Author: Zhixin Liu
UO ID: 951452405
Date: 5/21/2016

Project_2 Report

networkdriver.c

- *State:*

- successfully compiles
- compiles with no warnings
- work correctly in multiple tests
- test output for 1 hour contains no error message including “Error”, “Warning”, or “BUG”

- *Design Description:*

I. <u>Three global BoundedBuffer:</u>

- **buffer_send**: used when application send packet to network device via network driver
- **buffer_receive[PID_MIX + 1]**: used when network device send packet to specific application, each application hold an independent receive buffer
- **buffer_pool**: the storage of non-used packet descriptor, driver would first try to get new packet descriptor or return non-used packet descriptor to this pool, otherwise to the free store of packet descriptor

II. <u>Send side:</u>

- Providing non-blocking and blocking function for application to write packet to send buffer in driver
- Proving sending thread when driver sending packet to network device:
  1. enter the infinite loop
  2. use blocking read function to read packet descriptor from send buffer, blocking ensure the driver would get a packet descriptor successfully
  3. send the packet, continue trying the send_packet() until success or reach max trying time limitation
  4. return the non-used packet descriptor to pool via nonblocking write, if failed, pool is full and return the non-used packet descriptor to free store via nonblocking put function, nonblocking calls ensure the thread not block in return packet descriptor

III. <u>Receive side:</u>

- Providing non-blocking and blocking function for application to read packet from receive buffer in driver
- Providing receiving thread when driver receiving packet from network device:
  1. contains two packet descriptor pointer variable, one for current use , on for back-up

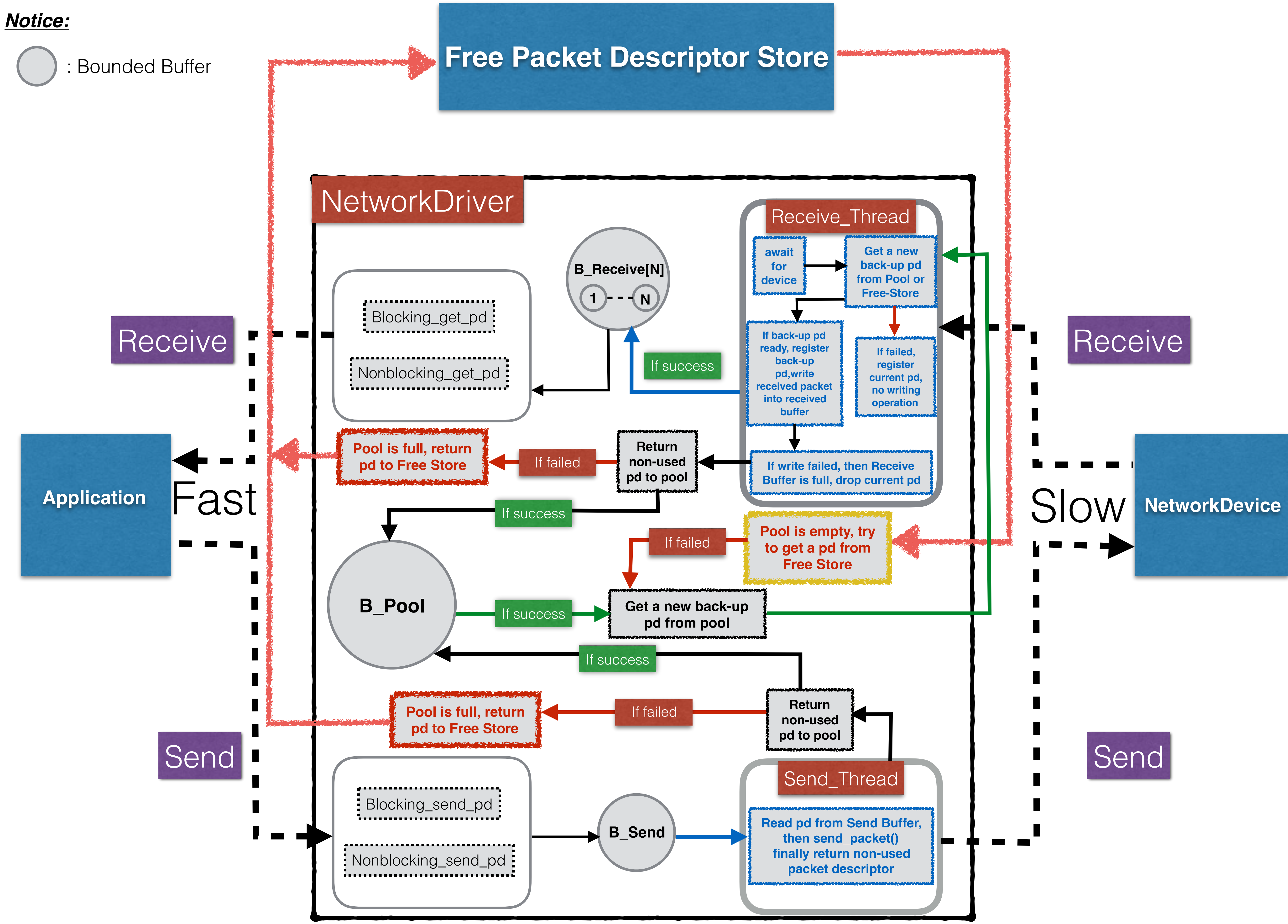
2. use blocking put function to get first packet descriptor from free store to start, initial and register this packet descriptor, blocking ensure the driver would get a packet descriptor successfully
3. enter the infinite loop
4. call await function to blocking wait for receiving incoming packet from the network device
5. ***Notice: all function calls in following part 6 are nonblocking to ensure the receiving thread never block here, because we wish not only to make network device always hold an empty packet descriptor to carry incoming packet, but also to receive packet as soon as possible and limit the packet loss to minimum.***
6. once the driver received packet, try to use nonblocking read function to read a new packet descriptor for back-up from the pool buffer, if failed, pool is empty, try it from the free store via non-blocking get function:
   - if cannot get a back-up packet descriptor, no writing operation, just initial and register the current packet descriptor we have, then continue the loop
   - if can get a back-up packet descriptor, first to initial and register the back-up packet descriptor to tell network device use this backup-one to carry next packet, then try to use non-blocking write function to write current packet descriptor into corresponding receive buffer (that is: first get the PID of the packet, then call non-blocking write function to write into the corresponding receive buffer):
     - if success, just make the pointer of our current packet descriptor point to the back-up one and continue the loop
     - if failed, the current packet descriptor become non-used, return it to the pool by using nonblocking write function, if it failed, pool is full and return the non-used packet descriptor to free store via non-blocking put function; lastly, make the pointer of the current packet descriptor point to the back-up one, and continue the loop

- *Explanation:*

1. Behavior when there is a shortage of PacketDescriptor for receiving packets:
   - In this case, the receiving thread in driver would never write received packet into receive buffer until it get an available back-up packet descriptor; what the receiving thread would do is just initial and register the current packet descriptor to the network device, and repeat trying to get an available back-up packet descriptor from the pool or the free store.
2. Blocking behavior matching the requirements:
   - Please see the blocking and non-blocking detail in the Send_Side and Receive_Side of Design Description and the Diagram below.

- *Diagram:*

(Notice: the detail of blocking and non-blocking behavior is addressed in Design Description part, the Diagram just show the basic workflow of the driver)


Notice:
: Bounded Buffer
Free Packet Descriptor Store
NetworkDriver
Receive_Thread
await for device
Get a new back-up pd from Pool or Free-Store
If back-up pd ready, register back-up pd,write received packet into received buffer
If failed, register current pd, no writing operation
If write failed, then Receive Buffer is full, drop current pd
B_Receive[N]
1
N
If success
Blocking_get_pd
Nonblocking_get_pd
Receive
Receive
Application
Fast
Slow
NetworkDevice
Pool is full, return pd to Free Store
If failed
Return non-used pd to pool
If success
If failed
Pool is empty, try to get a pd from Free Store
B_Pool
If success
Get a new back-up pd from pool
If success
Pool is full, return pd to Free Store
If failed
Return non-used pd to pool
Send
Send
Send_Thread
Blocking_send_pd
Nonblocking_send_pd
B_Send
Read pd from Send Buffer, then send_packet() finally return non-used packet descriptor