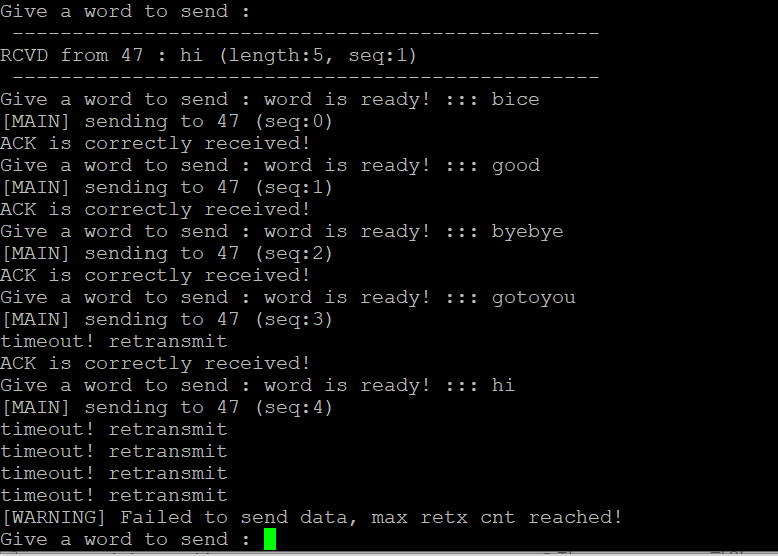
**실습 2 보고서**

201810900 권주환

**1. 결과 캡쳐**

 텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

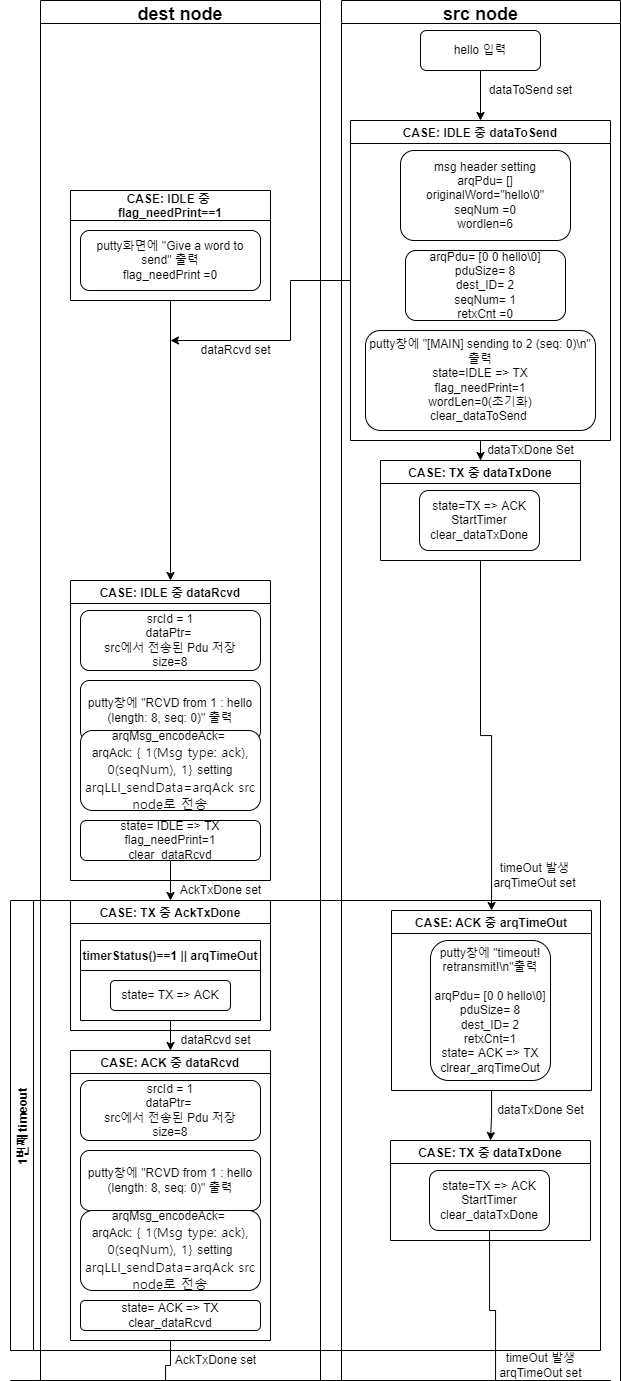
ARQ\_MAXRETRANMISSION=3으로 setting, timer가 만료될 때까지 ack를 수신하지 못하는 경

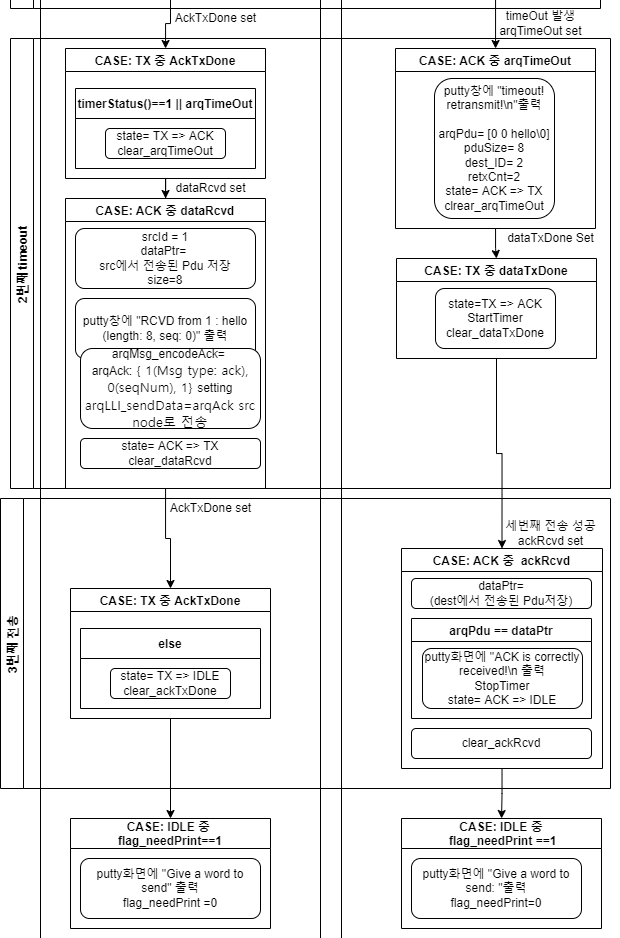
우 재전송을 3회까지하는 것으로 설정. 재전송 횟수를 나타내는 변수:retxCnt 0으로 초기화 되어있고 재전송 할때마다 1씩 증가. 그 값이 ARQ\_MAXRETRANMISSION의 값과 같아질 때 실패를 알리고 IDLE상태로 전이해야 재전송을 10회까지만 할 수 있습니다. 따라서 if문을 추가해 if문: retxCnt의 값을 확인하고 3을 넘었을 때 pc.printf("[WARNING] Failed to send data, max retx cnt reached!\n"); 명령을 추가해 사용에게 전송이 실패했음을 알리고, main\_state = MAINSTATE\_IDLE;를 통해 상태 전이. 전송이 실패했을 때 arqEvent\_dataTxDone flag의 경우 setting된 상태로 노드의 상태가 IDLE로 전이되면, 예외처리에 걸려 경고문을 출력합니다. Exercise2에서 원하는 것은 정상 동작이므로 arqEvent\_dataTxDone flag를 clear 해줍니다. Else문: 재전송 횟수를 초과하지 않는 경우에는 timer가 만료되어 재전송함을 사용자에게 알리고, 재전송을 수행하며, retxCnt를 1씩 증가

시키고, 노드의 상태를 Tx로 전이시킵니다.

**2. 동작과정 설명**

**>Flow chart**

****

****

**처음 두 노드의 상태**

Src 노드의 경우 endNode\_ID : 1, dest\_ID : 2, dest 노드의 경우 endNode\_ID : 2, dest\_ID : 1, 두노드 모두 flag\_needPrint : 1, prev\_state : 0 으로 초기화. 그리고 두 노드 전부

arqEvent\_clearAllEventFlag();를 수행하여 모든 event flag를 clear.

arqLLI\_initLowLayer(endNode\_ID) 함수를 통해 PHY MAC 주소를 초기화, 입력을 받을 수 있도록

pc.attach(&arqMain\_processInputWord, Serial::RxIrq);를 수행. 정상 실행을 가정하였으므로

prev\_state와 main\_state의 값이 달라도 debug\_if문을 실행하지 않고, prev\_state = main\_state;를 실행.

**Src node의 상태변화**

**0. case: MainSTATE\_IDlE 중 else if (flag\_needPrint==1)에 걸림**

putty창에 “Give a word to send: “출력 후 flag\_needPrint=0으로 만듬으로써, data 입력을 기다리는 상태.

**src노드에 ‘hello’를 입력하면, arqMain\_processInputWord()함수를 통해 그 입력을 받고, arqEvent\_dataToSend flag가 set**

**1. case: MainSTATE\_IDlE 중 else if (arqEvent\_checkEventFlag(arqEvent\_dataToSend))에 걸림**

보낼 Pdu header 세팅: PduSize =(arqPdu= [], originalWord="hello\0", seqNum =0 ,wordlen=6)

arqLLI\_sendData= (arqPdu= [0 0 hello\0], pduSize= 8, dest\_ID= 2)  
몇번째 data전송인지를 나타내기 위한 seqNum= 1   
재전송 횟수 초기화: retxCnt =0

putty창에 "[MAIN] sending to 2 (seq: 0)\n" 출력

상태 변환: IDLE => TX

전송의 모든과정이 끝난후 0번 과정을 하기위한 flage 세팅: flag\_needPrint=1  
data길이 초기화: wordLen=0  
그후 dataToSend event clear

Data를 dest node에 보낼 준비(setting)를 하고 = 보내는 과정이다.

Data를 보내고 나서 **arqEvent\_dataTxDone set**

**2. case MAINSTATE\_TX 중 else if (arqEvent\_checkEventFlag(arqEvent\_dataTxDone))에 걸림**

상태변환: TX => ACK

Ack에 대한 timer 시작

그 후 dataTxDone event clear

Src 노드가 dest노드로부터 ack 받을 준비를 하는 과정이다.

**3.첫번째 timeout발생 -> arqEvent\_arqTimeout set**

**case MAINSTATE\_ACK 중 else if (arqEvent\_checkEventFlag(arqEvent\_arqTimeout))에 걸림**

putty창에 "timeout! retransmit!\n"출력  
Pdu header 세팅: PduSize =(arqPdu= [], originalWord="hello\0", seqNum =0 ,wordlen=6(hello+\0의 길이))

arqLLI\_sendData= (arqPdu= [0 0 hello\0], pduSize= 8(header+wordlen), dest\_ID= 2)  
첫번째 재전송: retxCnt 0 -> 1

상태변환: ACK => TX

그 후 arqTimeout event clear

Timer의 timeout이 발생하여, 다시 dest노드에 보낼 준비를 하고 보내는 과정이다,

**4. Src node의 2번 과정과 동일**

**5.두번째 timeout발생-> arqEvent\_arqTimeout set**

**case MAINSTATE\_ACK 중 else if (arqEvent\_checkEventFlag(arqEvent\_arqTimeout))에 걸림**

putty창에 "timeout! retransmit!\n"출력  
Pdu header 세팅: PduSize =(arqPdu= [], originalWord="hello\0", seqNum =0 ,wordlen=6(hello+\0의 길이))

arqLLI\_sendData= (arqPdu= [0 0 hello\0], pduSize= 8(header+wordlen), dest\_ID= 2)  
두번째 재전송: retxCnt 1 -> 2

상태변환: ACK => TX

그 후 arqTimeout event clear

Timer의 timeout이 발생하여, 다시 dest노드에 보낼 준비를 하고 보내는 과정이다,

[3번과 retxCnt변수 값을 제외한 모든 값이 같음]

**6. Src node의 2번 과정과 동일**

**7.세번째 전송 ->성공 arqEvent\_ackRcvd set**

**case MAINSTATE\_ACK 중 if (arqEvent\_checkEventFlag(arqEvent\_ackRcvd))에 걸림**

DataPtr에 dest노드에서 받은 Pdu 저장

성공적으로 전송되었음을 가정하므로 dest노드에서 받은 pdu와 src노드가 보낸 pdu는 동일

-> if ( arqMsg\_getSeq(arqPdu) == arqMsg\_getSeq(dataPtr) )에 걸림

putty화면에 "ACK is correctly received!\n 출력  
Ack에 대한 Timer멈춤  
상태변환: ACK => IDLE

그 후 ackRcvd event clear

Src노드가 ack를 성공적으로 받아, ack를 받았음을 알리고, 다시 event를 기다리는 상태로 가기위한 과정이다.

**8. Src node의 0번 과정 진행 후 EVENT 대기 상태**

**Dest node의 상태 변화**

**Src node의 0번 과정 진행 후 EVENT 대기 상태일 때, src node로부터, data 받는 event 발생-> arqEvent\_dataRcvd set**

**1. case MAINSTATE\_IDLE 중 if (arqEvent\_checkEventFlag(arqEvent\_dataRcvd))**

받은 data 저장: srcId = 1(arqLLI\_getSrcId(); 함수를 통해, srcnode의 id저장) , dataPtr= arqLLI\_getRcvdDataPtr(); 함수를 통해, src에서 전송된 Pdu 저장 , size=8(arqLLI\_getSize() 함수를 통해, pdu header의 길이+wordlen)

putty창에 "RCVD from 1 : hello (length: 8, seq: 0)" 출력

 arqMsg\_encodeAck를 통해, arqAck : { 1(Msg type: ack), 0(seqNum), 1} 이 저장되고, arqLLI\_sendData(arqAck(앞서 저장한 값), ARQMSG\_ACKSIZE(arqMsg\_encodeAck함수를 통해 값 저장), srcId(1)); 함수를 통해 ack를 src 노드로 전송하여 **arqEvent\_ackTxDone flag가 set**

상태변환:IDLE=>TX

전송의 모든과정이 끝난후 0번 과정을 하기위한 flage 세팅: flag\_needPrint=1

그 후 dataRcvd event clear

Src node로부터 전송된 data를 받고, 받은 data저장 후, scr node로 전송할 ack를 setting 후, 전송하는 과정이다.

**2. 첫번째 timeout**

case MAINSTATE\_TX 중 if (arqEvent\_checkEventFlag(arqEvent\_ackTxDone))에 걸림

첫번째 timeout이 일어났으므로 if (arqTimer\_getTimerStatus() == 1 || arqEvent\_checkEventFlag(arqEvent\_arqTimeout))에 걸림

상태변환: TX => ACK

그 후 ackTxDone event clear

Timeout이 일어났으므로, ack를 다시 보내기 위한 상태로 가기 위한 과정이다.

**Src로부터 다시 data를 받았으므로, arqEvent\_dataRcvd set**

**3. case MAINSTATE\_ACK 중 else if (arqEvent\_checkEventFlag(arqEvent\_dataRcvd))에 걸림**

받은 data 저장: srcId = 1(srcnode의 id) , dataPtr= src에서 전송된 Pdu 저장 , size=8(pdu header의 길이+wordlen)

putty창에 "RCVD from 1 : hello (length: 8, seq: 0)" 출력

 arqMsg\_encodeAck를 통해, arqAck : { 1(Msg type: ack), 0(seqNum), 1} 이 저장되고, arqLLI\_sendData(arqAck(앞서 저장한 값), ARQMSG\_ACKSIZE(arqMsg\_encodeAck함수를 통해 값 저장), srcId(1)); 함수를 통해 ack를 src 노드로 전송하여 **arqEvent\_ackTxDone flag가 set**

상태변환:ACK=>TX

그 후 dataRcvd event clear

Dest node의 1번과정과 매우 유사하며, 1번과정과 마찬가지로 data를 받고, ack를 세팅 후 보내는 과정이다.

**4.두번째 timeout**

**Dest node의 2번과정과 동일**

**Src로부터 다시 data를 받았으므로, arqEvent\_dataRcvd set**

**5. Dest node의 3번과정과 동일**

**6. 세번째 전송 -> 성공** **arqEvent\_ackTxDone set**

case MAINSTATE\_TX 중 if (arqEvent\_checkEventFlag(arqEvent\_ackTxDone))에 걸림

ack 전송에 성공했으므로, else문에 걸려 상태변환: TX =>IDLE

data,ack 전송이 성공적으로 이루어졌고, 이를 알리고 다시 event를 기다리는 상태로 가기위한 과정이다.

**7. Src node의 0번과정과 동일**