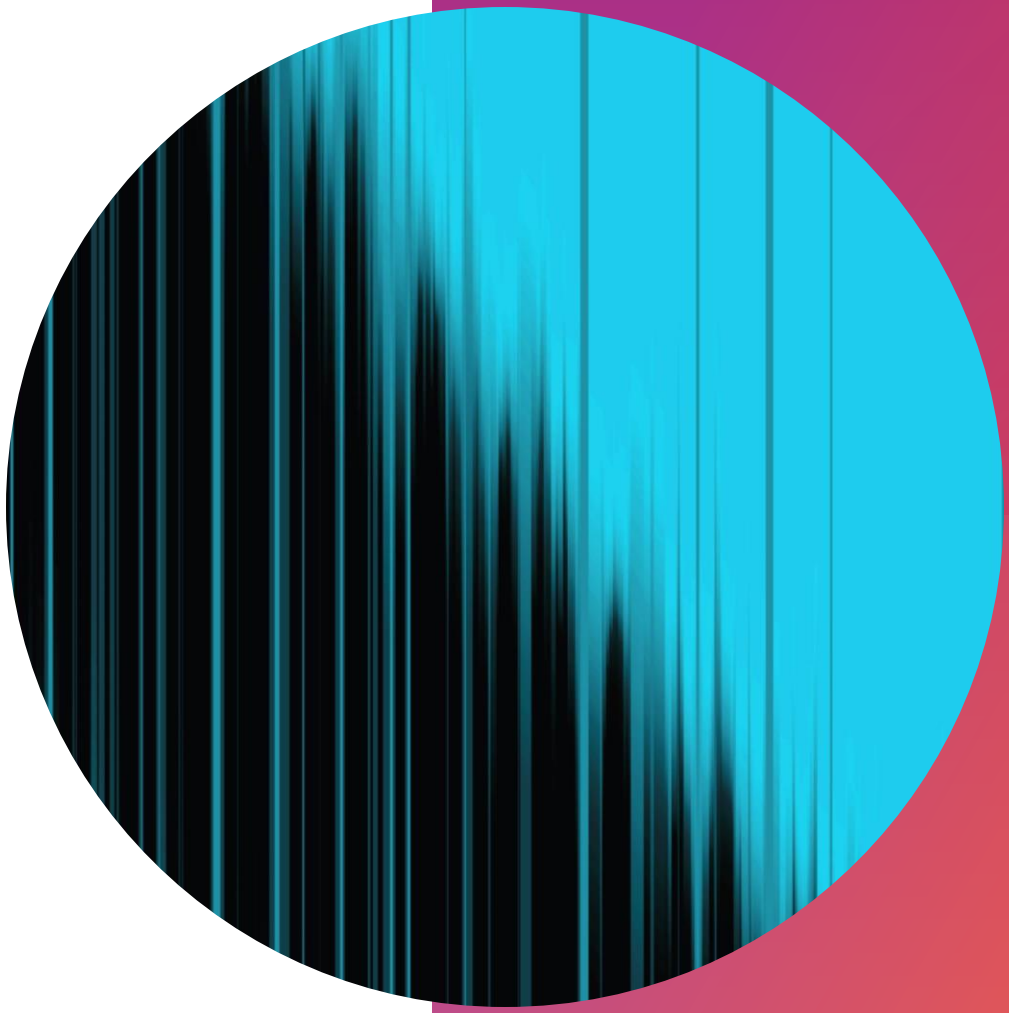
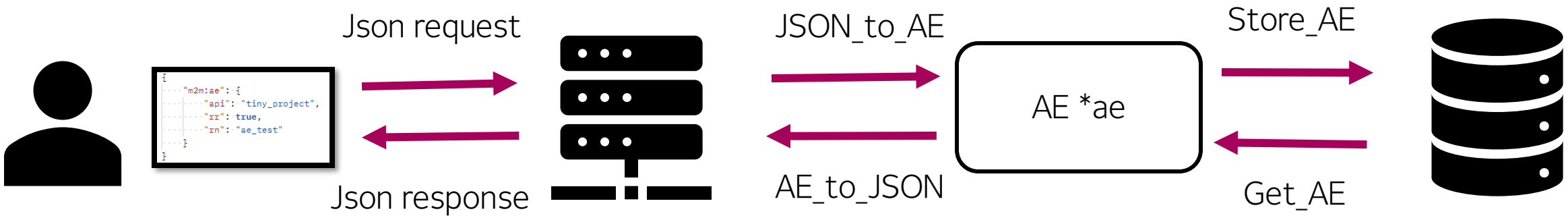
HTTP Daemon



ONEM2M TINY IOT PROJECT

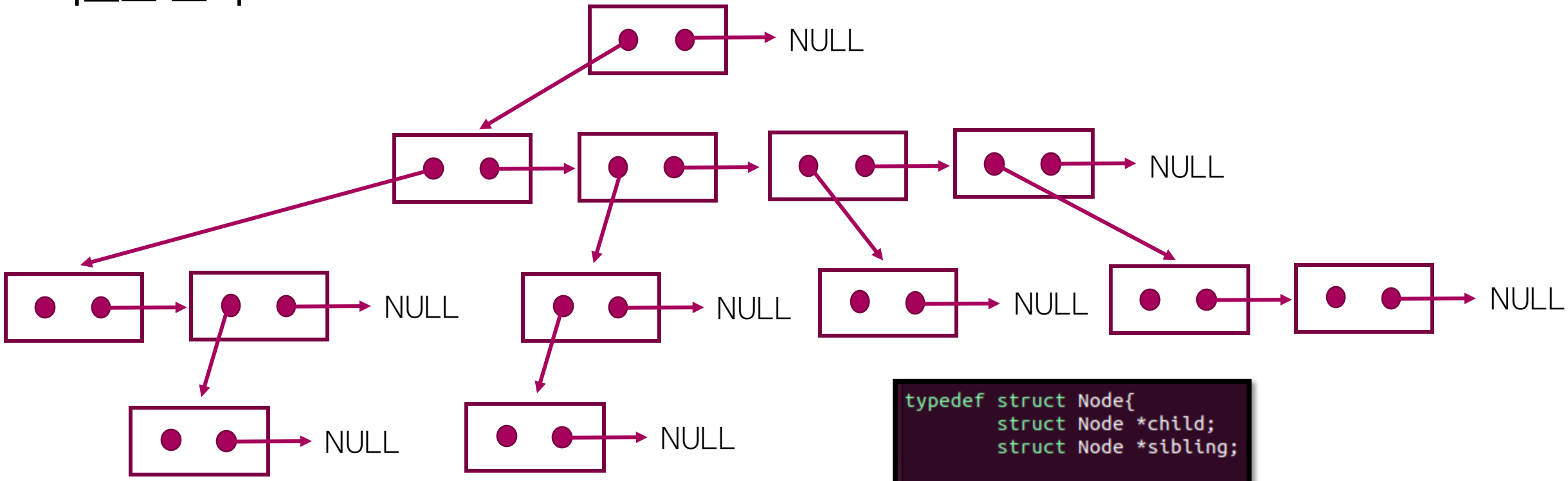
엄경호

이번 주 진행 상황



모든 과정이 원할하게 진행됨

리소스 트리




```
typedef struct Node{
        struct Node *child;
        struct Node *sibling;

        char *rn;
        char *ri;
        ObjectType ty;

        CSE *cse;
        AE *ae;
        CNT *cnt;
        CIN *cin;
}Node;
```

리소스 트리는 URI 계층 구조가 유효한지,
유효하다면 오브젝트 타입이 무엇인지,
rn와 ri를 맵핑해주는 것이 주된 역할임

Resource identifier 부여하는 규칙

2byte 씩 몇번째 child 인지 저장하는 방식

ri = "05020112" -> CSE (root)의 5번째 child의 2번째 child의 1번째 child의 12번째 child

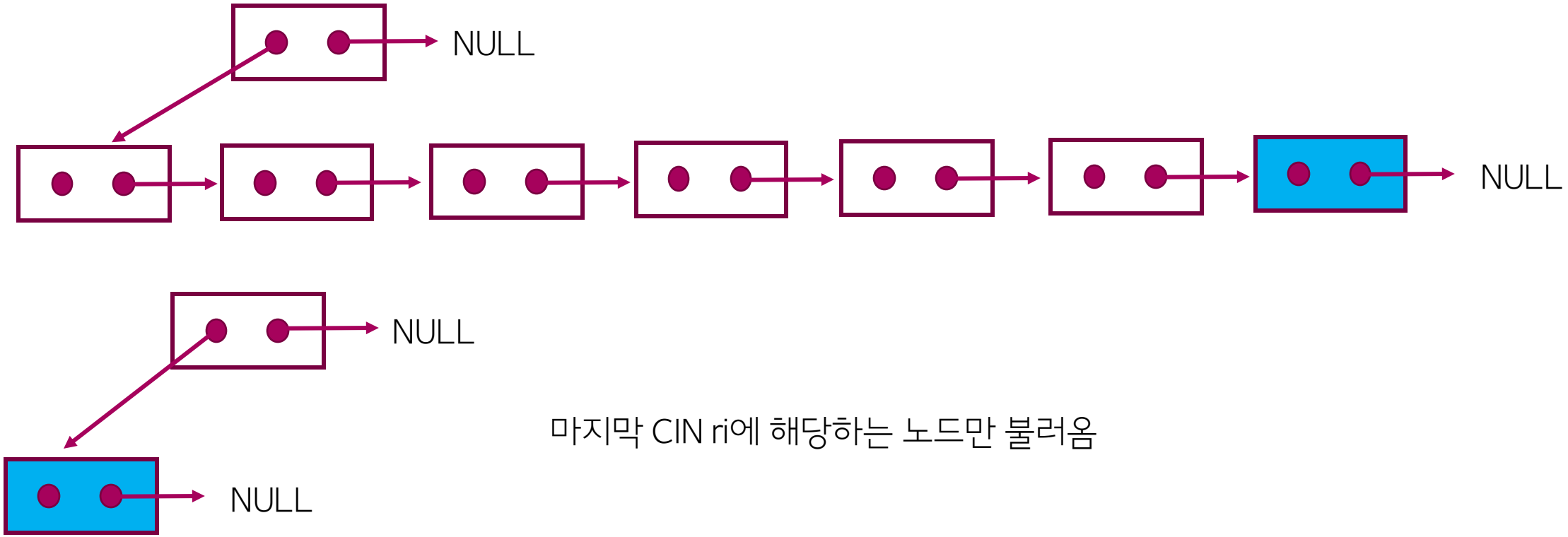
ri = "????????????07" -> ???? 부분은 pi임

전제 : DB Get에서 해당 ri에 해당하는 오브젝트가 없으면 NULL을 반환함

Get(pi+01)..Get(pi+02)..Get(pi+03)..Get(pi+04).. -> NULL을 반환할 때 까지 리소스 트리 add

생각해볼 점

CIN의 원래 리소스 트리 상태를 다시 불러올 필요가 있나?



마지막 CIN ri에 해당하는 노드만 불러옴

Delete는 어떻게?

DB에서 직접 그 값을 지우는 방법이 바로 떠오르지만

Delete 요청이 왔다면 DB에 해당 ri는 더 이상 유효하지 않음을 나타내는 값으로 변경하면 됨

이렇게 되면 DB는 수정하는 기능을 구현하면 됨

Update -> 해당 ri 오브젝트를 json body 값으로 수정

Delete -> 해당 ri 오브젝트를 유효하지 않은 값으로 수정

ex) 오브젝트 ty값을 0으로 변경

Resource Tree function

```
Node* Create_Node(char *ri, char *rn, ObjectType ty);
Node* Find_Node(RT *rt);
int Add_child(Node *parent, Node *child);
```

리소스 트리가 정상적으로 추가되지 않는 문제 발생

-> 로직을 계속 점검했지만 문제점 지속

fork() -> pthread

```
  // ACCEPT connections
  while (1) {
    addrlen = sizeof(clientaddr);
    clients[slot] = accept(listenfd, (struct sockaddr *)&clientaddr, &addrlen);

    if (clients[slot] < 0) {
      perror("accept() error");
      exit(1);
    } else {
      if (fork() == 0) {
        close(listenfd);
        respond(slot);
        close(clients[slot]);
        clients[slot] = -1;
        exit(0);
      } else {
        close(clients[slot]);
      }
    }

    while (clients[slot] != -1)
      slot = (slot + 1) % MAX_CONNECTIONS;
  }
}
```

```
pthread_t threadID;

pthread_create(&threadID, NULL, respondThread, (void*)&slot);
pthread_join(threadID, NULL);
```

```
void *respondThread(void *s) {
        int *slot = (int *)s;

        respond(*slot);
        close(clients[*slot]);
        clients[*slot] = -1;

        return NULL;
}
```

자식 프로세스와 부모 프로세스는 기본적으로 메모리를 공유하지 않음

기존 오픈소스는 클라이언트와 통신 후 자식 프로세스가 종료될 것을 전제로 필요한 소켓을 close, shutdown을 시키므로 이를 수정해주는 추가적인 작업이 필요했음

그러나 현재 가끔씩 http 요청을 받지 못하는 경우가 발생 -> 기존 오픈소스 이해와 발전된 pthread 설계가 필요함

Server

Postman -> https://www.postman.com

Server Host -> 223.131.176.101:3000/