DECEMBER 7, 2016

PROCESS MIGRATION

OPERATING SYSTEMS COURSE PROJECT

GROUP-23
KAIVALYA SHAH - 1401108
MOHIT VACHHANI - 1401073
RATNESH SHAH - 1401110
RIDDHESH SANGHVI - 1401074

Brief Description

Process Migration:

Process Migration is the method of transferring a live process from one host to another. This is crucial for the following:

- Dynamic Load Distribution:
  - Move processes from heavily loaded to lightly loaded systems
  - This increases the hardware resource consumption in servers
- Fault Tolerance:
  - Move process from a failed host to another to ensure High Availability
- Locality:
  - A process can be moved to another host to minimize East-West traffic

Overview of Process Migration:

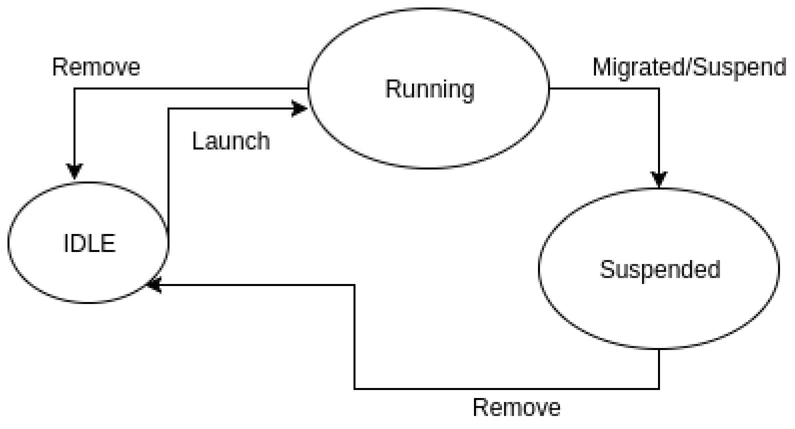
This method consists of retrieving the state of the process on the source host, transferring it to the destination host where a new instance of the process is created.

In this project, we have made 2 main executable classes, each for Server and Client. The Server's roles include:

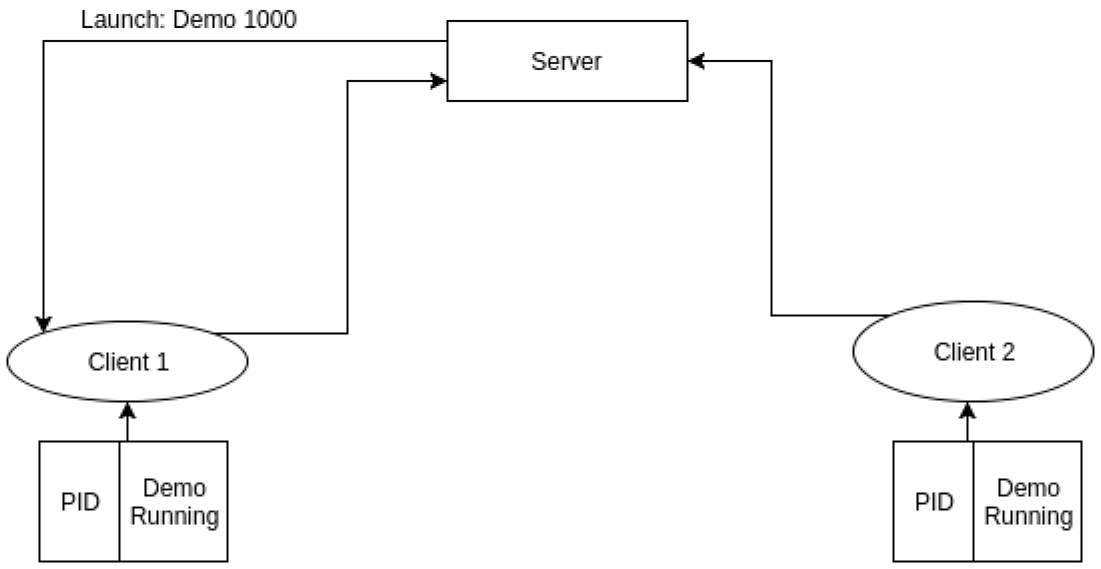
- Maintaining a continuous connection with all clients.
- Maintaining list of running processes and other details of each client.
- Instruct clients to perform the following operations:
  1. Launch a process: The server instructs a client with the class name and arguments to launch a process.
  2. Remove a process: The server instructs a client to remove a process with the process id.
  3. Migrate a process: The server instructs a client to migrate a process along with the destination client details.

Architecture

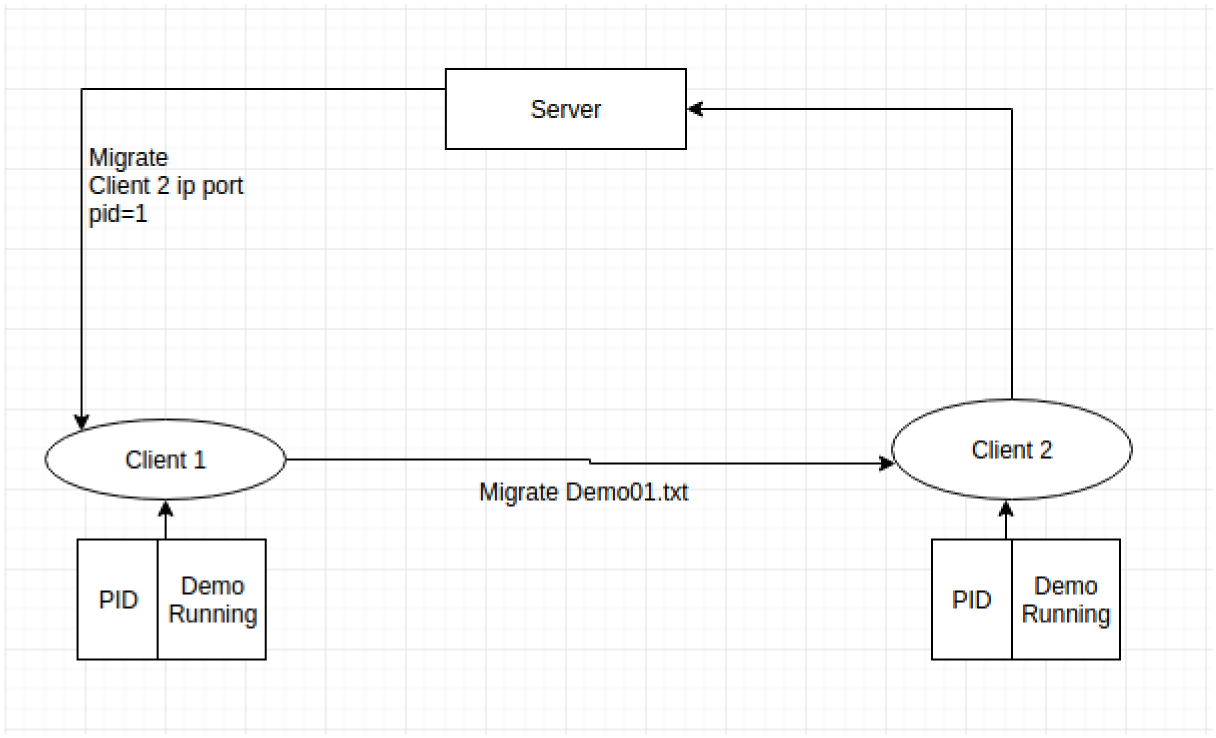
Client State Diagram:



Process Launching Flow:



Process Migrating Flow:



Technical Details

We have implemented Process Migration using Java and a Server-Client model. The server roles include establishing and maintaining client connections, instructing clients to launch, remove, or migrating a process.
The client executes the instructions sent by server.

We have made use of the following Java interfaces:

1. Runnable
2. Serializable

We have made use of the following Java classes:

1. Reflect
2. Socket
3. Logger
4. ObjectStream

The following data structures have been used:

1. Synchronized Map
2. Nested HashMap
3. ArrayList

We have implemented mutual exclusion of resources among threads by using:

1. Synchronized Map
2. Volatile variables
3. log.lck

Most of the data is stored using HashMaps to ensure a low access time.

All the threads are created on the JVM which is platform independent.

Migration Algorithm

1. A migration request is issued to the remote host including the process id, destination client IP address, and port.
2. A process is detached from its source host by suspending its execution and changing its state to migrating.
3. Connection is established between the source and destination clients.
4. The file containing the process state and information is sent to the destination client.
5. The process state is extracted that includes memory contents, processor state, and relevant kernel context.
6. The process is recreated from the above information.

Server Workflow

1. Server is started by running ServerAdmin class with port number as argument
2. ServerAdmin starts thread of ProcAdmin
3. ProcAdmin thread calls connect function
4. Connect function creates socket and accepts client connection
5. Connect function starts new thread of ClientProcConn
6. ClientProcConn thread listens for Hello packet from client and does the following:
   - Disconnects client on Hello timeout
   - Extracts client port number from Hello
7. ClientProcConn thread updates process and client HashMap entries which can be invoked by:
   - Message from client - Migrated/Terminated
   - Client disconnected or removed due to Hello timeout

Server-Client Interaction

1. Launch:
   Creates object of clientProcStructure with IP address, proc-name, pid, and port
   Create socket connection with client of the given IP and port
   Send command via PrintStream and close connection
2. Remove:
   Print the proc id, name and associated client from hashmap
   Ask for proc id to remove
   create socket connection with client of the given ip and port
   Send command via PrintStream and close connection
   Remove entry from hashmap containing that proc id
3. Migrate:
   Create socket connection with the host running the process
   Send migrate command via PrintStream and close connection
   Create socket connection with destination host

Send command via PrintStream to launch connection with source host
Set client details of new client

Client Workflow

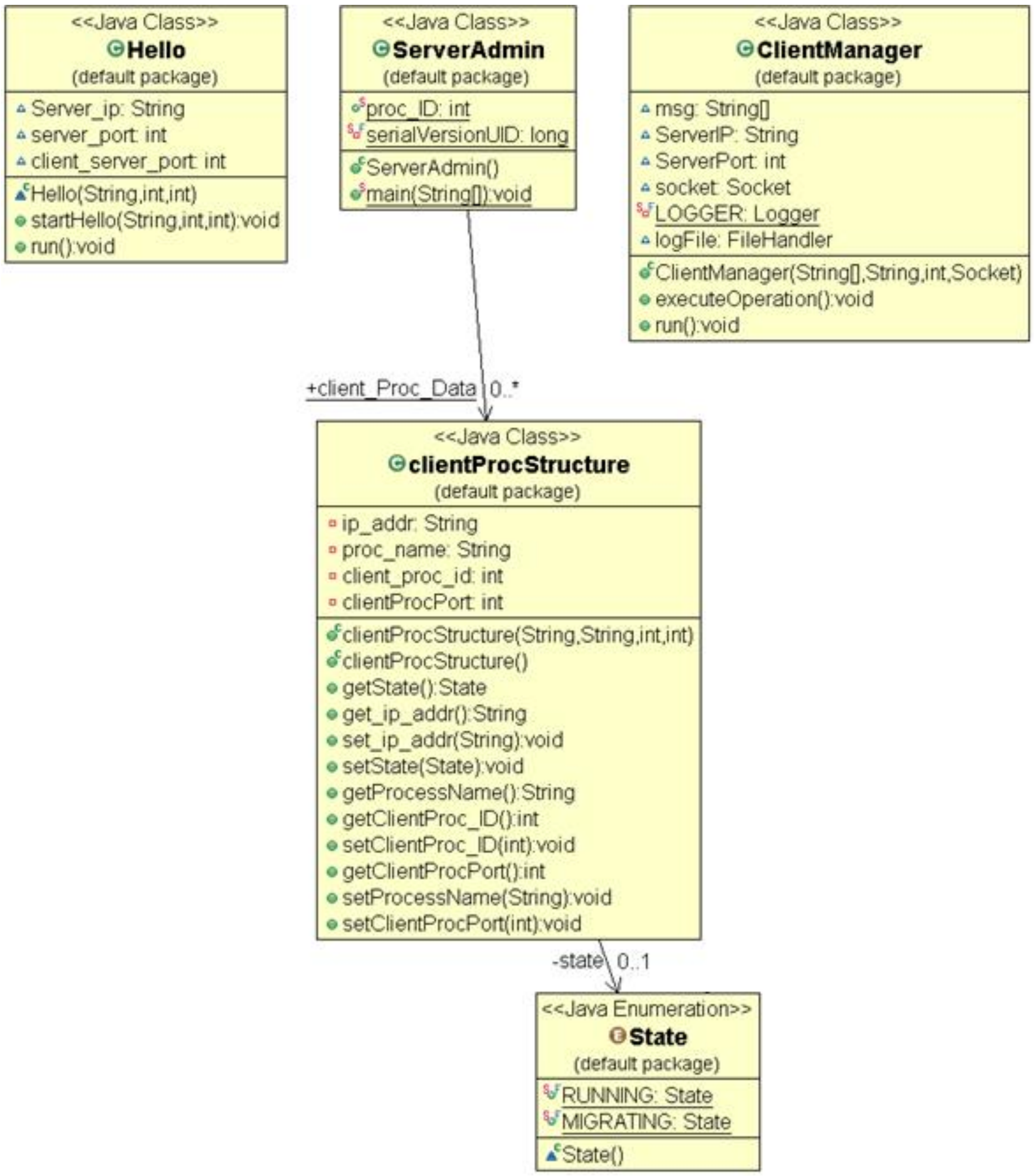
1. Client Console class --> serverip, serverport, clientServerport
2. Connection established with the server
3. Maintains list of process_id running on the client_id--->(ThreadIds)
4. Server Creates a new connection with the client to give the command
5. ClientManager Thread created
6. ClientManager thread performs operations for the received command from the server --> executeOperation() function called
7. Receives the command msg sent from the server
8. ********** Launch Command *********
9. Gets the process name and the required arguments from the server msg
10. A new instance of the class is created at the client
11. Thread is initiated for this class
12. Then waits for the thread to terminate and after it gets terminated it sends response to the server of the process name and process id
13. ********** Migrate Command *********
14. Work: We save a snapshot of the running thread on the source machine than make it serializable and then make the thread suspend in the source save the snapshot to a file and then transfer the file to the destination. At the destination we deserialize the object and then run it in a new thread
15. This message is received by the client from the server which includes -> command,dest-ip,dest-port, source-ip,source-port
16. Connection established between the source and the destination
17. gets the id of the running thread from the list
18. Thread.getAllStackTraces() returns a map containing the thread id as key and value containing the stack dump of all the live threads
19. The stack dump of the process to be migrated is obtained from this map using the key
20. At the source side the state of the migrating thread is changed to suspend.
21. The snapshot of the process is saved in the object file.

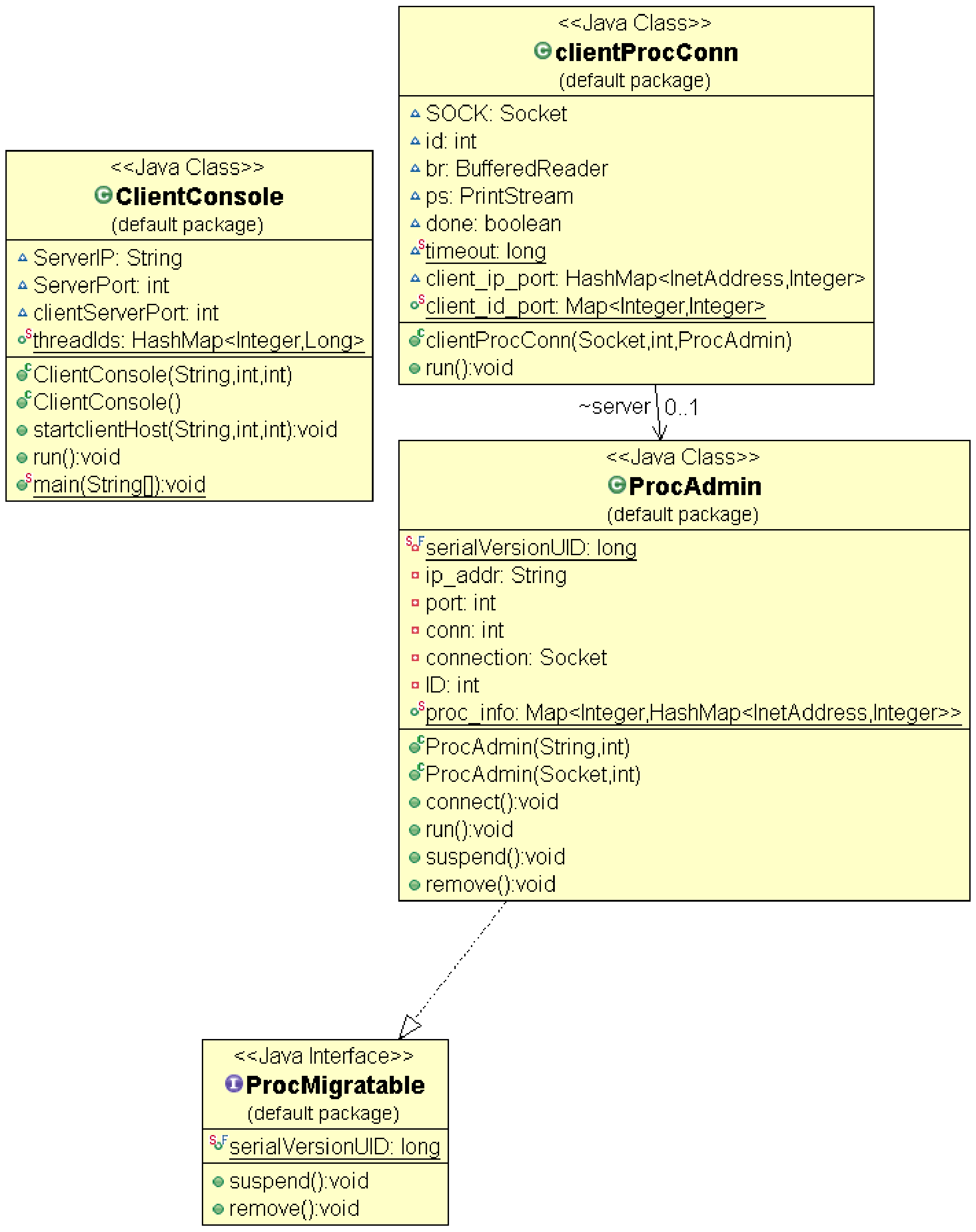
22. The file is sent to the destination client
23. The entry of this process is removed from the source running process list
24. After the migration is over message is sent to the destination client i.e (ReceiveProcess, proc_name, proc_ID, myIP, myPort, destIP, destPort, filename.txt)
25. Connection closed
26. *********** ReceiveProcess Command *************
27. This message is received at the destination client of the migration process. The message contains this clients ip as dest-address, dest-port, source-ip, source-port,filename.txt
28. The object file is deserialized at this receiving end and the thread is recreated and started at this end
29. The ThreadIds list is updated with this new thread entry
30. Then response message is sent to the source client that the migration is successful

Source Code Files

1. ClientConsole.java
2. ClientProcStructure.java
3. ServerAdmin.java
4. ProcAdmin.java

Class Diagram

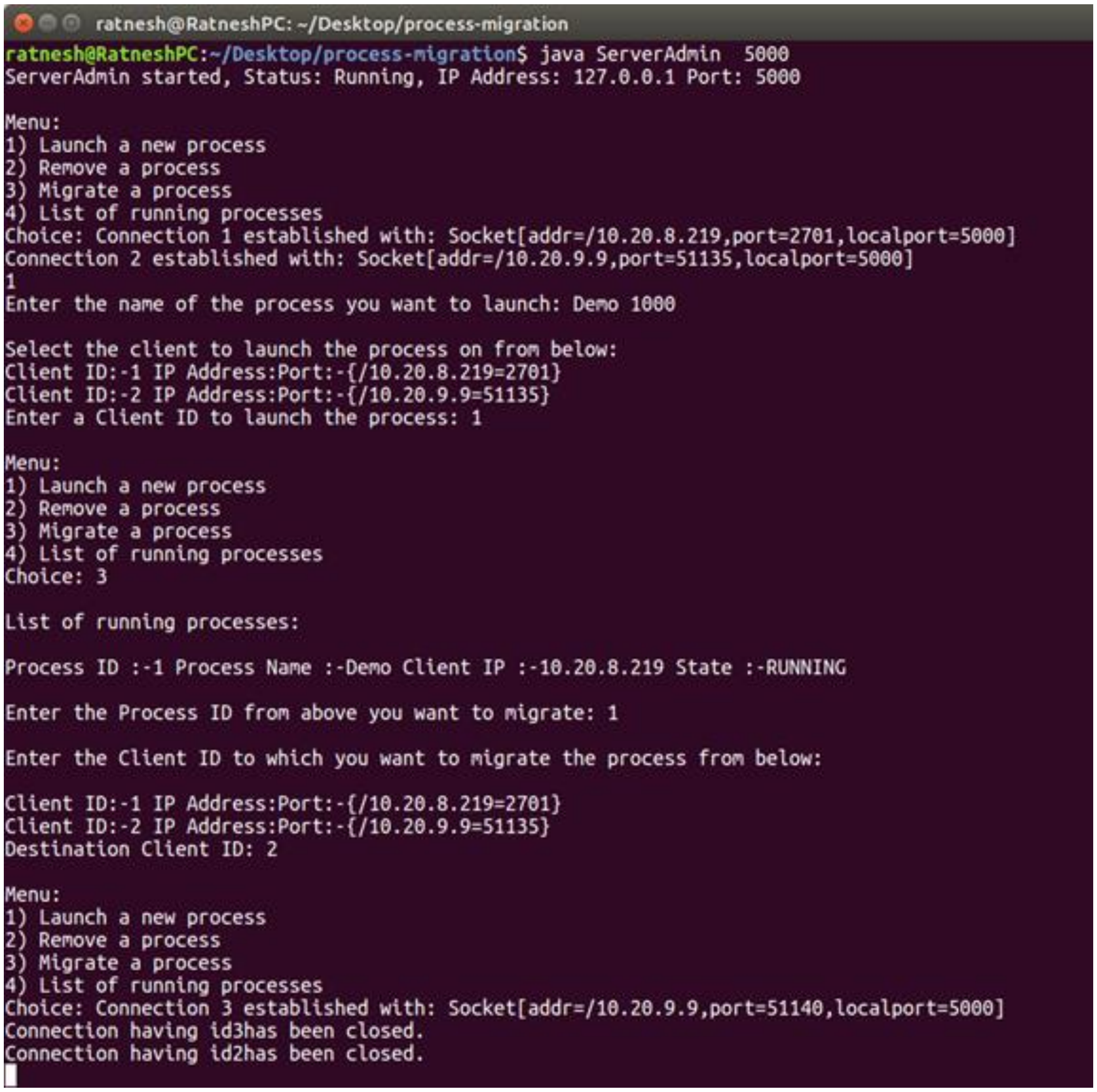




<<Java Class>>
ClientConsole
(default package)
ServerIP: String
ServerPort: int
clientServerPort: int
threadIds: HashMap<Integer,Long>
ClientConsole(String,int,int)
ClientConsole()
startclientHost(String,int,int):void
run():void
main(String[]):void
<<Java Class>>
clientProcConn
(default package)
SOCK: Socket
id: int
br: BufferedReader
ps: PrintStream
done: boolean
timeout: long
client_ip_port: HashMap<InetAddress,Integer>
client_id_port: Map<Integer,Integer>
clientProcConn(Socket,int,ProcAdmin)
run():void
~server
0..1
<<Java Class>>
ProcAdmin
(default package)
serialVersionUID: long
ip_addr: String
port: int
conn: int
connection: Socket
ID: int
proc_info: Map<Integer,HashMap<InetAddress,Integer>>
ProcAdmin(String,int)
ProcAdmin(Socket,int)
connect():void
run():void
suspend():void
remove():void
<<Java Interface>>
ProcMigratable
(default package)
serialVersionUID: long
suspend():void
remove():void

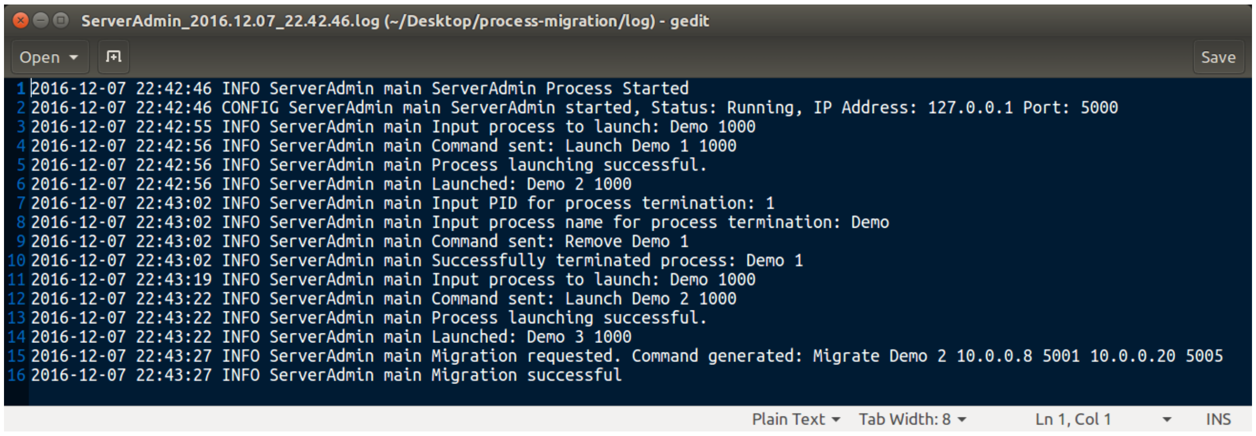
Test Results

Server:



```
ratnesh@RatneshPC: ~/Desktop/process-migration
ratnesh@RatneshPC:~/Desktop/process-migration$ java ServerAdmin  5000
ServerAdmin started, Status: Running, IP Address: 127.0.0.1 Port: 5000

Menu:
1) Launch a new process
2) Remove a process
3) Migrate a process
4) List of running processes
Choice: Connection 1 established with: Socket[addr=/10.20.8.219,port=2701,localport=5000]
Connection 2 established with: Socket[addr=/10.20.9.9,port=51135,localport=5000]
1
Enter the name of the process you want to launch: Demo 1000

Select the client to launch the process on from below:
Client ID:-1 IP Address:Port:-{/10.20.8.219=2701}
Client ID:-2 IP Address:Port:-{/10.20.9.9=51135}
Enter a Client ID to launch the process: 1

Menu:
1) Launch a new process
2) Remove a process
3) Migrate a process
4) List of running processes
Choice: 3

List of running processes:

Process ID :-1 Process Name :-Demo Client IP :-10.20.8.219 State :-RUNNING

Enter the Process ID from above you want to migrate: 1

Enter the Client ID to which you want to migrate the process from below:

Client ID:-1 IP Address:Port:-{/10.20.8.219=2701}
Client ID:-2 IP Address:Port:-{/10.20.9.9=51135}
Destination Client ID: 2

Menu:
1) Launch a new process
2) Remove a process
3) Migrate a process
4) List of running processes
Choice: Connection 3 established with: Socket[addr=/10.20.9.9,port=51140,localport=5000]
Connection having id3has been closed.
Connection having id2has been closed.
```

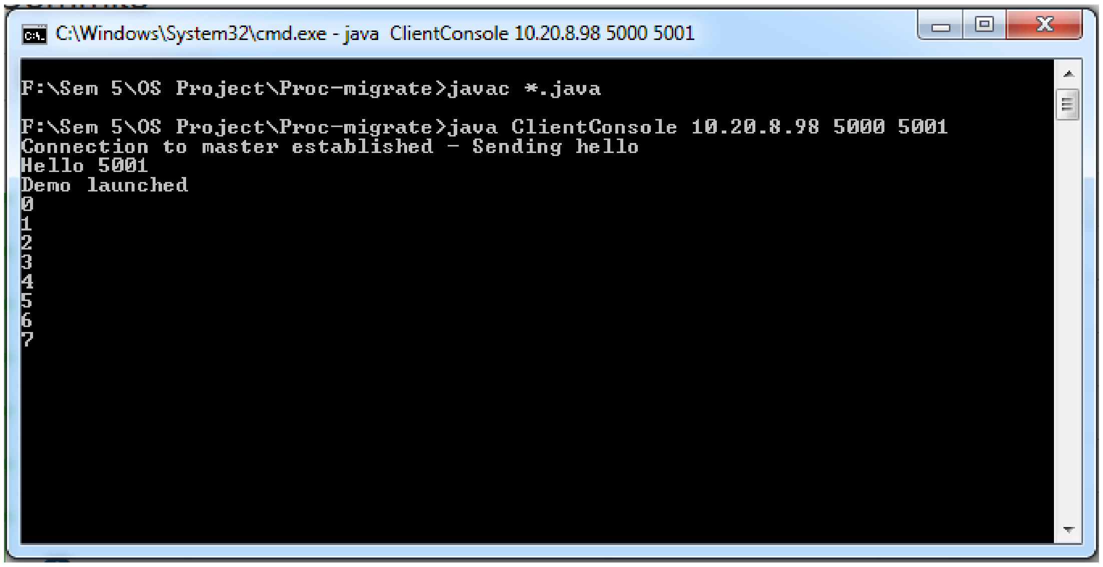
Server Log:



ServerAdmin_2016.12.07_22.42.46.log (~/Desktop/process-migration/log) - gedit

```
2016-12-07 22:42:46 INFO ServerAdmin main ServerAdmin Process Started
2016-12-07 22:42:46 CONFIG ServerAdmin main ServerAdmin started, Status: Running, IP Address: 127.0.0.1 Port: 5000
2016-12-07 22:42:55 INFO ServerAdmin main Input process to launch: Demo 1000
2016-12-07 22:42:56 INFO ServerAdmin main Command sent: Launch Demo 1 1000
2016-12-07 22:42:56 INFO ServerAdmin main Process launching successful.
2016-12-07 22:42:56 INFO ServerAdmin main Launched: Demo 2 1000
2016-12-07 22:43:02 INFO ServerAdmin main Input PID for process termination: 1
2016-12-07 22:43:02 INFO ServerAdmin main Input process name for process termination: Demo
2016-12-07 22:43:02 INFO ServerAdmin main Command sent: Remove Demo 1
2016-12-07 22:43:02 INFO ServerAdmin main Successfully terminated process: Demo 1
2016-12-07 22:43:19 INFO ServerAdmin main Input process to launch: Demo 1000
2016-12-07 22:43:22 INFO ServerAdmin main Command sent: Launch Demo 2 1000
2016-12-07 22:43:22 INFO ServerAdmin main Process launching successful.
2016-12-07 22:43:22 INFO ServerAdmin main Launched: Demo 3 1000
2016-12-07 22:43:27 INFO ServerAdmin main Migration requested. Command generated: Migrate Demo 2 10.0.0.8 5001 10.0.0.20 5005
2016-12-07 22:43:27 INFO ServerAdmin main Migration successful
```

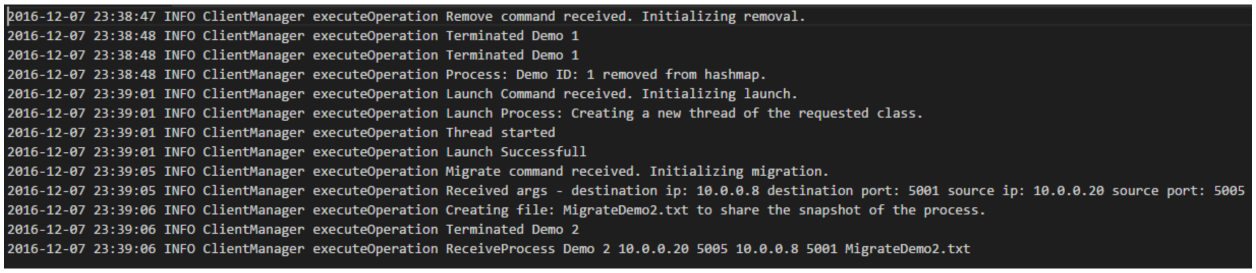
Client 1 console:



C:\Windows\System32\cmd.exe - java ClientConsole 10.20.8.98 5000 5001

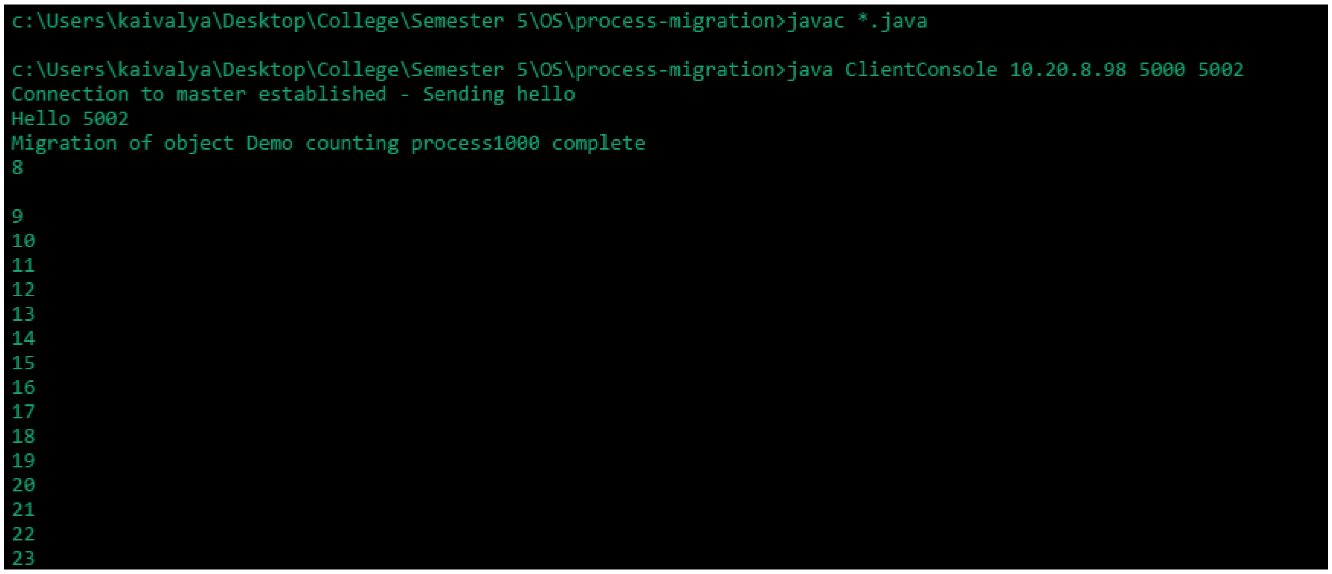
```
F:\Sem 5\OS Project\Proc-migrate>javac *.java

F:\Sem 5\OS Project\Proc-migrate>java ClientConsole 10.20.8.98 5000 5001
Connection to master established - Sending hello
Hello 5001
Demo launched
0
1
2
3
4
5
6
7
```

Client 1 log:



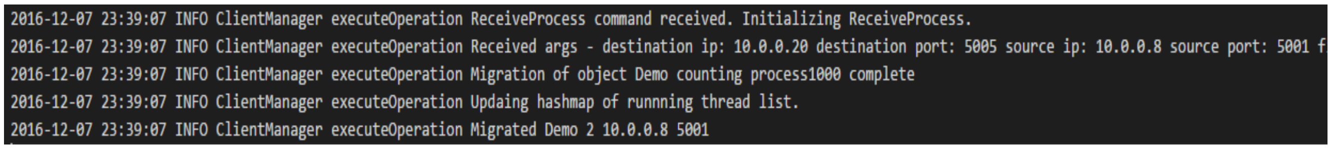
```
2016-12-07 23:38:47 INFO ClientManager executeOperation Remove command received. Initializing removal.
2016-12-07 23:38:48 INFO ClientManager executeOperation Terminated Demo 1
2016-12-07 23:38:48 INFO ClientManager executeOperation Terminated Demo 1
2016-12-07 23:38:48 INFO ClientManager executeOperation Process: Demo ID: 1 removed from hashmap.
2016-12-07 23:39:01 INFO ClientManager executeOperation Launch Command received. Initializing launch.
2016-12-07 23:39:01 INFO ClientManager executeOperation Launch Process: Creating a new thread of the requested class.
2016-12-07 23:39:01 INFO ClientManager executeOperation Thread started
2016-12-07 23:39:01 INFO ClientManager executeOperation Launch Successfull
2016-12-07 23:39:05 INFO ClientManager executeOperation Migrate command received. Initializing migration.
2016-12-07 23:39:05 INFO ClientManager executeOperation Received args - destination ip: 10.0.0.8 destination port: 5001 source ip: 10.0.0.20 source port: 5005
2016-12-07 23:39:06 INFO ClientManager executeOperation Creating file: MigrateDemo2.txt to share the snapshot of the process.
2016-12-07 23:39:06 INFO ClientManager executeOperation Terminated Demo 2
2016-12-07 23:39:06 INFO ClientManager executeOperation ReceiveProcess Demo 2 10.0.0.20 5005 10.0.0.8 5001 MigrateDemo2.txt
```

Client 2 console:



```
c:\Users\kaivalya\Desktop\College\Semester 5\OS\process-migration>javac *.java

c:\Users\kaivalya\Desktop\College\Semester 5\OS\process-migration>java ClientConsole 10.20.8.98 5000 5002
Connection to master established - Sending hello
Hello 5002
Migration of object Demo counting process1000 complete
8

9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

Client 2 log:



```
2016-12-07 23:39:07 INFO ClientManager executeOperation ReceiveProcess command received. Initializing ReceiveProcess.
2016-12-07 23:39:07 INFO ClientManager executeOperation Received args - destination ip: 10.0.0.20 destination port: 5005 source ip: 10.0.0.8 source port: 5001 f
2016-12-07 23:39:07 INFO ClientManager executeOperation Migration of object Demo counting process1000 complete
2016-12-07 23:39:07 INFO ClientManager executeOperation Updaing hashmap of runnning thread list.
2016-12-07 23:39:07 INFO ClientManager executeOperation Migrated Demo 2 10.0.0.8 5001
```

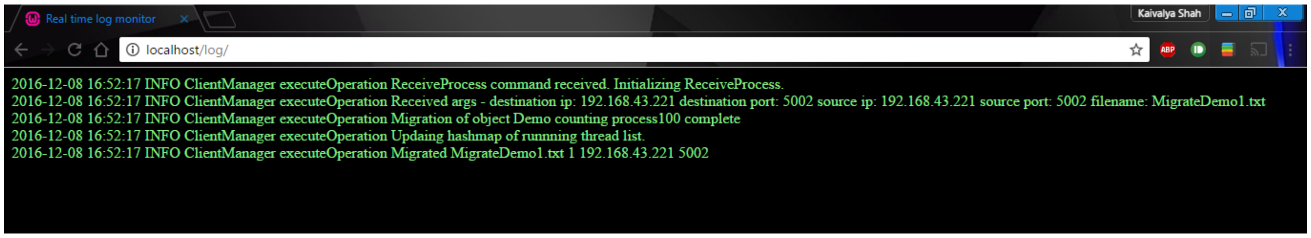
Real-Time Logging

We have implemented logging in this project on the server as well as client which has 4 types of log messages:

1. Config - Shows the configuration and initial setup
2. Info - Shows normal execution messages
3. Warning - Warns about potential errors which can be recovered
4. Severe - Shows error that cause process termination or other fatal operations

Apart from the log text file, a log.lck file is also created at runtime to ensure mutual exclusion among threads.

We have also made a Real-Time Log monitoring website which shows the log onto a hosted server and is capable of notifying severe messages to the system admin. This is achieved using AJAX (Asynchronous JavaScript and XML) and PHP.



```
2016-12-08 16:52:17 INFO ClientManager executeOperation ReceiveProcess command received. Initializing ReceiveProcess.
2016-12-08 16:52:17 INFO ClientManager executeOperation Received args - destination ip: 192.168.43.221 destination port: 5002 source ip: 192.168.43.221 source port: 5002 filename: MigrateDemo1.txt
2016-12-08 16:52:17 INFO ClientManager executeOperation Migration of object Demo counting process100 complete
2016-12-08 16:52:17 INFO ClientManager executeOperation Updaing hashmap of runnning thread list.
2016-12-08 16:52:17 INFO ClientManager executeOperation Migrated MigrateDemo1.txt 1 192.168.43.221 5002
```

References

1. Process Migration - DEJAN S. MILOJI, HP Labs; FRED DOUGLIS, AT&T Labs–Research; YVES PAINDAVEINE, TOG Research Institute; RICHARD WHEELER, EMC; SONGNIAN ZHOU, University of Toronto and Platform Computing - ACM Computing Surveys, Vol. 32, No. 3, September 2000, pp. 241–299
2. https://docs.oracle.com/javase/7/docs/api/
3. http://www.indiastudychannel.com/resources/101340-Process-Management-Process-Migration.aspx
4. http://vega.cs.kent.edu/~mikhail/classes/aos.f03/l15migration.PDF