OS Lab – 5

Madhav Bhardwaj (CS23B035)

September 2025

Explanation

- **kernel/e1000.c**
  - **e1000_transmit()**: This function programs the Transmit (TX) Descriptor . It must first check the DD (Descriptor Done) status bit on the current TDT (TX Descriptor Tail) slot before reusing it. the buffer from the previous transmission on that slot is freed (kfree) once DD is observed. The new packet's address and length are then written, EOP (End-of-Packet) and RS (Report Status) bits are set in the command field, and TDT is advanced to signal the NIC.
  - **e1000_recv()**: This function drains the Receive (RX) Descriptor by iterating over descriptors that have both DD and EOP set. For each valid packet:
    1. The filled buffer and packet length are saved.
    2. A fresh buffer is allocated (kalloc()) and installed into the ring slot to maintain the flow of reception.
    3. The RDT (RX Descriptor Tail) is advanced.
    4. The e1000_lock is **released** before calling net_rx() (which may block or acquire other locks) and immediately **reacquired** after, preventing a nested-lock deadlock.

```
int e1000_transmit(char *data, int length)
{
  acquire(&e1000_lock);
  uint32 tail_index = regs[E1000_TDT];
  struct tx_desc *tx_entry = &tx_ring[tail_index];
  int wait_cycles = 0;

  while ((tx_entry->status & E1000_TXD_STAT_DD) == 0)
  {
    // ... spin loop with timeout ...
    if (++wait_cycles > 1000000)
    {
      release(&e1000_lock);
      return -1;
    }
```

```
  }

  if (tx_bufs[tail_index])
  {
    kfree(tx_bufs[tail_index]);
    tx_bufs[tail_index] = 0;
  }

  tx_entry->addr = (uint64)data;
  tx_entry->length = (uint16)length;
  tx_entry->cmd = E1000_TXD_CMD_EOP | E1000_TXD_CMD_RS;
  tx_entry->status = 0;
  tx_bufs[tail_index] = data;

  __sync_synchronize();
  regs[E1000_TDT] = (tail_index + 1) % TX_RING_SIZE;

  release(&e1000_lock);
  return 0;
}

static void e1000_recv(void)
{
  acquire(&e1000_lock);

  for (;;)
  {
    uint32 current_index = (regs[E1000_RDT] + 1) % RX_RING_SIZE;
    struct rx_desc *rx_entry = &rx_ring[current_index];

    if ((rx_entry->status & E1000_RXD_STAT_DD) == 0)
      break;

    if ((rx_entry->status & E1000_RXD_STAT_EOP) == 0)
    { /* handle fragments and continue */ }

    int packet_length = rx_entry->length;
    char *packet_data = rx_bufs[current_index];
    char *new_buffer = kalloc();

    if (new_buffer == 0) { /* handle kalloc failure */ }

    rx_bufs[current_index] = new_buffer;
    rx_entry->addr = (uint64)new_buffer;
    rx_entry->status = 0;
    regs[E1000_RDT] = current_index;
```

```
    release(&e1000_lock);
    net_rx(packet_data, packet_length);
    acquire(&e1000_lock);
  }
  release(&e1000_lock);
}
```

Test results

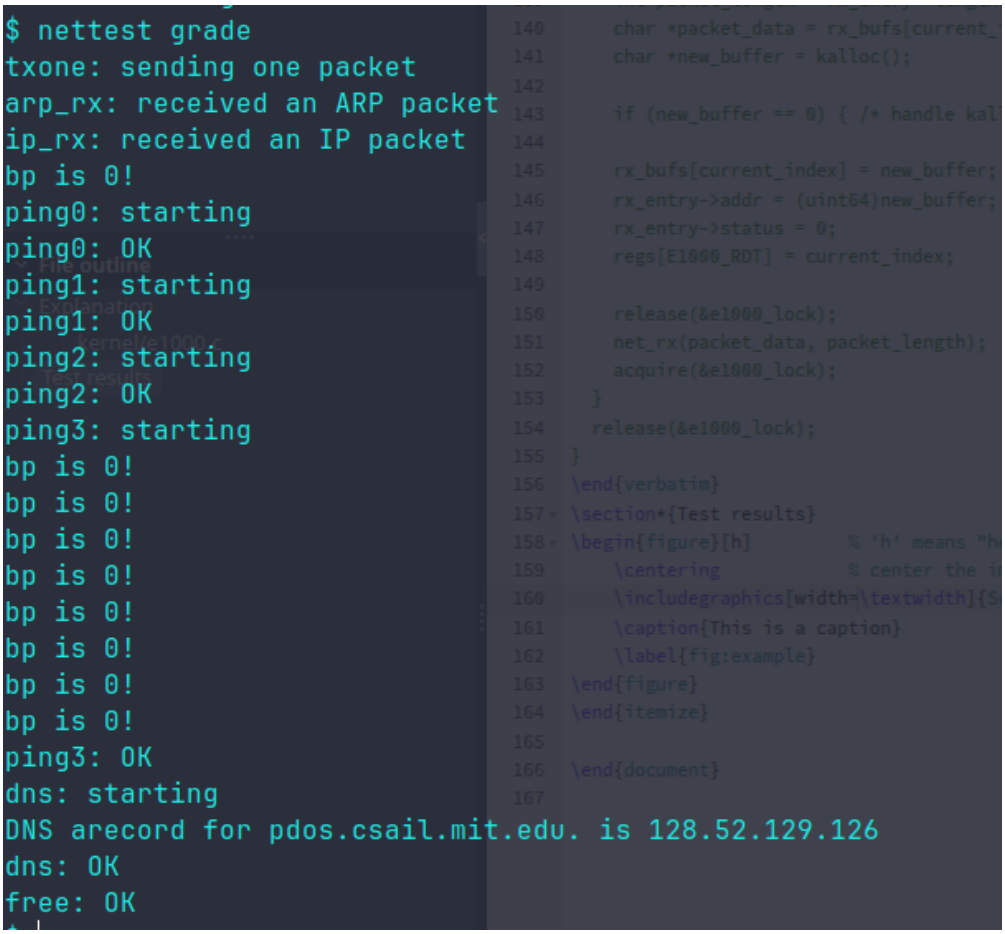

Figure 1: Test results