Network Application

hw6

20190532이상윤

2024

**Manchester encoding**

* NRZ방식으로 인코딩 한 데이터와 현재 slave의 clock을 xor시켜서 바꾼다.

텍스트, 폰트, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

* 이 방식은 NRZ나 NRZI로 만든 신호와 똑같은 정보를 보내기 위해서 2배의 대역폭을 사용 해야 한다는 단점이 있다.
* 하지만 이 방식은 수신자가 송신자의 clock을 꺼낼 수 있기 때문에 clock 동기화를 서로 할 수 있다는 매우 큰 장점이 있다.
* manchester가 아닌 인코딩을 사용하면 Master는 downlink를 통해서 주기적으로 clock 정보를 보내야 하는 overhead가 발생하는데, manchester는 간단한 연산을 통해서 수신자가 송신자가 보낸 데이터 clock을 추출 할 수 있어서 clock 정보를 보내는 overhead가 사라진다.
* 우리 상황에서는 주기적으로 데이터도 보내고, 순간 많은 데이터를 보내야 하는 등, 다양한 데이터 전송이 있기 때문에 클락을 맞추는 작업에 시간을 쏟으면 안된다고 생각한다.

**Packet Format**

|Preamble|src|dst|type|seqNum|len|Payload|CRC|

* Preamble (8byte)
  + 송신자와 수신자의 동기화를 위해 사용한다. 10101010 으로 되는 8개의 비트열을 전달한다
  + 이 시간에 맨채스터 인코딩으로 얻은 clock 정보를 사용한다.
* src (2byte)
  + 송신자 번호
* dst (2byte)
  + 수신자 번호
* type (1bit)
  + 패킷의 유형
  + ACK일때는 0, ACK가 아닐때는 1
* len (1byte)
  + payload의 길이
* payload
  + 실제 데이터
  + ACK일때는 ACK 순서 번호를 여기에 담아서 전송한다.
* CRC (2byte)
  + 오류 검출 코드

**마스터 → 슬레이브**

* 특정 주파수 대역을 이용해서 신호를 전송한다.
* 슬레이브가 데이터를 받을때 다른 주파수에서 오기 때문에 주파수 분할을 이용해 송신과 수신을 동시에 수행할 수 있다.

**슬레이브 → 마스터**

* uplink와는 다른 주파수 대역을 이용해서 신호를 전송한다.
* down link 주파수 대역을 슬레이브 개수만큼 나누고, 정해진 주파수대역만 각 슬레이브가 사용한다.
* 마찬가지로 마스터는 주어진 주파수에서 데이터를 추출한다.
* 마스터가 데이터를 받을 때 다른 주파수에서 오기 때문에 주파수 분할을 이용해서 송신과 수신을 동시에 수행할 수 있다.

**오류 검출**

* 각 패킷에 CRC를 사용해서 데이터 무결성을 확인한다.
* 마스터와 슬레이브는 모두 오류가 있는 경우 중복 ACK를 전송한다.
* 중복 ACK를 받으면 해당 순서 번호의 ACK를 재전송한다.

**패킷 손실**

* 패킷을 보낼때 마다 타이머를 건다.
* 타임아웃이 발생하면 해당 패킷을 재전송한다.
* Ack를 받으면 타임아웃을 해제한다.

**재전송 매커니즘**

* 패킷을 재전송할 때, 동시에 재전송하면 충돌이 발생하기 때문에 대책이 필요하다.
* 따라서 타임아웃이 발생하면 다음과 같이 재전송을 수행한다.
  1. 초기 대기시간을 설정한다. 폰트, 로고, 그래픽, 상징이(가) 표시된 사진

     자동 생성된 설명
  2. 타임아웃이 발생하면 대기 시간을 두배로 증가하되 무작위 Jitter를 추가한다.
  3. 대기 시간 T는 다음과 같이 계산한다.
     1. 텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

        자동 생성된 설명random factor는 -0.5에서 0.5사이 무작위 값이다.
  4. 해당 jitter가 추가된 대기시간만큼 기다린 다음에 재전송을 시도한다.
  5. 성공하면 초기 대기시간으로 돌아간다.
  6. 실패하면 collision count를 1 올리고 다시 대기 시간을 계산하고 대기한다.
* 이 방식은 슬레이브 마스터 모두 적용한다.
* 이 방식은 네트워크에서 충돌을 줄이고, 재전송의 무작위성을 추가해서 충돌 확률을 더욱 낮춘다.