Xnode로 배우는

센서 네트워크

9 Zigbee 통신 확장

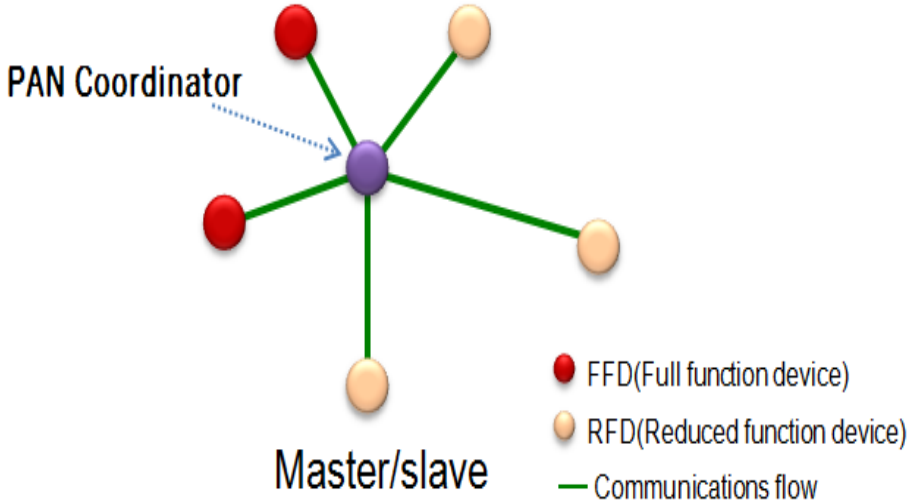
네트워크 토폴로지

- 네트워크 토폴로지
  - 네트워크에 참여한 노드들의 기하학적인 연결 형태
  - IEEE 802.15.4 응용 요구사항에 따라 스타(Star)와 피어투피어(Peer to peer) 토폴로지에서 작동 가능

네트워크 토폴로지

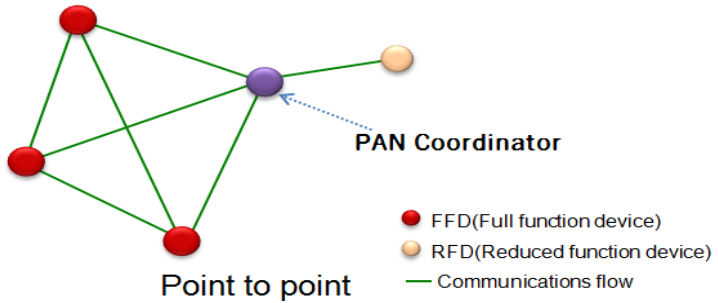
- 스타 토폴로지
  - PAN 코디네이터와 디바이스 간의 연결
  - 일부 디바이스에 설치된 응용프로그램은 네트워크 통신의 시작이나 종료에 관여
  - PAN 코디네이터는 네트워크의 초기화나 종료 및 네트워크 간의 경로 지정을 위해 특별한 응용프로그램을 사용
  - PAN 코디네이터는 PAN의 메인 컨트롤러이며 특정 토폴로지의 네트워크에서 운영되는 모든 디바이스는 64비트의 고유 주소를 갖습니다.
  - 하나의 PAN 코디네이터에 다수의 네트워크 디바이스가 참여하는 가장 일반적인 형태의 로컬 네트워크 구현
  - PAN 코디네이터를 중심으로 통신이 수행되므로 네트워크 구현 역시 가장 간단

네트워크 토폴로지


PAN Coordinator
Master/slave
FFD(Full function device)
RFD(Reduced function device)
Communications flow

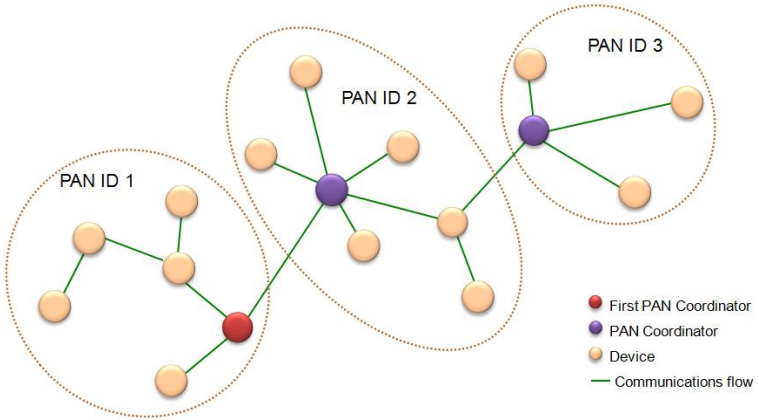
네트워크 토폴로지

- 피어투피어(Peer to Peer) 토폴로지
  - FFD로 구성되며, RFD가 포함될 때는 PAN 코디네이터를 추가
  - 로컬 네트워크에 주로 사용
  - 대부분의 네트워크 디바이스가 FFD로 구성되므로 전력소모가 많다는 단점이 있음
  - 모든 네트워크 디바이스간 통신 경로를 설정하기 위해 라우팅 알고리즘의 적용 필요
  - 피어투피어는 메쉬(Mesh) 토폴로지로도 불림



네트워크 토폴로지

- 클러스터 트리 토폴로지
  - 스타 토폴로지로 각각의 로컬 네트워크를 구성한 후 로컬 네트워크와 로컬 네트워크를 피어투피어 형태로 연결해 대규모 네트워크를 구성하는 토폴로지
  - 각 로컬 네트워크마다 PAN 코디네이터가 존재
  - 각 로컬 네트워크를 피어투피어로 연결하기 위해 복잡한 라우팅 알고리즘 적용

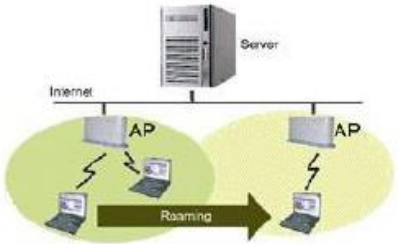


무선 통신 모드

- 무선 LAN이나 LR-WPAN과 같은 무선 기기 간의 두가지 통신 모드
  - 중계기나 기지국과 같은 별도의 기반 시설을 이용하는 인프라스트럭처 (Infrastructure)
    - 무선 공유기나 액세스 포인터(Access Pointer)를 휴대폰은 기지국(Base station)을 기반 시설로 이용해 기기 간에 통신을 수행
    - 기반 시설은 일정 거리 이상에서도 통신이 수행되도록 높은 안테나 출력을 유지
    - 대규모 무선 기기 간의 통신에 필요한 관리 기능을 제공
    - 주로 대량의 고속 통신 운영에 활용

무선 통신 모드

- 기반 시설 없이 직접 통신이 가능한 애드혹(Ad-Hoc)
  - 애드혹 모드는 기반 시설 없이 무선 기기 간에 직접 통신하는 방법
  - 기반 시설을 이용한 중앙 집중화 된 관리가 필요 없음
    - Xnode와 같이 네트워크 구성이 유동적이며 안테나 출력이 제한된 환경에서 주로 사용
  - 노트북에 내장된 무선 LAN을 애드혹 모드로 운영하면 액세스 포인터나 무선 공유기 없이 노트북 간에 직접 통신을 수행하는 것이 가능
    - 전송 속도나 통신에 참여 가능한 노트북 개수는 제한



애드혹 네트워크

- 애드혹 기반의 네트워크 특징
  - 보안
    - 통신 기기 간에 주고받는 정보에 대한 무결성 및 도청방지를 제공하는 기능
    - 정보에 대한 암호화 구현
  - 라우팅(Routing)
    - 라우팅은 통신 거리 확장 및 장애 회피를 위해 정보의 이동을 관리하는 기능으로 다양한 라우팅 알고리즘을 이용해 구현
  - 이동성

애드혹 네트워크

- 애드혹 네트워크의 보안 요구 조건은 다른 통신 네트워크와 동일
  - 무선 통신은 매체를 신뢰할 수 없는 상황에서 보안에 필요한 암호를 사용
  - 암호 키에 크게 의존
  - 키 사이에 신뢰할 수 있는 관계를 형성하고 이를 애드혹 네트워크 전반에 분배하는 것이 중요

애드혹 네트워크

- 애드혹 통신에서 사용하는 암호는 공개키 방식을 주로 사용
  - A, B, C 세 개의 그룹이 존재한다고 할 때 그룹 A의 대표 노드가 서버 노드 역할을 수행하여 신뢰 위임(Trust delegation) 절차를 주도
  - B, C 그룹의 대표 노드들은 자기 그룹이 정한 공개키를 다른 그룹과 교환해 그룹 간 신뢰 관계를 형성

애드훅 네트워크

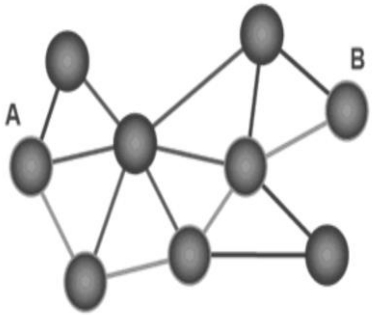
- 노드가 이동할 수 있는 애드훅 네트워크의 경우
  - 노드 간에 주고받는 정보가 네트워크 내에서 원활하게 교환하도록 관리하는 라우팅 기능 필요
    - 라우팅 : 네트워크에 참여한 노드들이 출발지에서 보낸 정보의 목적지까지 경로 배정 기능
  - 노드의 이동성이 심한 경우
    - 네트워크에 일반적인 정보 대신 라우팅 정보만 가득 찰 수 있음
    - 노드의 배터리 소모가 급속도로 증가하기도 함
    - 해결책 : 라우팅이 필요한 경우에만 경로 배정을 하는 반응에 따른 경로 배정 기법 필요

Mesh 네트워크

- 메시 네트워크
  - 네트워크의 각 노드가 주변의 다른 노드에 연결되는 토폴로지
- 메시 네트워크의 세 가지 기능
  - Routing : 메시지가 최종 목적지에 도달할 때까지 노드에서 호핑하여 경로를 따라 메시지가 전파됨
  - Ad-hoc network creation : 사람의 개입 없이 전체 노드 네트워크를 자동 생산하는 프로세스
  - Self-healing : 네트워크에서 누락된 노드를 자동 파악하고 복구하도록 네트워크 재구성

Mesh 네트워크

- 메시지를 전달하기에 충분한 노드가 있다면 두 노드 사이의 거리는 중요하지 않음
  - 노드간 통신시 네트워크는 자동으로 최상의 경로 계산
- 메시 네트워크는 안정적이며 중복성 제공
  - 노드가 네트워크에서 더 이상 작동 할 수 없는 경우
    - 나머지 노드는 여전히 직접 또는 중간 노드를 통해 서로 통신 가능



Mesh 네트워크 통신

- Mesh 네트워크 통신 예제

  | 1 | Xnode 3대 준비

  | 2 | 각각 Coordinator, Router, End Device로 설정.

  각각의 NI를 Coordinator, Router, EndDevice로 설정

  | 3 | Coordinator는 End Device 로 지속적으로 데이터를 전송하는 코드와 수신된 데이터를 출력하는 코드 실행

  | 4 | End Device는 수신된 데이터를 반송하는 코드를 실행

Mesh 네트워크 통신

| 5 | Router는 코드 실행 없음

| 6 | Coordinator와 End Device가 서로 데이터를 주고받을 수 없을 정도의 거리에 위치시킴

| 7 | Coordinator와 End Device 사이에 Router를 위치시켜 전원을 켜고, 껐을 때 Coordinator에서 출력되는 메시지 확인하여 Mesh 네트워크 동작여부 확인

Mesh 네트워크 통신

- Xnode 설정
  - 1개의 Coordinator, 1개의 Router, 1개의 End Device 필요
  - Xnode 3개를 다음과 같이 설정한 후 'WR' command 실행
  - 모든설정이 완료된 후 reset버튼을 눌러 네트워크 검색

Command	Coordinator	Router	End Device
NI	'Coordinator'	'Router'	'EndDevice'
CE	0x01	0x00	0x00
ID	0x15	0x15	0x15
JV	-	0x01	0x01
SM	-	-	0x04

Mesh 네트워크 통신

Coordinator

```
01:  from pop import xnode
02:
03:  xnode.atcmd('NI', 'Coordinator')
04:  xnode.atcmd('CE', 0x01)
05:  xnode.atcmd('ID', 0x15)
06:  xnode.atcmd('WR')
```

Router

```
01:  from pop import xnode
02:
03:  xnode.atcmd('NI', 'Router1')
04:  xnode.atcmd('CE', 0x00)
05:  xnode.atcmd('ID', 0x15)
06:  xnode.atcmd('JV', 0x01)
07:  xnode.atcmd('WR')
```

End Device

```
01:  from pop import xnode
02:
03:  xnode.atcmd('NI', 'Router2')
04:  xnode.atcmd('CE', 0x00)
05:  xnode.atcmd('ID', 0x15)
06:  xnode.atcmd('JV', 0x01)
07:  xnode.atcmd('SM', 0x04)
08:  xnode.atcmd('WR')
```

Mesh 네트워크 통신

- 설정이 완료된 후 Coordinator에서 네트워크를 검색
- 2개의 장치가 검색되는 것 확인

Coordinator

```
01:     from pop import xnode
02:
03:     for i in xnode.discover():
04:         print(i)
```

```
{'rssi': -20, 'node_id': 'EndDevice', 'device_type': 1179648, 'parent_nwk': 0, 'sender_nwk': 8607, 'sender_eui64': b'\x00\x13\xa2\x00A\xae\\\x07', 'node_type': 2}
{'rssi': -26, 'node_id': 'Router', 'device_type': 1179648, 'parent_nwk': 65534, 'sender_nwk': 42278, 'sender_eui64': b'\x00\x13\xa2\x00A\xae\\\x90', 'node_type': 1}
```

Mesh 네트워크 통신

- Coordinator 코드
  - Coordinator에서 End Device로 1초 간격으로 데이터를 전송
  - 수신된 메세지가 있다면 출력

Mesh 네트워크 통신

```
01:  from pop import xnode
02:  import time
03:
04:  for i in xnode.discover():
05:      if i['node_id'] == EndDevice':
06:          addr = i['sender_eui64']
07:          print("End Device addr : ", addr)
08:
09:  pre_stime = time.ticks_ms()
10:  Count = 0
11:
12:  while True:
13:      try:
14:          if time.ticks_ms()-pre_stime>1000:
15:              payload = 'Count' + str(Count)
16:              xnode.transmit(addr, payload)
17:              print('send msg : ', payload)
18:              Count += 1
19:              pre_stime = time.ticks_ms()
20:
21:          msg = xnode.receive()
22:          if msg['sender_eui64']==addr:
23:              print('received msg : ', msg['payload'])
24:
25:      except:
26:          pass
27:
28:      time.sleep(0.1)
```

Mesh 네트워크 통신

- End Device 코드
  - End Device에 수신된 payload를 송신 주소에게 다시 전송

```
01:  from pop import xnode
02:  import time
03:
04:  while True:
05:      try:
06:          msg = xnode.receive()
07:          if msg:
08:              addr = msg['sender_eui64']
09:              payload = msg['payload']
10:              xnode.transmit(addr, payload)
11:              print("send addr / msg : ", addr, "/", payload)
12:
13:      except:
14:          pass
15:
16:      time.sleep(0.1)
```

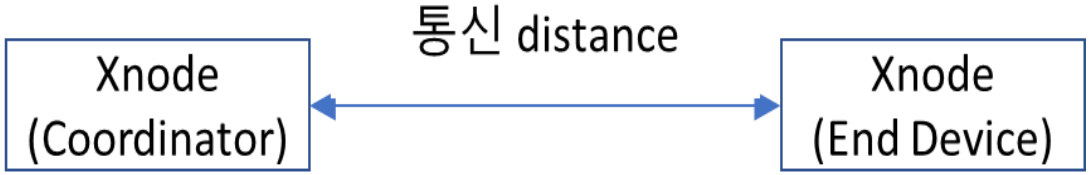
Mesh 네트워크 통신

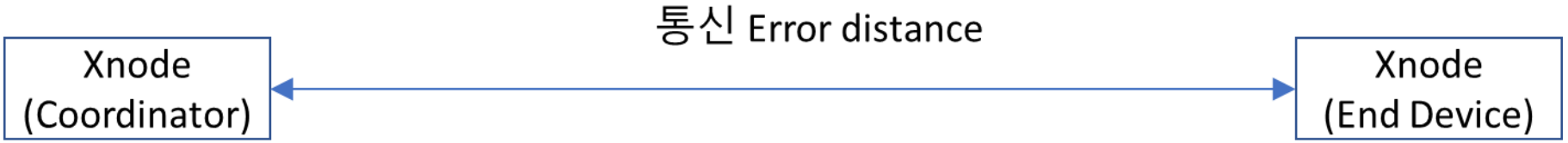
- Coordinator와 End Device가 서로 통신가능한 거리내에 있는 경우 Coordinator 출력

```
End Device addr :  b'\x00\x13\xa2\x00A\xae\\\x07'
send msg :  Count0
received msg :  b'Count0'
send msg :  Count1
received msg :  b'Count1'
```

Mesh 네트워크 통신

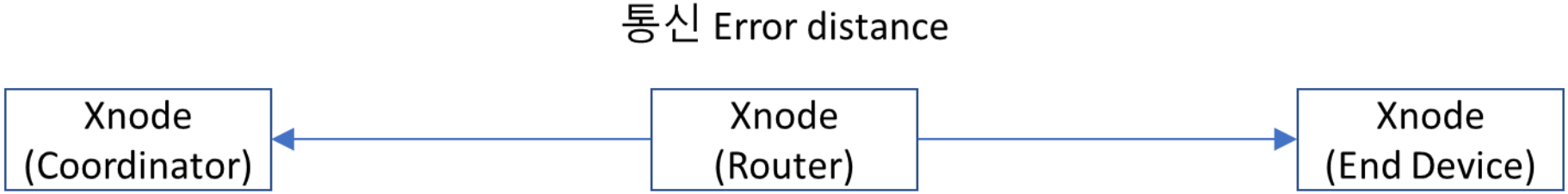
- Mesh 네트워크
  - Coordinator와 End Device가 실행중인 상태에서 서로 간의 거리를 점점 멀리 떨어뜨려 더 이상 데이터통신이 되지 않도록 두 장치를 위치시킴





Mesh 네트워크 통신

- Router 장치를 Coordinator와 End Device 장치 사이에 놓고 전원을 켰을 때 Coordinator와 End Device 장치 간에 통신이 이루어 지는지 확인




```
send msg :  Count0
received msg :  b'Count0'
send msg :  Count1
received msg :  b'Count1'
```