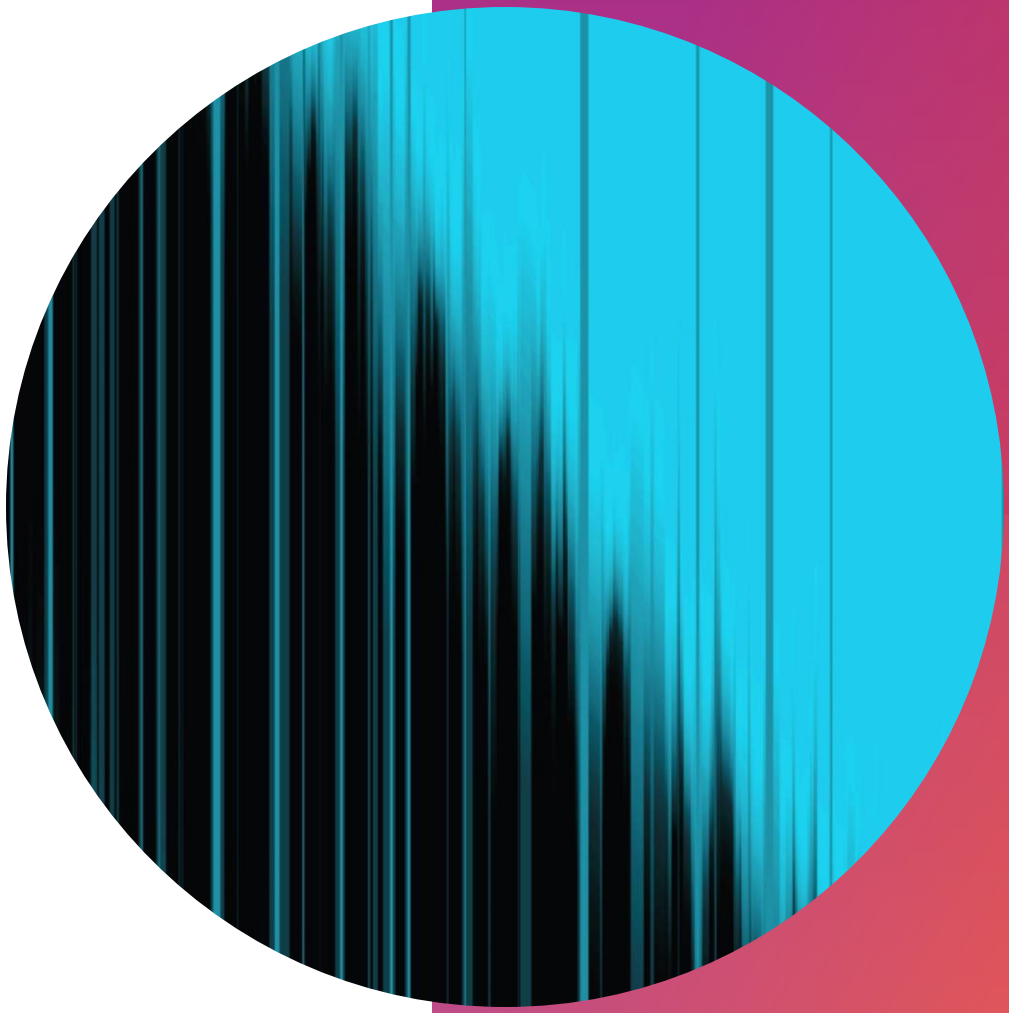
HTTP Daemon



ONEM2M TINY IOT PROJECT

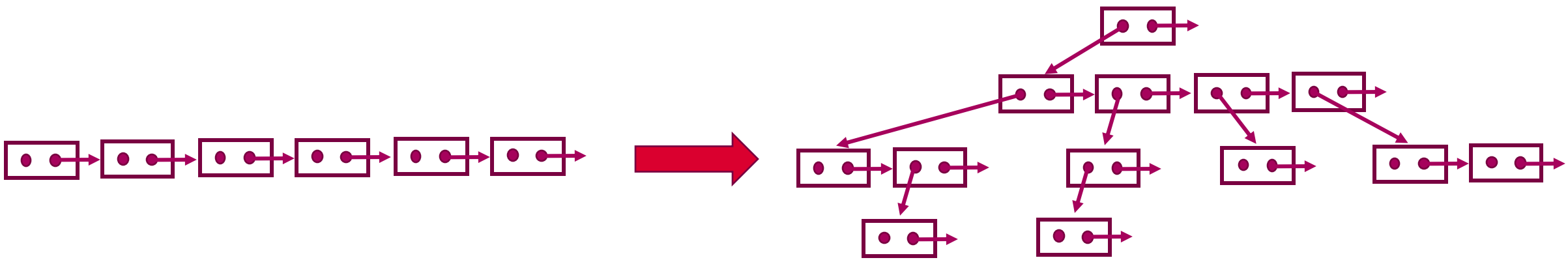
엄경호

이번주 진행 상황

db에 저장된 값을 토대로 리소스 트리 재조립하는 과정 구현 완료

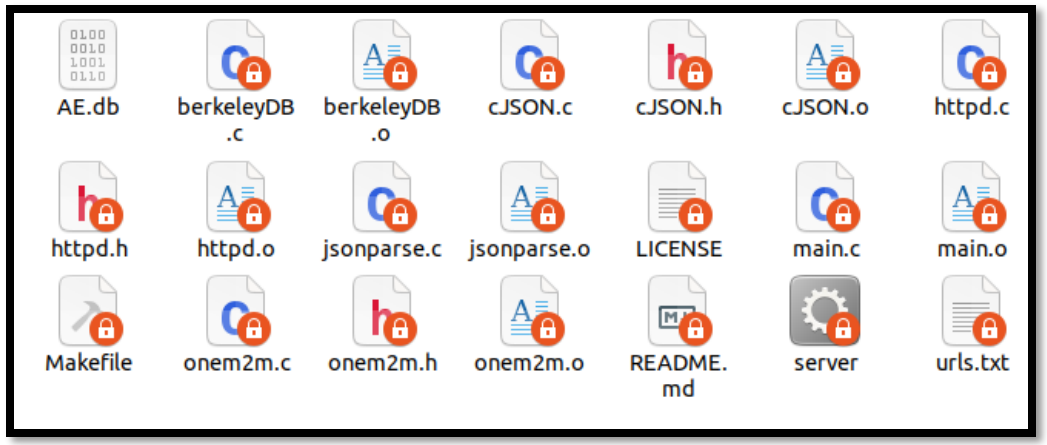
전체적인 소스코드 파일 분할 및 리팩토링

이번주 진행 상황



모든 오브젝트 노드를 불러온 후 pi값을 토대로 조립

이번주 진행 상황



main.* -> 서버 내 라우팅 관련 함수

onem2m.* -> 리소스 관련 구조체 및 함수들 (CRUD, 리소스 트리 등)

berkeleyDB.* -> DB 관련 함수들 (Store, Get, Delete 등)

jsonparse.* -> json parser 관련 함수들 (json_to_object 등)

httpd.* -> httpd 오픈 소스

cJSON.* -> cJSON 오픈 소스

이번주 진행 상황

```
void route() {
        Node* pnode = Validate_URI(rt);
        if(!pnode) {
                HTTP_500;
                printf("Invalid URI\n");
                return;
        }

        char *json_payload;

        if(payload_size > 0) {
                if(!(json_payload = Parse_Request_JSON())) {
                        HTTP_500;
                        return;
                }
        }

        Operation op = Parse_Operation();

        switch(op) {

        case o_CREATE:
                Create_Object(json_payload, pnode); break;

        case o_RETRIEVE:
                Retrieve_Object(pnode); break;

        case o_UPDATE:
                //Update_Object(); break;

        case o_DELETE:
                Delete_Object(pnode); break;

        default:
                HTTP_500;
        }
}
```

uri 검증, 유효하다면 해당 오브젝트 Node 반환

Request Body Json 파싱

Operation 파싱 후 분기

이번주 진행 상황

```
switch(op) {
case o_CREATE:
        Create_Object(json_payload, pnode); break;
case o_RETRIEVE:
        Retrieve_Object(pnode); break;
case o_UPDATE:
        //Update_Object(); break;
case o_DELETE:
        Delete_Object(pnode); break;
default:
        HTTP_500;
}
```

```
void Create_Object(char *json_payload, Node *pnode) {
        ObjectType ty = Parse_ObjectType();

        switch(ty) {

        case t_AE :
                Create_AE(json_payload, pnode);
                break;

        case t_CNT :
                Create_CNT(json_payload, pnode);
                break;

        case t_CIN :
                //Create_CIN(json_payload, pnode);
                break;
        case t_CSE :
                /*No Definition such request*/
        }
}
```

```
void Retrieve_Object(Node *pnode) {

        switch(pnode->ty) {

        case t_CSE :
                //Retrieve_CSE(pnode);
                break;

        case t_AE :
                Retrieve_AE(pnode);
                break;

        case t_CNT :
                Retrieve_CNT(pnode);
                break;

        case t_CIN :
                //Retrieve_CIN(pnode);
                break;
        }
}
```

```
void Delete_Object(Node* pnode) {
        Delete_Node(pnode,1);
        HTTP_200;
        printf("Deleted");
}
```

전체 점검

Create	Retrieve	Update	Delete	트리 재조립
Json_to_Object	Object_to_Json	Delete + Create	Delete_CSE	Get_All_AE
Object_to_Json	Get_CSE	적절히 조합할 예정	Delete_AE	Get_All_CNT
Store_CSE	Get_AE		Delete_CNT	Get_All_CIN
Store_AE	Get_CNT			
Store_CNT	Get_CIN			
Store_CIN				

AE로 모든 계층 구조가 이루어 졌을 때 CRUD, 리소스 트리 재조립 모두 가능

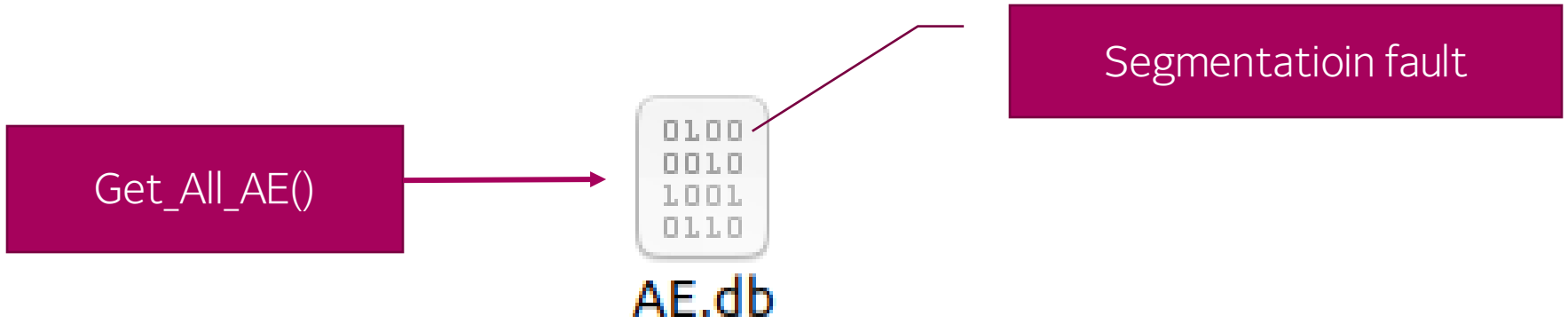
→ 같은 포맷으로 다른 오브젝트 만들면 됨

고쳐야 할 에러들

트리 재조립 시 Get_All_AE()를 호출하는데 이 때 AE.db가 존재하지 않으면 AE.db를 생성한다.

하지만 위의 AE.db를 통한 모든 CRUD 과정에서 DB Open 시 Segmentation fault가 발생함.

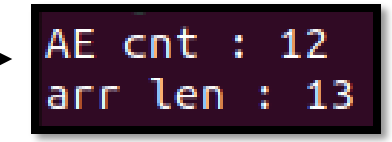
이전 Store_AE()로 생성된 AE.db를 사용해야 함. → 이 AE.db에 대해선 트리 재조립 및 CRUD 정상 작동



고쳐야 할 에러들

트리 재조립 시 Get_All_AE()에서 반환하는 AE** 배열 크기가 변함

```
AE** new_ae = (AE**)malloc(sizeof(AE*)*cnt);
for (int i = 0; i < cnt; i++) {
    new_ae[i] = (AE*)malloc(sizeof(AE));
}
    fprintf(stderr,"AE cnt : %d\n",cnt);
```



```
AE cnt : 12
arr len : 13
```

구체적으로는 리소스 등록 후 서버 재실행 시 배열 크기가 변하고
리소스 등록하지 않고 재실행 하면 원래 크기로 돌아옴

고친 에러 : URI 처리 과정과 DB접근 과정에서 사용되는 모든 함수 인자가 독립적인데 URI 길이가 길어지면 DB 접근에서 에러 발생 → URI 처리 과정에서 malloc 크기 수정