# Question I:

I know that for a network card, OS must allocate tx/rx rings for it so that when OS wants to receive/transmit packets, the network card will know where the packets are and which packets are to be transmit.

And when I read about DMA, I see something named DMA ring buffer. Are the DMA ring and tx/ rx ring the same thing? Or what is the relationship?

Answer: -

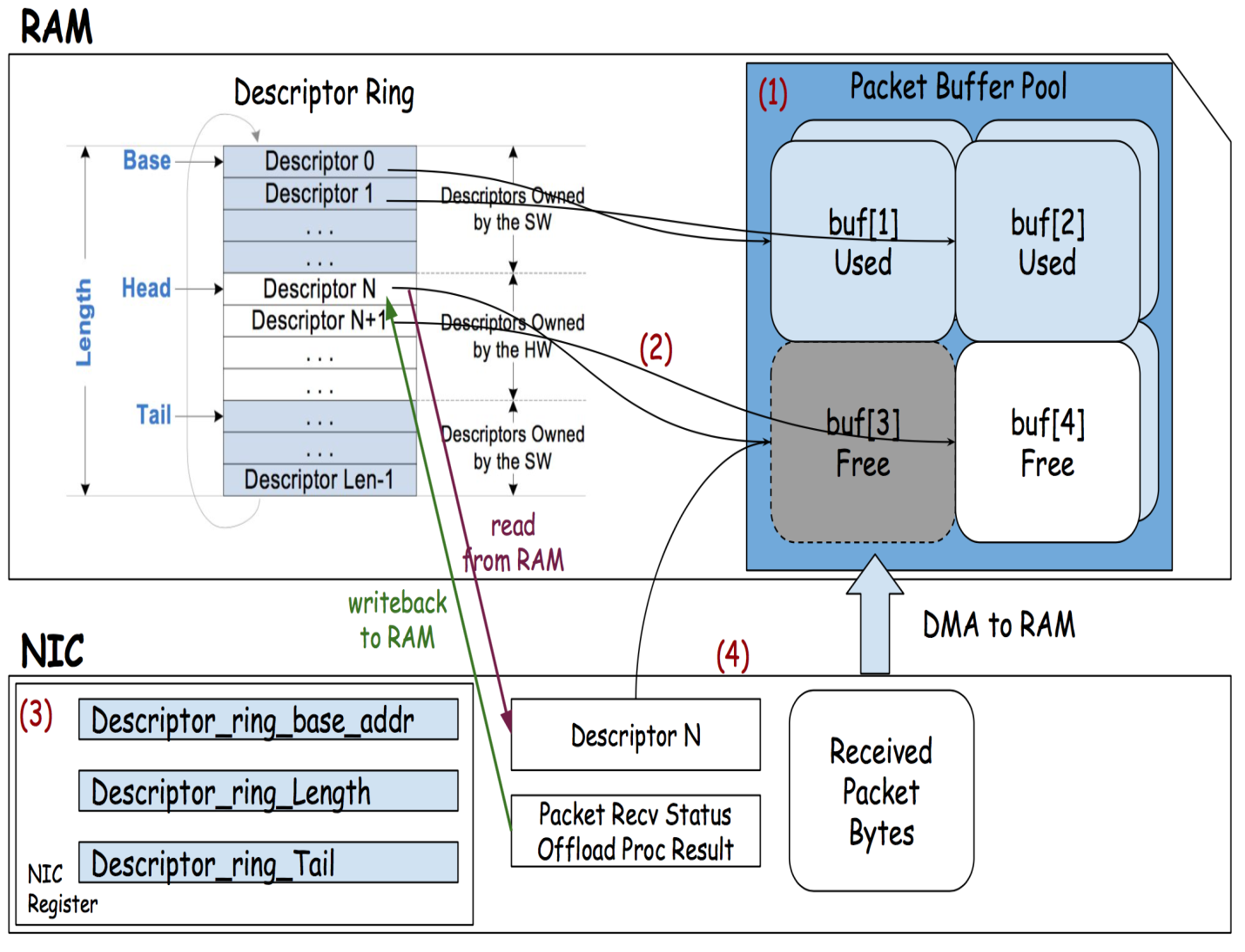
Reference: - <https://stackoverflow.com/questions/47450231/what-is-the-relationship-of-dma-ring-buffer-and-tx-rx-ring-for-a-network-card>

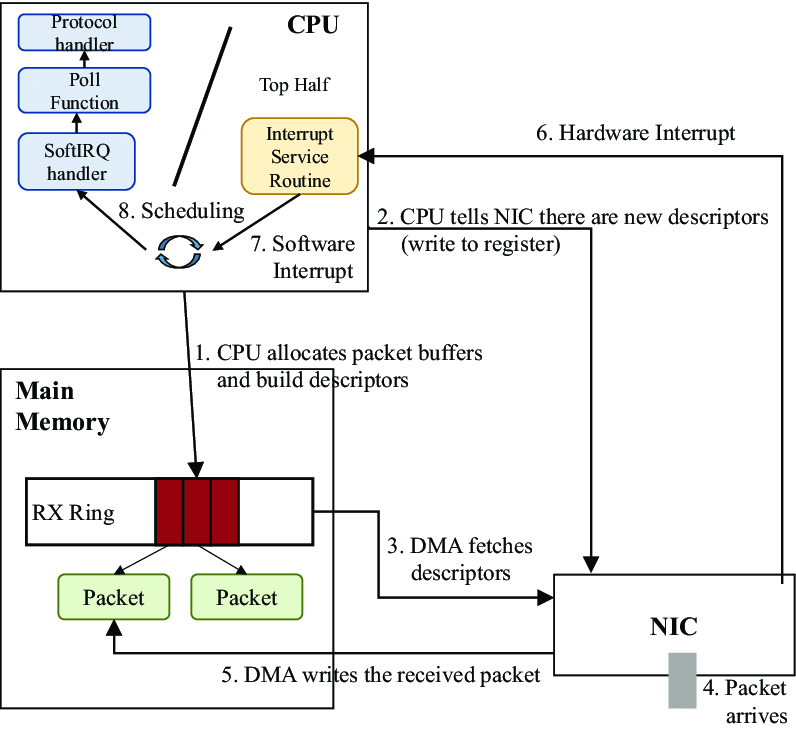
1. Ring Buffer Contains Start and End Address of Buffer in RAM. TX Ring will contain addresses of Buffer in RAM that contains data to be transmitted. RX Ring will contains address of Buffer in RAM where NIC will place data.

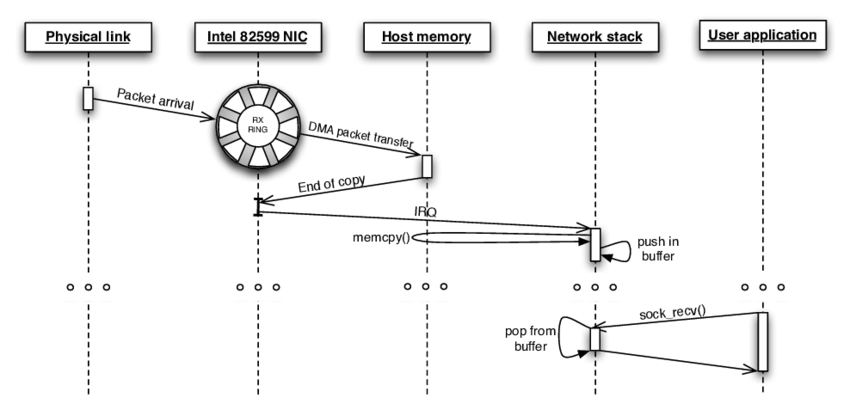
These rings are present in RAM.

1. TX buffer and RX buffer are are in RAM pointed by TX/RX rings.
2. Now Network Card Register has Location of Rings Buffer in RAM .

If 1 and 2 are DMA able buffer, they are called DMA TX/RX ring and DMA TX/RX buffer. Now since RX/TX ring must remain throughout they are made as consistent/coherent DMA type of memory. While Buffers are made streaming/Single DMA type of memory.







Network cards often expect to see a circular buffer (often called a DMA ring buffer) established in memory shared with the processor; each incoming packet is placed in the next available buffer in the ring, and an interrupt is signaled. The driver then passes the network packets to the rest of the kernel, and places a new DMA buffer in the ring.

The DMA ring allows the NIC to directly access the memory used by the software. The software (NIC's driver in the kernel case) is allocating memory for the rings and then mapping it as DMA memory, so the NIC would know it may access it. TX packets will be created in this memory by the software and will be read and transmitted by the NIC (usually after the software signals the NIC it should start transmitting). RX packets will be written to this memory by the NIC and will be read and processed by the software (usually after an interrupt is issued to signal there's work).

DMA ring buffer and TX/RX rings are the same thing.  
DMA has two type of ring buffers

* TX ring buffer - used for transmitting data from kernel to device
* RX ring buffer - used for receiving data from device to kernel

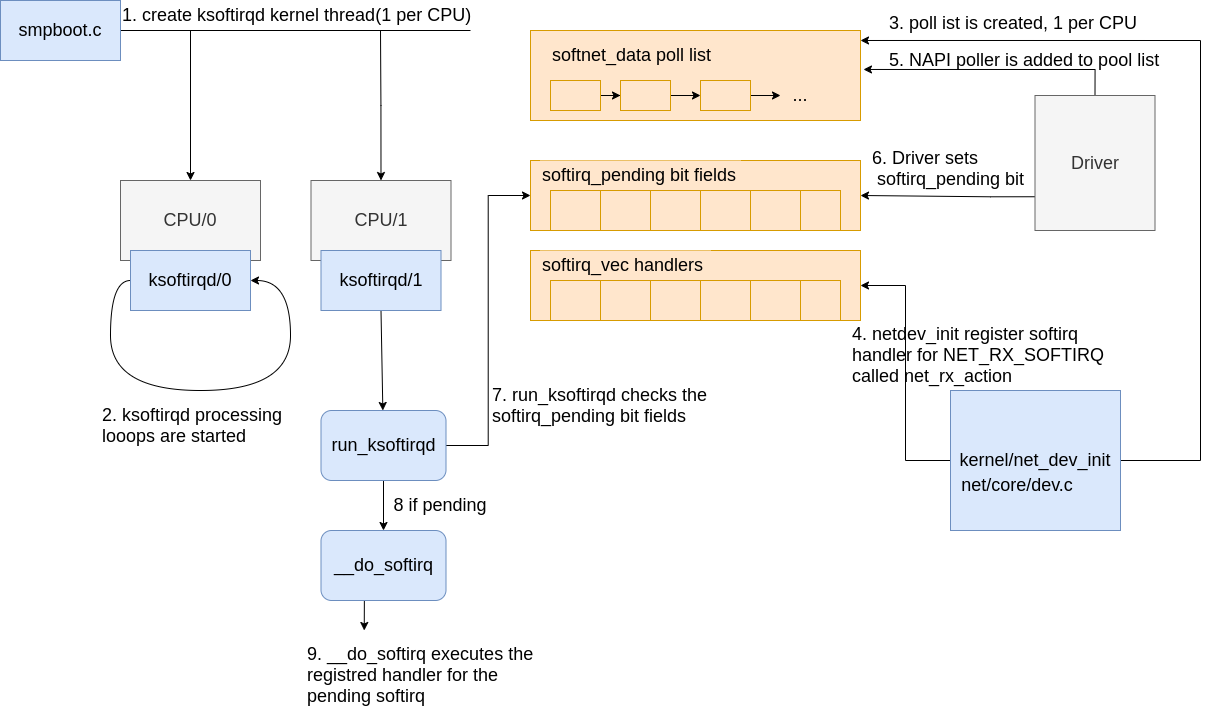
# Question II: Network driver and its Softirq

TODO

References: -

<https://www.lmax.com/blog/staff-blogs/2016/05/06/navigating-linux-kernel-network-stack-receive-path/>

<https://maxnilz.com/docs/004-network/005-linux-rx/>



1. Data is received by the NIC from the network.
2. The NIC uses DMA to write the network data to RAM.
3. The NIC raises an IRQ.
4. The NIC driver’s [registered IRQ handler is executed](https://packagecloud.io/blog/monitoring-tuning-linux-networking-stack-receiving-data/#interrupt-handler).
5. The IRQ is cleared on the NIC, so that it can generate IRQs for new packet arrivals.
6. NAPI softirq poll loop is started with [a call to napi\_schedule](https://packagecloud.io/blog/monitoring-tuning-linux-networking-stack-receiving-data/#napi-and-napischedule).

The call to napi\_schedule triggers the start of steps 5 - 8 in Fig.3 above. As we’ll see, the NAPI softirq poll loop is started by simply flipping a bit in a bitfield and adding a structure to the poll\_list for processing. No other work is done by napi\_schedule and this is precisely how a driver defers processing to the softirq system.

Continuing on to the diagram in the Fig.3, using the numbers found there:

1. The call to napi\_schedule in the driver adds the driver’s NAPI poll structure to the poll\_list for the current CPU.
2. The softirq pending bit is set so that the ksoftirqd process on this CPU knows that there are packets to process.
3. run\_ksoftirqd function (which is being run in a loop by the ksoftirq kernel thread) executes.
4. \_\_do\_softirq is called which [checks the pending bitfield](https://packagecloud.io/blog/monitoring-tuning-linux-networking-stack-receiving-data/#dosoftirq), sees that a softirq is pending, and calls the handler registered for the pending softirq: net\_rx\_action which does all the heavy lifting for incoming network data processing.

It is important to note that the softirq kernel thread is executing net\_rx\_action, not the device driver IRQ handler.