이동 에이전트를 이용한 일정관리 데이터 동기화 기법

이대희°, 김재곤, 김구수, 엄영익
성균관대학교 정보통신공학부
tricksters@hanmail.net°, {angel77, gusukim, yieom}@ece.skku.ac.kr

An Efficient Schedule Data Synchronization using Mobile Agents

Dae Hee Lee°, Jae-gon Kim, Gu Su Kim, Young Ik Eom
School of Information and Communication Engineering, Sungkyunkwan University

요 약

유비쿼터스 환경은 사용자가 컴퓨터나 네트워크를 의식하지 않으며 장소에 구애받지 않는 컴퓨팅 환경을 제시하고 있어 사용자는 스케줄러를 항상 소지하지 않고도 컴퓨터에 접속하여 일정을 관리할 수 있다. 그러나 그러한 환경에서는 일정관리 데이터가 불일치한 채로 분산되어 관리되는 경우가 빈번히 발생하며 그러한 문제를 해결하기 위한 데이터 동기화 기법이 필요하다. 기존의 데이터 동기화 기법들은 클라이언트-서버 구조를 가지고 있어 서버에 트래픽이 집중되는 단점이 있다. 본 논문에서는 이동 에이전트를 이용하여 트래픽을 특정 서버에 집중시키지 않고 일정관리 데이터를 동기화하는 기법을 제안한다. 본 제안 기법을 적용함으로써 일정관리 데이터뿐만 아니라 다양한 데이터의 동기화를 수행할 수 있을 것이다.

1. 서 론

일상생활 속에서 일정을 관리하기 위해서는 수첩이나 PDA(Personal Digital Assistance)를 항상 소지하고 다녀야 했으나, 앞으로 다가올 유비쿼터스 컴퓨팅 환경에서는 사용자가 컴퓨터나 네트워크를 의식하지 않는 상태에서 장소에 구애받지 않고 컴퓨터에 접속하여 일정을 관리할 수 있다[1].

그러나 그러한 환경에서, 일정관리 데이터가 분산되어 관리되는 경우가 빈번히 발생할 것이다. 예를 들어, 한 사람이 사무실에서 노트북으로 일정을 입력하고, 퇴근하는 길에 또 다른 일정이 생겨 소지하고 있던 PDA에 일정을 입력했다면, 같은 사람의 기기인 노트북과 PDA에 저장된 각각의 일정관리 데이터가 서로 불일치하게 된다. 이를 해결하기 위해서는 기기들 간의 일정관리 데이터 동기화 기법이 필요하다.

기존의 데이터 동기화 기법들은 클라이언트-서버 구조를 가지고 있어 서버에 트래픽이 집중되는 단점이 있다. 본 논문에서는 이동 에이전트를 이용하여 일정관리 데이터를 동기화하는 기법을 제안한다. 본 제안 동기화 기법을 적용함으로써 기기들 간에 트래픽을 분산시키면서 일정관리 데이터의 동기화를 수행할 수 있다[2].

2. 관련 연구

본 장에서는 이동 에이전트와 데이터 동기화 기법에 대해서 설명한다.

2.1 이동 에이전트

이동 에이전트란 네트워크상에서 스스로 이동하면서 사용자나 다른 개체의 작업을 대신하는 컴퓨터 프로그램을 말한다[3]. 이동 에이전트는 목적지 호스트로 코드를 이동시켜 목적지의 자원을 이용하여 작업을 수행하며, 사용자의 별도 지시 없이 스스로 판단하여 작업을 수행하는 것을 주요 특징으로 갖는다. 이동 에이전트는 실행 환경 변화에 동적이고 유연하게 대응하며, 네트워크 트래픽을 줄이는 장점을 갖는다[4].

2.2 데이터 동기화

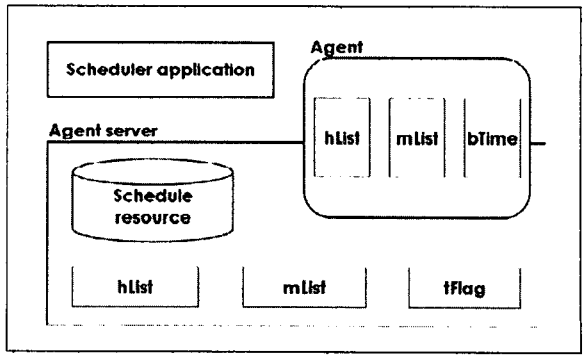
데이터 동기화 기법으로는 동기화 서버를 두고 클라이언트들이 동기화 서버에 접속하는 기법이 있다. 한 클라이언트가 자신의 데이터 변동사항을 서버에 기록하면 서버에 접속하는 다른 클라이언트들이 변동사항을 전달받아 데이터의 동기화가 이루어지는 기법으로 SyncML 등의 프로토콜이 있다[5, 6, 7]. 그러나 이러한 기법은 별도의 서버가 필요하고, 연산과 네트워크의 부하가 서버에 집중되는 단점이 있다.

3. 제안 기법

본 장에서는 제안 시스템의 아키텍처와 메커니즘에 대해서 설명한다.

3.1 시스템 아키텍처

제안 시스템의 아키텍처는 그림 1에서 보이는 바와 같이 스케줄러 애플리케이션과 에이전트 서버 그리고 에이전트로 구성된다.



(그림 1) 시스템 아키텍처

(1) 스케줄러 애플리케이션

스케줄러 애플리케이션은 사용자에게 일정관리 기능을 제공하는 프로그램이다. 사용자는 이 프로그램을 통해 일정관리와 동기화 명령을 내린다.

(2) 에이전트 서버

에이전트 서버는 에이전트를 생성하여 이주시키고 다른 호스트에서 이주해온 에이전트에게 실행환경을 제공하는 소프트웨어이다. 에이전트 서버는 schedule resource, hList, mList, tFlag 등의 자료구조를 갖는다. schedule resource는 일정관리 데이터에 대한 데이터베이스와 그에 대한 인터페이스를 제공한다. hList(host List)는 에이전트가 순회해야 할 호스트들의 주소가 저장된다. mList(method List)에는 에이전트가 순회하면서 수행할 작업으로서, 사용자가 일정을 관리하기 위해 실행한 Schedule resource의 메쏘드의 이름과 매개변수가 저장된다. tFlag(time Flag)는 순회중인 에이전트가 호스트를 예약하기 위해 자신의 생성시각을 기록하는 곳으로서, 기본 값은 -1이다.

(3) 에이전트

에이전트는 호스트들을 순회하면서 동기화를 수행하는 소프트웨어이다. 에이전트는 생성 시 에이전트 서버로부터 hList와 mList를 전달받고, 자신의 생성시각인 bTime(birth Time)을 갖는다.

3.2 동기화 메커니즘

제안기법은 호스트 예약과 데이터 동기화를 위한 두 번의 에이전트 순회로 이루어진다. 에이전트는 첫 번째 순회에서 자신의 트랜잭션의 보장을 위해 호스트들을 예약한 다음, 두 번째 순회에서 실제적인 동기화를 수행한 후, 호스트들의 예약을 해제한다. 본 절에서는 예약 알고리즘과 동기화 알고리즘을 설명한다.

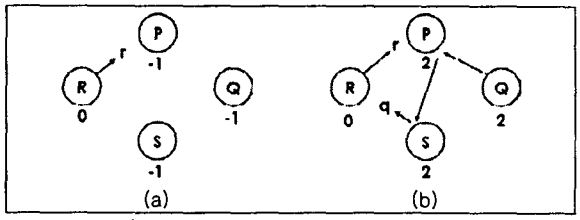
(1) 호스트 예약 알고리즘

첫 번째 순회에서 에이전트는 자신의 트랜잭션을 보장하기 위해 자신보다 늦게 생성된 다른 에이전트가 순회하지 못하도록 호스트를 예약한다. 방문하는 각 호스트에서 실행되는 에이전트의 호스트 예약과정을 알고리즘 1에서 보인다.

```
// AS : Agent server
if ((AS.tFlag != -1)
  and (AS.tFlag < bTime of agent))
{
    migrate to the home agent server;
    wait(T_random);
    restart the host reservation algorithm;
}
AS.tFlag ← bTime of agent;
migrate to the next host;
```

(알고리즘 1) 호스트 예약 알고리즘

알고리즘 1에서 에이전트는 에이전트 서버의 tFlag값과 자신의 bTime을 비교하고 자신보다 먼저 생성되어 순회중인 또 다른 에이전트가 있다면, 에이전트는 자신을 생성한 홈 에이전트 서버로 돌아가서 불특정 시간동안 대기 후 다시 첫 번째 순회를 시도한다. 그렇지 않다면 에이전트는 자신의 bTime을 에이전트 서버의 tFlag에 기록함으로써 예약을 수행하고 다음 목적지로 이주한다. 그림 2에서 호스트 예약 알고리즘의 동작과정을 보인다.



(그림 2) 예약 알고리즘의 동작과정

그림 2에서 알파벳 대문자, 알파벳 소문자, 숫자는 각각 에이전트 서버, 에이전트, 에이전트 서버의 tFlag값을 의미한다. 그림 2의 (a)는 에이전트 서버 R로부터 시간 0에 생성된 에이전트 r이 아직 첫 번째 목적지인 P에 도착하기 이전의 상황이다. R의 tFlag가 r의 bTime인 0으로 설정되어 있다. 그림 2의 (b)는 r이 아직 P에 도착하기 전에, Q에서 시간 2에 생성된 q가 이미 P와 S를 예약한 상황이다. P와 S의 tFlag에 q의 bTime인 2가 설정된 것을 볼 수 있다. 이때 q는 R을 예약하려하나 R의 현재 tFlag값인 0이 자신의 bTime값인 2보다 작으므로, q는 자신보다 먼저 생성되어 활동 중인 에이전트가 있음을 인지하고 Q로 돌아가 대기한 후 다시 예약 알고리즘을 수행한다. 이러한 방법으로 r은 트랜잭션을 보장받는다.

(2) 동기화 알고리즘

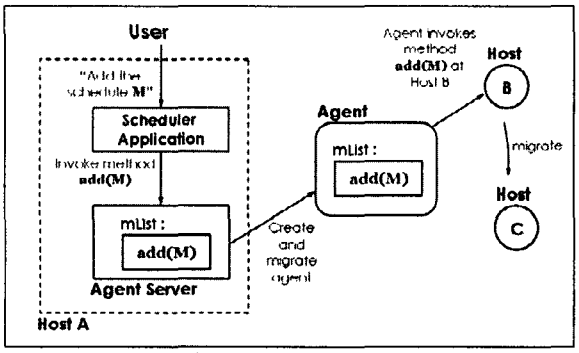
방문하는 각 호스트에서 실행되는 에이전트의 일정관리 데이터 동기화 과정을 알고리즘 2에서 보인다.

```
// AS : Agent server
// SR : Schedule resource in agent server
while (method M exists in mList)
{
    invoke method M with SR;
}
AS.tFlag ← -1;
migrate to the next host;
```

(알고리즘 2) 동기화 알고리즘

알고리즘 2에서 mList에는 사용자가 일정관리를 위해 실행한 메쏘드 목록이 저장되어 있다. 에이전트는 hList의 호스트들을 순회하면서 사용자를 대신하여 mList의 메쏘드들을 실행한다. 즉, 에이전트는 사용자가 실행한 것과 같은 매개변수와 같은 메쏘드를 목적지 호스트에서 실행함으로써 동기화를 수행한다. 동기화를 마친 에이전트는 에이전트 서버의 tFlag를 -1로 설정하여 호스트의

예약을 해제한 후 다음 호스트로 이주한다. 에이전트의 동기화 과정을 그림 3에서 보인다.

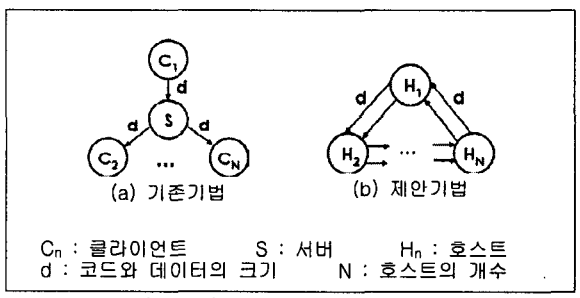


(그림 3) 동기화 과정

그림 3에서 사용자는 호스트 A에서, 일정관리 데이터 M을 입력하고 다른 호스트 B, C와 동기화하려 한다. 사용자가 스케줄러 애플리케이션을 통하여 일정관리 데이터 M을 입력하면, Schedule resource의 메쏘드 'add(M)'이 호출되고 메쏘드 이름과 매개변수가 에이전트 서버의 mList에 기록된다. 이때 사용자가 동기화 명령을 내리면 에이전트 서버는 mList를 갖는 에이전트를 생성하고, 생성된 에이전트는 알고리즘 1의 호스트 예약 과정을 거친다. 첫 번째 순회를 마친 에이전트는 동기화를 위해 두 번째 순회를 시작한다. 호스트 B로 이주한 에이전트는 mList에 기록된 메쏘드 'add(M)'를 호출하여 호스트 B를 호스트 A와 동기화 시킨다. 호스트 B에서 동기화를 마친 에이전트는 호스트 B의 예약을 해제하고 다음 목적지인 호스트 C로 이주하여 호스트 B에서와 같은 수행과정을 거친다.

4. 결 론

본 논문에서는 이동 에이전트를 이용한 일정관리 데이터의 동기화 기법을 제안하였다. 그림 4에서 본 논문에서 제안하는 동기화 기법의 트래픽 흐름도와 기존 기법의 트래픽 흐름도를 보인다.



(그림 4) 각 기법의 트래픽 흐름

그림 4의 (a)에서 클라이언트 C_1은 크기 d의 일정관리 데이터와 명령을 서버에 기록하고 다른 클라이언트들은 서버로 부터 C_1의 데이터를 전달받는다. 그림 4의 (b)에서 크기 d의 에이전트는 각 호스트들을 두 번 순회한다. 각 기법의 트래픽 비교를 표 1에서 보인다.

(표 1) 트래픽 비교표

기존기법	제안기법
$T_s = T_{in} + T_{out}$ $= d + d(N-1)$ $= dN$	$T_H = T_{in} + T_{out}$ $= 2d + 2d$ $= 4d$
T_{in} : *In-coming Traffic* T_{out}: *Out-going Traffic*	

표 1에서 보이는 바와 같이 기존 기법은 호스트의 개수가 늘어날수록 서버에 트래픽이 집중되는 반면, 제안 기법은 호스트의 개수에 상관없이 트래픽이 일정한 것을 알 수 있다.

본 제안 기법을 사용함으로써 내용이 불일치된 채로 분산된 일정관리 데이터를, 네트워크 트래픽을 특정 기기에 집중시키지 않고 동기화 할 수 있다. 또한 본 기법을 적용함으로써 유비쿼터스 컴퓨팅 환경에서 일정관리 데이터뿐만 아니라 다양한 데이터의 동기화를 수행할 수 있을 것이다.

참 고 문 헌

[1] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," Commun. ACM, vol. 36, no. 7, pp. 75-84, July 1993.

[2] P. Marques, P. Simoes, L. Silva, F. Boavida, and J. Silva, "Providing Applications with Mobile Agent Technology," Open Architectures and Network Programming Proceedings, 2001 IEEE , pp. 129-136, 27-28 April 2001

[3] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding Code Mobility," Software Engineering, IEEE Transactions on, Volume: 24, Issue: 5, pp. 342-361, May 1998.

[4] V. A. Pham and A. Karmouch, "Mobile Software Agents: An Overview," IEEE Communication Magazine, pp. 29-37, July 1998.

[5] Ligang Ren and Junde Song, "Data Synchronization in the Mobile Internet," Computer Supported Cooperative Work in Design, 2002. The 7th International Conference on , pp. 95-98, 25-27 Sept. 2002.

[6] S. Agarwal, D. Starobinski and A. Trachtenberg, "On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices," Network, IEEE , Volume: 16 , Issue: 4 , pp. 22-28, July-Aug. 2002.

[7] SyncML Group, "Building an Industry-Wide Mobile Data Synchronization Protocol, SyncML White Paper," http://www.syncml.org/.