바이노미얼 트리를 이용한 이동 에이전트 기반 소프트웨어 업데이트 시스템 설계[1]

송창환, 김연우, 최현우, 장현수, 엄영익
성균관대학교 정보통신공학부
e-mail : {eerien, daroobil, iskraskk, jhs4071, yieom}@ece.skku.ac.kr

Design of Mobile Agent based Software Update System by Using Binomial Trees

Song Changhwan, Youn-Woo Kim, Hyunwoo Choi, Hyun-Su Jang, Young Ik Eom
School of Information and Communication Engineering, Sungkyunkwan University

요 약

기업 네트워크에 대한 보안 위협이 지속적으로 발생함에 따라 네트워크 관리가 중요시 되고 있다. 기업 네트워크의 보안성을 강화하기 위해서는 네트워크상의 각 노드에 설치된 보안 소프트웨어나 취약성이 있는 소프트웨어의 업데이트가 요구된다. 그러나, 일반적인 서버/클라이언트 형태의 소프트웨어 업데이트 서비스는 서버의 부하가 크고 시간효율적이지 못하다. 따라서 본 논문에서는 네트워크의 논리적인 형태를 노드의 상태정보에 따라 바이노미얼 트리로 구성하고, 이를 기반으로 이동 에이전트가 각 노드를 순회하며 소프트웨어를 업데이트할 수 있는 시스템을 제안한다. 제안 시스템을 이용함으로써 서버에 집중되는 부하를 줄이면서 빠른 시간 내에 네트워크 상의 모든 노드의 소프트웨어를 업데이트 할 수 있게 된다.

1. 서론

기업 기밀을 목표로 하거나 악성코드 유포지/경유지로 이용하기 위한 기업 네트워크 보안 위협이 지속적으로 발생하여 안전한 기업 네트워크의 관리가 중요시 되고 있다[1]. 네트워크의 보안을 강화하기 위해 네트워크 내 각 노드에 보안 소프트웨어를 설치하고 취약성이 있는 소프트웨어를 수정하는 작업이 요구된다. 또한, 시시각각 변화하는 보안 위협사항의 유형과 방법에 따라 보안 소프트웨어를 업데이트 하거나 패치하는 등의 작업이 요구된다. 그러나, 기존의 서버/클라이언트 방식은 서버에서 모든 노드에 업데이트를 제공해야 하므로 서버의 부하가 크고 시간효율적이지 못하다.

본 논문에서는 바이노미얼 트리를 이용한 이동 에이전트 기반 소프트웨어 업데이트 시스템을 제안한다. 제안 시스템은 네트워크의 논리적인 형태를 각 노드의 현재 상태와 버전 정보를 바탕으로 바이노미얼 트리로 구성하고, 트리의 각 노드를 이동 에이전트가 이주하며 노드에 필요한 업데이트 및 패치 작업을 수행하도록 구성된다. 트리구조를 통해 병렬적으로 데이터의 전송 및 업데이트 또는 패치를 진행할 수 있으므로 서버의 부하를 줄이면서 빠르게 각 노드에 필요한 업데이트 또는 패치 등의 작업이 가능하다.

본 논문의 구성은 다음과 같다. 2 장에서는 이동 에이전트와 바이노미얼 트리에 대해 설명하고, 3 장에서는 본 논문에서 제안하는 시스템의 구성과 업데이트 프로세스를 설명한다. 4 장에서는 기존 시스템과 본 논문에서 제안한 시스템을 비교 평가해 보고, 5 장에서는 결론 및 향후 연구 과제를 설명한다.

2. 관련 연구

2.1 이동 에이전트

이동 에이전트는 네트워크를 통해 자율적으로 이동하며 작업을 수행하는 소프트웨어 프로그램을 말한다[2][3]. 이동 에이전트는 실행 코드를 목적지 호스트로 이동시켜 독립적으로 작업을 수행할 수 있어 분산 시스템을 위한 새로운 패러다임으로 각광 받고 있다[4]. 본 논문에서는 이동 에이전트 시스템의 이동성과 자율성을 반영하여 시스템을 설계하였다.

2.1 바이노미얼 트리

바이노미얼 트리는 재귀적으로 정의되는 순서 트리로서, One-to-All 브로드캐스팅에 대해, $O(\log_2 n)$의 시간 복잡도를 갖는 최적의 위상을 가진다[5][6]. 본 논문에서 제안하는 시스템은 바이노미얼 트리의 각 노드를 이동 에이전트의 이주 경로로 하여 병렬적으로 에이전트를 각 노드에 이주시킨다. 따라서 빠르게 전체 노드를 순회할 수 있다.

3. 소프트웨어 업데이트 시스템

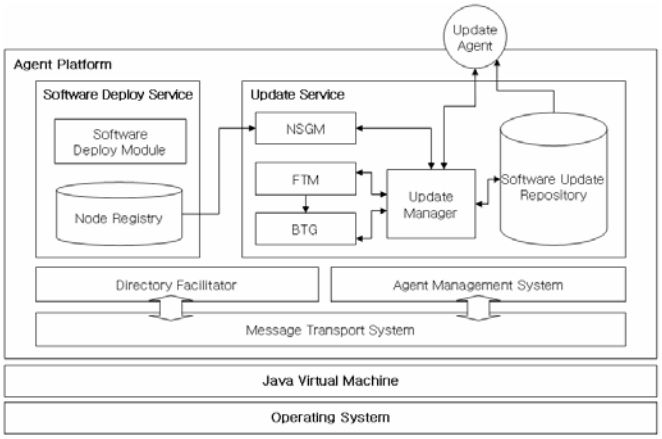
3.1 시스템 구성

본 논문에서 제안하는 소프트웨어 업데이트 시스템

[1] "본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음" (IITA-2007-(C1090-0701-0046))

은 Software Update Management Server(SUMS)와 각 노드의 Local Update Manager 로 구성된다. 제안 시스템은 이동 에이전트 시스템을 기반으로 하며, 에이전트들을 관리하기 위한 Agent Management System, 에이전트의 서비스 정보를 갖는 Directory Facilitator, 메시지 통신을 위한 Message Transport System 을 포함한다 [7][8]. SUMS 는 업데이트 데이터와 각 노드의 정보를 유지하며, 전체 업데이트 프로세스를 관리한다.

그림 1 은 SUMS 의 구성을 보인다. Software Deploy Service 는 각 노드에 소프트웨어를 배포하는 서비스이고, Update Service 는 각 노드의 소프트웨어 업데이트를 위한 핵심 기능을 제공한다.



(그림 1) Software Update Management Server

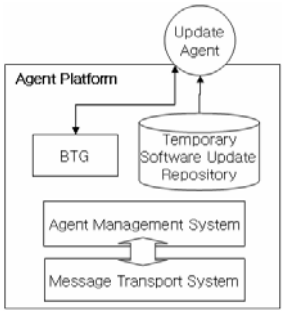
- **Update Manager**: 업데이트 프로세스 관리
- Software Deploy Module: 소프트웨어 배포 모듈
- **Node Registry**: 소프트웨어를 배포한 노드의 주소와 소프트웨어 정보를 보관

Node Status Gathering Module(NSGM): Node Registry 의 엔트리를 참조하여 각 노드의 현재 상태 정보를 수집

- **Software Update Repository**: 업데이트 데이터와, 업데이트 사이의 의존성 관계 저장
- Fault Tolerance Module(FTM): 노드에 결함 발생시 결함감내 기능 수행
- **Binomial Tree Generator(BTG)**: NSGM에 의해 수집된 정보를 기반으로 바이노미얼 트리를 구성
- **Update Agent**: 각 노드에 이주하여 업데이트 진행

Local Update Manager 는 업데이트 프로세스 전체를 총괄한다. NSGM 과 Software Update Repository 로부터 필요정보를 수집하고 BTG 를 통해 바이노미얼 트리를 생성한 후, 업데이트 프로세스를 수행한다.

그림 2 는 각 노드의 Local Update Manager 의 구성을 보인다.



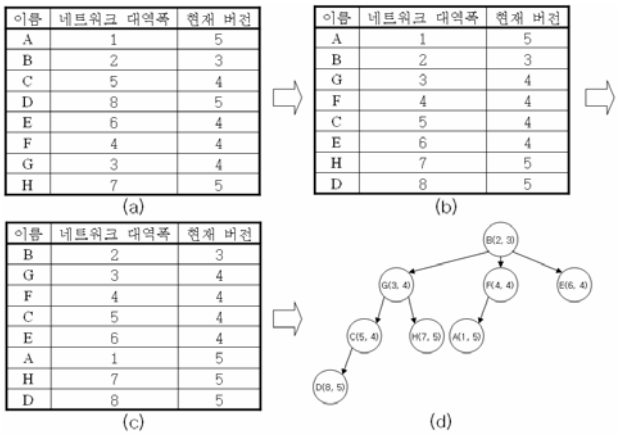
(그림 2) Local Update Manager

- **Temporary Software Update Repository**: 바이노미얼 트리 상의 부모 노드로부터 전송된 소프트웨어 업데이트 데이터를 임시로 저장

3.2 바이노미얼 트리 구성

바이노미얼 트리는 NSGM 에 의해 수집된 각 노드의 디스크 공간, 네트워크 대역폭, 현재 버전 정보를 기반으로 구성한다. 트리를 구성하기 위한 조건은 다음과 같다.

- 트리를 구성하는 각 노드는 업데이트 데이터를 저장하고 업데이트를 수행하기 위한 충분한 디스크 공간이 남아있을 것
- 부모 노드는 자식 노드들에 비해 되도록 네트워크 대역폭이 높을 것
- 부모 노드가 자식 노드 보다 업데이트가 요구되는 소프트웨어 버전이 낮거나 같을 것



(그림 3) 바이노미얼 트리 구성 예시

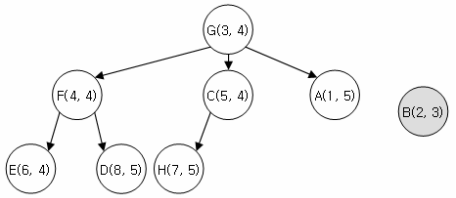
따라서, 우선 네트워크 대역폭 순위 기준으로 노드 리스트를 내림차순으로 정렬(b)한다. 그리고 소프트웨어의 현재 버전을 기준으로 내림차순으로 안정적 정렬(c)한 후, 이를 통해 바이노미얼 트리를 구성(d)한다.

그림 3 은 총 8 개 노드에 설치된 소프트웨어를 6 버전으로 업데이트하고자 할 때를 가정한 바이노미얼 트리 구성 예시를 보인다. 네트워크 대역폭은 순위로 표시하였다. 그림 3 과 같이 바이노미얼 트리를 구성하면, 부모 노드가 자식 노드에게 Update Agent 를 전달할 때, 자식 노드에게 필요한 업데이트 데이터만을 추출하여 전달할 수 있게 되어 효율적으로 업데이트

데이터를 배포할 수 있다.

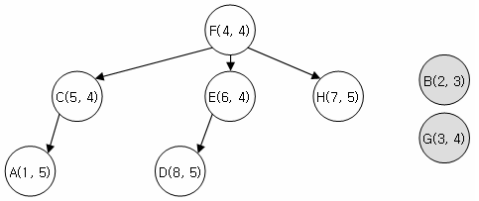
그러나 아직까지는 디스크 공간과 요구 대역폭을 충분히 갖추지 못한 노드가 트리의 상단으로 올라갈 가능성이 있다. 이를 해결하기 위해서 다음과 같은 프로세스를 통해 최종 트리를 구성한다.

- **디스크 공간 및 요구 대역폭 충족 여부 검사:** 그림 3의 바이노미얼 트리를 순회하며 디스크 공간과 대역폭 충족 여부를 검사한다. 검사 결과, B노드의 대역폭이 G, F, E 노드에 대한 요구 대역폭에 미치지 못하기 때문에 B노드를 제외하고 트리를 수정한다.



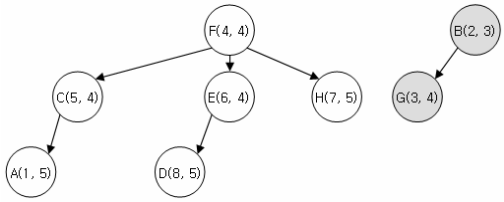
(그림 4) 요구 대역폭을 충족시키지 못하는 노드 B 를 제외

- **요구 조건 충족 여부 반복 검사:** 다시 트리를 순회하여 검사한 결과 G노드의 대역폭이 F, C, A 노드에 대한 요구 대역폭에 미치지 못하기 때문에 G노드를 제외하고 트리를 재구성한다.

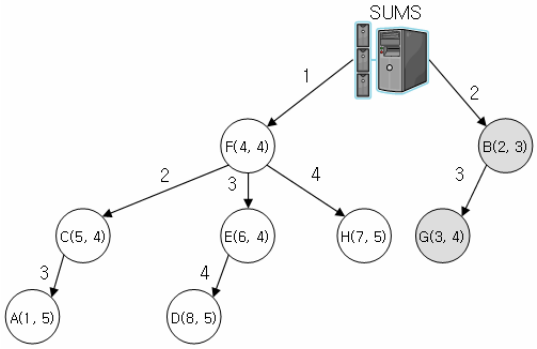


(그림 5) 요구 대역폭을 충족시키지 못하는 노드 G 를 제외

- **트리 재구성:** 모든 노드에 대한 요구 조건 충족이 확인 되면, 지금까지 제외된 노드로 새로운 트리를 생성하고 새로운 트리 역시 모든 요구 조건을 충족하는 트리인지를 검사한다.



(그림 6) 노드들의 트리 재구성



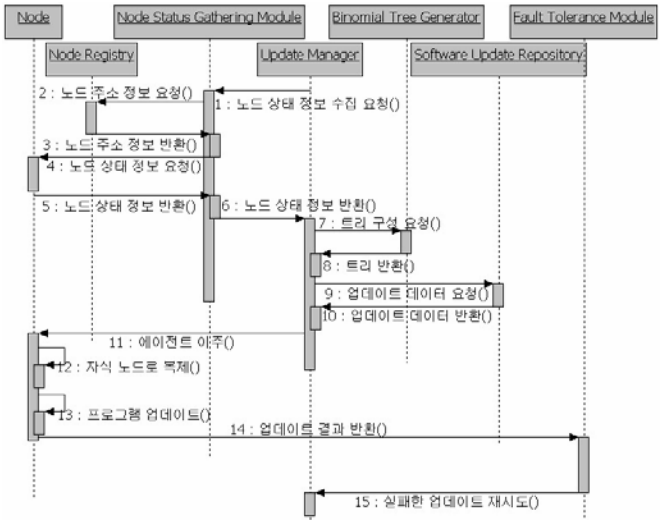
(그림 7) 요구 조건을 모두 충족하는 트리 구성

- **최종 업데이트 트리 구성:** 생성된 모든 트리가 요구 조건을 충족하는 트리임이 확인되면 서버를 루트로 하여 최종 트리를 구성한다.

이렇게 구성된 바이노미얼 트리는 연결선에 표시된 순서대로 업데이트가 진행된다. 먼저 SUMS 에서 F 노드로 에이전트가 이주하여 업데이트를 진행하고, 그 다음, SUMS 로부터 B 노드로, F 노드로부터 C 노드로의 에이전트의 이주와 복제, 그리고 업데이트가 병렬적으로 진행된다. 이를 통해 서버의 부하를 줄이면서 기존의 서버/클라이언트 방식에 비해 빠른 속도로 업데이트를 완료할 수 있다.

3.3 업데이트 프로세스

정책이나 관리자의 요청에 따라 시스템은 업데이트 프로세스를 시작하게 되는데, 그 과정은 아래와 같다.



(그림 8) 업데이트 프로세스 시퀀스 다이어그램

먼저, 업데이트 프로세스가 시작되면 Update Manager 가 NSGM 에게 노드 상태 정보 수집을 요청하고, NSGM 은 Node Registry 에서 업데이트 하고자 하는 소프트웨어가 설치된 노드의 주소 목록을 가져온다. 이 주소정보를 바탕으로 각 노드의 상태 정보를 수집한다. NSGM 이 수집하는 노드 상태 정보의 자료구조는 그림 8 과 같다.

Node Name	Disk Space	Network Bandwidth	Program Name	Current Version
A	15GB	200Mbps	Program A	5
...	...	...	...	...
H	20GB	1000Mbps	Program A	5

(그림 9) 노드 상태 정보 자료 구조

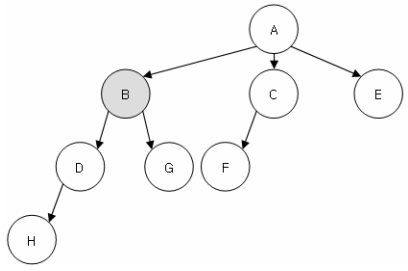
이렇게 수집한 정보를 Update Manager 에게 반환하면, Update Manager 는 수집된 정보를 BTG 로 보내 트리를 구성한다. Update Agent 는 Software Update Repository 에 저장된 업데이트 데이터와 BTG 에서 생성된 트리를 가지고 각 노드로 이주한다.

각 노드에 이주한 에이전트는 우선 자식 노드에 자신을 복제하는데, 자식 노드에게 필요한 업데이트 데이터를 추출하여 자식 노드가 업데이트 데이터를 전달해야 할 노드들의 트리와 함께 전달한다. 복제가 완료되면 소프트웨어를 업데이트 하고 완료된 결과를 SUMS 의 FTM 에 보고한다. 만약 자식 노드에 결함을 발견했을 경우 결함이 발생했음을 SUMS 에 보고한다. 한번의 업데이트 사이클이 완료되면 FTM 에 의해, 업데이트에 실패한 노드에 대해서 다시 업데이트를 시도한다.

3.3.1 결함 내구성

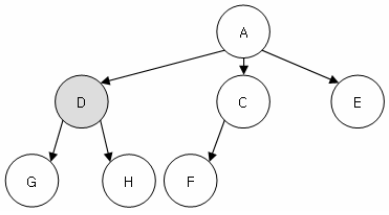
트리 위상에서 리프 노드에 결함이 발생하면 해당 노드에 결함이 있다는 것을 SUMS 의 FTM 에 알리는 것으로 해결된다. 그러나, 리프 노드가 아닌 노드에 결함이 발생한 경우에는 결함이 발생한 노드의 자식 노드에게 업데이트 데이터를 전달해 줄 수 없게 된다. 이 경우에는 결함이 발생한 노드의 자식 노드들로 트리를 재구성하여 업데이트 데이터를 전달해야 한다.

- **A노드가 B노드의 결함을 탐지했을 경우**



(그림 10) A 노드가 B 노드의 결함을 탐지

- **B노드의 상태를 서버에 알리고, D, G, H노드를 이용하여 트리를 재구성한 후, 업데이트 진행**



(그림 11) 결함에 대처하여 트리 재구성

4. 비교 평가

본 논문에서 제안한 시스템은 클라이언트/서버 형태의 업데이트 시스템과 두 가지 면에서 비교될 수 있는데, 첫 번째는 서버에 걸리는 부하이다. 클라이언트/서버 형태의 업데이트 시스템은 서버가 모든 클라이언트에게 업데이트 데이터를 전달해주어야 한다. 그러나 본 논문에서 제안한 시스템은 하나의 노드, 또는 트리가 분리되었을 경우 일부의 노드에만 업데이트 데이터를 전송해 주어도 전체 클라이언트를 업데이트 할 수 있다.

두 번째는 소요되는 시간이다. 본 논문에서 제안하는 시스템은 One-to-All 멀티캐스팅에 대해 최적의 전송 시간을 갖는 바이노미얼 트리를 바탕으로 하였기 때문에 빠른 속도로 업데이트를 진행할 수 있다.

5. 결론 및 연구 과제

기존의 소프트웨어 업데이트에 사용되는 클라이언트/서버 환경의 업데이트 시스템은 서버에 부하가 집중되는 단점이 있다.

본 논문에서는 바이노미얼 트리와 이동 에이전트를 이용해 업데이트 데이터와 코드를 이동시켜 업데이트 서비스를 제공하여 서버에 집중되는 부하를 줄이고 빠른 속도로 소프트웨어 업데이트를 수행할 수 있는 시스템을 제안했다. 향후 본 논문에서 제안한 시스템의 실험적인 효율성 입증을 수행할 것이다.

참고 문헌

1. 한국정보보호진흥원, 인터넷침해사고 동향 및 분석 월보, 2007. 5.

2. A. Aneiba and J. S. Rees, “Mobile Agent Technology and Mobility”, Proc. of the 5th Annual Post graduate Symposium on the Convergence of Telecommunications, Net-working and Broadcasting(PGNet2004), 2004.

3. V. A. Pham and A. Karmouch, “Mobile Software Agents: An Overview”, IEEE Communications Magazine, Vol. 36, Issue 7, pp. 26-37, Jul, 1998.

4. ObjectSpace, Inc., Dallas, TX., “ObjectSpace Voyager Core Technology”, 1999.

5. D. Kouvatsos, I. H. Mkwawa, and I. Awan, “One-to-All Broadcasting Scheme for Hypercubes with Background Traffic”, Proc. of the 36th Annual Simulation Symposium, 2003.

6. J. E. Jang, "An Optimal Fault-Tolerant Broadcasting Algorithm for a Cube-Connected Cycles Multiprocessor", IEEE, 1990.

7. FIPA Abstract Architecture Specification, SC00001L, http://www.fipa.org/repository/standardspecs.html. 2002.

8. FIPA Agent Management Specification, SC00023, http://www.fipa.org/repository/standardspecs.html, 2004.