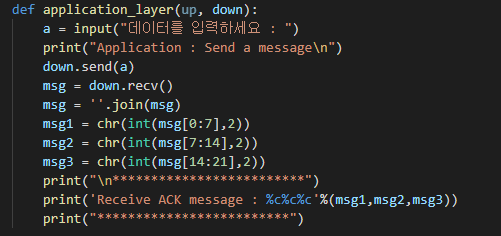
**Function block description**

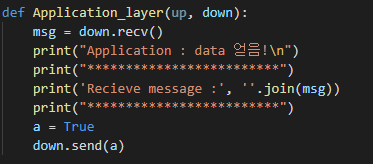
**20192234 방수경**

1. **Code block의 구조**
2. **함수의 종류**
3. **Application\_layer - Sender**



* 데이터를 입력 받고, down.recv으로 Transport\_layer로 내려주는 역할을 한다. receiver에서 ACK 메시지를 받으면 ‘ ’.join으로 문자열로 만들고, 각 자리 수를 2진수로 바꾸고, chr 함수를 사용하여 ASCII코드로 바꿔준다.

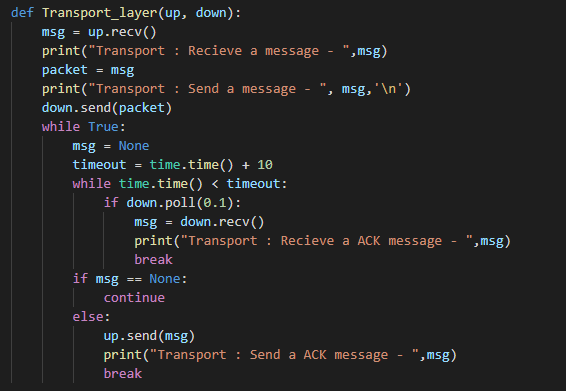
1. **Application\_layer – Receiver**



* Transport\_layer에서 받은 msg를 문자열로 출력해준다. 그리고 Transport\_layer로

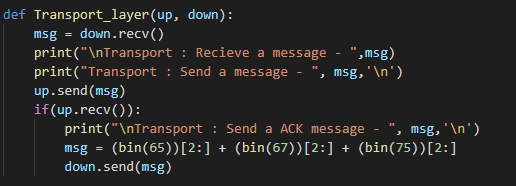
True를 내려서 원활한 통신을 할 수 있게 도와준다.

1. **Transport\_layer – Sender**



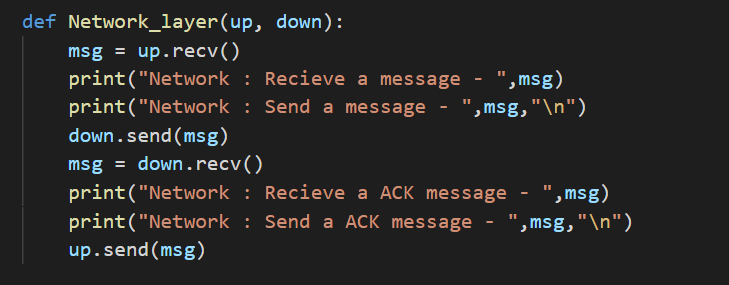
* Application\_layer에서 받은 msg를 packet을 변환시킨다. 그리고 stop and wait를구현했다. Timer를 10초로 설정했다. Network\_layer로 packet을 보낸다. 그리고 Receiver에서 ACK message를 받을 때는, Network\_layer에 msg가 있다면 받아서 출력하고, Application\_layer로 보낸다. 만약, Network\_layer에 msg가 없다면 있을 때까지 기다린다.

1. **Transport\_layer – Receiver**



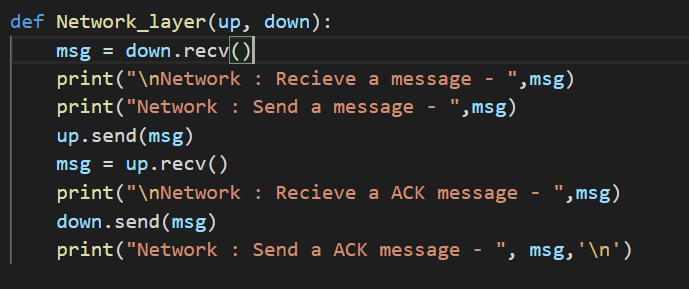
* Network\_layer에서 보낸 msg를 받는다. 그리고 Sender로 다시 보낼 때, B에서 받은 값이 있으면 ‘ACK’ message를 2진수로 만들어서 Network\_layer로 보낸다.

1. **Network\_layer – Sender**



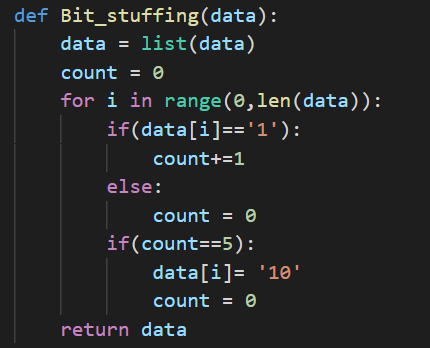
* Transport\_layer에서 보낸 packet을 msg로 받아서 Datalink\_layer로 보낸다. Receiver에서 받을 때는, Datalink\_layer에서 보낸 msg를 받아서 Transport\_layer에 보낸다.

1. **Network\_layer – Receiver**



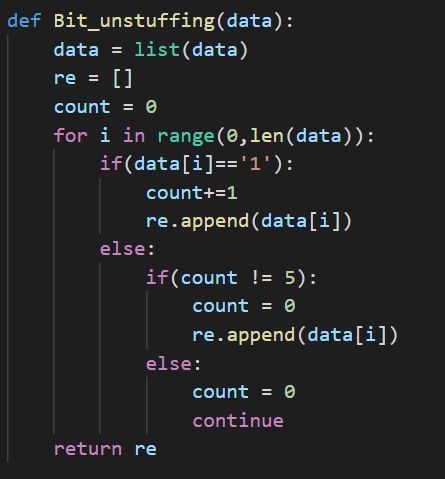
* Datalink\_layer에서 받은 msg를 Transport\_layer로 보낸다. Sender로 보낼 때, Transport\_layer에서 받는 msg를 Datalink\_layer로 보낸다.

1. **Bit\_stuffing**



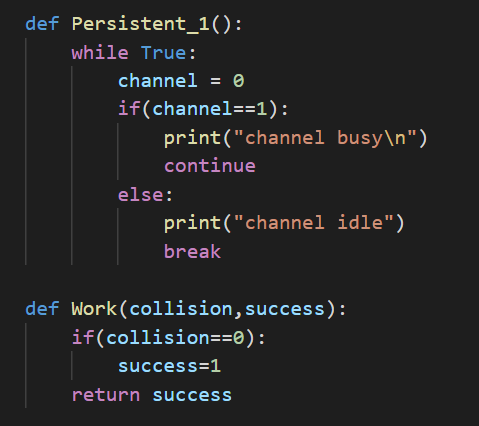
* Bit를 받아서 1이 5개 이상 들어가면 0을 넣는 Bit stuffing을 구현했다.

1. **Bit\_unstuffing**



* 1이 5개 있으면 0을 넣는 Bit stuffing에서 다시 0을 뺀다.

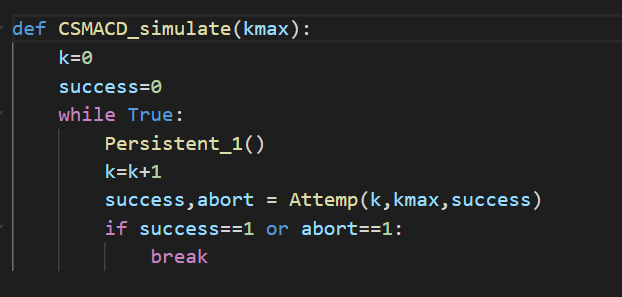
1. **Persistent\_1, Work, Attemp, CSMACD\_simulate -> CSMA/CD**



* idle할 때까지 계속 시도하는 Persistent1을 사용했다. 그리고 collision이 발생하지 않으면 성공했다는 Work를 구현했다.

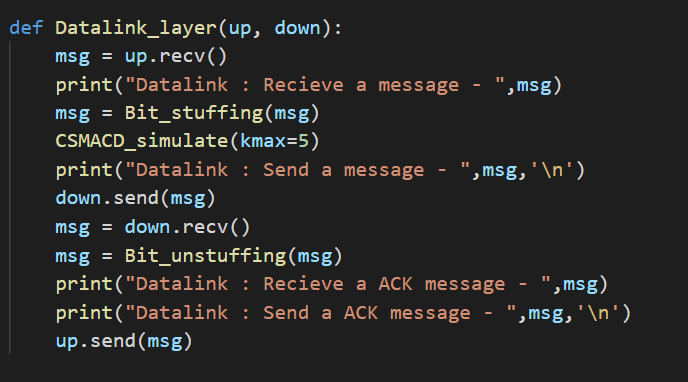


* Collision이 발생하지 않았고, 아직 성공하지 않으면 기다린다. 그리고 collision이 발생하지 않으면 성공했으며, 발생했으면 jamming signal을 보낸다. 시도횟수가 kmax를 넘으면 버리고, 다시 실행한다.



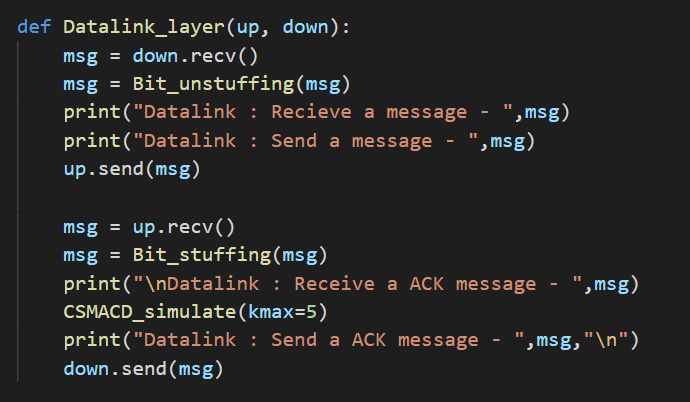
* 위의 함수들을 다 실행하는 CSMA/CD를 돌리는 CSMACD\_simulate 함수이다. 여기서는 원활한 통신을 위해 무조건 Persistent\_1가 idle하고, collision이 발생하지 않게 세팅하였다.

1. **Datalink\_layer – Sender**



* 위의 함수들을 사용한다. 먼저 Network\_layer에서 받은 msg를 Bit\_stuffing에 넣어서 bit stuffing을 해준다. 그리고 CSMA/CD를 사용하여, idle한 상태에서 msg를 Physical layer로 Simple Protocol을 사용하여 보낸다. 그리고 Receiver에서 받을 때는, Physical\_layer에서 받은 msg를 Bit\_unstuffing 함수에 넣어 bit unstuffing한다. 그리고 msg를 Network\_layer로 보낸다.

1. **Datalink\_layer – Receiver**



* Physical\_layer에서 받은 msg를 Bit\_unstuffing하여 Network\_layer로 보낸다.

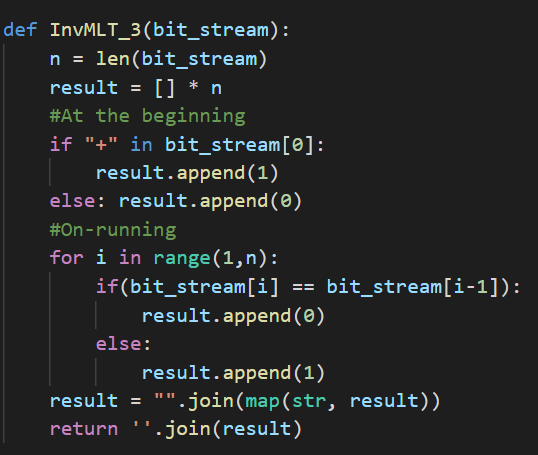
Sender로 보낼 때는, Network\_layer에서 받은 msg를 Bit\_stuffing하여 CSMACD\_simulate를 사용하여 idle할 때 simple protocol을 사용하여 보낸다.

1. **MLT\_3**



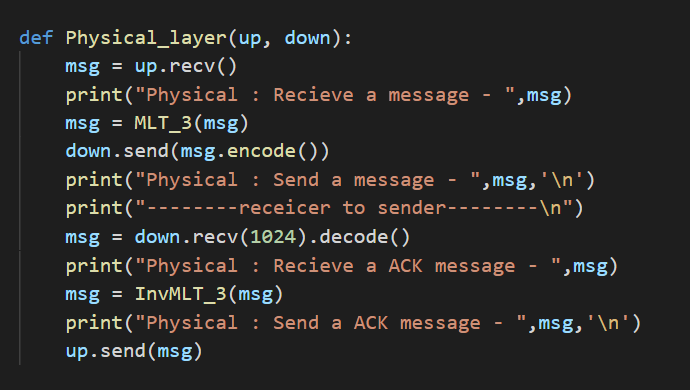
* 2진수의 Bit stream을 MLT\_3 알고리즘을 사용하여 +,0,-로 바꾸는 함수이다.

1. **InvMLT\_3**



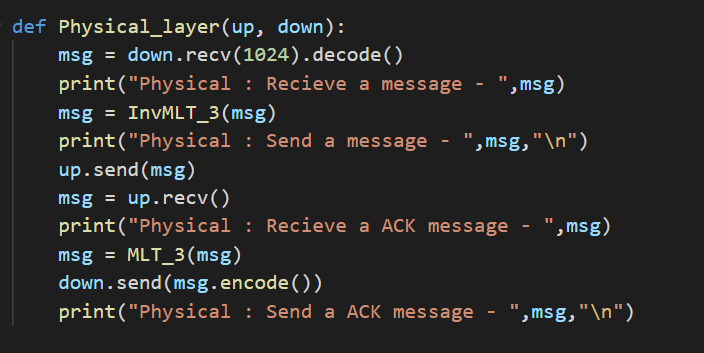
* MLT\_3를 사용하여 +,0,-로 구성된 Bit stream을 다시 2진수의 형태로 바꾸는 함수이다.

1. **Physical\_layer – Sender**



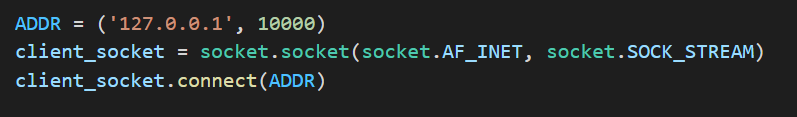
* 위의 MLT\_3 함수를 사용하여 Datalink\_layer에서 받은 msg를 바꾼다. 그리고 socket을 사용하여 encode해서 Receiver에게 보낸다. 그리고 Receiver에서 받을 때도 socket을 사용하여 받는다. 그리고 decode하여 받은 msg를 InvMLT\_3를 사용하여 원래 2진수 값으로 바꾼다. 바꾼 msg를 Datalink\_layer로 보낸다.

1. **Physical\_layer – Receiver**



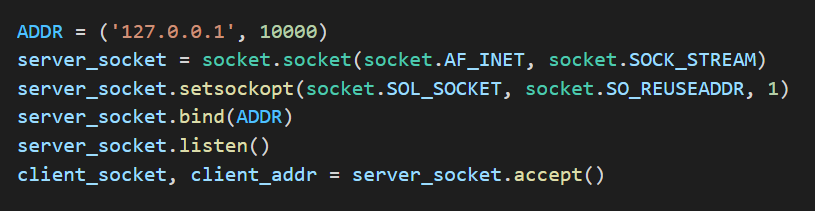
* Sender에서 socket통신으로 decode하여 받은 msg를 InvMlt\_3를 사용하여 원래 2진수 값으로 바꿔준다. 그리고 Datalink\_layer로 보낸다. Sender로 보낼 때는, MLT\_3을 사용하여 바꿔주고 socket 통신을 사용하여 encode하여 보낸다.

1. **Socket 통신 – sender**



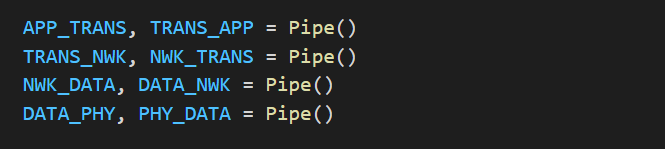
* Port를 10000으로 정하고 client\_socket로 연결한다.

1. **Socket 통신 – Receiver**



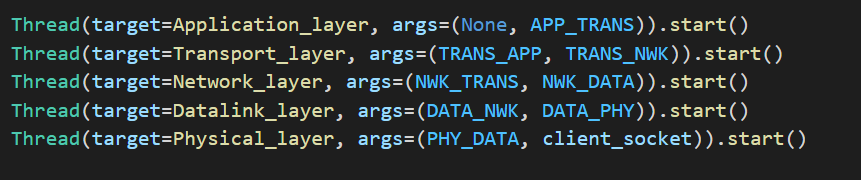
* Port를 10000으로 정하고, server\_socket를 설정한다. Bind를 사용하여 소켓이 네트워크와 port 번호에 연결하도록 한다. 그리고 sender가 연결하면 listen으로 혀용하고 accept한다.

1. **Pipe 통신**



* APP\_TRANS는 Application\_layer를 시작으로, Transport\_layer를 끝으로 연결해주고 TRANS\_APP는 Transport\_layer를 시작, Application\_layer를 끝으로 연결해준다.
* TRANS\_NWK는 Transport\_layer를 시작으로, Network\_layer를 끝으로 연결해주고 NWK\_TRANS는 Network\_layer를 시작으로, Transport\_layer를 끝으로 연결해준다.
* NWK\_DATA는 Network\_layer를 시작으로, Datalink\_layer를 끝으로 연결해주고DATA\_ NWK는 Datalink\_layer를 시작으로, Network\_layer를 끝으로 연결해준다.
* DATA\_PHY는 Datalink\_layer를 시작으로, Physical\_layer를 끝으로 연결해주고 PHY\_ DATA는 Physical\_layer를 시작으로, Datalink\_layer를 끝으로 연결해준다.

1. **Thread**



* 각 함수들의 매개변수에 위의 pipe통신을 넣어주고 Thread를 시작한다.

1. **Code block의 구조**

**sender receiver**

**Physical\_layer**

**(MLT-3, InvMLT-3)**

**Datalink\_layer**

**(Bit\_stuffing, Bit\_unstuffing, Persistent\_1, Work, Attemp, CSMACD\_simulate)**

**Network\_layer**

**Transport\_layer**

**(stop-and-wait 방식)**

**Application\_layer**

**Datalink\_layer**

**(Bit\_stuffing, Bit\_unstuffing, Persistent\_1, Work, Attemp, CSMACD\_simulate)**

**Physical\_layer**

**(MLT-3, InvMLT-3)**

**Network\_layer**

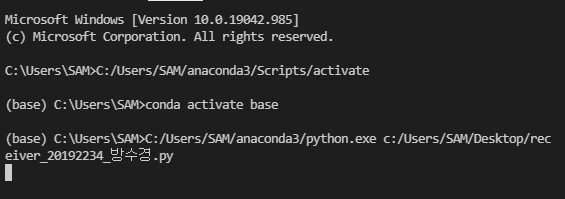
**Application\_layer**

**Transport\_layer**

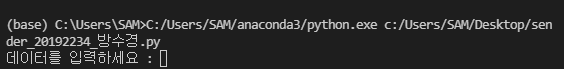
**(stop-and-wait 방식)**

1. **How to run**

1번) Receiver 코드 실행 -> Sender를 기다린다.



2번) Sender 코드 실행 -> Receiver와 연결된다.



3번) Sender에서 Receiver로 보낼 데이터를 입력 -> 바로 연결되어 값이 출력된다.

* **Sender의 경우**



* **Receiver의 경우**



이렇게 Sender와 Receiver가 통신하는 것을 볼 수 있습니다.

그럼 이상으로 보고서를 마치겠습니다.

감사합니다.