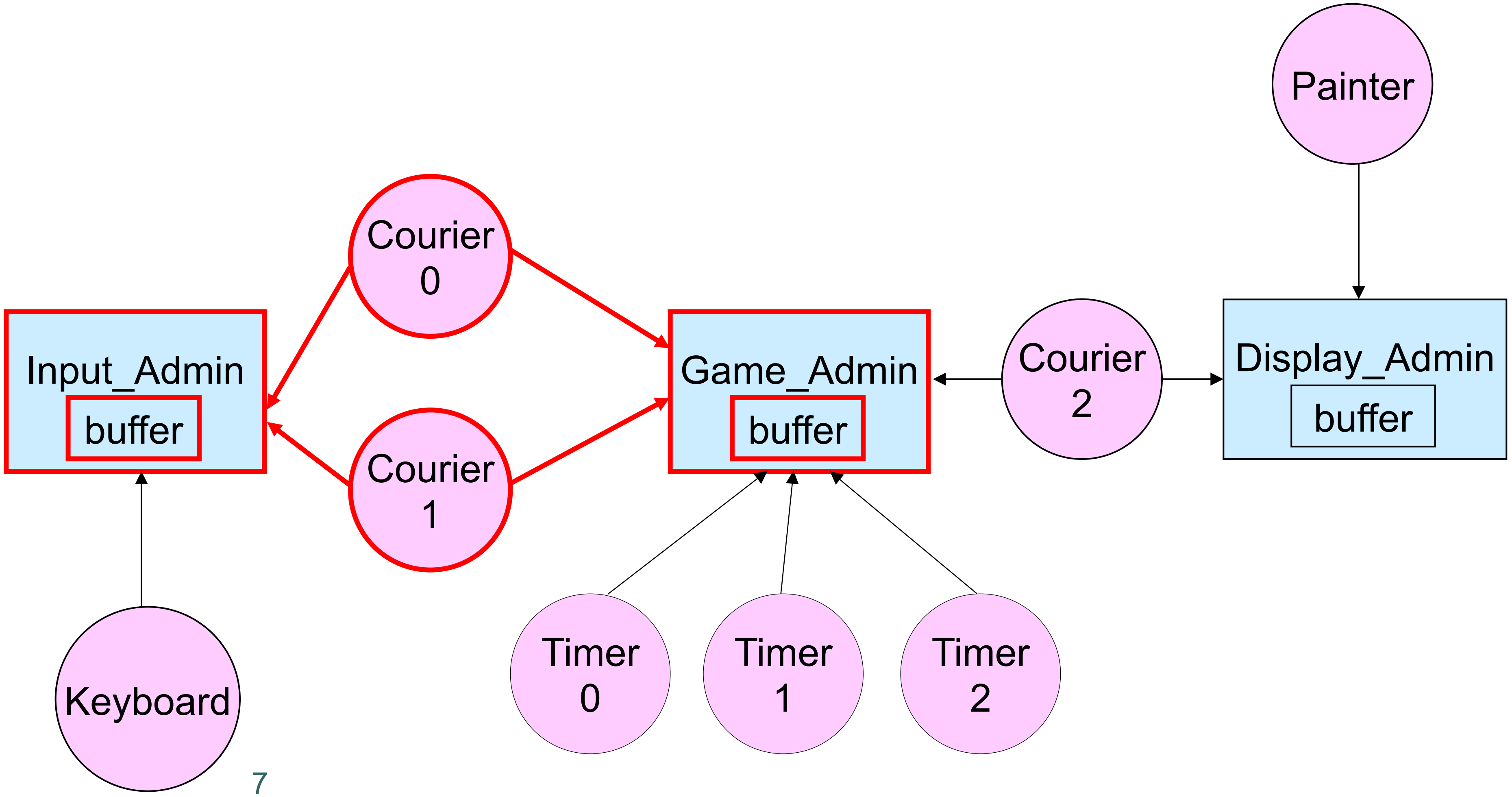
Processes Design



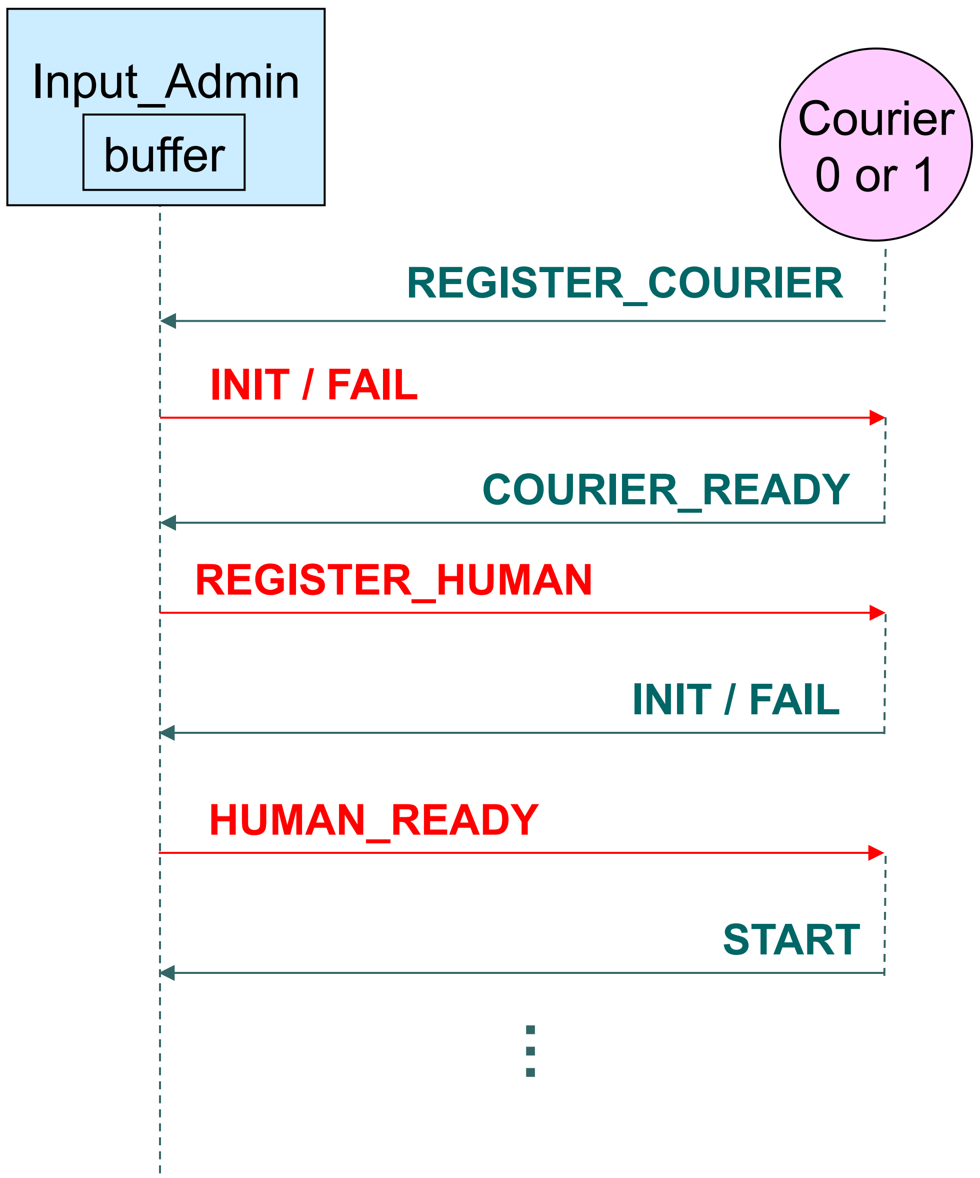
Game_Admin and Timer

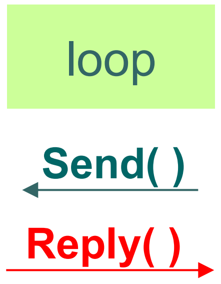


Processes Design

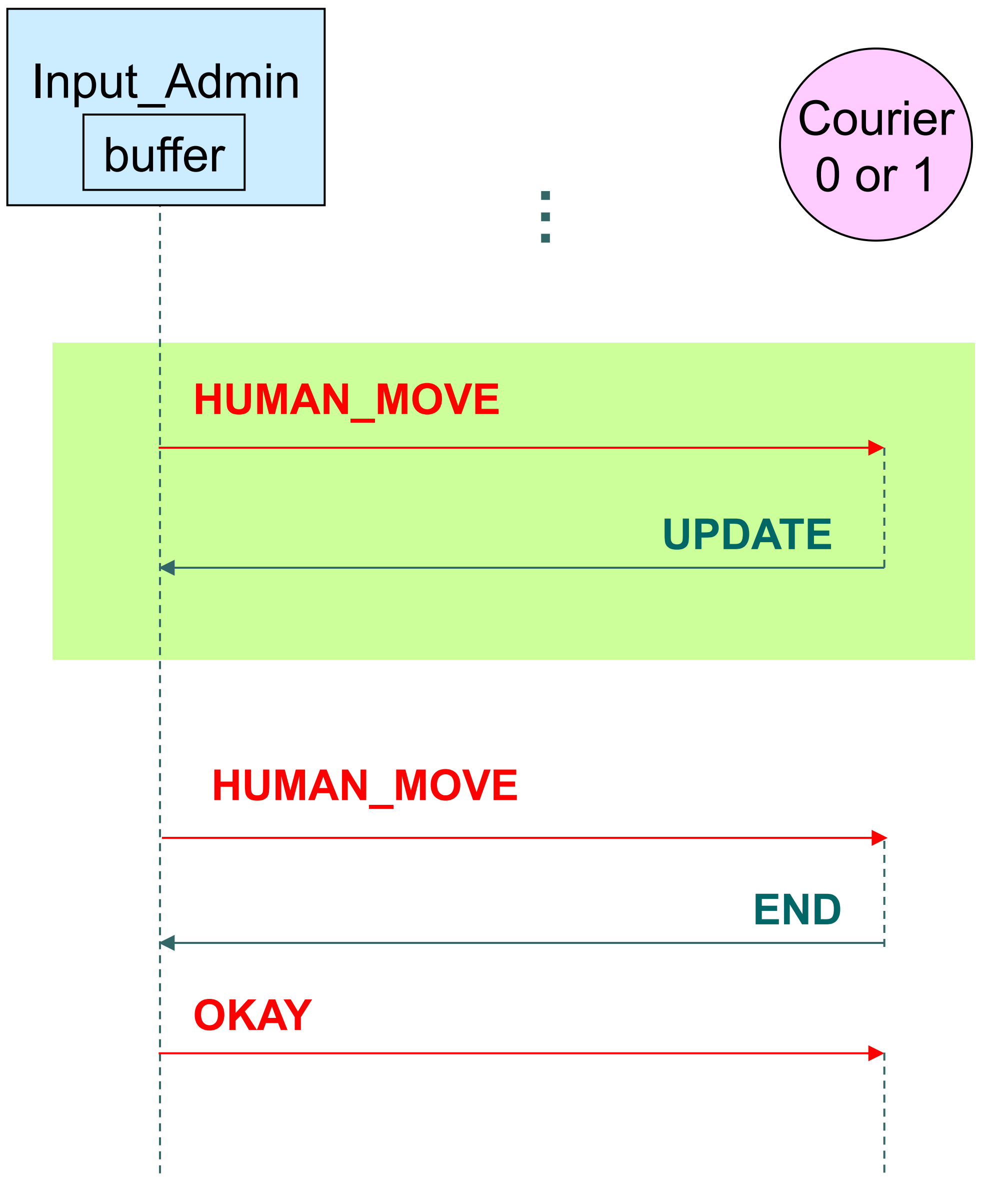


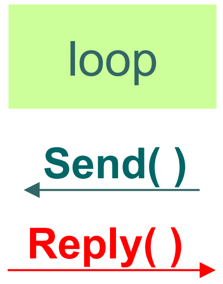
Input_Admin and Courier 0/1



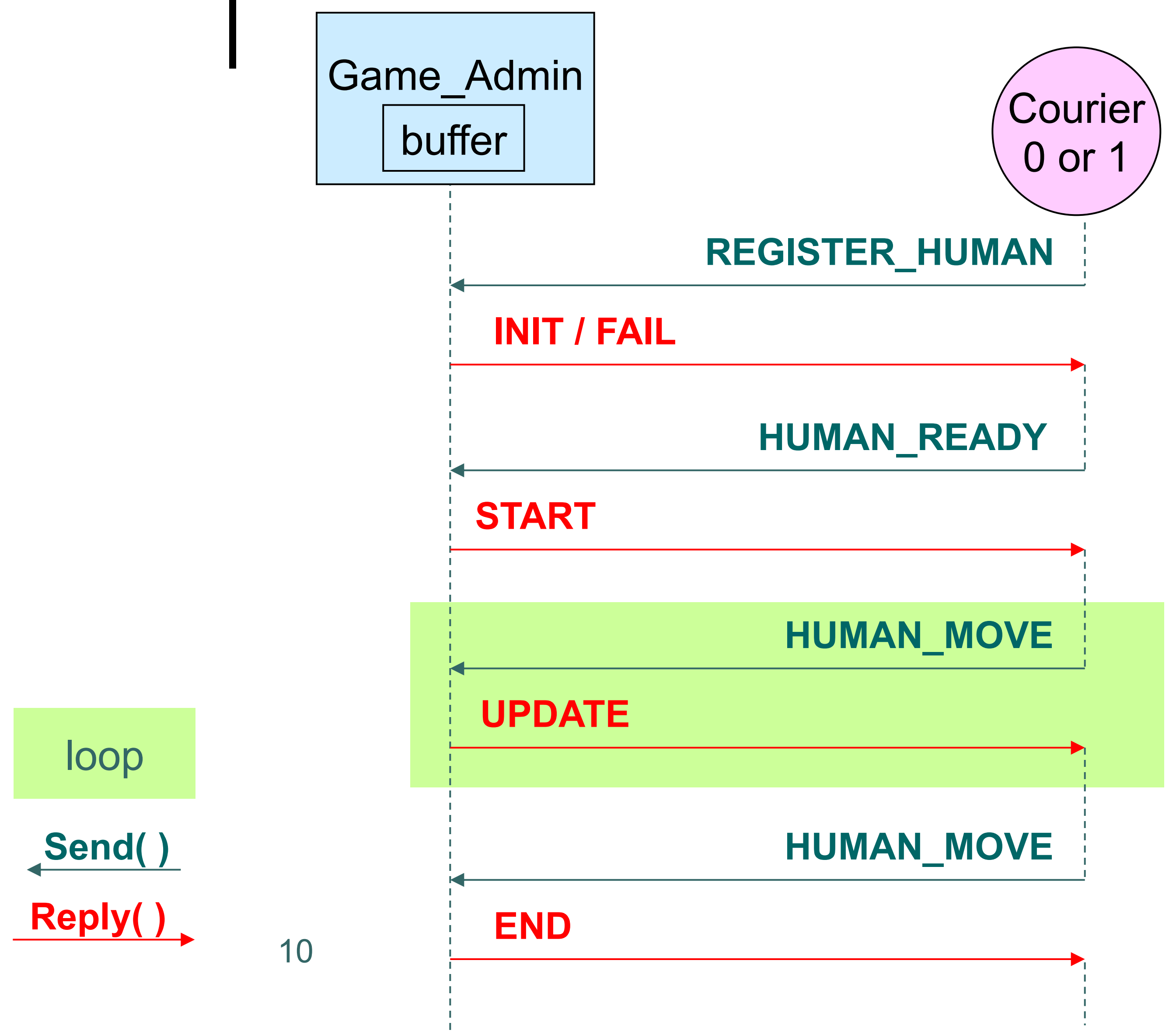


Input_Admin and Courier 0/1

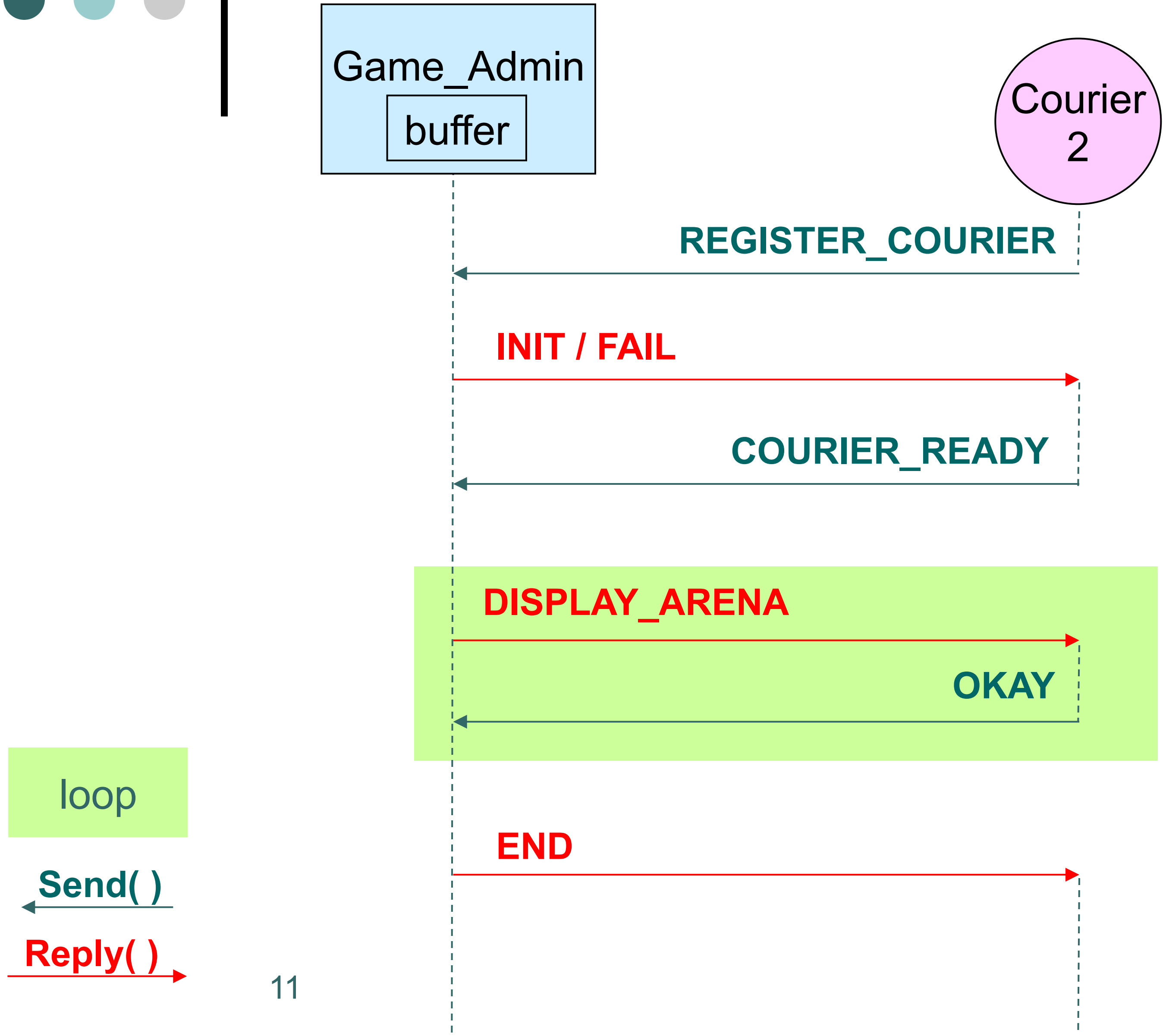




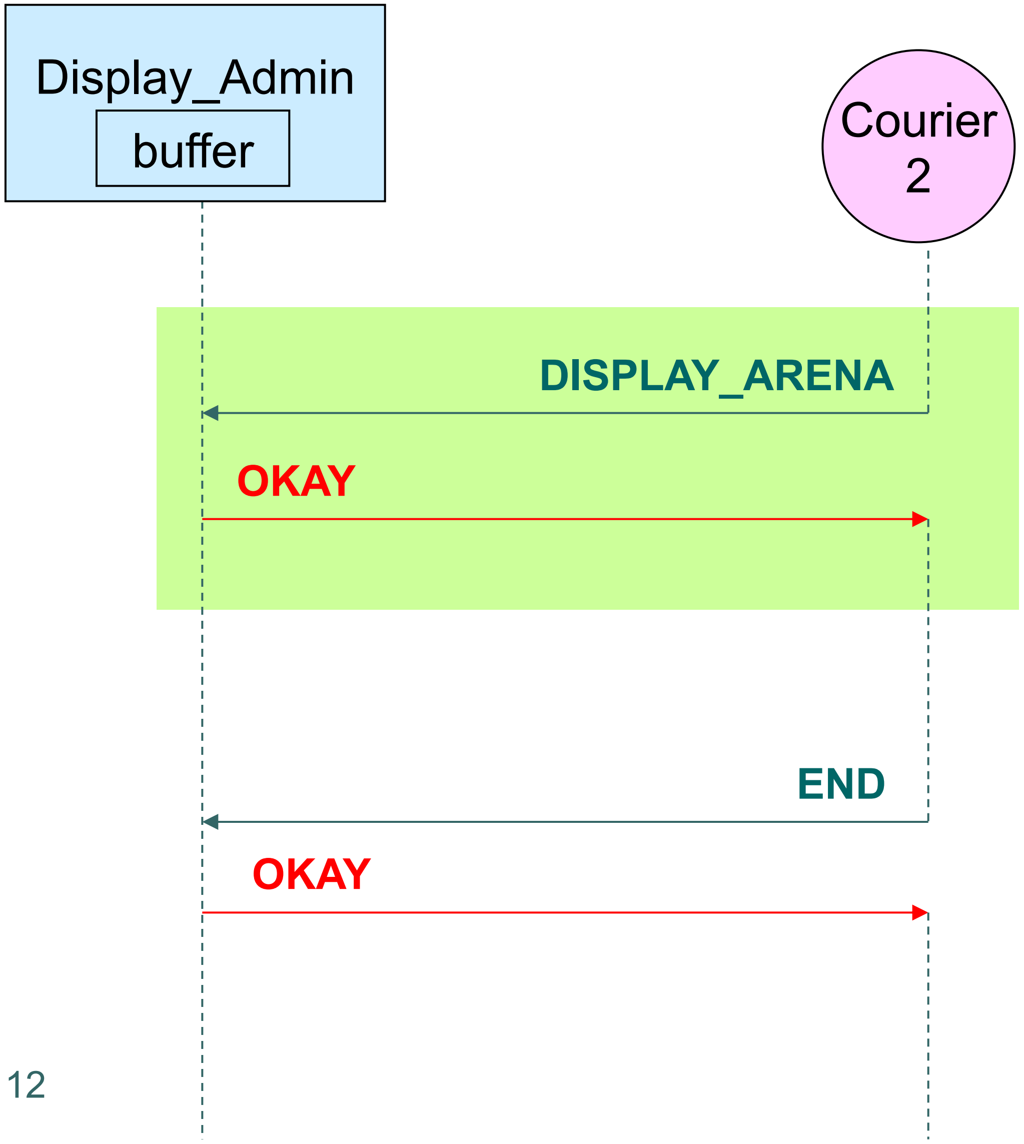
Game_Admin and Courier 0/1

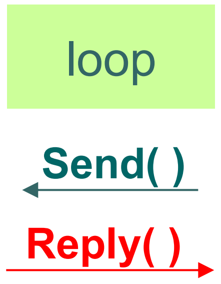


Game_Admin and Courier 2

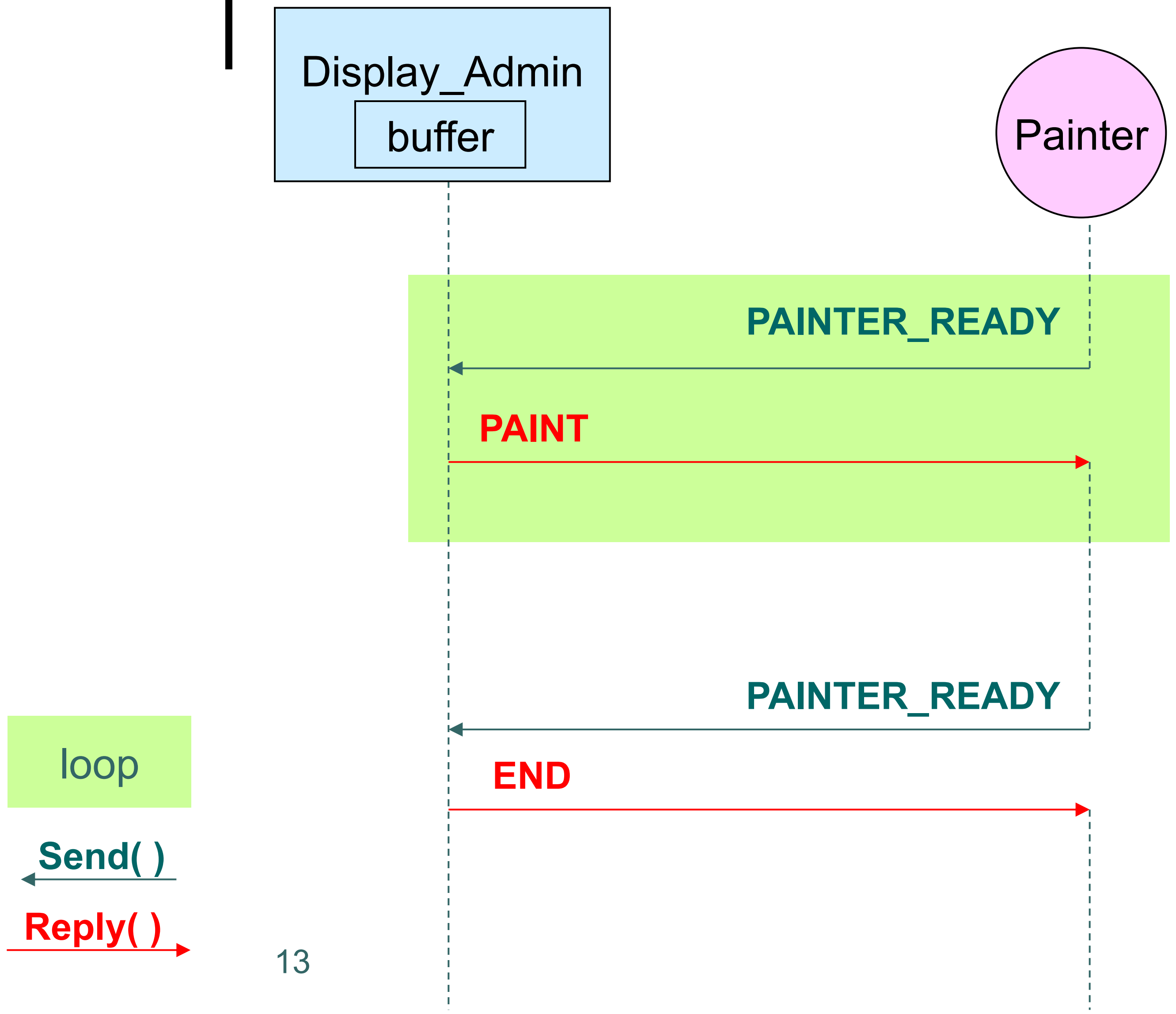


Display_Admin and Courier 2

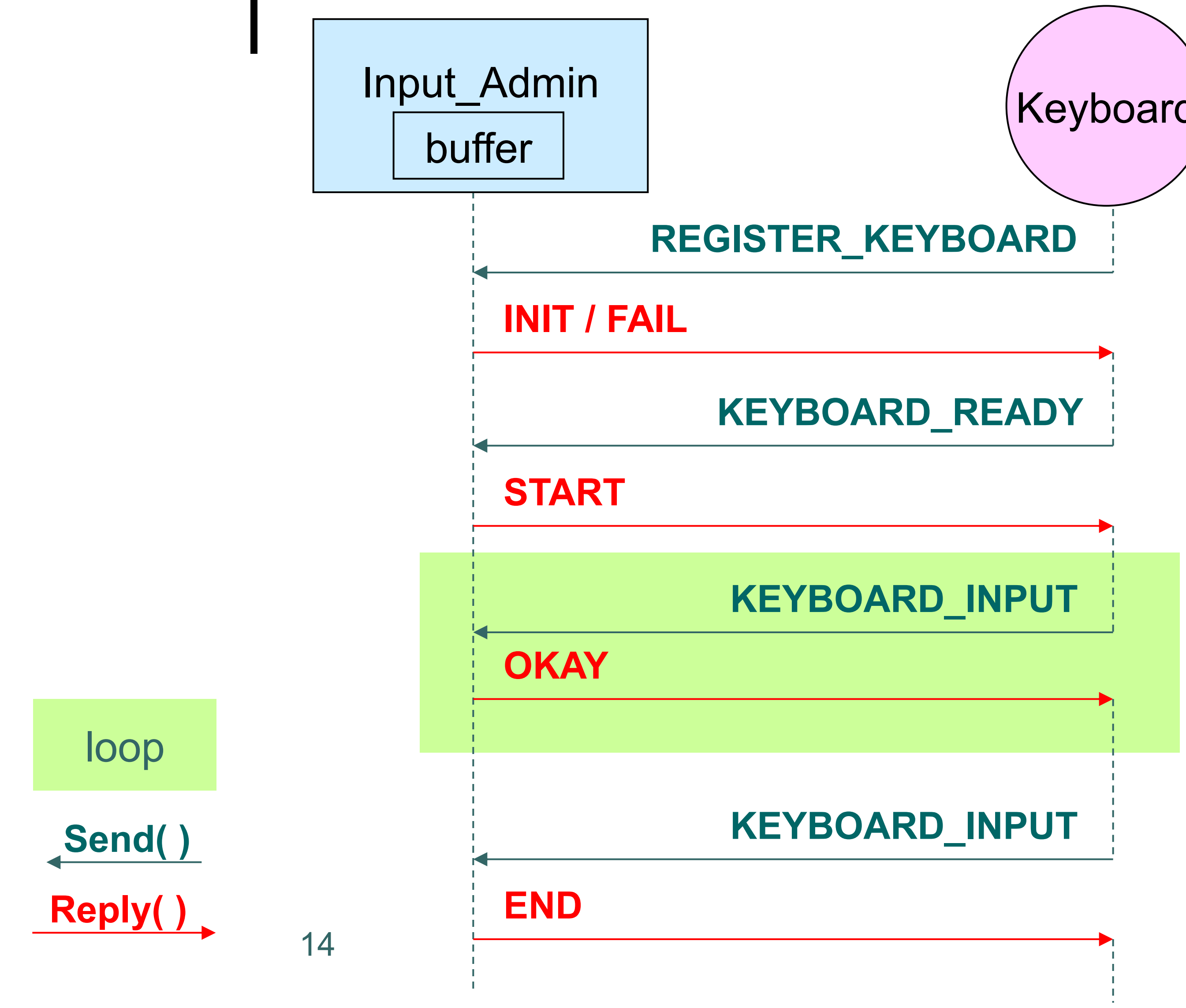




Display_Admin and Painter



Input_Admin and Keyboard



Implementation Details

- Your programs should be compatible with the sample
- Same communication protocol
- Incremental development
  - Implement each of the program units individually
  - Test your program unit by replacing it in the sample
- Submit the source files, the makefile, and the run script (you can reuse and submit the makefile and script file provided by us).

Sleep

- Timer has to sleep for a certain time
- The sleep() function takes a value in seconds
- The ***u***sleep() function takes a value in microseconds

```
#include <unistd.h>
int usleep(useconds_t useconds);
```

What's My Error?

- A useful function in SIMPL library which returns the error string

```
char *whatsMyError();
```

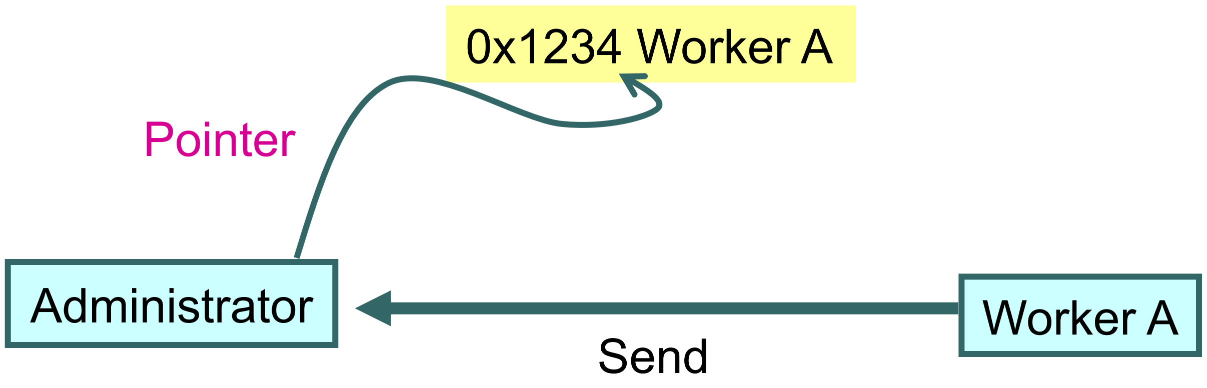
- Use it when the return value of the SIMPL functions return -1 (usually denoting an error)

Process Identification

- Each process must register a unique name before invoking other SIMPL functions
  - `name_attach()`
- You can get the id of the administrator processes by using the process name
  - `name_locate()`
- The id can be used to send messages to the administrator processes
  - `Send()`

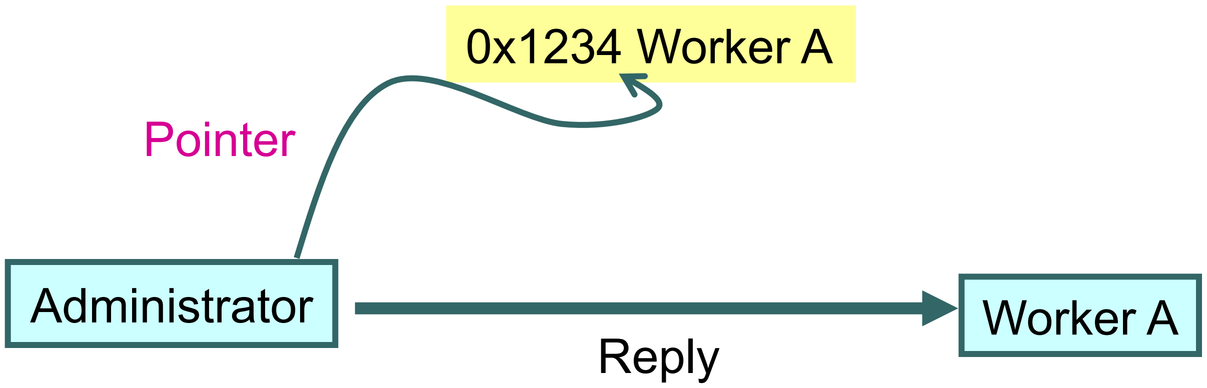
Process Identification

- Upon the administrator process receiving a message, a pointer is provided for the identification of the sender



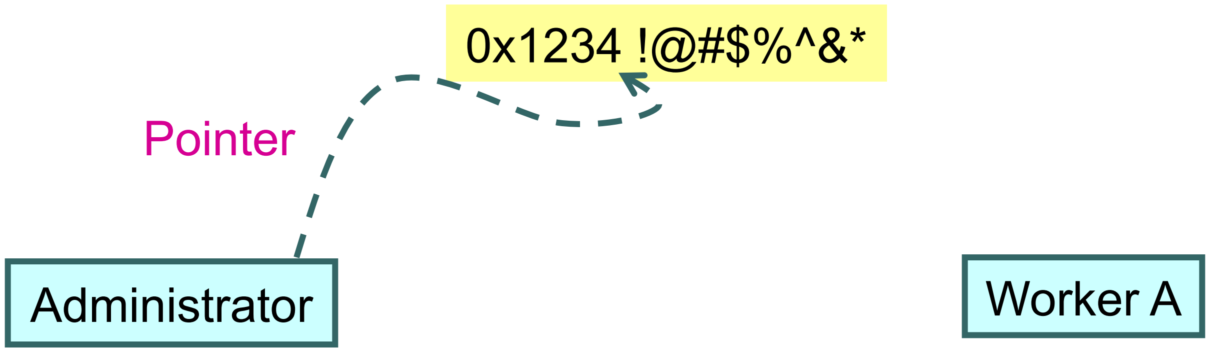
Process Identification

- The administrator process can use the pointer when replying the message



Process Identification

- Once the message is replied, the pointer can *no longer* identify the original sender
- Administrator processes **cannot** reply without receiving a message



Process Identification

- Sometimes, an administrator process *does **not** wish* to reply to messages immediately
- You can **save** the pointers for later use

Process Identification

- Sometimes, an administrator process *does wish* to send messages to a worker before receiving a message
- NO! No reply without receive

- Have to wait for a worker's message
  - COURIER_READY, PAINTER_READY

Process Identification

- **For each user**, the process name is unique on each machine
  - The data is stored under `$FIFO_PATH` (`~/fifo/`)
  - All processes must be launched to run on the same machine
  - Use a batch file to launch all processes or use different terminals to run different processes

Process Running

- `./Game_Admin &`
  - Append & after the command to make the process run in *background*

- Resources and the process name are consumed even when the process runs in background

- Keep track of the processes you are running

- Use “`ps -u [user id]`” to list out all your running processes

- Use “`kill [PID]`” to terminate any unwanted processes

Make – Build Management

- Main idea: specifying dependencies with `makefile`
- Example:

```
prog.o: prog.c
[tab] gcc -c prog.c -o prog.o
prog: prog.o
[tab] gcc prog.o -o prog -lcurses
```

- Support wild cards (e.g. %) and special macros (e.g. $?, $@)
- Read the man page or look for online resources