**ELC 2080**

**Network Communication Simulation (First Draft)**

|  |  |  |  |
| --- | --- | --- | --- |
| BN | Section | Student ID | اسم الطالب |
| **37** | **1** | **9230242** | **أمير اشرف لولي عباس** |
| **38** | **1** | **9230243** | **أمير سامح فانوس عطية** |

# System Design

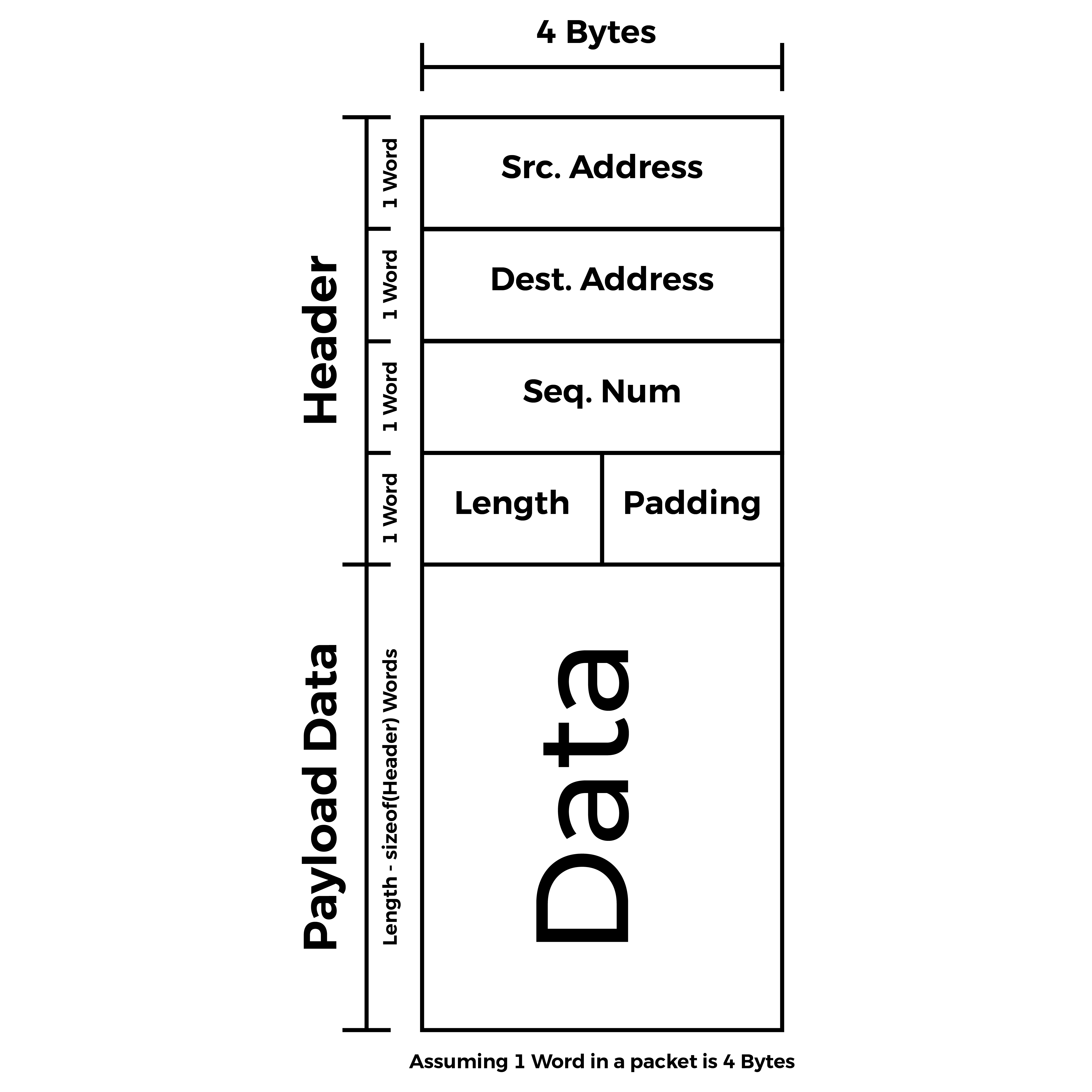
The system consists of 5 main nodes , 2 sender nodes , 2 receiver nodes , 1 switch node (figure 2). Each sender node is required to generate a packet every 200ms attaching with it the source it has been sent from (node 1 or node 2) and the destination where it should be received (node 3 or node 4) and the sequence number of this packet then the packet is forwarded to the switch node.

The switch node receives the packet in a queue and sends it to the desired destination after 200ms of receiving it from the sender. Considering the switch to be a lossy communication link, it should have the probability of losing some packets while sending it to each destination. Also we may account for some error where the switch may send the data to the wrong receiver node.

Each receiver node is forwarded a packet from the switch node, then it checks whether this packet is in the right destination (if this is not the case the receiver may drop a log error). Also the receiver determines the number of lost packets using the sequence number attached with each packet.

**Packet implementation:**

The packet consists of a Header: Containing Source Address, Destination Address, Sequence number and a variable containing the entire length of the packet. The remainder of the packet is allocated for the payload data having a random length for each generated packet. Making the total Length of the packet randomly generated between L1 and L2 defined in the code as 1000 Bytes and 2000 Bytes



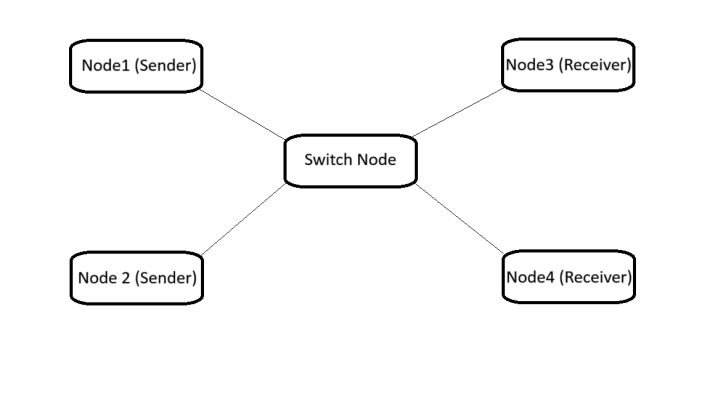


Figure 1: packet struct Format Figure 2: System Overall Design

**Nodes implementation:**

In order to implement this system we need to define a struct to hold each node properties such as current task , queues , timers and semaphores. Also we need 3 main tasks (1 for senders, 1 for receivers , 1 for the switch).

Each sender node is attached to the sender task and has 1 timer for the packet generation (each 200ms) and 1 queue to store the packets to be forwarded.

Each receiver node is attached to the receiver task and has 1 queue for received packets.

The switch node is attached to it’s task and has 1 timer for the forwarding delay (200ms) and 1 queue to store received packets to be forwarded to their destinations.

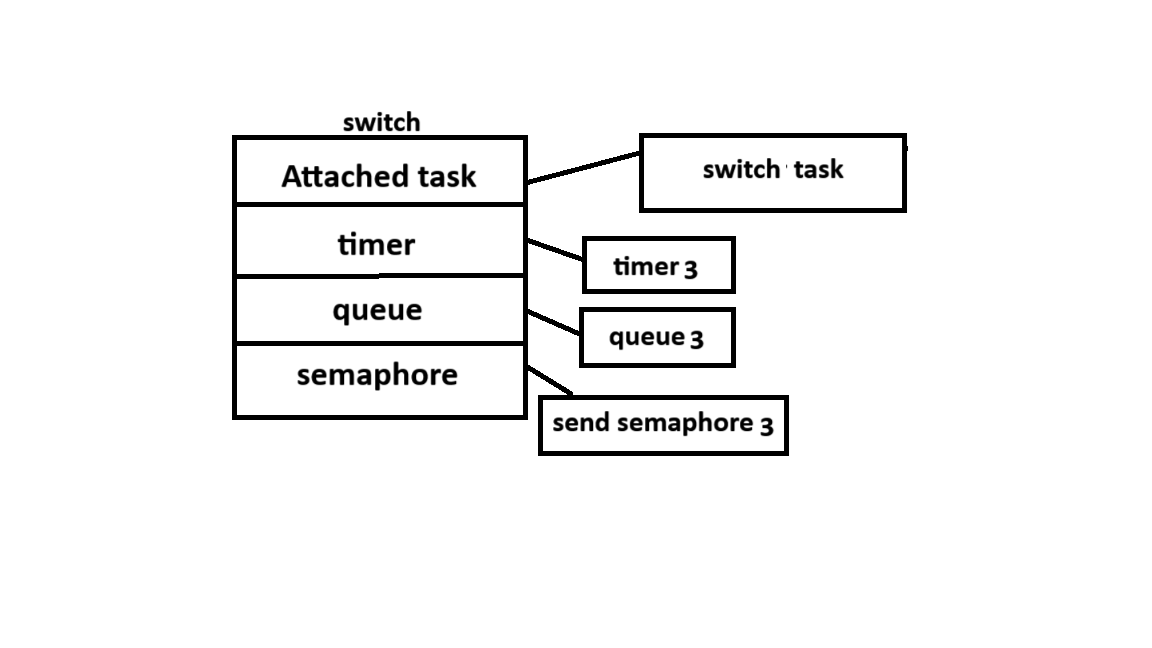
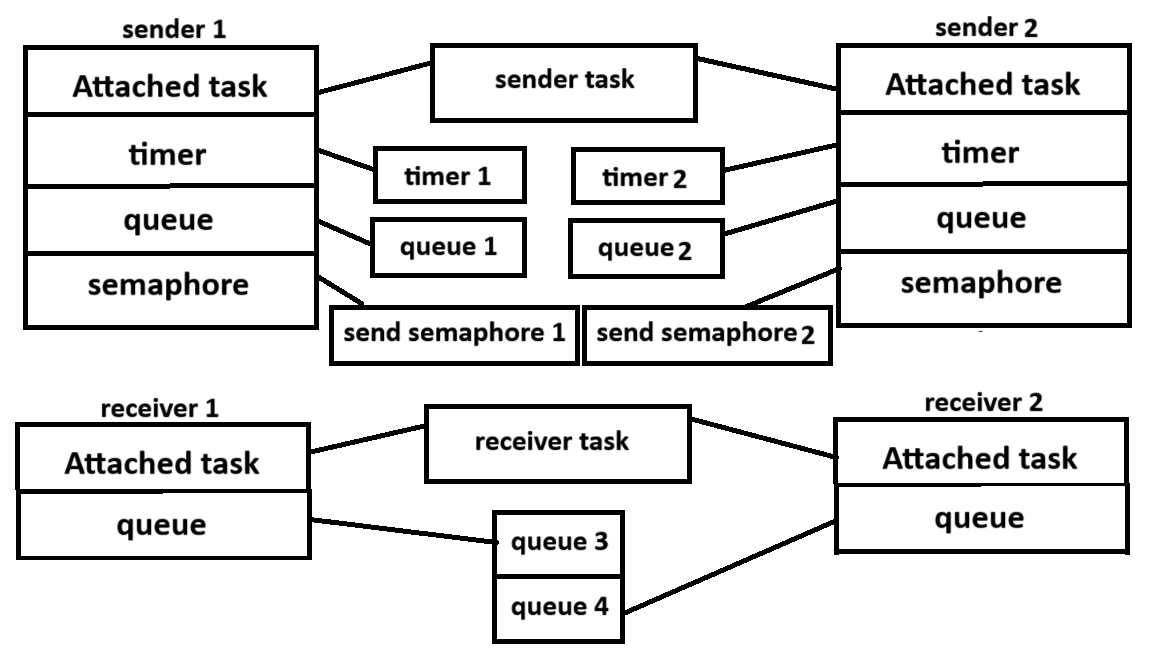
****

Figure 3: Nodes structs Format and their attached objects

The sender task starts the timer and attempts to take the SendData Semaphore (Which is initialized as taken) and remains blocked until the timer executes its callback and gives the semaphore allowing the task to generate the packet to a random receiver with a random length and sends it to the switch (This is how the periodic data send is implemented), additionally a small delay as added at the beginning for node 1 such that both tasks don’t start sending at the same time.(sender queues may be used for ACKs in the future).

The switch task is waiting until its queue gets some packet, then it calculates the drop probability and decides whether to drop the packet or forward it. If the packet is to be forwarded the switch starts its timer to wait 200ms until the switch semaphore allows the task to continue in forwarding the packet. A last additional probability is calculated upon which the switch decides whether to send the packet to the right destination or to account for false destination error. After that the packet is forwarded to its corresponding destination considering the destination error that might appear.

The receiver task waits until one of the queues gets some packet then it checks whether the received packet is in the right destination or not. If the packet is in the right destination the task compares the sequence number of this packet and compare it with the last received packet from the same sender to count the number of lost packets. If the packet is in the wrong destination the receiver logs some error and stores the number of wrong received packets from the same sender.

The switch frees dropped packets as well as the receiver frees all packets after processing to prevent memory leaks.

**The packet drop and false destination probability:**

the total number of lost packets has to be about 1% of the total number of sent packets, so an additional function is called in the switch task before the packet is forwarded to it’s destination. If the function returned true the packet is dropped, otherwise it is forwarded. The function implementation is about generating some random value and compare it with the drop probability p=0.01.

False destination probability is applied the same way as the drop one but with p=0.005

# Results and Discussion

To test the system we counted the packets received at each receiver node and printed each node statistics after it receives 2000 packet in total from the 2 senders combined. This test was applied without the false destination probability ,The results are shown in table 1.

Table 1: node 3,4 statistics

| statistics | Node 3 | Node 4 |
| --- | --- | --- |
| Total packets sent from node 1 | 998 | 993 |
| Total packets received from node 1 | 991 | 980 |
| Total packets lost from node 1 | 7 | 13 |
| Lose rate from node 1 | 0.7% | 1.3% |
| Total packets sent from node 2 | 1002 | 1007 |
| Total packets received from node 2 | 986 | 999 |
| Total packets lost from node 2 | 16 | 8 |
| Lose rate from node 2 | 1.59% | 0.79% |
| Total sent from nodes 1,2 combined | 2000 | 2000 |
| Total received from nodes 1,2 combined | 1977 | 1979 |
| Total lost from nodes 1,2 combined | 23 | 21 |
| Total lose rate | 1.1% | 1.05% |

From table 1 we deduce that the total packets received by the switch are 4000 packets. The total packets dropped by the switch are 44 packets.

The lose rate is 1.1%

If we increased the sample to be more than 4000 packets the lose rate gets closer to 1%

We ran another simulation with the addition of the false destination probability, the results are shown in table 2.

Table 2: node 3,4 combined statistics

| statistics | Node 3 &4 combined |
| --- | --- |
| Total packets sent | 4000 |
| Total packets received | 3940 |
| Total packets lost | 60 |
| Total diverted packets | 18 |
| Total dropped packets | 42 |
| Total drop rate | 1.05% |
| Total diversion rate | 0.45% |

# Design demonstrations:

1. Structs decleration:

typedef uint8\_t Payload\_t;

typedef struct {

QueueHandle\_t sender;

QueueHandle\_t reciever;

SequenceNumber\_t sequenceNumber;

uint16\_t length;

uint16\_t padding; // To make sure the header 16 bytes

} header\_t;

typedef struct {

header\_t header;

Payload\_t\* data;

} packet;

typedef struct {

TaskHandle\_t\* CurrentTask;

QueueHandle\_t CurrentQueue;

TimerHandle\_t CurrentTimer;

TimerHandle\_t ACKToutTimer;

SemaphoreHandle\_t SendDataSema;

} NodeType\_t;

1. packet drop and false destination probability:

BaseType\_t checkProb(double prob)

{

if(prob <= 0.0 || prob >= 1.0)

{

return pdFALSE;

}

double random\_value = (double)rand() / RAND\_MAX;

if(random\_value < prob)

{

return pdTRUE;

}

else

{

return pdFALSE;

}

}