TCT-시스템&솔루션개발
실기형 문제지

[2023년 1차]

사번		성명	
유의 사항	1. 공정한 평가를 위해 동료를 도와주는 행위, 보여주는 행위를 금지하고 있습니다. 2. 부정행위 적발 시, 응시한 평가는 0점 처리됩니다. 3. 본 시험지는 응시장 외부로 유출할 수 없으며, 시험 종료 후 감독관에게 제출해야 합니다.		

LG CNS

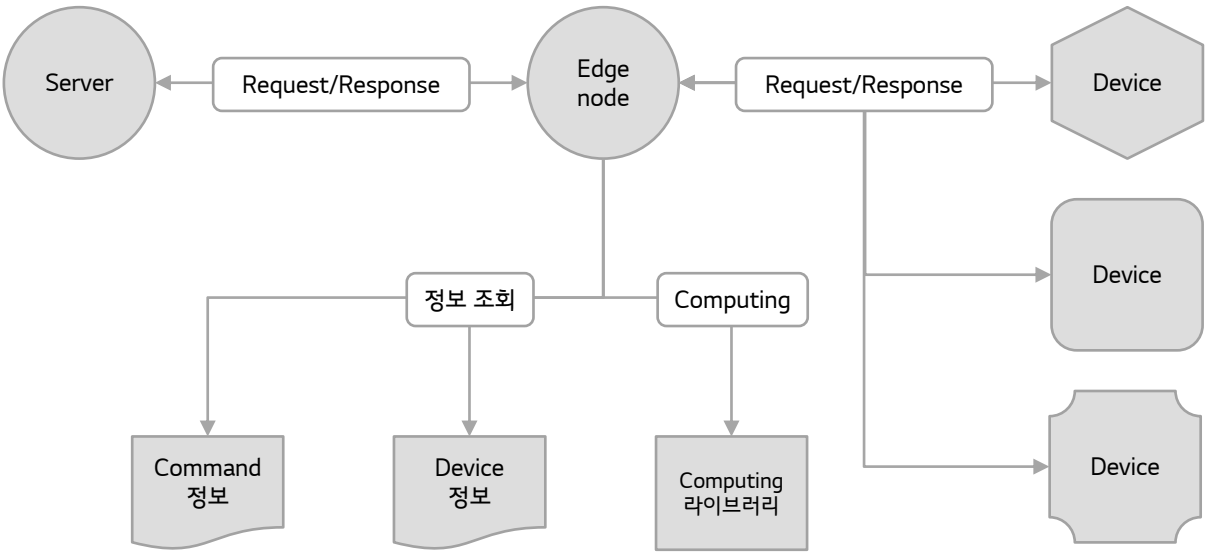
Edge Computing

개요

해당 시스템 구현은 동해 요구사항 분석, 파일처리, 데이터관리, HTTP Server/Client 구현 등의 기술역량 및 프로그램 구현 역량을 측정하기 위한 문제입니다.

설명

본 시스템은 'Edge node'가 'Server' 또는 'Device'로부터 Request를 수신(콘솔/파일 입출력, HTTP 통신)하여 요구사항을 처리하는 'Edge Computing' 시스템입니다.



[기능 요약] (Edge node 관점)

- 'Server'로부터 Request가 수신되면 'Edge node'는 'Device'로 Request를 전달하고, 'Device'로부터의 Response를 취합하여 'Server'의 Response로 송신한다.
- 'Edge node'는 'Device'의 병렬 처리 개수를 고려하여 Request를 송신한다.
- 'Device'로부터 Request가 수신되면 'Edge node'는 동적 라이브러리 로딩 기술을 적용하여 'Computing'을 수행하고 필요시 그 결과를 'Server'로 전달한다.

주의사항

실행 결과로 평가하고 부분점수는 없으므로 아래사항을 필히 주의해야 함

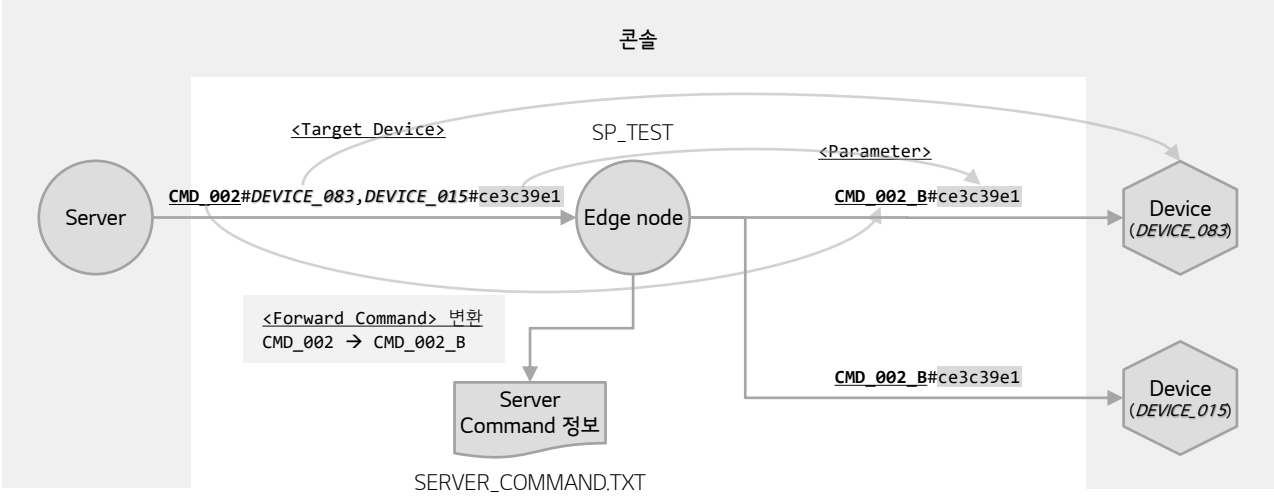
- 구현된 프로그램은 실행 완결성 필수 (명확한 실행&종료 처리, 정확한 결과 출력, 동상의 실행 시간)
- 소 문항별 결과 검수 필수 (선행문항 오류 시, 후속문항 전체에 오류가 발생할 수 있음)
- 제시된 조건이 없는 한 선행요구사항 유지 필수
- 프로그램 실행 위치 및 실행결과출력 (위치, 파일명, 데이터포맷)은 요구사항과 정확히 일치 필수
- 제시된 모든 위치는 상대경로 사용 필수 (프로그램 실행 위치 기준)
- 종료조건에 맞는 자동종료 처리 필수 (불필요한 종료방해처리(pause/입력대기 등)를 하면 안됨)
- 모든 문자는 대소문자 구분 필수

문제

아래 제시된 문항은 문항번호가 증가할 수록 점진적 개선을 요구하는 방식으로 구성되어 있으며, 제시된 문항번호 별로 각각 **구현된 소스와 컴파일 된 실행파일을 제출**하시오.
cf) 1번 구현 → 1번 소스복사 → 2번 구현 → 2번 소스복사 → ...

1. Server로부터 Request가 수신되면 Target Device로 Request를 Forward하는 Edge node를 구현하시오. (20점)

상세설명



※ 'Server로부터의 Request' 처리

- Server로부터 Request가 수신(콘솔 입력)되면 <Target Device>로 Request를 Forward(콘솔 출력) ('※ 콘솔 입/출력' 참조)
- Server로부터 수신된 Request의 <Command>는 'Server Command 정보 파일'을 이용하여 <Forward Command>로 변환 ('※ Server Command 정보 파일' 참조)

형식정보

※ Request 형식

- Request 형식 : <Command>
  + "#" + <Target Device> + "," + <Target Device> (Target Device 개수만큼 반복)
  + "#" + <Parameter>
- 예> CMD_002#DEVICE_083,DEVICE_015#ce3c39e1
  <Command> <Target Device> <Parameter>
- 'Server도부터의 Request'에만 <Target Device> 존재

```
<Request - Server로부터의 Request>
CMD_001#DEVICE_069#fe303904
CMD_002#DEVICE_083,DEVICE_015#ce3c39e1

<Request - Device로의 Request>
CMD_001_A#fe303904              ← <Target Device> 없음
CMD_002_B#ce3c39e1              ← <Target Device> 없음
```

※ Server Command 정보 파일

- 파일명 : SERVER_COMMAND.TXT (INFO 폴더 내)
- 파일 형식 : <Command> + "#" + <Forward Command>
- 예> CMD_002#CMD_002_B
  Server도부터 "CMD_002" Request를 수신하면 Device도 "CMD_002_B" Request 송신

```
CMD_001#CMD_001_A
CMD_002#CMD_002_B
```

※ 콘솔 입/출력

- 콘솔에 'Server도부터의 Request'가 입력되면 'Device도의 Request' 출력 후 프로그램 종료
- 입력 형식 : <Server도부터의 Request>
- 출력 형식 : <Target Device> + ":" + <Device도의 Request>
  (Target Device 개수만큼 반복)

```
C:\>SP_TEST<엔터키>
CMD_001#DEVICE_069#fe303904<엔터키>
DEVICE_069:CMD_001_A#fe303904
C:\>

C:\>SP_TEST<엔터키>
CMD_002#DEVICE_083,DEVICE_015#ce3c39e1<엔터키>
DEVICE_083:CMD_002_B#ce3c39e1
DEVICE_015:CMD_002_B#ce3c39e1
C:\>
```

평가대상

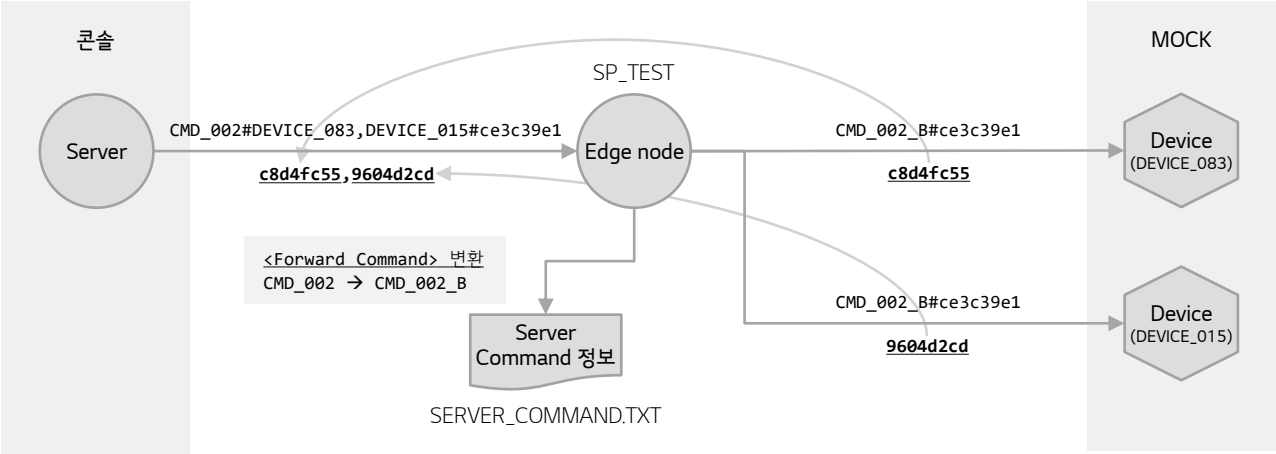
프로그램 정상 실행, 콘솔 출력 결과

문제

2. 위 1번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 Edge node를 구현하시오. (15점)

- Response 추가
- 'Server로부터의 Request' 수신 및 'Server로의 Response' 송신은 콘솔에서 처리
- 'Device로의 Request' 송신 및 'Device로부터의 Response' 수신은 파일로 처리

상세설명



※ Response 추가

- Edge node가 <Target Device>로 송신한 Request에 대한 Response를 수신하여 이를 Server의 Response로 송신 ('※ Response 형식 추가' 참조)
- <Target Device>가 2개 이상인 경우 'Device로부터의 Response'의 <Result>들을 하나로 취합하여 'Server로의 Response'로 송신 ('Device로부터의 Response'는 1개의 <Result>만 존재)
- 'Server로의 Response'에서 <Result>의 순서는 'Server로부터의 Request'의 <Target Device> 순서와 동일해야 함 (상단 이미지 참조)

※ 'Server로부터의 Request' 수신 및 'Server로의 Response' 송신은 콘솔에서 처리

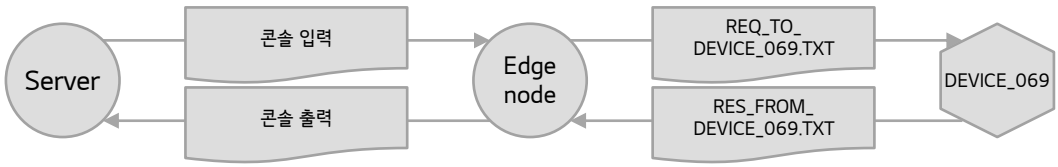
- Edge node가 'Server로부터의 Request' 수신은 콘솔 입력으로, 'Server로의 Response' 송신은 콘솔 출력으로 수행 ('※ 콘솔 입/출력' 참조)

※ 'Device로의 Request' 송신 및 'Device로부터의 Response' 수신은 파일로 처리

- Edge node가 'Device로의 Request' 송신 및 'Device로부터의 Response' 수신 수행 시
  1) <Request 파일>에 'Device로의 Request'를 기록하고,
  2) <Response 파일>에서 'Device로부터의 Response'를 읽는 방식으로 수행
  (MOCK이 <Request 파일>로부터 Request를 읽고 <Response 파일>에 Response를 기록하는 역할을 수행)
- 파일명 : 아래 표 참조 (DEVICE 폴더 내)

Request/Response 구분	Request/Response 파일명
Device로의 Request	REQ_TO_<Device>.TXT
Device로부터의 Response	RES_FROM_<Device>.TXT

상세설명 (계속)



※ 'Device도의 Request' 송신 및 'Device도부터의 Response' 수신 예시

- "DEVICE_069"도 "CMD_001_A" Request 송신

REQ_TO_DEVICE_069.TXT

```
CMD_001_A#fe303904
```

RES_FROM_DEVICE_069.TXT

```
20b95f9c
```

※ MOCK 관련 참고

1) SP_TEST 실행 전, **MOCK 종료 및 재실행 필수**
   (MOCK은 실행하면 모든 <Request 파일>과 <Response 파일>이 삭제됨)
2) SP_TEST는 <Request 파일>에 Request Write하고 **500ms sleep** 후 <Response 파일>에서 Response Read 수행
   (MOCK은 <Request 파일>에 Request가 기독되면 500ms 내에 <Response 파일>에 Response를 기독함)

형식정보

※ Response 형식 추가

- Request 형식은 기존과 동일 (1번 문항 '※ Request 형식' 참조)
- Response 형식 : <Result> + "," + <Result> (Result 개수만큼 반복)
- 예>

```
c8d4fc55,9604d2cd
```

※ 콘솔 입/출력

- 콘솔에 'Server도부터의 Request'가 입력되면 'Server도의 Response' 출력 후 프로그램 종료
- 입력 형식 : <Server도부터의 Request>
- 출력 형식 : <Server도의 Response>

```
C:\>SP_TEST<엔터키>
CMD_001#DEVICE_069#fe303904<엔터키>
20b95f9c
C:\>

C:\>SP_TEST<엔터키>
CMD_002#DEVICE_083,DEVICE_015#ce3c39e1<엔터키>
c8d4fc55,9604d2cd
C:\>
```

평가대상

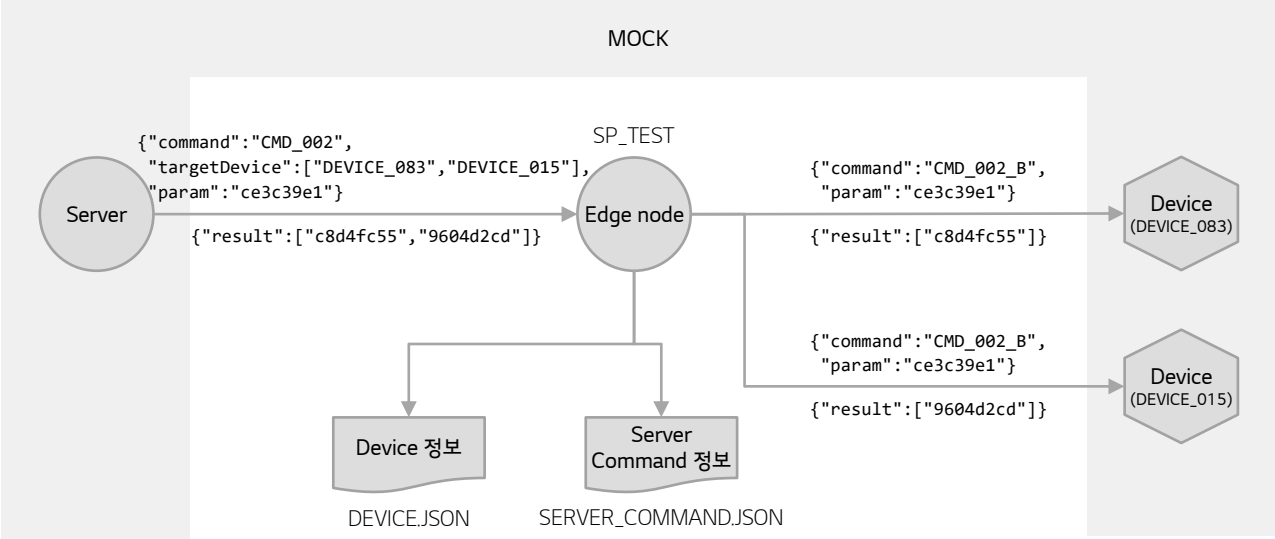
프로그램 정상 실행, 콘솔 출력 결과

문제

3. 위 2번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 Edge node를 구현하시오. (15점)

- Request/Response를 HTTP로 처리하도록 변경
- Request/Response를 JSON 형식으로 변경
- 'Server Command 정보 파일'을 JSON 형식으로 변경
- 'Device 정보 파일' 추가

상세설명



※ Request/Response를 HTTP로 처리하도록 변경

- Edge node는 "127.0.0.1" hostname과 "8010" port를 사용하며,
  Device의 hostname과 port는 'Device 정보 파일' 참조 ('※ Device 정보 파일 추가' 참조)
- HTTP 요청/응답은 다음의 URI로 수행

구분	URI
Server로부터의 Request	POST http://127.0.0.1:8010/fromServer
Device로의 Request	POST http://<Device hostname>:<Device port>/fromEdge

형식정보

※ Request/Response를 JSON 형식으로 변경

- 변경내역 : TEXT 형식을 JSON 형식으로 변경
  ('Server로부터의 Request'에만 "targetDevice" 존재)

```
<Request>
{"command":"<Command>","targetDevice":<Target Device 배열>,"param":"<Parameter>"}

<Response>
{"result":<Result 배열>}
```

형식정보 (계속)

※ Request/Response를 JSON 형식으로 변경 (계속)

- 예>

```
<Request – Server로부터의 Request>
{"command":"CMD_002","targetDevice":["DEVICE_083","DEVICE_015"],"param":"ce3c39e1"}

<Request – Device로의 Request>
{"command":"CMD_002_B","param":"ce3c39e1"}     ← "targetDevice" 없음

<Response – Server로의 Response>
{"result":["c8d4fc55","9604d2cd"]}

<Response – Device로부터의 Response>
{"result":["c8d4fc55"]}                        ← <Result> 1개만 존재
```

※ Server Command 정보 파일 변경

- 변경내역 : TEXT 형식을 JSON 형식으로 변경
- 파일명 : SERVER_COMMAND.JSON (INFO 폴더 내)
- 파일 형식 : JSON 형식

```
{  "serverCommandInfo":[
      {"command":"<Command>", "forwardCommand":<Forward Command>"},
      ...
   ]
}
```

- 예>

```
{  "serverCommandInfo":[
      {"command":"CMD_001", "forwardCommand":"CMD_001_A"},
      {"command":"CMD_002", "forwardCommand":"CMD_002_B"}
   ]
}
```

※ Device 정보 파일 추가

- 파일명 : DEVICE.JSON (INFO 폴더 내)
- 파일 형식 : JSON 형식

```
{  "deviceInfo":[
      {"device":"<Device>", "hostname":"<Device hostname>", "port":<Device port>},
      ...
   ]
}
```

- 예>

```
{  "deviceInfo":[
      {"device":"DEVICE_069", "hostname":"127.0.0.1", "port":9010},
      {"device":"DEVICE_083", "hostname":"127.0.0.1", "port":9020},
      {"device":"DEVICE_015", "hostname":"127.0.0.1", "port":9030}
   ]
}
```

평가대상

프로그램 정상 실행, 제공프로그램(MOCK.BAT) 실행결과 정답/오답 여부

문제

4. 위 3번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 Edge node를 구현하시오. (20점)

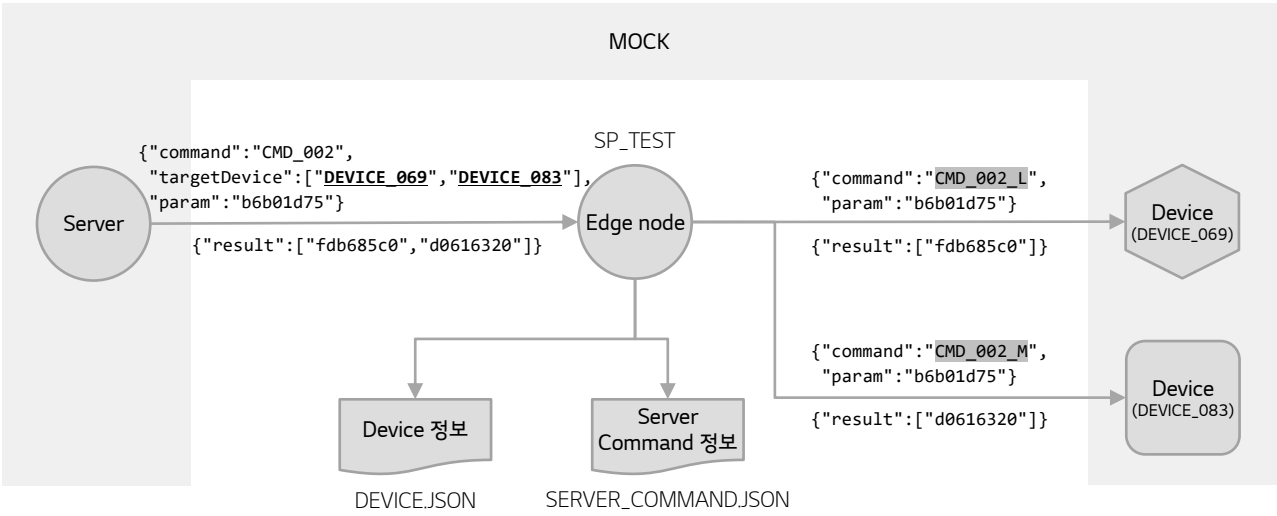
- <Device 유형>에 따라 <Forward Command> 변환
- 'Device로의 Request' 순차 송신
- 'Device로의 Request' 병렬 송신

상세설명

※ <Device 유형>에 따라 <Forward Command> 변환

- 'Device 정보 파일'과 'Server Command 정보 파일'에 <Device 유형> 추가
  ('※ Device 정보 파일 변경', '※ Server Command 정보 파일 변경' 참조)
- Server로부터 Request 수신 시 <Target Device>의 <Device 유형>을 참조하여 <Forward Command> 변환

- 예> Server로부터 "CMD_002" Request 수신 시
  1) <Target Device>의 <Device 유형> 참조 ('Device 정보 파일' 참조)
     DEVICE_069 → *DEVICE_A*
     DEVICE_083 → *DEVICE_B*
  2) <Device 유형>별로 <Forward Command> 변환 ('Server Command 정보 파일' 참조)
     CMD_002 → *DEVICE_A* → CMD_002_L
     CMD_002 → *DEVICE_B* → CMD_002_M



'Device 정보 파일'

```
{  "type":"DEVICE_A",
   "deviceList":[
      {"device":"DEVICE_069", ..., "port":9010},
      {"device":"DEVICE_198", ..., "port":9050}
   ]
},
{  "type":"DEVICE_B",
   "deviceList":[
      {"device":"DEVICE_083", ..., "port":9020}
   ]
}
```

'Server Command 정보 파일'

```
{  "command":"CMD_002",
   "forwardCommandInfo":[
      {"type":"DEVICE_A", "forwardCommand":["CMD_002_L"]},
      {"type":"DEVICE_B", "forwardCommand":["CMD_002_M"]},
      ...
   ]
}
```

상세설명 (계속)

※ 'Device도의 Request' 순차 송신

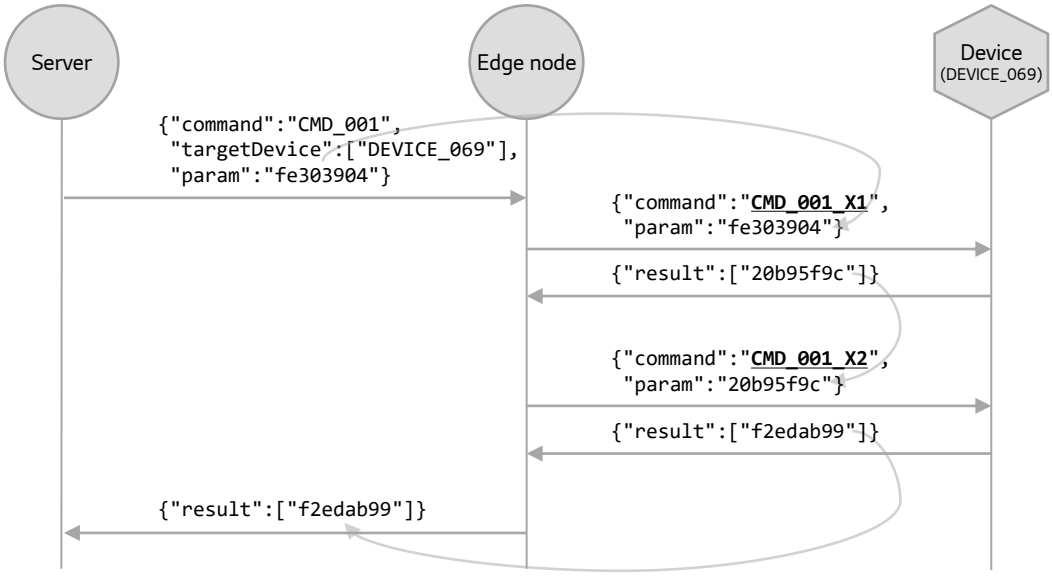
- 'Server Command 정보 파일'의 "forwardCommand"에 두개 이상의 <Command>가 설정된 경우 Device도 Request를 순서대도 송신 ('※ Server Command 정보 파일 변경' 참조)

'Server Command 정보 파일'

```
{  "serverCommandInfo":[
      {  "command":"CMD_001",
         "forwardCommandInfo":[
            {"type":"DEVICE_A", "forwardCommand":["CMD_001_X1","CMD_001_X2"]},
...
```

2개 이상의 <Command> 설정

- 'Device도의 Request'를 순서대도 송신 시 이전 Response의 <Result>를 다음 Request의 <Parameter>도 입력
- 예> Server도부터 "DEVICE_A" 유형의 <Target Device>도 "CMD_001" Request 수신된 경우



※ 'Device도의 Request' 병렬 송신

- 각각의 Device는 <Device 유형>별도 설정된 <병렬 처리 개수>만큼 Edge node도부터의 Request를 동시에 처리 가능 (' ※ Device 정보 파일 변경' 참조)
- Server도부터 Request가 수신되면 Edge node는 <Target Device>의 <병렬 처리 개수>를 고려하여 병렬도 Request를 즉시 송신해야 함
  - 병렬 처리가 가능함에도 직렬 처리를 수행하면 처리 시간 초과도 MOCK에서 FAIL 처리
  - Device의 <병렬 처리 개수>를 초과하는 Request를 요청하면 MOCK에서 FAIL 처리
- 단, 각각의 Device는 동일한 <Command> Request를 동시에 처리할 수 없음
  - Request를 처리중인 Device에 동일한 <Command> Request를 송신하면 MOCK에서 FAIL 처리

형식정보

※ Server Command 정보 파일 변경

- 변경 내역 : 기존 "forwardCommand"를 <Device 유형>별 <Command 배열>로 구성된 "forwardCommandInfo"로 대체

```
{  "serverCommandInfo":[
      {  "command":"<Command>",
         "forwardCommandInfo":[
            {"type":"<Device 유형>", "forwardCommand":<Command 배열>},
            ...
         ]
      },
      ...
   ]
}
```

- 예>

```
{  "serverCommandInfo":[
      {  "command":"CMD_001",
         "forwardCommandInfo":[
            {"type":"DEVICE_A", "forwardCommand":["CMD_001_X1","CMD_001_X2"]},
            {"type":"DEVICE_B", "forwardCommand":["CMD_001_P"]},
            {"type":"DEVICE_C", "forwardCommand":["CMD_001_W"]},
            {"type":"DEVICE_D", "forwardCommand":["CMD_001_H"]}
         ]
      },
      {  "command":"CMD_002",
         "forwardCommandInfo":[
            {"type":"DEVICE_A", "forwardCommand":["CMD_002_L"]},
            {"type":"DEVICE_B", "forwardCommand":["CMD_002_M"]},
            {"type":"DEVICE_C", "forwardCommand":["CMD_002_Q1","CMD_002_Q2","CMD_002_Q3"]},
            {"type":"DEVICE_D", "forwardCommand":["CMD_002_U"]}
         ]
      }
   ]
}
```

형식정보 (계속)

※ Device 정보 파일 변경

- 변경 내역

1) <Device 유형> 추가

2) <병렬 처리 개수> 추가

(<Device 유형>에 속하는 각각의 Device가 동시에 처리 가능한 Request 개수)

```
{  "deviceInfo":[
      {  "type":"<Device 유형>",
         "parallelProcessingCount":<병렬 처리 개수>,
         "deviceList":[
            {"device":"<Device>","hostname":"<Device hostname>","port":<Device port>},
            ...
         ]
      },
      ...
   ]
}
```

- 예> "DEVICE_152"는 동시에 2개의 Request 처리 가능
("DEVICE_C" 유형의 <병렬 처리 개수>가 2)

```
{  "deviceInfo":[
      {  "type":"DEVICE_A",
         "parallelProcessingCount":1,
         "deviceList":[
            {"device":"DEVICE_069", "hostname":"127.0.0.1", "port":9010},
            {"device":"DEVICE_198", "hostname":"127.0.0.1", "port":9050}
         ]
      },
      {  "type":"DEVICE_B",
         "parallelProcessingCount":2,
         "deviceList":[
            {"device":"DEVICE_083", "hostname":"127.0.0.1", "port":9020}
         ]
      },
      {  "type":"DEVICE_C",
         "parallelProcessingCount":2,
         "deviceList":[
            {"device":"DEVICE_015", "hostname":"127.0.0.1", "port":9030},
            {"device":"DEVICE_152", "hostname":"127.0.0.1", "port":9040}
         ]
      },
      {  "type":"DEVICE_D",
         "parallelProcessingCount":1,
         "deviceList":[
            {"device":"DEVICE_274", "hostname":"127.0.0.1", "port":9060}
         ]
      }
   ]
}
```

평가대상

프로그램 정상 실행, 제공프로그램(MOCK.BAT) 실행결과 정답/오답 여부

문제

5. 위 4번 문항까지 구현된 내용을 기준으로, 아래 사항을 추가로 반영한 Edge node를 구현하시오. (10점)

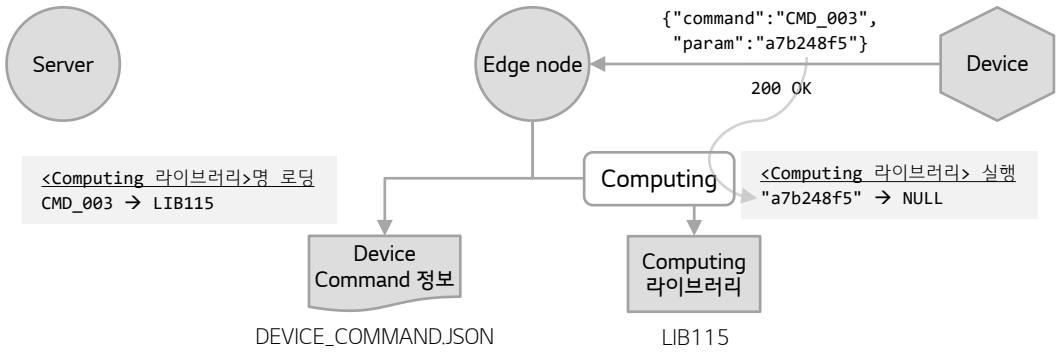
   - 'Device로부터의 Request' 처리(Computing) 추가

상세설명

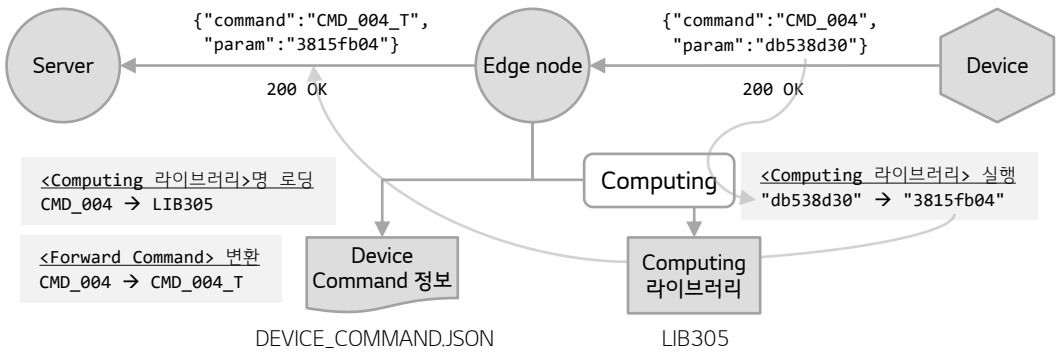
※ 'Device로부터의 Request' 처리(Computing) 추가

- 'Device로부터 Request'가 수신되면 'Device Command 정보 파일'에 <Command>별로 설정된 <Computing 라이브러리>를 이용하여 Computing 수행 ('※ Device Command 정보 파일 추가' 참조)
- <Computing 라이브러리>는 동적 라이브러리 로딩 기술을 사용하여 실행 (언어별 인터페이스는 README.TXT 참조)
- <Computing 라이브러리> 실행 시 Device로부터 수신한 Request의 <Parameter>값을 입력

1) <Computing 라이브러리> 실행 결과가 NULL인 경우, Device로 Body 없이 200 OK 응답



2) <Computing 라이브러리> 실행 결과 NULL이 아닌 경우, Computing 결과(리턴된 데이터)를 Server로 Forward (<Forward Command> 변환 수행)



상세설명 (계속)

※ 'Device도부터의 Request'를 HTTP도 처리

- Server와 Edge node는 고정된 hostname과 port 사용

Node	hostname	port
Server	127.0.0.1	7010
Edge node	127.0.0.1	8010

- HTTP 요청/응답은 다음의 URI도 수행

구분	URI
Device로부터의 Request	POST http://127.0.0.1:8010/fromDevice
Server로의 Request	POST http://127.0.0.1:7010/fromEdge

형식정보

※ Device Command 정보 파일 추가

- 파일명 : DEVICE_COMMAND.JSON (INFO 폴더 내)
- 파일 형식 : JSON 형식

```
{  "deviceCommandInfo":[
      {  "command":"<Command>",
         "forwardCommand":"<Forward Command>",
         "computingLib":"<Computing 라이브러리>"
      },
      ...
   ]
}
```

- 예>

```
{  "deviceCommandInfo":[
      {  "command":"CMD_003",
         "forwardCommand":"CMD_003_S",
         "computingLib":"LIB115"
      },
      {  "command":"CMD_004",
         "forwardCommand":"CMD_004_T",
         "computingLib":"LIB305"
      }
   ]
}
```

평가대상

프로그램 정상 실행, 제공프로그램(MOCK.BAT) 실행결과 정답/오답 여부

폴더 정보

※ 프로그램 및 파일 위치 정보 (실행위치 기반 상대경로 사용 필수)

- 구현할 프로그램 위치 및 실행 위치 : 각 소문항 홈 (SUB1/SUB2/SUB3/SUB4/SUB5)
- 자가 검수용 참고 파일명 : COMPARE 폴더 내 CMP_CONSOLE.TXT (SUB1/SUB2)
- 제공되는 MOCK 프로그램 파일명 : 각 소문항 홈 아래 MOCK.BAT (SUB2/SUB3/SUB4/SUB5)

*** 제공되는 파일들은 문항에 따라 다를 수 있음**

실행 방식

※ 구현할 프로그램 형식

- 프로그램 형태 : 콘솔(Console) 프로그램
- 프로그램 파일명 : SP_TEST
- 실행 방식(문항1) : 콘솔 실행→결과처리→자동종료

```
C:\>SP_TEST<엔터키>                                ← 구현한 프로그램 실행 (Argument 없음)
CMD_001#DEVICE_069#fe303904<엔터키>
DEVICE_069:CMD_001_A#fe303904
C:\>

C:\>SP_TEST<엔터키>                                ← 구현한 프로그램 실행 (Argument 없음)
CMD_002#DEVICE_083,DEVICE_015#ce3c39e1<엔터키>
DEVICE_083:CMD_002_B#ce3c39e1
DEVICE_015:CMD_002_B#ce3c39e1
C:\>
```

- 실행방식(문항2) : 콘솔 실행→결과처리→자동종료

```
C:\>SP_TEST<엔터키>                                ← 구현한 프로그램 실행 (Argument 없음)
CMD_001#DEVICE_069#fe303904<엔터키>
20b95f9c
C:\>

C:\>SP_TEST<엔터키>                                ← 구현한 프로그램 실행 (Argument 없음)
CMD_002#DEVICE_083,DEVICE_015#ce3c39e1<엔터키>
c8d4fc55,9604d2cd
C:\>
```

- 실행방식(문항3~5) : 콘솔 실행→실시간 HTTP 요청/응답 처리 (종료 없음)

```
C:\>SP_TEST<엔터키>                                ← 구현한 프로그램 실행 (Argument 없음)
...
```

제공 및 제출

✓ 각 언어별 제공파일 압축 해제 후 자동 생성된 폴더 사용 필수

✓ 제공되는 주요 내용

- 샘플 파일
- 제공 프로그램 실행파일 (MOCK.BAT)
- 제출시 사용할 문항별 폴더 구조

✓ 제출 파일 및 폴더 상세 내용 (각 언어별 실기 가이드 참고)

<주의사항>

제출 파일 관련 내용 (폴더위치, 파일명, 프로그램명 등) 이 틀린 경우 및 상대경로를 사용하지 않은 경우에는 평가 시 불이익이 발생할 수 있으므로 반드시 요구되는 내용과 일치시켜 제출해야 함.

테스트 방법

※ 자가 검수를 위해 제공되는 샘플은 검수용 데이터와 다를 수 있음

자가 검수를 위한 샘플 결과 파일은 각 문항 출력 폴더(COMPARE)에 사전 제공됨

[문항1]

- SP_TEST를 실행한 후 콘솔 입/출력 결과를 샘플 결과 파일(CMP_CONSOLE.TXT)과 동일한지 비교

```
C:\>SP_TEST<엔터키>
CMD_001#DEVICE_069#fe303904<엔터키>
DEVICE_069:CMD_001_A#fe303904
C:\>
```

[문항2]

- MOCK.BAT를 종료 및 재실행 후, SP_TEST 실행
- 콘솔 입/출력 결과를 샘플 결과 파일(CMP_CONSOLE.TXT)과 동일한지 비교

```
C:\>SP_TEST<엔터키>
CMD_001#DEVICE_069#fe303904<엔터키>
20b95f9c
C:\>
```

테스트 방법 (계속)

[문항3]

- MOCK.BAT를 실행한 후, SP_TEST 실행
  (MOCK.BAT의 콘솔에서 <엔터키>를 입력하면 테스트가 시작됨)
- MOCK.BAT의 콘솔에 출력되는 테스트 결과 확인
  (SP_TEST가 문항의 요건을 만족하지 못하는 경우, MOCK.BAT의 콘솔에 테스트 Fail의 사유 또는 Stack trace가 출력되므로 자가 검수에 참고 요망)

```
C:\>MOCK.BAT<엔터키>

[MOCK] 시나리오 테스트 준비(SP_TEST 실행)가 완료되면 <엔터키>를 입력하세요.

[MOCK] Server/Device HTTP 서버 시작

2023-05-17 18:48:18.662:INFO::main: Logging initialized @10846ms to
org.eclipse.jetty.util.log.StdErrLog
...

[1번 시나리오] Server가 CMD_001 Request 송신

[시나리오][Server] Request 송신 (URI:http://127.0.0.1:8010/fromServer,
Body:{"command":"CMD_001","targetDevice":["DEVICE_069"],"param":"fe303904"})
[시나리오][DEVICE_069] Request 수신 (URI:http://127.0.0.1:9010/fromEdge,
Body:{"command":"CMD_001_A","param":"fe303904"})
[시나리오][DEVICE_069] Response 송신 (Status Code:200, Body:{"result":["20b95f9c"]})
[시나리오][Server] Response 수신 (Status Code:200, Body:{"result":["20b95f9c"]})

[2번 시나리오] Server가 CMD_002 Request 송신

[시나리오][Server] Request 송신 (URI:http://127.0.0.1:8010/fromServer,
Body:{"command":"CMD_002","targetDevice":["DEVICE_083","DEVICE_015"],"param":"ce3c39e1"})
[시나리오][DEVICE_083] Request 수신 (URI:http://127.0.0.1:9020/fromEdge,
Body:{"command":"CMD_002_B","param":"ce3c39e1"})
[시나리오][DEVICE_083] Response 송신 (Status Code:200, Body:{"result":["c8d4fc55"]})
[시나리오][DEVICE_015] Request 수신 (URI:http://127.0.0.1:9030/fromEdge,
Body:{"command":"CMD_002_B","param":"ce3c39e1"})
[시나리오][DEVICE_015] Response 송신 (Status Code:200, Body:{"result":["9604d2cd"]})
[시나리오][Server] Response 수신 (Status Code:200, Body:{"result":["c8d4fc55","9604d2cd"]})

[정답] 테스트에 성공했습니다!
```

테스트 방법 (계속)

[문항4]

- MOCK.BAT를 실행한 후, SP_TEST 실행
  (MOCK.BAT의 콘솔에서 <엔터키>를 입력하면 테스트가 시작됨)
- MOCK.BAT의 콘솔에 출력되는 테스트 결과 확인
  (SP_TEST가 문항의 요건을 만족하지 못하는 경우, MOCK.BAT의 콘솔에 테스트 Fail의 사유 또는 Stack trace가 출력되므로 자가 검수에 참고 요망)

```
C:\>MOCK.BAT<엔터키>

[MOCK] 시나리오 테스트 준비(SP_TEST 실행)가 완료되면 <엔터키>를 입력하세요.

[MOCK] Server/Device HTTP 서버 시작

2023-05-17 18:48:18.662:INFO::main: Logging initialized @10846ms to
org.eclipse.jetty.util.log.StdErrLog
...

[1번 시나리오] Server가 CMD_002 Request 송신
  - '※ <Device 유형>에 따라 <Forward Command> 변환' TEST

[시나리오][Server] Request 송신 (URI:http://127.0.0.1:8010/fromServer,
Body:{"command":"CMD_002","targetDevice":["DEVICE_069","DEVICE_083"],"param":"b6b01d75"})
[시나리오][DEVICE_069] Request 수신 (URI:http://127.0.0.1:9010/fromEdge,
Body:{"command":"CMD_002_L","param":"b6b01d75"})
[시나리오][DEVICE_083] Request 수신 (URI:http://127.0.0.1:9020/fromEdge,
Body:{"command":"CMD_002_M","param":"b6b01d75"})
[시나리오][DEVICE_083] Response 송신 (Status Code:200, Body:{"result":["d0616320"]})
[시나리오][DEVICE_069] Response 송신 (Status Code:200, Body:{"result":["fdb685c0"]})
[시나리오][Server] Response 수신 (Status Code:200, Body:{"result":["fdb685c0","d0616320"]},
Elapsed time:1,131ms)

...

[정답] 테스트에 성공했습니다!
```

❖ [문항 4]의 [6번 시나리오] 참고

- 아래 테이블과 같이 첫번째 Request는 약 3,000ms, 두번째 Request는 약 3,500ms 소요
  (MOCK에서는 여유시간 포함하여 3,300ms / 3,800ms 내에 실행 완료 체크)

Elapsed time	Request / Response	DEVICE_083 (<병렬 처리 개수> 2개)		DEVICE_015 (<병렬 처리 개수> 2개)	
0ms	첫번째 Request 송신	CMD_002_M		CMD_002_Q1	
500ms	두번째 Request 송신				
1,000ms			CMD_002_M	CMD_002_Q2	CMD_002_Q1
1,500ms					
2,000ms				CMD_002_Q3	CMD_002_Q2
2,500ms					
3,000ms	첫번째 Request 완료				CMD_002_Q3
3,500ms					
4,000ms	두번째 Request 완료				

테스트 방법 (계속)

[문항5]

- MOCK.BAT를 실행한 후, SP_TEST 실행
  (MOCK.BAT의 콘솔에서 <엔터키>를 입력하면 테스트가 시작됨)
- MOCK.BAT의 콘솔에 출력되는 테스트 결과 확인
  (SP_TEST가 문항의 요건을 만족하지 못하는 경우, MOCK.BAT의 콘솔에 테스트 Fail의 사유 또는 Stack trace가 출력되므로 자가 검수에 참고 요망)

```
C:\>MOCK.BAT<엔터키>

[MOCK] 시나리오 테스트 준비(SP_TEST 실행)가 완료되면 <엔터키>를 입력하세요.

[MOCK] Server/Device HTTP 서버 시작

2023-05-17 18:48:18.662:INFO::main: Logging initialized @10846ms to
org.eclipse.jetty.util.log.StdErrLog
...

[1번 시나리오] Device(DEVICE_069)가 CMD_003 Request 송신

[시나리오][DEVICE_069] Request 송신 (URI:http://127.0.0.1:8010/fromDevice,
Body:{"command":"CMD_003","param":"a7b248f5"})
[시나리오][DEVICE_069] Response 수신 (Status Code:200, Body:없음, Elapsed time:113ms)

[2번 시나리오] Device(DEVICE_069)가 CMD_004 Request 송신

[시나리오][DEVICE_083] Request 송신 (URI:http://127.0.0.1:8010/fromDevice,
Body:{"command":"CMD_004","param":"db538d30"})
[시나리오][Server] Request 수신 (URI:http://127.0.0.1:7010/fromEdge,
Body:{"command":"CMD_004_T","param":"3815fb04"})
[시나리오][Server] Response 송신 (Status Code:200, Body:없음)
[시나리오][DEVICE_083] Response 수신 (Status Code:200, Body:없음, Elapsed time:1,102ms)

[정답] 테스트에 성공했습니다!
```