반복적인 SNTP를 이용한 공장화 시스템에서의 시간 동기화

정 승 한*1, 권 오 봉*
*전북대학교 컴퓨터공학부
1e-mail: jshan0731@gmail.com

Clock Synchronization in Plants System using Repetitive SNTP

Seung-Han Jeong*1, Oubong Gwun*
*Dept. of Computer Engineering, Chon-Buk National University

요 약

최근 임베디드 시스템이 이용되는 이동체나 자동화 공장에서 각 디바이스들간의 분배된 작업이 정확한 시간에 맞추어 작동 할 것을 요구하고 있어 시간 동기화는 중요하게 인식되고 있다. 각각의 시스템은 자기의 시계를 가지고 처리되기 때문에 이들간의 시간 정보의 오차를 적게 하여야 한다. 이에 본 논문에서는 SNTP 프로토콜을 이용해 타임서버로부터 시간의 단위를 정하여 단위 간격마다 시간을 여러 번 확인하여 오차를 조정하고, 실험을 통해 평균오차와 최대오차를 구하여 시간 동기화 시점을 확인하였다. 이를 통해 시간 동기화 시점을 미리 조정하여 다음 장치들 간의 통신 시점에서 동기화를 할 경우 오차를 줄이는 방법을 제안한다. 본 논문에서 제안한 방법을 Rabbit Board 6700 으로 구현하고 검증하여 이를 시간적 정밀함이 요구되는 응용분야에 적용할 수 있게 한다.

1. 서 론

최근 임베디드 시스템이 사용되는 이동체, 자동화 공장 등에서 각 디바이스들간의 시간 동기화는 매우 중요하다. 각 디바이스들에서 부품을 만들고 조립하는 공정이 정해진 시간에 정확하게 작동하지 않는다면 부품을 생산하는 시간의 지연과, 지연에 따른 부품들의 부적합한 조합으로 인해 품질과 성능이 크게 떨어질 가능성이 크기 때문이다. 시간의 지연은 생산력저하로 경제성에 큰 타격을 주게 된다. 따라서 타임 서버로부터 정확한 시간을 얻어오는 것도 중요하지만 시간을 여러 번 체크하여 각 디바이스들간의 오차율을 줄일 수 있다면 정확성을 필요로 하는 무 정지 공정과 같은 환경에서 이더넷/무선[1]을 이용한 어플리케이션이나 컨베이어 시스템 등을 보다 신속하고 효율적으로 관리할 수 있을 것이다.

2 관련 연구

서론에서 제시한 것과 같이 이러한 문제를 해결하기 위한 동기화 방법으로는 NTP(Network Time Protocol)[2], PTP(Precision Time Protocol)[4], SNTP(Simple Network Time Protocol)]이 있다. NTP와 PTP는 Master-Slave[5] 구조고 SNTP와 유사하지만 slave 장치가 자신의 클럭과 master 장치의 클럭 간에 offset을 결정하게 해주는데 목적이 있다. Offset을 o(t)라고 하고 나타낸 식은 o(t) = s(t) - m(t)로 s(t)는 slave 장치의 시간, m(t)는 master장치의 시간을 나타낸다. 이 프로토콜에서 마스터 장치는 주기적으로 slave와 메시지 교환을 하며 slave가 offset을 재계산하는데 도움을 준다. 이 프로토콜은 offset의 교환이 매우 작거나 slave에서 master로 가는 시간과 master에서 slave로 가는 시간이 같다는 가정하에 설계되었다. NTP와 PTP는 정밀한 시간 동기화에 쓰이지만 Client와 Server를 넓은 범위의 네트워크에서 사용하면 신호가 늦어지는 문제가 있을 수 있다. 그리고 여기서 SNTP Client의 최상위층의 서브넷에서 시간을 받아 각 디바이스에 보내는 구조의 응용시스템에 적용하기 위하여 SNTP를 이용하였다.

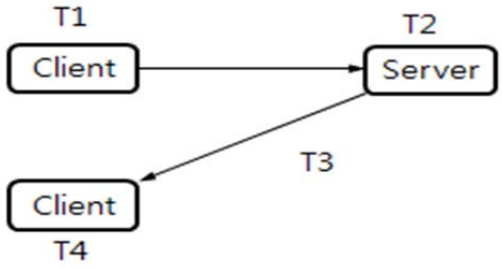
3. 반복적인 시간동기화

3 장에서는 본 논문을 제시하기에 앞서 먼저 SNTP의 기본구조에 대한 설명을 하고 이를 확

장하여 SNTP를 이용한 시간의 동기화 방법에 대해 제안한다.

3.1 SNTP 기본 구조

SNTP(Simple Network Time Protocol)는 RFC2030 에 정의 된 네트워크 장비의 시간 동기화 프로토콜로써 UDP 123 포트를 사용하며 네트워크 장비의 전원 사용 여부에 관계없이 항상 동작한다. 시간은 초 단위까지 동기화가 가능하며 프로토콜 단계에서는 NTP와 동일하지만 시간 오프셋 및 분석을 결정하는 SNTP의 타이밍 알고리즘과 중복성 기능들은 NTP와 다르다. SNTP의 기본구조는 (그림 1)과 같다.

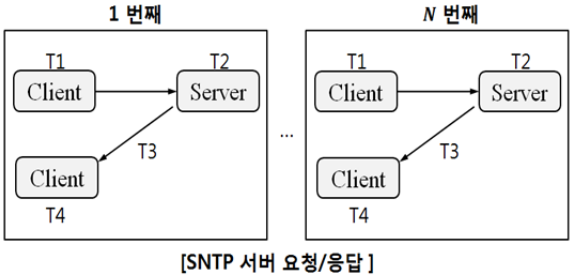


(그림 1) SNTP의 기본 구조

SNTP의 기본 알고리즘에 따르면, T1 은 Originate Time Stamp, T2 는 Receive Time Stamp, T3 는 Transmit Time Stamp, T4 는 Destination Time Stamp 이다. T1 은 Client에 의해 시간 요청을 보낸 것, T2 는 Server에 의해 시간 요청을 받은 것. T3 는 Server에 의해 시간정보를 보낸 것, T4 는 Client가 Server로부터 시간 정보를 받은 것을 나타낸다. Client는 현재 시간을 설정하고, Server에게 시간 요청을 한다. 시간 요청을 받은 Server는 가지고 있는 표준시간을 Client에게 응답한다. Client는 응답 받은 시간을 현재 시간과 비교하여 현재시간을 설정한다.

3.2 반복적으로 체크하는 시간 동기화

3.1 에서 설명한 바와 같이 SNTP의 구조를 통해 Client가 Server로부터 시간을 반복적으로 얻어오는 방법을 타임스탬프 T에 대하여 (그림 2)로 나타내었다.



(그림 2) 반복적인 시간의 요청 구조

앞에서 설명한 것과 같이 시간 정보를 얻어오는 방법에는 여러 가지가 있으나 본 논문에서는 기본적인 SNTP구조를 확장하였다. NTP는 송신시간과 액세스 시간을 예측하기 어려운 문제점을 갖고 있어[4] 본 논문에서는 SNTP를 채용하였다. 기본적인 SNTP는 서버로부터 시간 정보를 단순히 한번만 얻어오기 때문에 네트워크에 지연이 있는 경우 그 다음 요청과의 오차율이 얼마나 되는지 측정하기 어렵다. 그렇기 때문에 SNTP를 확장하여 시간의 정보를 여러 번 얻어와 오차율을 줄일 수 있는 방법을 제안 한다.

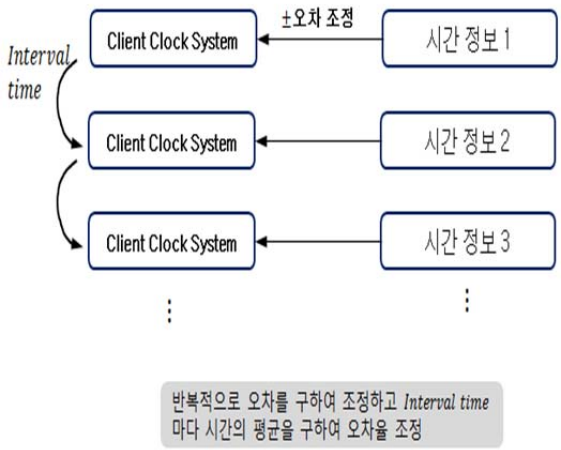
제안한 방법은 Server에게 1 번째 요청부터 10 번째 요청까지 여러 번의 요청을 하여 Server로부터 요청한 시간의 정보를 받게 된다. 10 번째 이상을 요청해도 무방하나 컴퓨터의 전체 가용시간에 대하여 표준시간을 얻어오고 이를 처리하는데 걸리는 시간을 고려하여 10 번으로 한정하였다 위에서 설명한 내용을 식으로 표현하면 다음과 같다.

$$\theta = \left[\frac{(T2 - T1) + (T3 - T4)}{2}\right] \cdots ①$$

$$avg = \frac{\sum_{i=0}^{n}\left[\frac{(T2_i - T1_i) + (T3_i - T4_i)}{2}\right]}{N} \cdots ②$$

위 식을 분석하면 ①식은 Client가 Server에게 한번의 요청을 하고 이를 이용해서 얻은 타임 스탬프 오프셋을 말한다. 그리고 ②식의 분자는 ①식을 첫 번째부터 10 번까지 요청했을 때 전체 합을 나타낸다. 따라서 전체 평균(avg)을 내기 위해서 ②식을 전체 요청한 횟수 10 으로 나누면 된다. SNTP는 앞서 말한 바와 같이 초단위로 계산을 하기 때문에 시간의

초까지 계산을 한다. 다음 단계는 10 으로 나눈 전체 시간의 평균(avg)을 통해 Client가 가지고 있는 클럭 시스템과 비교를 한다. 만약 비교를 했을 시에 현재 클럭 시스템의 오차가 있다고 가정한다면 발생 된 오차만큼 ±를 한다. 그리고 주기적인 Interval time을 가지고 체크를 한다. 예를 들어 Interval time이 1 시간이라고 가정한다면, 1 시간 뒤에 위의 avg식을 통해 다시 한번 시간정보를 얻어온다. 그 후 다시 Client의 클럭 시스템과 비교하여 발생된 오차만큼 ±하여 시간을 조정한 뒤 이를 하루 주기로 체크를 하고 하루 동안 얻은 값들을 통해 클럭 시스템을 ±를 판별하여 속도를 빠르고 느리게 조정한다. 위 설명에 대한 내용을 (그림 3)과 같이 나타내었다.

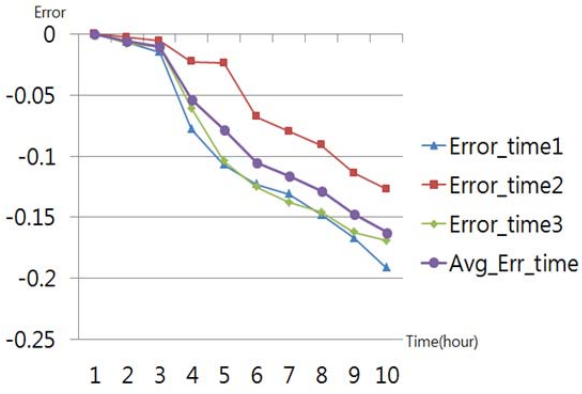


(그림 3) 시간 정보를 이용한 오차 조정

오차 조정을 한 후, 시스템 클럭의 주파수를 조절한다. 사용한 보드는 oscillator 주파수가 최저 전력인 32kHz에서 한 사이클의 500 번까지 호출 되도록 되어있다[5]. 따라서 이 환경에서 장치 별 Frequency를 조정하여 시간동기화를 하였다.

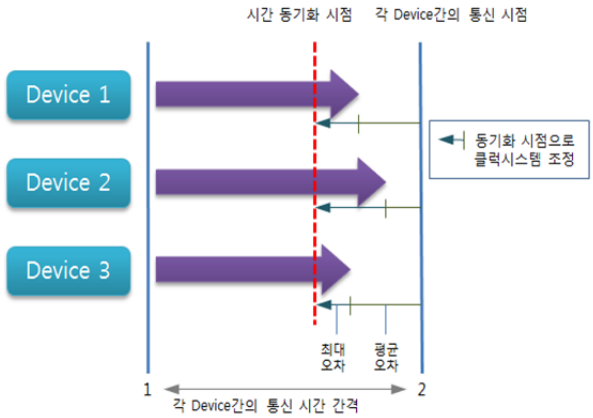
4. 실험 및 결과

본 논문의 연구는 임베디드 시스템 환경을 가지기 위해 임베디드 시스템으로 구성된 보드인 Rabbit Board 6700 에서 실험했다. 다음은 시간에 따라 얻은 오프셋 결과이다.



(그림 4) 시간에 따른 오프셋 확인

(그림 4)는 시간에 따라 얻은 오프셋 주기의 결과이다. 클럭의 각 장치들의 시간에 따른 Error를 각각 구하였다. (그림 4)를 보면 알 수 있듯이 각 장치의 클럭이 점점 느린 것을 알 수 있고 이 것을 이용하여 각 Error_time을 통해 평균을 얻을 수가 있다. 장치들 간의 통신에서 오차를 줄이기 위해서는 주기 단위로 확인하여 줄이는 것이 효율적이기 때문에 한 시간 단위로 장치들 간의 통신을 하도록 하였다.



(그림 5)동기화 시점으로 클럭 시스템 조정

(그림 5)는 장치들 간의 통신을 하게 되면 어느 시점에서 클럭 시스템이 빠르거나 느릴 수 있기 때문에 (그림 4)에서 얻게 되는 값들을 통해 시간 동기화 시점을 미리 앞당겨 주는 방법을 나타낸다. 이 방법을 통해 장치들 간의 시간 동기화 시점은 같아지고 다음 장치들의 통신 시점에서 동기화를 할 때 좀 더 오차를 줄일 수 있다. 시간 동기화의 시점을 구하는 식은 다음 ③식과 같다.

시간동기화 시점 = 평균오차 + 최대오차 …③

<표 1>은 시스템 클럭에서 최저전력 32kHz에서 최대 500 번이하에서 각 Frequency별 클럭 시스템을 조정 하여 얻은 결과이다.

<표 1>장치 별 Frequency(단위 :Hz)

Device B1	162463744
Device B2	40599552
Device B3	81199104
Device B4	60915712

<표 1>에서 얻은 결과를 통해 장치 별 클럭 차이를 이용하여 각각의 Device의 Clock Doubler를 조정하여 장치 별로 시간이 빠르면 Doubler를 빠르게 조정하고 시간이 느리면 Doubler를 느리게 조정하여 연산속도에 따른 주파수 조정을 하였더니 시간의 오차를 좀더 줄일 수 있었다.

5. 결 론

본 논문에서는 SNTP을 이용한 장치들 간의 오차를 개선하기 위한 방법으로 시간 단위마다 시간을 확인하여 평균을 구하여 오차를 조정하였고 각 장치 간의 시간 동기화 시점을 미리 조정하여 다음 장치들 간의 통신 시점에서의 동기화를 할 때 오차를 줄이게 함으로써 보다 정확한 결과를 얻는 실험을 하였다. 그리고 클럭 시스템을 조정한 값들의 시간 간격마다 평균을 이용하여 시스템 clock frequency를 설정함으로써 한번 얻어 왔을 때의 시간의 불안정성을 개선하여 오차율을 줄이고, 보드에서 얻은 frequency 조정 값을 이용해 시스템을 저전력 상태에서 효율적으로 운영할 수 있는 방법을 제안하였다. 위 방법은 이동체나 공장 자동화 시스템뿐만 아니라 농업 시스템에서 광원의 투광시간을 정밀조정하거나 방송용 시스템에서 가정이나 회사로 채널 별 송수신을 정확한 시간에 해야 할 경우 등 여러 분야에서 쓰일 수 있을 것으로 기대한다.

참 고 문 헌

[1]노진홍, 홍영식. ”무선 임베디드 환경에서의 시간동기화”,정보과학회논문지,제 32 권,제 6 호, 2005,pp.668-675

[2]이상엽, 김영호, “NTP 상의 동적 지연 시간 측정”, 한국정보과학회, 가을 학술발표논문집 제 24 권 제 2 호(III), 1997

[3]조현태, 조봉래, 진영우, 백윤주, “무선 센서 네트워크를 위한 PTP 기반 시각 동기 시스템”, 대한전자공학회 하계종합학술대회, 2009

[4]이윤준, 홍영식, “무선환경에서 분산 임베디드 시스템을 위한 시간 동기화 기법”, 학술발표논문집III

[5]MiniCore RCM6700 Series 관련 URL http://www.digi.com/products/wireless-wired-embedded-solutions/solutions-on-module/rabbit-minicore/rcm6700#docs