Commission phase: Join end device

1. Router as communication center (boundary device)
   Connect to web server (django server)
   Package refer to Zigbee protocol
1. Pivot (Zigbee device)
   Authority
1. End Device (Zigbee device)
   Beacon broadcast to request join

OTA firmware update

1. Pivot send starting packet (end device handshake)
2. Pivot send new firmware segment by segment
3. End device checks after complete
4. Request the lost segment to pivot

For this work, what language did you use … is it C ?

C for the firmware, python for django server

OTA firmware

End device will have some code

Pivot has some code

End Device Code

```
If (OTA start beacon rcv){
        Pkt = rcv()
        if( auth(Pkt) ){ // magic number
                Seq = decode( Pkt)
                Write Pkt in the i-th place in flash
        }
}
if( OTA end beacon){
        Check completeness of flash // ECC
        if( segment i is lost){
                Send resend request to pivot
        }
}
```

Pivot Code

```
Send OTA start beacon()
// new fw is store in flash
for( all segment i in flash){
        i = add_informtion(i) // sequential number, magic number
        Broadcast (i)
}
Send OTA end beacon
If (re-resend()){
        Lost number = i
        send (packet i)
}
```

The requirement is to implement a web-server. How would you do it using sockets….just pseudo code.

Main loop to receive requests from clients, send response and go back to receiving...how will this look like

```
Server()
        listen(port)
        while(1){
                Request = listen.receive()
                Hand
        }
```

What did you do in django?

```
<div >
        <div id=switch>
                <div id=button value=ON></>
        </>
</>
```

File:
Packet 1: <dst-addr>, <src-addr>
Packet 2:
…

Packet n:

10000 million packets in the file

Dst-addr is 32-bits (4 bytes)
Src…

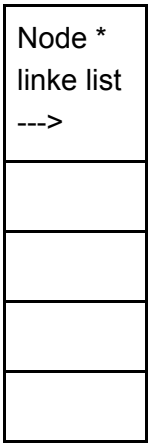
1000 unique combination of dst-addr, src-addr, count number of packets for each 1M unique pair?

```
Structure info{
        Int dst;
        Int src;
        Int cnt;
}
unordered_map<int,vector<info>> m;
for(i=0;i<file.size();i++){
        Dst = packet[i].dst;
        info.src = packet[i].src;
        Info.cnt = 1;
        if(m.find(making_pair<dst, src>) != m.end()){
                if( !m[dst].find(src) ){
                        m[dst].append(src);
                }else{
                        Cnt ++;
                }
        }else{
                m.insert(making_pair<dst, src>);

        }
}
```

If you have to implement map (or a hash table). How do you do it?
Key, value… pseudo code of map

```
Node {
        Int key;
        Int value;
        Int index;
}

Class map{
```

```
        Node i;
        Int Tree[];
        Node Root;

        map(){
                Root.index = 0;
        }

        Private:
        rotate();

        public:
        insert(node){
                Node tmp = root;
                Int Index = tmp.index;
                while(tmp != null){
                        if( array[index] < node.key){
                                Tmp = array[index*2+1]
                        }elif (array[index > node.key]){
                                Tmp = array[index*2+2]
                        }
                }
                Array[index] = node;
        }

        find(){

        }

}

Node *node_table[prime_numnber];

insert(node)
{
        index = hash(node.key) % prime_number;
        Head = &Node_table[index];
        While (tail.node != u
}
```



Each element in hahs table is a linked list
Node -> node -> node

No collisions - linked list will have 1 elemenet onely

Have you worked on any device drivers ?

How did you use I2C ? pseudo code for how you used I2C

I2C.freq()
I2C.bitrate()

What did you in school project for Linux kernel ?
Did you do a project for fifo scheduling?

```
Struct p{
        Int pid;
        Int ticket;
}

Struct ptable{
        Int MAX_num
        Struct p ptable[MAX_num]
}
```

Each process:
        process(100 ticket)

```
        process(200 ticket)
        process(300 ticket)

In scheduler:
        while(1){
                For (i in MAX_num)
                        Total_ticket += p[i].ticket
                Random (1:Total_ticket)
        }
```