분산 시스템의 실시간 지원을 위한 전역시간 동기화 방법

백봉현*, 안병철

영남대학교 컴퓨터공학과
e-mail : wefbbh@hanmail.net

Global-Time Synchronization Method for Real-Time Support in Distributed System

Bong-Hyun Back*, Byoung-Chul Ahn

Dept of Computer Engineering, Yeung-Nam University

요약

멀티미디어 응용의 확산에 따라 MOD와 같은 서비스와 이를 뒷받침하기 위해 네트웍 환경이 급증하고 있다. 이러한 서비스는 네트웍을 이용하므로 실시간성과 최적의 스케줄링 알고리즘이 필요하다. 본 논문에서는 가변적인 네트웍 상황을 고려한 전역시간 동기화 방법을 제안한다. 제안한 알고리즘은 UDP환경 하에서의 설계 및 구현하였다. 성능 측정은 여러 가지 네트웍 환경에서 측정하였다. 제안한 알고리즘을 통해 측정된 데이터분석에 의해 서버 시간에 근접한 시간동기화를 얻을 수 있으므로 클라이언트의 QoS를 높일 수 있다.

1. 서론

현재 분산환경과 인터넷의 보급으로 네트워크를 이용한 데이터 송수신이 보편화 되었다. 특히 인터넷을 이용한 MOD(Multimedia On Demand), 공동 작업들이 급속하게 증가 하였다. 클라이언트와 서버 사이에 정확한 전송시간 서비스를 요구하게 되었다. 클라이언트는 자기가 가진 지역시간과는 관계없이 서버의 시간에만 의존하게 된다. 그리고 원거리에 있는 클라이언트들이 MOD나 VOD (Video On Demand)서버에 접속해서 서비스를 원할 때 서버에서는 각각의 클라이언트들이 서버에 도착한 시간을 가지고 스케줄링하게 된다. 즉 각각의 클라이언트들이 언제 서비스 요구를 했는지, 네트웍의 상황이 어떤지는 고려하지 아니하고, 단지 서버에 먼저 도착한 클라이언트들에게 먼저 서비스를 하는 형태를 가졌다. 이러한 서비스정책은 먼저 서비스를 요청함에도 불구하고 다른 클라이언트들에 비해 늦게 서비스를 받을 수가 있다. 그리고 Time Server에서는 전역시간을 요구해오는 모든 클라이언트들에게 전역시간을 계산해 주어야 하므로 서버는 부하를 가질 수 있다. 여기서는 단지 요구에 대한 응답을 줌으로써 클라이언트에서 전역시간을 계산한다.

2. 기존연구

기존의 전역시간 동기화 알고리즘으로는 Berkeley 알고리즘과 Cristian 알고리즘들이 있다.

2.1 Berkeley 알고리즘

Berkeley알고리즘은 Time Server의 데몬이 활성화된 후 각각의 클라이언트들에게 서버 시간을 알려준다. 클라이언트에서는 서버의 시간과 비교 후 다시 오차를 Time Server에게 전송한다. 비교되어진 시간들은 모두 합해지고, 이를 N개의 클라이언트와 서버의 합으로 나누어진다. 나누어진 시간을 다시 각각의 클라이언트들에게 전송하여 이를 전역시간화 한다.

이러한 Berkeley 알고리즘의 문제점은 근거리와 원거리의 네트웍 지연 시간을 고려하지 않은 일률적인 네트웍 환경에서 사용하는 알고리즘이다.

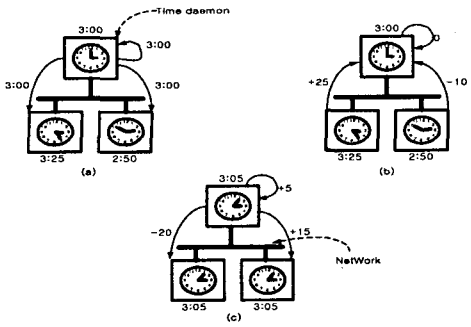

그림1. Berkeley 알고리즘

2.2 Cristian 알고리즘

Cristian 알고리즘은 클라이언트의 초기에 얻은 전역 시간이 T_0이고, 다음으로 타이머에게 요청하여 얻은 전역 시간이 T_1이라면 단 방향 전송 메시지 전송 기간은 식 (1)과 같이 표현된다.

$$T_p = (T_1 - T_0)/2 \qquad (1)$$

또한 서버의 인터럽트 조작 시간 I를 알고 있다면 정확한 전송 시간을 식 (2)와 같이 얻을 수 있다.

$$T_p = (T_1 - T_0 - I)/2 \qquad (2)$$

식 (1)과 (2)는 Cristian 알고리즘에서 단방향 전송 시간으로 추정을 한다. 식 (1) 또는 식 (2)의 단방향 전송시간과 요구 응답시 전송된 서버의 시간 조합이 식 (3)과 같이 전역 시간을 얻을 수 있다.

$$Gt = Tp + St \qquad (3)$$

식(3)에서 $G(t)$는 전역시간이며 St는 서버시간을 의미한다. Cristian 알고리즘에서 얻은 전역시간 Gt는 네트웍 상에서의 전송시간을 고려하였다. 그러나 Cristian알고리즘은 전송시간을 추가하여 계산하였으나 네트웍의 지연시간을 고려하지 않았다. 네트웍상에서의 전송은 상당히 가변적임으로 일률적인 시간 정책 또한 높은 시간오차를 가져다 줄 수가 있다. 전역시간 계산에 있어서 모든 시간을 Time Server에서 계산한다면 서버측에서는 여러개의 클라이언트에서 접속시 시간 계산에 관하여 많은 부하를 가질수 있다. 또한 서버의 작업에 관한 인터럽트 처리시간을 더욱 증가시킬 수가 있다.

본 논문에서는 Cristian 알고리즘을 수정함으로서 가변적인 네트웍 환경을 고려한 알고리즘을 제시한다.

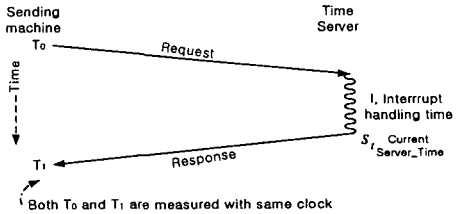

그림2. Cristian 알고리즘

3. 제안 알고리즘

Cristian 알고리즘의 변형인 제안 알고리즘이 그림 3에 주어져 있다. 제안 기법에서는 클라이언트에서 서버로 타임패킷을 한번이 아닌 여러번의 규칙적인 시간 간격으로 전송한다. T_1, T_2 . . . . .T_n까지의 양방향 전송시간 중에 특정 시간내에 수신된 시간을 실제 지연이 없는 양방향전송시간으로 본다. 양방향 전송시간을 이용하여 식(4)와 같은 단방향 시간을 계산한다. 이렇게 만들어진 단방향 전송 시간과 서버에서 전송되어진 시간의 합으로 전역시간을 형성한다.

네트워크에서 바운스 시간 정책에서는 B_1, B_2 . . . . B_n 까지의 n개의 양방향 전송 시간들을 바운스 테이블에 배열을 한다. 값 중 최대 값과 최저 값들은 생략한 평균값을 산출한다. 산출된 값을 양방향 전송시간으로 추정한다. 클라이언트에서 패킷을 보낸 후 서버의 응답을 수신할 때까지의 시간을 지연시간 이라 하면 수신된 지연시간의 차를 지연시간 차 B라고 정의한다.

$$G_t = S_t + \frac{1}{2m}\sum_{i=1}^{m} B_i \qquad (4)$$

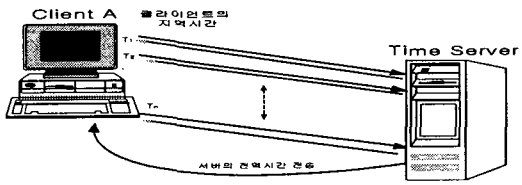

그림. 3 제안 알고리즘

4. 개발 환경

본 알고리즘 구현 환경은 클라이언트 측인Client_A와 Client_B, Client_C는 운영체제 Win98을 사용하였다. 구현 코드 및 어플리케이션을 컴파일하기 위해서

Java1.2를 사용하였다. 서버 측인 Time Server는 Win2000의 운영체제를 사용하였으며, Java Network 에서의 UDP환경에서 패킷전송을 하였다. 그림 4는 알고리즘 구현 및 테스트 환경이다. UDP는 IP를 사용하는 네트웍 내에서 컴퓨터들간에 메시지들이 교환될 때 제한된 서비스만을 제공하는 통신 프로토콜이다. 그러나 UDP는 TCP와는 달리, 메시지를 데이터그램으로 전송하므로 서버의 부하를 줄일 수 있다. 특히 도착하는 데이터 패킷들의 순서를 제공하지 않는 단점이 있다.

본 논문에서는 교환해야할 데이터가 매우 적으므로 서버의 부하가 적게 걸리는 UDP를 사용하였다.

서버의 타임 서비스는 타임 서버가 가진 시간을 클라이언트에게 전송해주는 서비스다. 이 서비스는 해당 소켓으로 들어오는 요청 패킷에 대해 ASCII 텍스트 형식의 현재 날짜 시간정보와 long 타입의 시간을 보내어준다. 그림 4와 같이 long 타입의 시간은 millisecond까지의 시간표현이 가능하다.

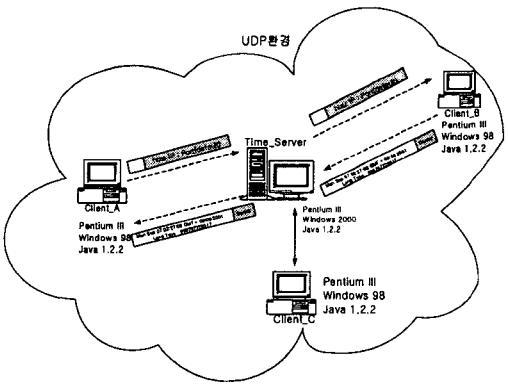

그림 4. 개발 환경

5. Time Server 설계

알고리즘을 적용한 서버의 기본 알고리즘은 그림 5와 같다.

DatagramSocket 타입의 socket을 생성하고 이 디폴트 포트에서 대기시킨 후 UDP 패킷을 받을 때 사용할 DatagramPacket 타입의 packet를 생성한다. receive()를 호출하여 패킷이 오기를 기다린다. 패킷이 수신되면, 이 패킷을 보낸 호스트의 정보를 출력한 후에 현재의 시간 정보가 담긴 바이트 배열인 outBuffer를 새로 만든다. 마지막으로 send()를 통해 원래의 호스트에게 시간을 전송해준다.

```
public static void main (String[] args) throws IOException {
 if (args.length > 1)
 throw new IllegalArgumentException
                ("Syntax: TimeServer[<port>]");
 DatagramSocket socket = new DatagramSocket
   (args.length == 0 ? DEFAULT_PORT : Integer.parseInt (args[0]));
DatagramPacket packet = new DatagramPacket (new byte[1], 1);
 while (true) {
   socket.receive (packet);
   System.out.println
   ("Received from: " + packet.getAddress () + ":" + packet.getPort ());
   byte[] outBuffer = new java.util.Date ().toString ()
     .getBytes ("latin1");
   packet.setData (outBuffer);
   packet.setLength (outBuffer.length);
   socket.send (packet);
 }
}
```

그림 5. 타임 서버 알고리즘

6. 실행 결과

그림 6은 테스트결과이다.

제안 알고리즘에서는 예측할 수 없는 네트웍 환경에서 가변적인 수신 지연 값을 줌으로써 수신 지연 오차 내에 들어온 시간들의 평균을 전송시간으로 한다면 서버에 근접한 시간을 산출 할 수가 있다.

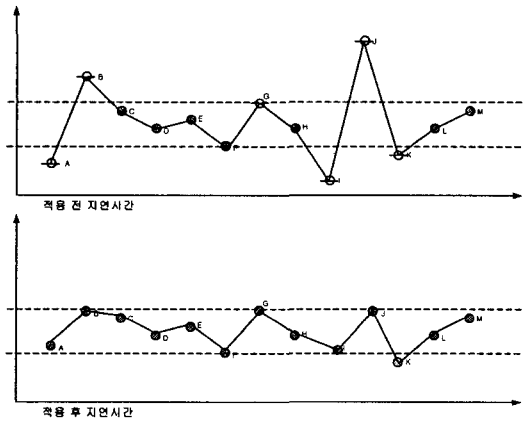

그림 6. 테스트 결과

7. 결 론

분산 시스템환경에서의 서버와 클라이언트와의 전역시간 동기화를 위해서 많은 알고리즘이 사용되었다.

이러한 알고리즘들은 가변적인 네트웍 환경을 고려하지 않았기에 전역시간 동기화에 있어서 정확하지 않은 시간을 가지게 되었다. 또한 타임서버에서 이루어지는 시간 계산은 서버의 부하를 야기할 수 있는 문제점들을 가졌다. 그러므로, 본 논문은 기존의 알고리즘이 가진 문제점을 보완하여 새로운 수정알고리즘을 제안하였다. 이 알고리즘은 서버와 클라이언트들의 실시간 서비스 수행에 앞서 전역시간 동기화를 이룰 수가 있었다.

본 연구의 동기화는 기존 알고리즘의 동기화 기법보다 오차가 적은 전역시간 구현을 형성하였으며, 지연 전 패킷의 감소가 약 80%까지 감소 시킬수 있었다.

앞으로 연구로는 멀티미디어 서비스를 위한 이기종 환경에서 전역시간 동기화 서비스를 위한 연구를 하고자 한다.

참고문헌

[1] Umesh Bellur, Merlin Hughes, Michael Shoffner, Derek Hamner.
"JAVA Network Programming", 인포북, 2000

[2]Andrew S. Tanenbaum. "Distributed Operating Systems", PRENTICE HALL

[3]Deitel & Deitel. "JAVA HOW TO PROGRAMM", PRENTICE HALL, 2000

[4]김명희, "CORBA 환경에서 실시간 응용 지원을 위한 분산 객체그룹 플랫폼의 설계 및 구현", 한국정보처리 학회, 2000.4월

[5]http://web-svr.borland.co.kr/visibroker/ssl/

[6]http://www.wisefree.com/~jhpark/jncreport.html