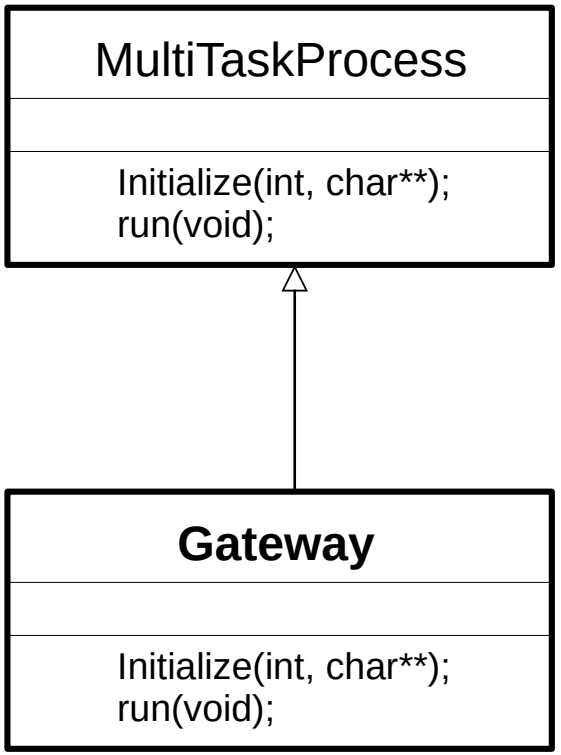
MQTT-SN Gateway Overview

The Gateway is constructed and invoked from within **mainGateway.cpp**

```
24
25 /*
26  *  Gateway Application
27  */
28 Gateway gateway;
29 PacketHandleTask  task1(&gateway);
30 ClientRecvTask    task2(&gateway);
31 ClientSendTask    task3(&gateway);
32 BrokerRecvTask    task4(&gateway);
33 BrokerSendTask    task5(&gateway);
34
35 int main(int argc, char** argv)
36 {
37     gateway.initialize(argc, argv);
38     gateway.run();
39     return 0;
40 }
41
```

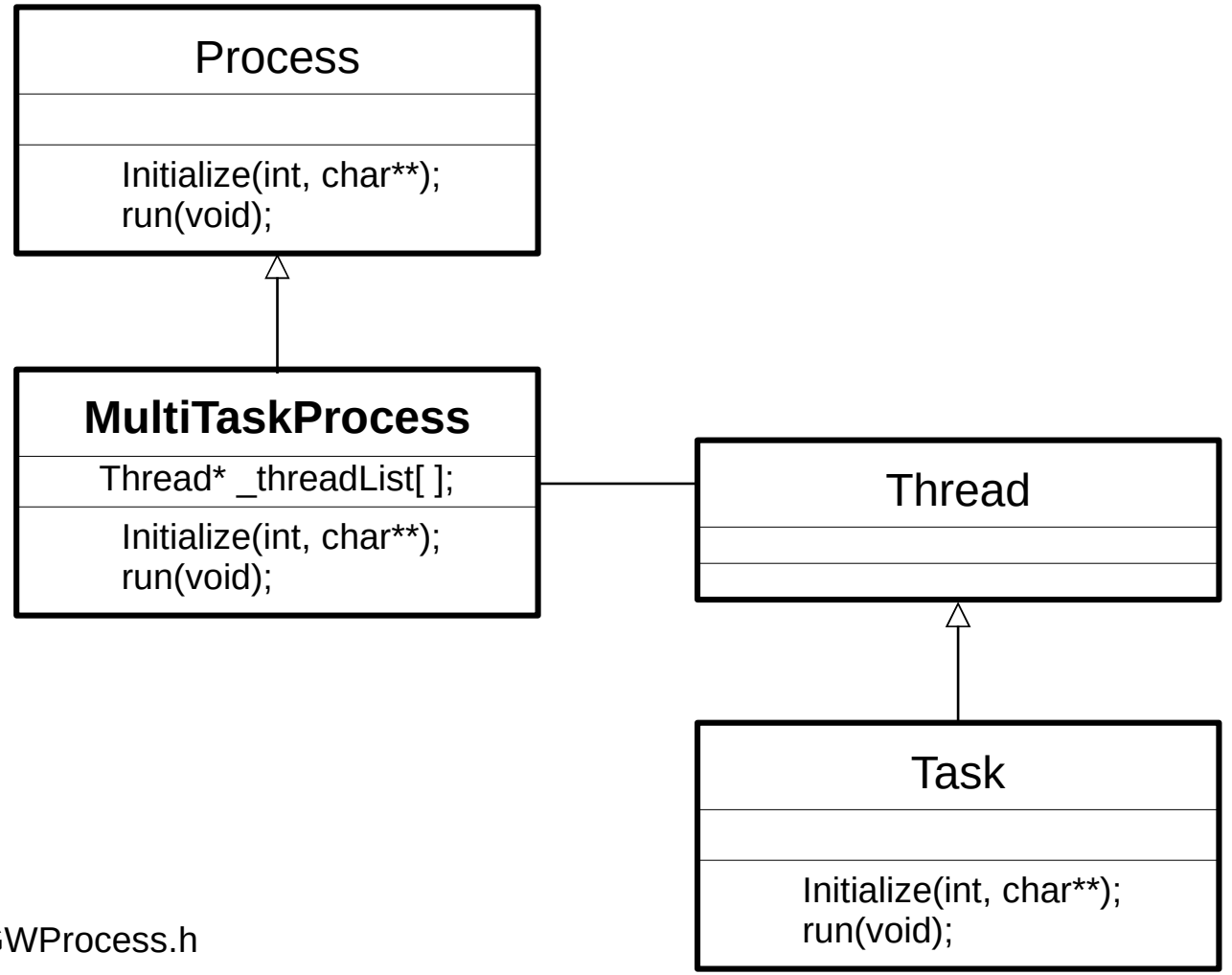
1. Model

1-1. Gateway Class



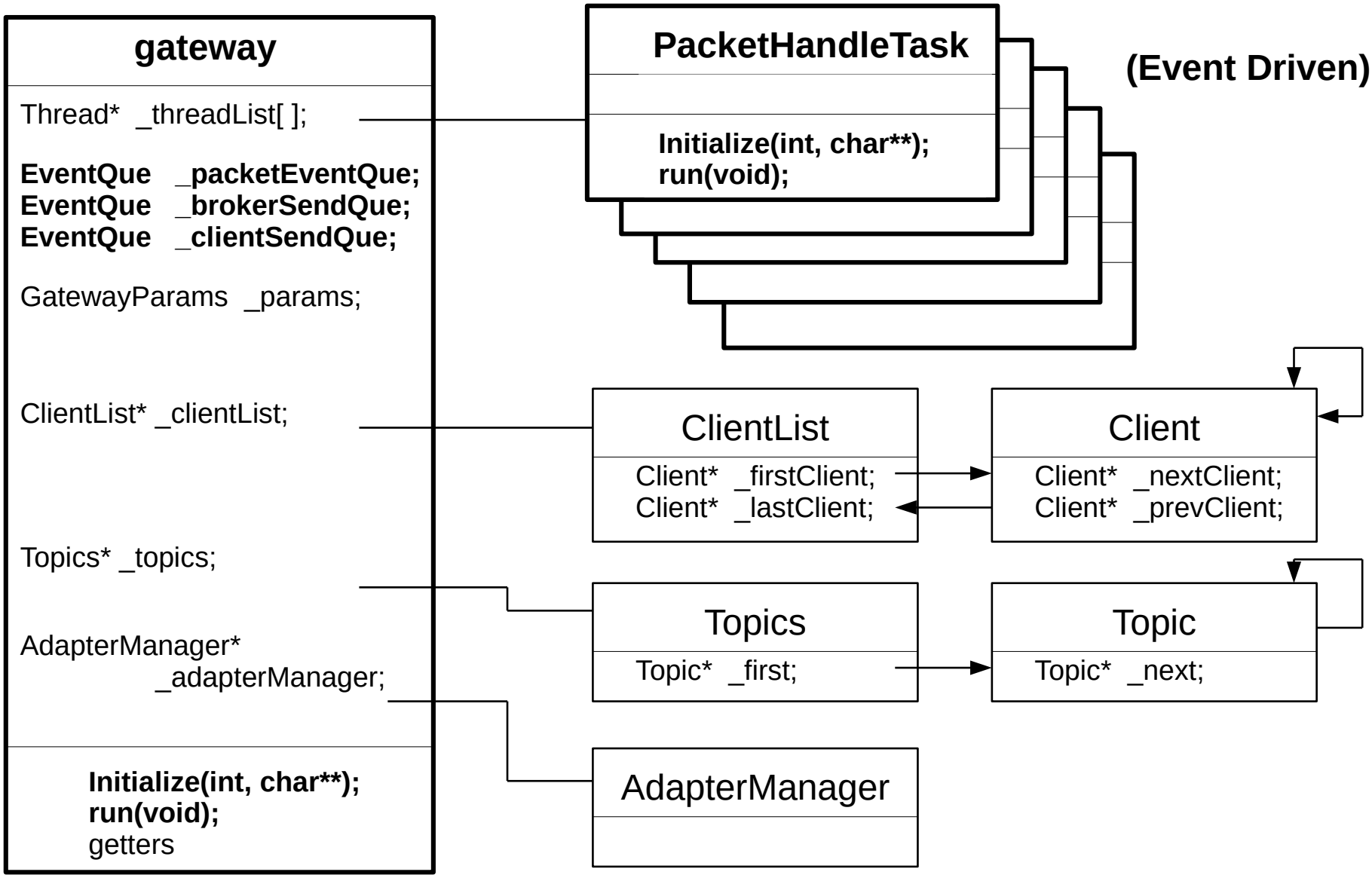
See MQTTSNGateway.h

1-2. MultiTaskProcess Class

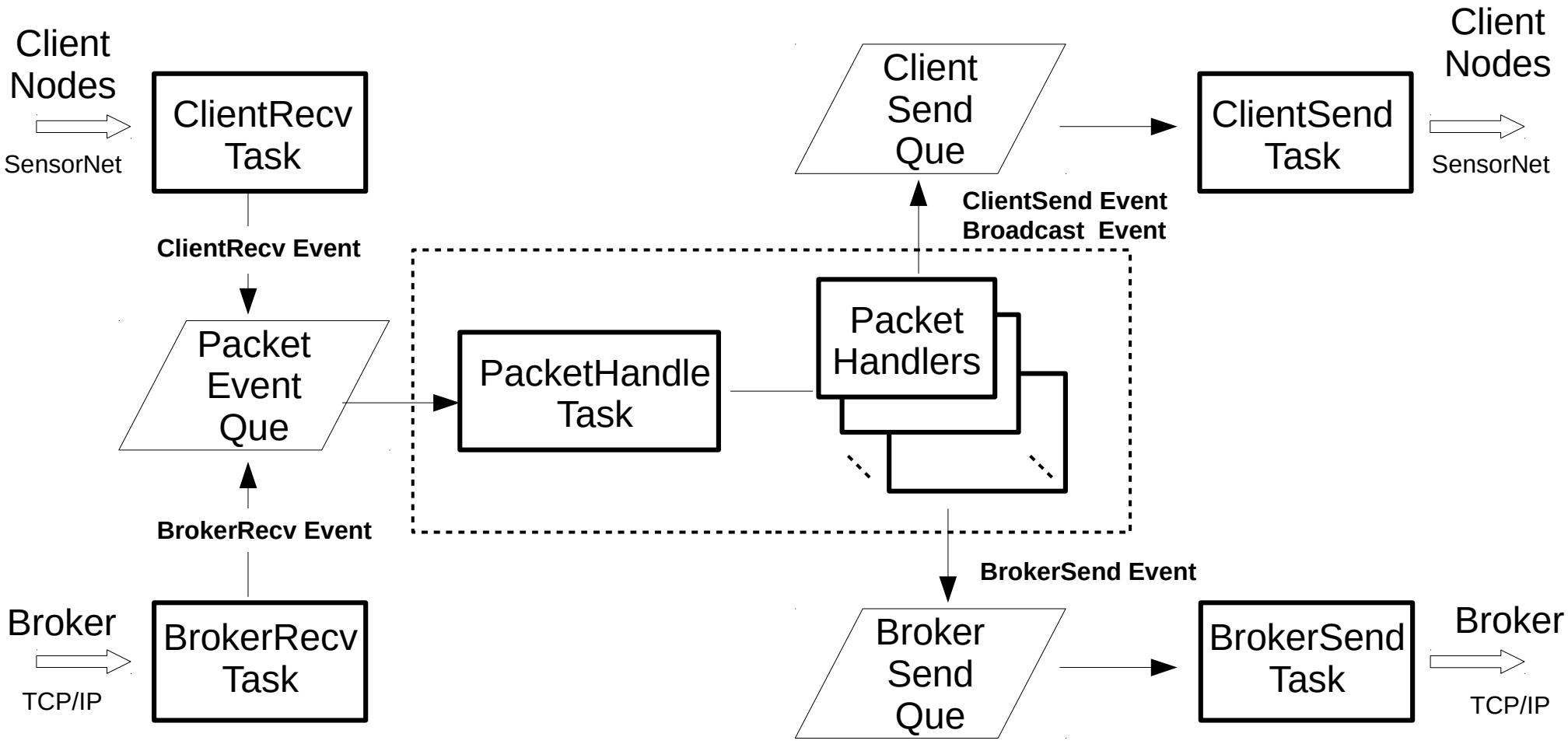


See MQTTSNGWProcess.h

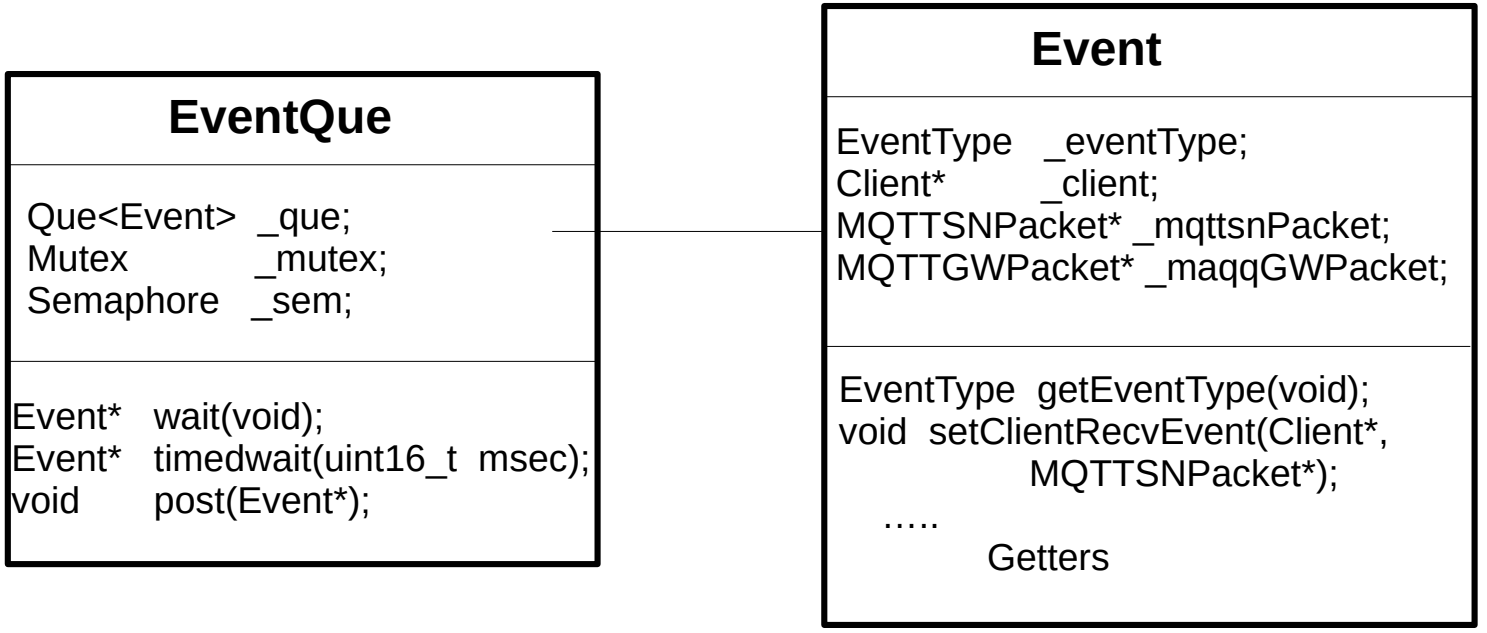
1-3. Instances created by ‘mainGateway.cpp’



1-4. Event Flow and Task Linkage

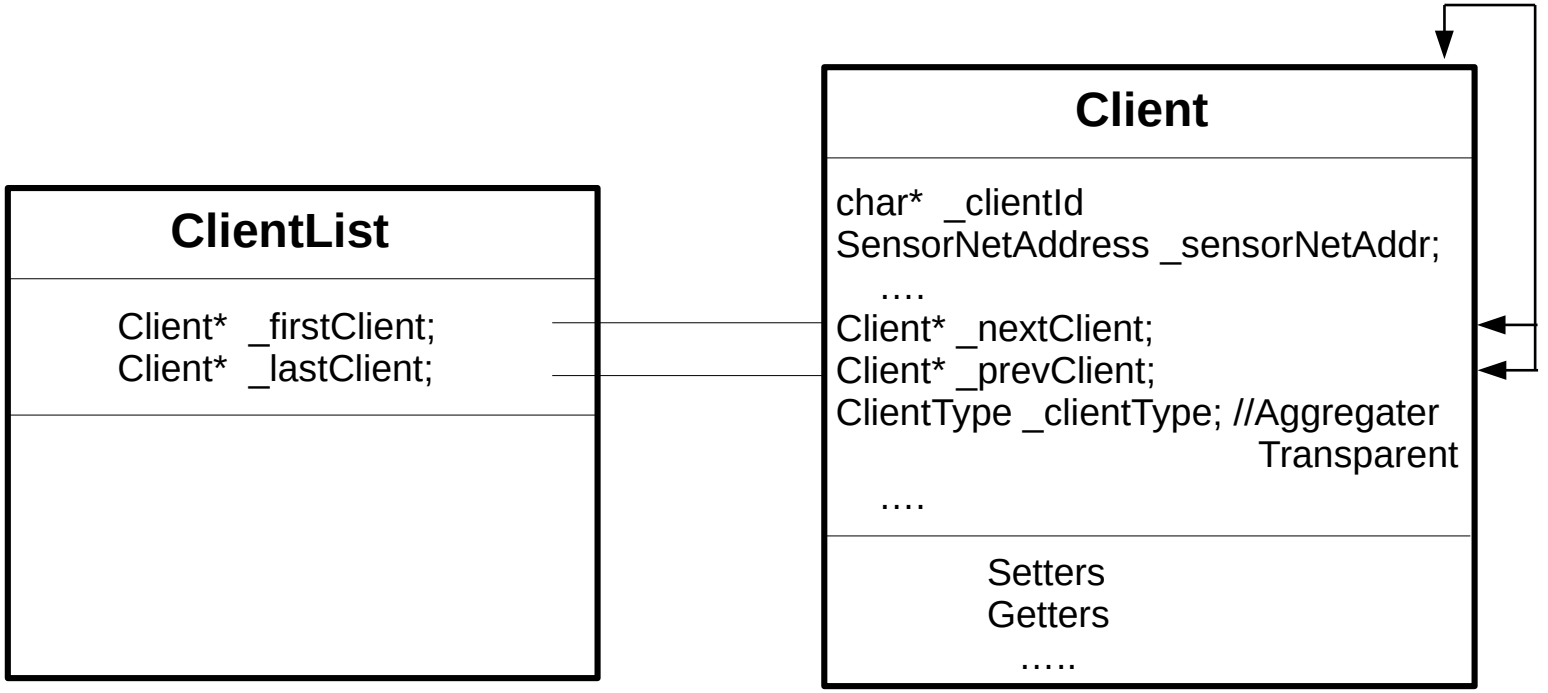


1-5. Event and Event Que



See MQTTSNGateway.h
MQTTSNGWProcess.h

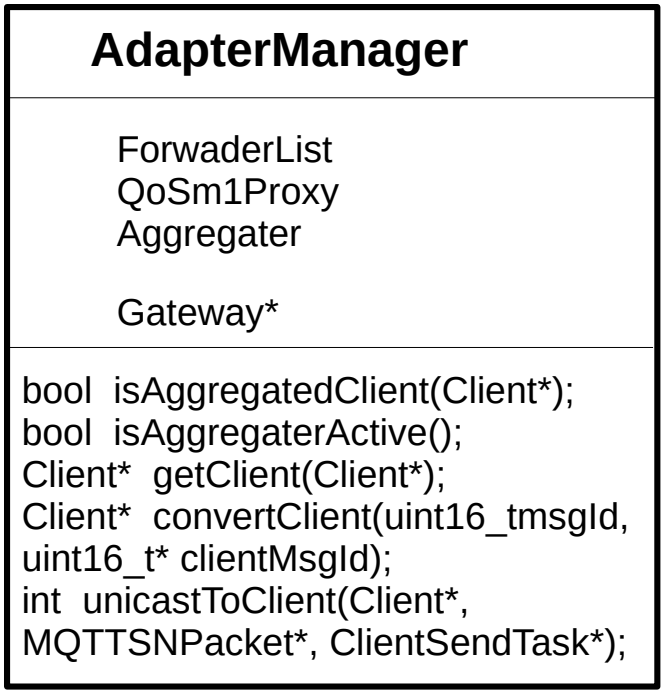
1-6. Client and ClientList Class



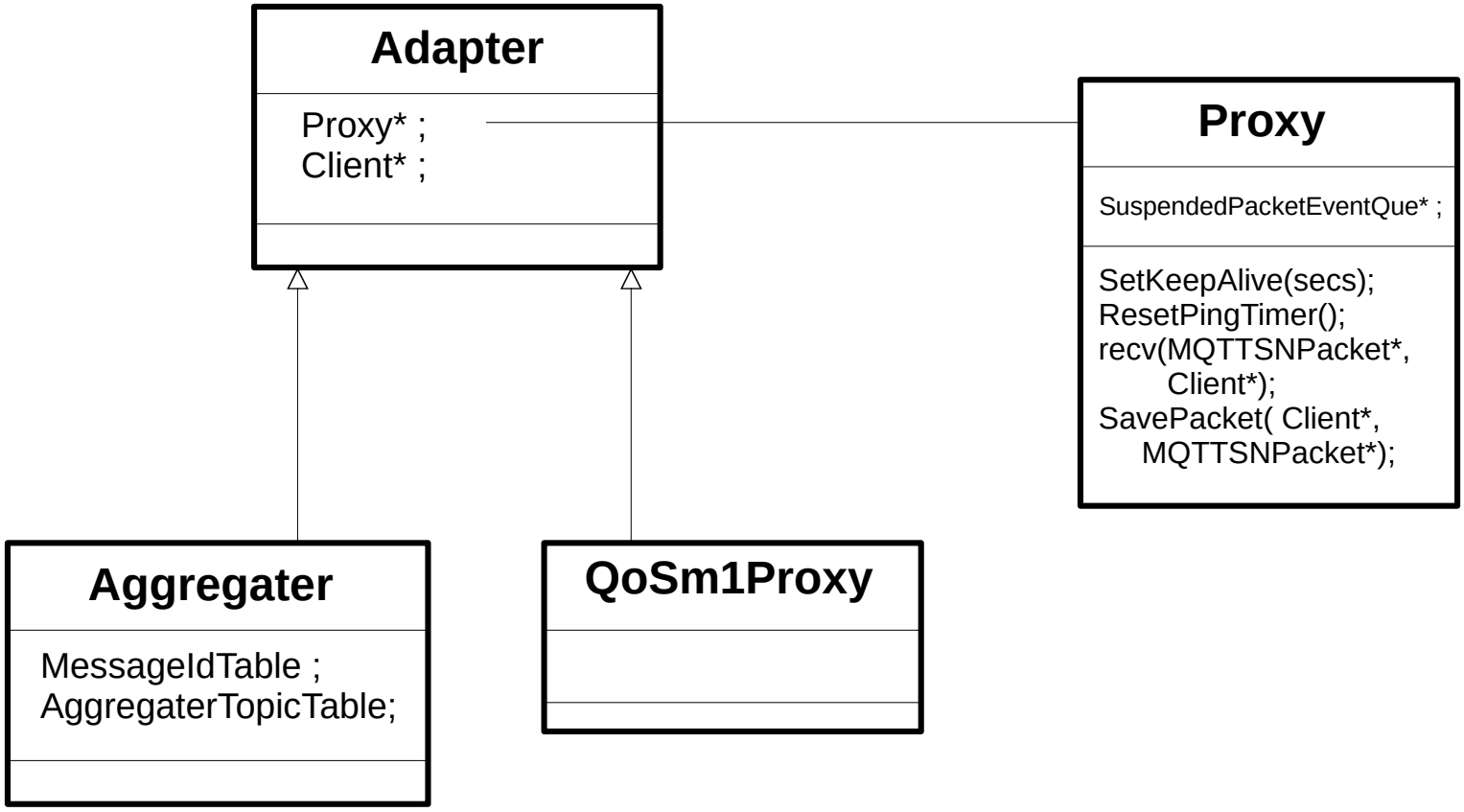
Client is a most complicated Class.
It should be refactered.

See MQTTSNGWClientList.h
MQTTSNGWClient.h

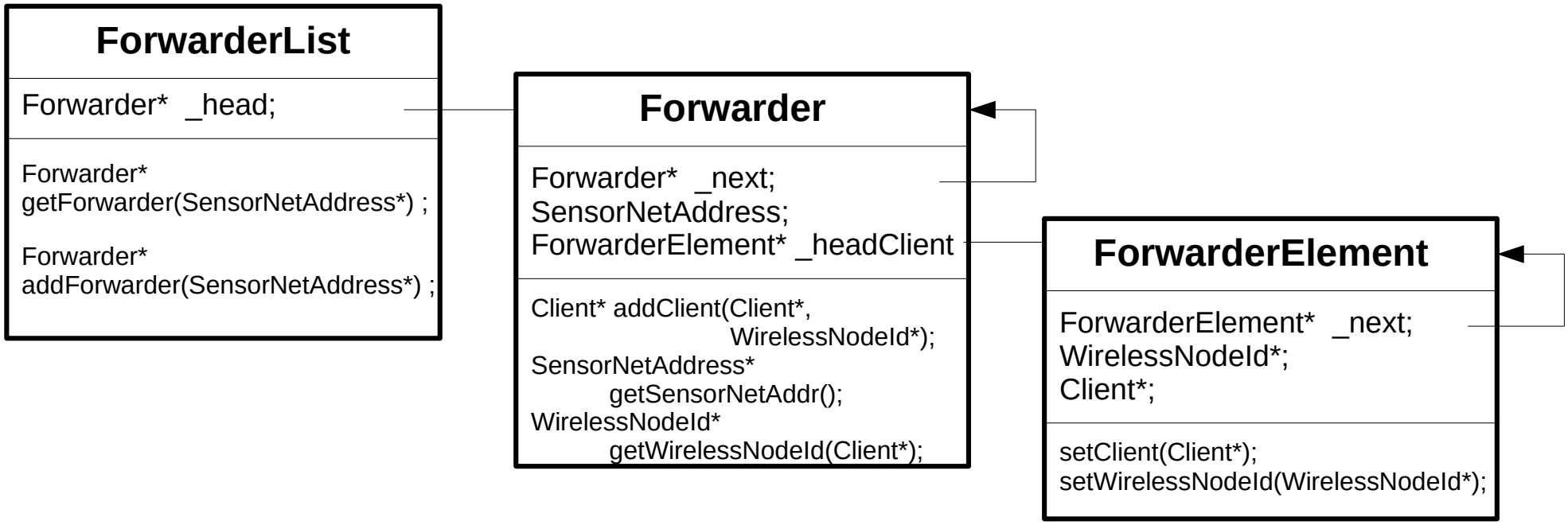
1-7. AdapterManager



1-8. Aggregater & Q0Sm1Proxy



1-9. Forwarder



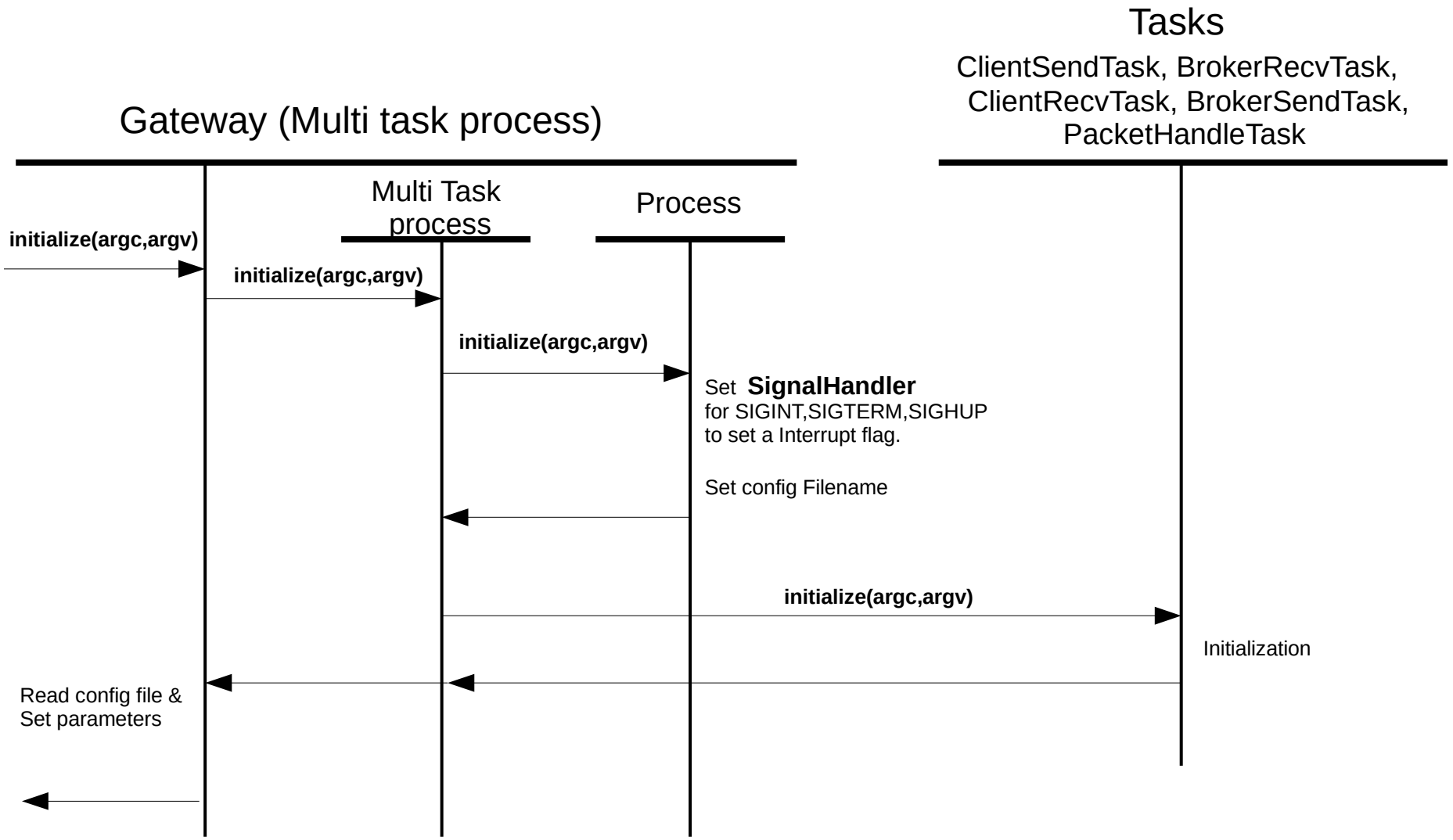
2. Application Framework

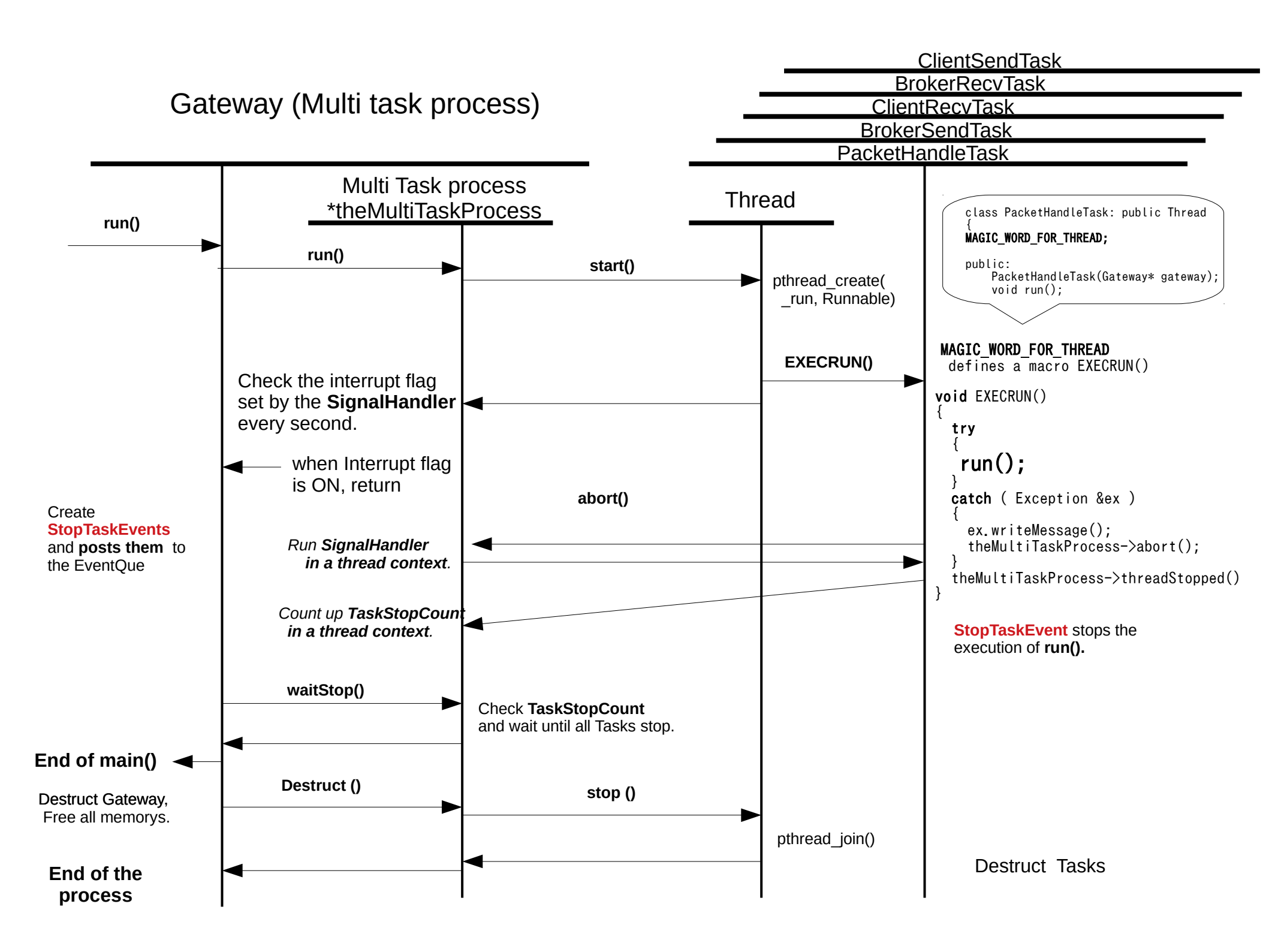
```
 /*  Gateway Application */

Gateway                 gateway;
PacketHandleTask    task1(&gateway);
ClientRecvTask        task2(&gateway);
ClientSendTask        task3(&gateway);
BrokerRecvTask       task4(&gateway);
BrokerSendTask       task5(&gateway);

int main(int argc, char** argv)
{
    try
    {
        gateway.initialize(argc, argv);
        gateway.run();
    }
    catch (Exception &ex)
    {
        ex.writeMessage();
        WRITELOG("ABORT Gateway!!!\n\n\n");
    }
}
```

2-1 Process execution




Gateway (Multi task process)
ClientSendTask
BrokerRecvTask
ClientRecvTask
BrokerSendTask
PacketHandleTask
Multi Task process
*theMultiTaskProcess
Thread
run()
run()
start()
pthread_create(
_run, Runnable)
class PacketHandleTask: public Thread
{
MAGIC_WORD_FOR_THREAD;
public:
PacketHandleTask(Gateway* gateway);
void run();
MAGIC_WORD_FOR_THREAD
defines a macro EXECRUN()
EXECRUN()
void EXECRUN()
{
try
{
run();
}
catch (Exception &ex)
{
ex.writeMessage();
theMultiTaskProcess->abort();
}
theMultiTaskProcess->threadStopped()
}
Check the interrupt flag
set by the SignalHandler
every second.
when Interrupt flag
is ON, return
abort()
Create
StopTaskEvents
and posts them to
the EventQue
Run SignalHandler
in a thread context.
Count up TaskStopCount
in a thread context.
StopTaskEvent stops the
execution of run().
waitStop()
Check TaskStopCount
and wait until all Tasks stop.
End of main()
Destruct ()
stop ()
Destruct Gateway,
Free all memorys.
pthread_join()
End of the
process
Destruct Tasks

2-2 Stop Task Event