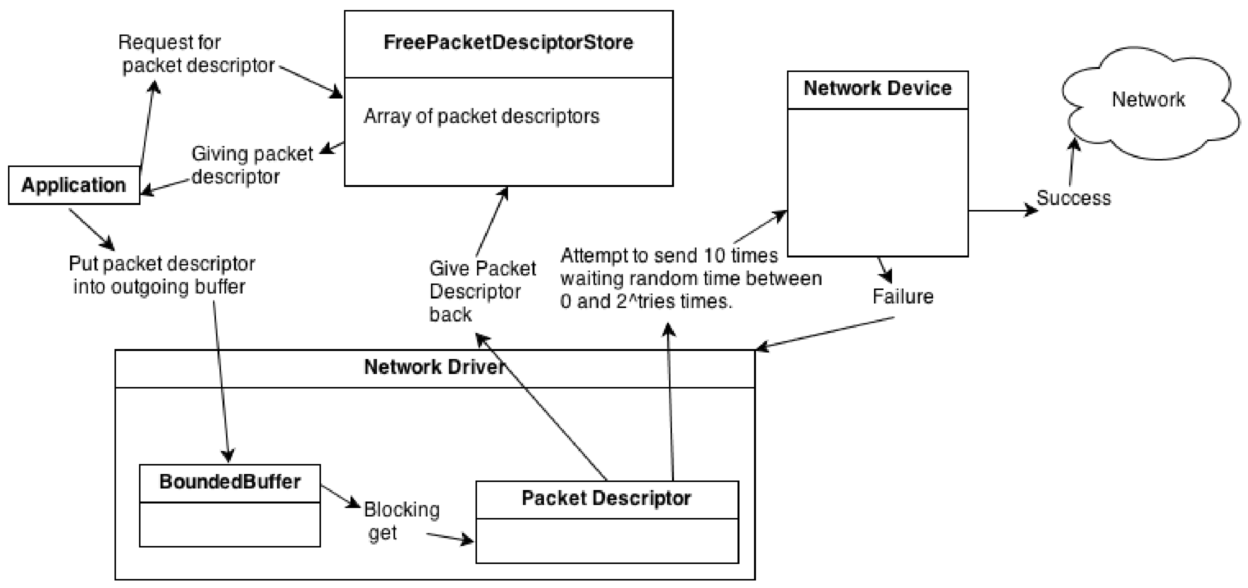
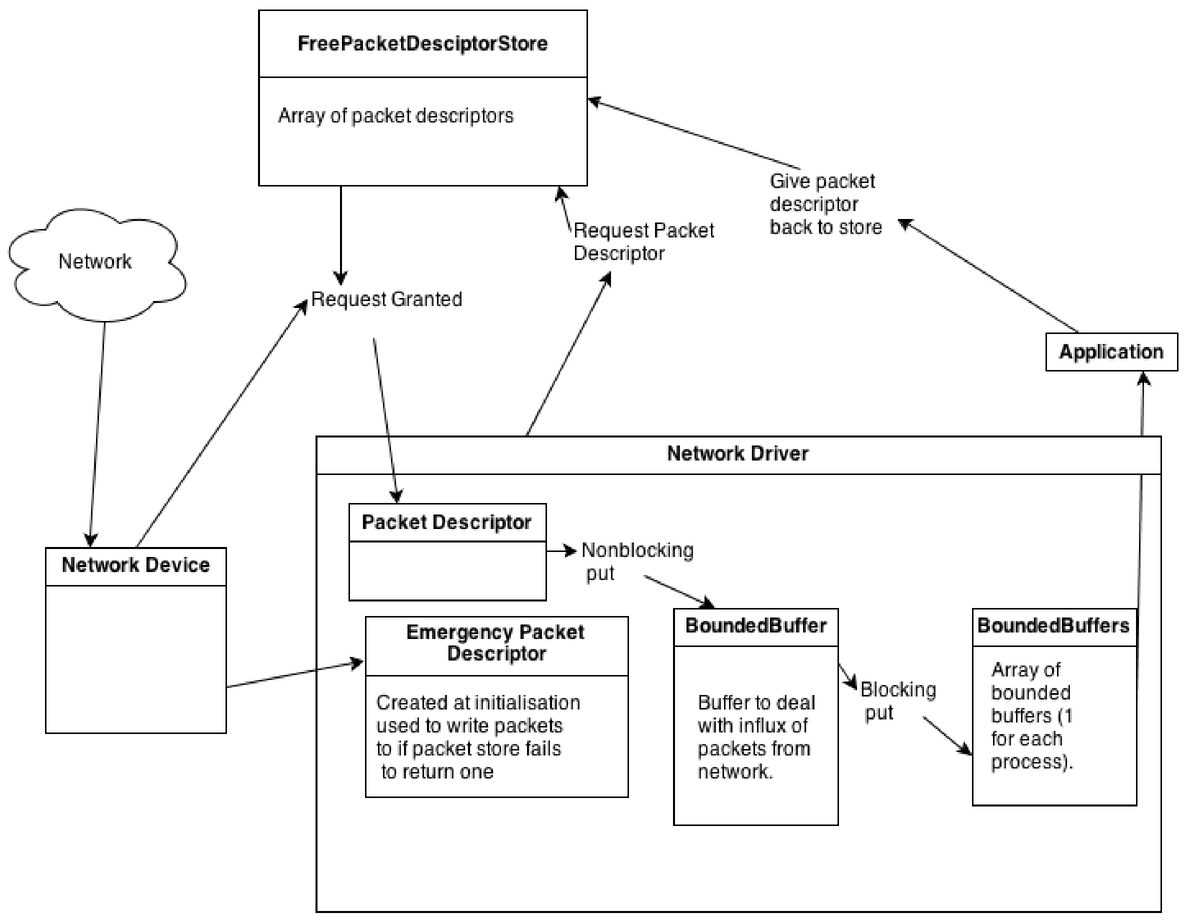
Applications Sending Packets



FreePacketDesciptorStore
Array of packet descriptors
Request for packet descriptor
Giving packet descriptor
Application
Put packet descriptor into outgoing buffer
Give Packet Descriptor back
Attempt to send 10 times waiting random time between 0 and 2^tries times.
Network Device
Network
Success
Failure
Network Driver
BoundedBuffer
Blocking get
Packet Descriptor

Applications Retrieving Packets



FreePacketDesciptorStore
Array of packet descriptors
Network
Request Granted
Request Packet Descriptor
Give packet descriptor back to store
Application
Network Driver
Packet Descriptor
Nonblocking put
Network Device
Emergency Packet Descriptor
Created at initialisation used to write packets to if packet store fails to return one
BoundedBuffer
Buffer to deal with influx of packets from network.
Blocking put
BoundedBuffers
Array of bounded buffers (1 for each process).

When receiving packets and finding a shortage of packet descriptors the driver will continually write over an emergency packet until a fresh packet descriptor becomes available. This means that if one were to be so inclined one would be able to handle this drop of packet, whereas if instead the driver say blocked until it had a descriptor the software would never be aware that it had missed a packet and could therefore never handle the loss.

My receiving thread never blocks except upon initialisation which in theory should never block as at start up there should always be free packet descriptors, so apart from this technicality this meets the requirements.

Upon receiving a packet from the network if the driver is successful in getting a packet descriptor it will attempt to non-blocking write into a bounded buffer, this is a non-blocking write so that this thread is never idle whilst it could be awaiting packets. If successful the packet will then be moved by a different thread by a blocking write into an array of bounded buffers (each of which represents the cubbyhole hole from each process goes to retrieve their packets) this is a blocking write because at this stage there is nothing to handle packet loss and this thread has nothing else to do but wait until it is able to write.

When attempting the send a packet there is no reason for the system not to blocking read from the buffers therefore my sending thread blocks until it is able to read from the outgoing buffer it then attempts to send the packet across the network, if this fails it backs of for a random number of microseconds in the range of 0 to (2 to the power of the number of tries). It will attempt to send 10 times after which it will discard the packet, blocking to put the descriptor back into the store. The process then begins again.