분산 컴포넌트 시스템에서 동기화된 시간의 정밀도를 높일 수 있는 기법

박수환 ○ 이창건 ○ 하은용
서울대학교 컴퓨터공학과
cordiallys@gmail.com, cglee@snu.ac.kr, eyha@anyang.ac.kr

A Clock Synchronization Protocol to Enhance the Clock Accuracy in Distributed Component Systems

Soo-Hwan Park○ Chang-Gun Lee○ Eun-Yong Ha
Dept of Computer Engineering, Seoul National University

요 약

분산 컴포넌트 시스템에서 여러 컴포넌트가 맞물려 서비스를 수행할 때 기준이 되는 시간이 필요하고 각 컴포넌트 별로 시간 오차가 발생하는 상황에서 시간 동기화 과정이 필요하다. 안정성이 중요시되고 실시간성을 보장하고자 하는 시스템에서 동기화된 시간의 정밀도는 중요한 이슈가 되고 있는데 현재까지 제안된 시간 동기화를 그대로 사용할 경우 발생할 수 있는 딜레이 요소들로 인해 동기화된 시간의 정밀도가 떨어진다. 따라서 본 논문에서는 실제 환경에 시간 동기화가 이루어질 때 오차를 발생시키는 요소들을 지적하고 보완할 수 있는 방법들은 제안함으로써 시간 동기화의 정확도를 높인다.

1. 서 론

분산 컴포넌트 시스템이란 모듈로 구성된 컴포넌트가 네트워크로 연결되어 상호 간에 데이터를 주고 받으며 컴퓨팅을 하는 형태의 집단을 일컫는다. 로봇과 같은 복잡한 분산 컴포넌트 시스템에서는 수많은 컴포넌트가 존재하고 그 컴포넌트들 간에 교환되는 데이터의 양도 점점 커지고 있는데 이를 위해 반드시 필요한 것 중 하나가 컴포넌트 간의 시간 동기화이다. 그리고 수많은 컴포넌트 간에 복잡한 데이터 교환을 요하는 로봇 시스템과 같은 경우에는 밀리초 혹은 마이크로초 단위까지의 정교한 시간 동기화가 필요하다.

분산 시스템에서의 시간 동기화에 대한 연구가 그 동안 많이 되어 왔지만 그 정확도는 10ms, 100ms이상의 오차를 보이거나 마이크로 컨트롤러에서 발생하는 블록, 네트워크 트래픽에 따른 딜레이 등의 주변 상황에 대해서 영향을 받는 정도에 대한 연구와 이를 극복할 수 있는 방안에 대해서는 많이 진척된 바가 없어 본 논문에서는 이에 대해 언급한다. 실제로 시간 동기화에 영향을 주는 딜레이 요소들을 면밀히 분석하고 이를 극복하기 위한 기법들을 제안함으로써 시간 동기화시 발생할 수 있는 오차를 최소한으로 줄인다.

본 논문은 서론에 이어 현재 사용되고 있는 대표적인 동기화 기법에 대하여 2 장 관련 연구에서 소개하겠으며 3 장에서는 시간 동기화에 오차를 일으키는 요소를 밝히고 이를 최소화 시킬 수 있는 방안을 제안한다. 그리고 마지막 장에서는 본 논문을 요약 및 정리한다.

2. 관련 연구

시간 동기화를 위한 연구는 그 동안 많이 이루어져왔다. 고전적인 Cristian알고리즘[7]으로 시작하여 최근에 IEEE1588 Standard인 Precision Time Protocol(PTP)까지 네트워크를 통하여 컴포넌트 간에 시간을 맞추는 여러 가지 방법이 제안되어왔다. 대부분의 시간 동기화 방법은 다음과 같다. 기준이 되는 컴포넌트의 시간을 정하고 나머지 컴포넌트들이 네트워크망을 통해 기준 시간을 자신의 시간으로 동기화시키는 방법을 따르고 있다.

기존의 연구에서 시간 동기화를 위한 여러 가지

방법을 제시하고는 있지만 이를 실제 환경에서 적용할 때는 이론적인 수치와는 달리 오차가 발생한다. 단일 컴포넌트 시스템이 아닌 분산 시스템에서 기준이 되어야 할 시간 자체에서 오차가 발생한다면 적절한 서비스를 하는데 문제가 발생하며 안정성이나 실시간성을 매우 중요하게 요구하는 시스템에서는 매우 치명적이다. [4],[6]

3. 일반적인 동기화 기법과 정확도를 떨어뜨리는 요소

제안한 동기화 기법의 기본적인 틀은 Christian algorithm[7]에서 제안하는 Master와 Slave구조를 따른다. Master와 Slave로 구성되어 있고 Slave는 일정한 주기마다 Master에게 시간을 요구하고 Master는 Slave측에서 시간을 받으면 자기의 시간을 Slave에게 보낸다. Slave는 Master에게서 받은 시간에 라운드 딜레이[2]의 절반을 더하여 자신의 시간으로 셋팅한다. 시간 동기화를 위해 Slave에서 주기적으로 실행되는 주도 코드는 다음과 같다.

```
Procedure SynchronizeTime()
var
  startTime, endTime, masterTime;
begin
  startTime = GetSlaveTime;
  masterTime = GetMasterTime;
  endTime = GetSlaveTime;
  SetClientTime(MasterTime + ( endTime - startTime) / 2);
end
```

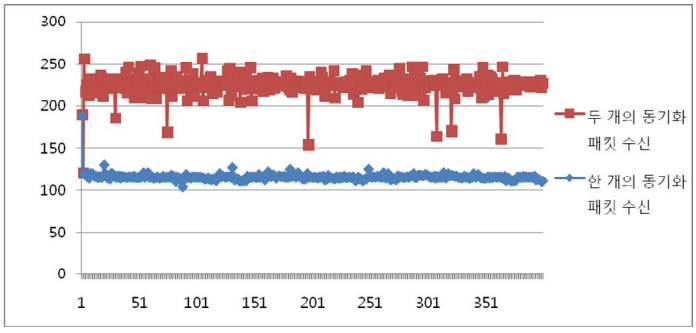
그리고 동기화 주기는 각 Slave 마이크로 컨트롤러에서 clock drift 발생의 원인이 되는 크리스탈의 ppm을 기기 별로 제공된 문서를 살펴 보면 구할 수 있다. 크리스탈의 1ppm은 초당 1us의 오차까지 발생할 수 있음을 의미한다. 예를 들어 10ppm이라 함은 초당 10us로 발생할 수 있음을 의미하며 100 초 후에 worst case로 1ms 정도의 오차가 발생함을 할 수 있다. 따라서 해당 슬레이브의 동기화된 시간 오차를 1ms이내로 유지하고자 한다면 동기화 주기를 100 초 이내로 하면 될 것이다.

제안된 동기화 방법이 제대로 돌아가기 위해선 위 과정 중에서 다른 어떠한 딜레이 요소가 없다는 가정이 있을 때 비로소 성립할 수 있다. 즉, 실제 환경에서 시간 동기화 과정이 이루어 질 때 네트워크 상에는 동기화 관련 패킷만 존재하게 되는 것이 아니라 여러 컴포넌트들이 동기화 이외의 데이터들이 송수신하고 있어 시간 동기화 관련 패킷 이외의 네트워크 트래픽이 존재한다. 또한 Master에서 병렬식으로 시간 동기화 패킷을 동시에 처리할 수 있는 것이 아니므로 Slave에서 동시에 Master측에 시간 동기화 패킷을 송신하여 시간을 요구할 경우에 Master측에서 충돌이 일어날 수 있다. 뿐만 아니라 Master와 Slave측에서도 동기화 관련 프로세스만 돌고 있는 것이 아니고 이 밖에 다른 프로세스들이 함께 돌아가고 있기 때문에 시간 동기화 프로세스가 각 노드의 커널 레벨의 스케쥴링에 의해서 블록될 여지가 있다.

이론적인 구상에서 더 나아가 실제 환경에서 시간 동기화 과정이 일어날 때 정확도를 떨어뜨리는 요소가 위에서 전체적으로 언급되었다. 다음 장에서는 위에서 언급된 요소를 종류별 구분하여 이를 극복할 수 있는 방안들을 제안한다.

4. 정확도를 떨어뜨리는 요소들을 최소화하기 위한 방안

Master에서 동시에 두 개 이상의 시간 동기화 패킷이 수신될 경우 발생할 수 있는 round delay는 다음과 같다.



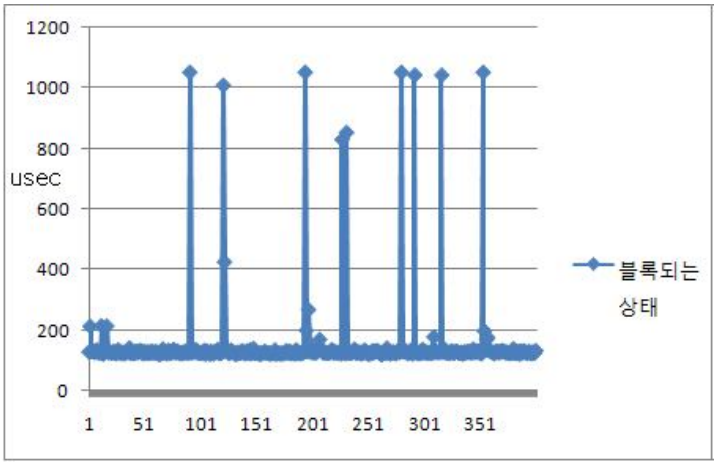
동시에 수신되는 패킷의 양이 늘어날 수록 slave에서 얻어지는 round delay는 시간 동기화 패킷 하나만 수신하는 상황에 비교하여 다소 불규칙하고 delay양이

늘어남을 볼 수 있다. 이와 같이 불규칙하고 늘어난 round delay가 의미하는 바는 Master측에서 언제 Slave의 시간 동기화 메세지를 받았는지 불분명함을 의마하고 송수신 round delay가 대칭임을 보장할 수 없는데 (asymmetric), 이러한 상황에서 round delay/2 를 Master의 시간에 더한 값을 Slave의 동기화 시간으로 설정하기에는 적합하지 않음을 뜻한다.[2],[4]

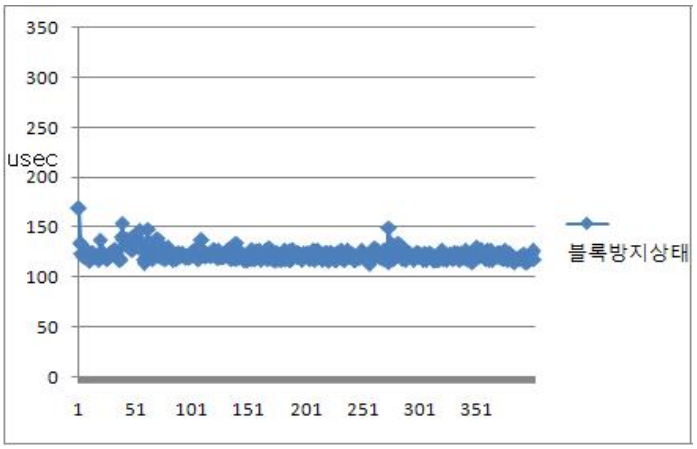
그러므로 각 Slave의 동기화 주기를 고려하여 충돌을 일으키지 않도록 각 Slave마다의 동기화 시작에 offset을 주는 방법을 제안한다. 예를 들어 2 초, 2 초, 3 초의 동기화 주기를 갖는 Slave 3 개가 시간 동기화 패킷을 Master에게 송신하고 있다고 가정할때 각 Slave가 동시에 시간 동기화 패킷을 보내는 것이 아니라 0.1, 0.2, 0.3 초의 offset을 가지고 시작을 한다면 Master에 송신되는 시간 동기화 패킷은 단위 시간당 하나씩이 되어 충돌이 일어나지 않는다.

다음으로는 각 노드 내의 커널레벨에서 시간 동기화에 관련된 프로세스가 다른 프로세스에 의해 블록되어 지연되는 시간에 대한 문제이다. 실시간성을 중요시하는 분산 컴포넌트 시스템에서 프로세스들은 동기화된 시간을 기준으로 진행된다. 이 기준이 되는 시간이 서로 어긋 낫을때 컴포넌트간의 고리는 서로간에 얽혀 제대로 시스템이 제대로 돌아가지 않는 상황이 발생하므로 시간 동기화에 관련된 프로세스가 가장 높은 우선 순위를 가지고 동작해야 한다.

노드에 시간 동기화 이외의 프로세스가 돌고 있을 때 커널 레벨의 스케쥴링에 의해 블록되어 발생하는 round delay는 다음과 같다.

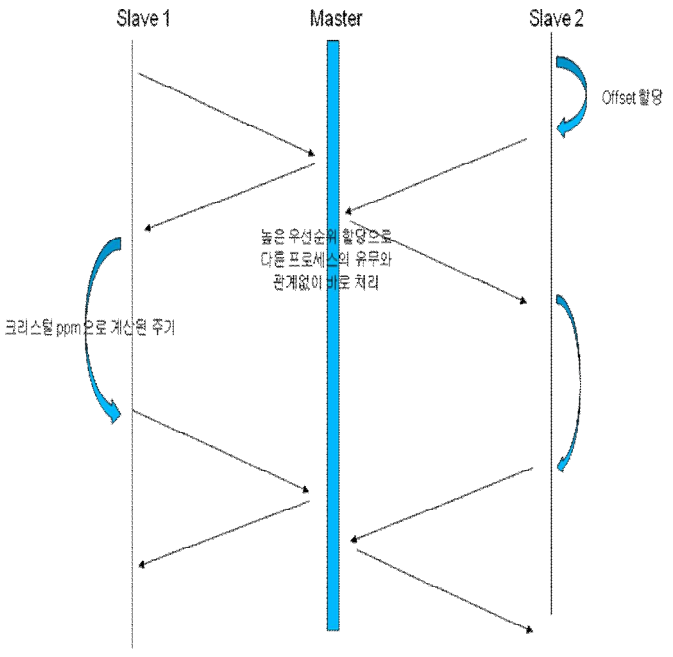


시간 동기화 관련 프로세스 이외에 다른 프로세스가 노드 내에 돌고 있을 때 시간 동기화 프로세스가 블록되는 현상을 지연된 round delay를 관찰함으로써 알 수 있다. 그러나 시간 동기화 관련 프로세스에 최고 우선순위를 할당함으로써 커널이 스케쥴링을 할 때 다른 프로세스에 블록당하지 않게 할 수 있다. 이러한 처리를 했을 경우에 결과는 다음과 같다.



그래프에서 알 수 있듯이 시간 동기화 관련 프로세스에 높은 우선 순위의 할당한 경우 round delay가 거의 일정하게 유지되고 있음을 관찰할 수 있다.

시간 동기화의 오차를 최소화 하기 위해 본 논문에서 제안한 방안을 적용한 시간 동기화의 전체적인 구조는 다음과 같다.



5. 결론

본 논문에서는 일반적인 Master와 Slave구조의 시간 동기화 방법에 기반하여 시간 동기화가 실제 환경에 적용되었을때 오차를 발생시키는 요소들을 지적하고 그것들을 극복할 수 있는 방안을 제시하였다.

우선 시간 동기화 관련 프로세스에 높은 우선 순위를 부여함으로써 각 노드 내의 다른 프로세스에 의해 블록되는 현상을 차단하였으며, Slave마다 시간 동기화를 시작함에 앞서 offset 할당하여 Master에서 발생할 수 있는 패킷의 충돌을 회피하여 정확하고 안정적인 시간 동기화가 이루어질 수 있는 방안을 제시하였다. 이와 같이 오차를 발생시키는 요소들을 제거함으로써 시간 동기화의 정밀도를 높여서 적절한 서비스를 제공하고 실시간성을 요하는 분산 컴포넌트 시스템에서도 또한 노드별로 동기화된 시간을 충분히 신뢰할 수 있도록 하고 안정성을 보장한다.

6. 참고 문헌

[1] J. Escobar, C. Patridge, “Flow Synchronization Protocol”, IEEE/ACM Transactions on Networking, 1993

[2] J. Bolot, “End-toEnd Packet Delay and Loss Behavior in the Internet”, SIGCOMM ’93, 1993

[3] K. H. Kim, Chansik Im and Prasad Athreya, “Realization of a Distributed OS Component for Internal Clock Synchronization in a LAN Environment”, Object-Oriented Real-Time Distributed Computing, 2002.

[4] John C. Eidson, Michael C. Fischer, Joe White, IEEE-1588 standard for a precision clock synchronization protocol for networked measurement and control systems, http://ieee1588.nist.gov

[5] http://standards.ieee.org

[6] PTPd Source Code Documentation, http://ptpd.sourceforge.net/doc.html

[7] Cristian’s Algorithm @ everything, http://everything2.com/index.pl?node_id=1435974