이벤트 등록 스패닝트리의 역경로 정보를 이용한 이벤트 라우팅 알고리즘

류 기 열[†] · 이 정 태[††]

요 약

이벤트에 기반한 약결합의 분산 응용이 인터넷과 같은 광역네트워크 상에서의 주요 응용분야로 대두되고 있다. 이를 지원하기 위해 이벤트 통지서비스가 필요하다. 광역 네트워크상에서 이벤트 통지 서비스를 구현하는 방법으로 내용기반 이벤트 라우팅이 최근 활발히 연구되고 있다. 본 논문에서는 효과적인 내용기반 이벤트 라우팅 알고리즘을 구현하기 위해 기존에 발표된 대표적인 이벤트 통지서비스 시스템인 SIENA에 바탕을 두고, SIENA에서의 이벤트 라우팅 알고리즘의 문제점을 분석하고 이를 개선한 라우팅 알고리즘을 제안한다.

Event Routing Algorithms Using the Reverse Paths of Event Subscription Spanning Trees

Ki-Yeol Ryu[†] · Jung-Tae Lee[††]

ABSTRACT

A new class of applications based on event interactions are emerging for the wide-area network such as Internet, which is characterized as loose coupling, heterogeneity, and asynchrony. Content-based publish/subscribe systems are widely being studied to implement the event notification service for wide-area networks. In this paper, we analyze some problems of the content-based routing algorithm in SIENA, a recently developed as a representative event notification service architecture, and develop an enhanced routing algorithm.

키워드 : 이벤트(event), 이벤트 통지 서비스(event notification service), 내용기반 등록/발표 시스템(content-based subscribe/publish system), 멀티캐스팅(multicasting), 라우팅 알고리즘(routing algorithm)

1. 서 론

인터넷과 같은 광역 네트워크가 발전함에 따라 이러한 네트워크상에서 운용되는 새로운 클래스의 응용분야가 생겨난다. 이들 중 한가지로 최근 많은 연구가 진행되고 있는 분야가 인터넷의 비동기성(asynchrony), 이질성(heterogeneity), 약결합성(loose-coupling)의 특성을 잘 반영하는 이벤트-기반 분산 응용(event-based distributed applications)이다[3-5]. 이러한 응용시스템은 응용을 이루는 분산 컴포넌트(또는 응용)가 이벤트의 형태로 메시지를 주고받으면서 상호 작용을 한다. 이벤트에 의한 상호작용의 일반적인 형태는 이벤트의 등록자(event subscriber)는 자신이 받고자 하는 이벤트를 이벤트 브로커(event broker)에게 등록하고 이벤트 발표자(event publisher)는 이벤트를 이벤트 브로커에게 발표(publish)한다. 이벤트 브로커는 적절한 메카니즘을 통해 발표된 이벤트를 수신자에게 전달한다. 이러한 이벤트 서비스를 이벤트 통지 서비스(event notification service)라고 부른다.

기존에 이벤트 통지 서비스를 구현하는 방식은 주로 그룹기반 등록/발표(group-based subscribe/publish) 방식이었다. 그룹기반 방식은 미리 정해진 한정된 그룹(채널(channel), 주제(topic, subject)와 같은)을 정해두고 이벤트 발표자는 모든 이벤트를 특정 그룹을 선택해 발표한다. 이벤트 수신자는 하나의 그룹에 이벤트를 등록하고 이 그룹으로 발표되는 이벤트를 모두 수신한다. 그룹개념이 바로 이벤트 브로커에 해당한다. 이러한 그룹기반 방식은 이벤트 라우팅 면에서는 효율적이나 그룹이 미리 한정되어 있다는 점과 그룹 내에 속한 이벤트의 선별기능의 면에서 융통성이 부족하다는 단점이 있다. 이에 속하는 시스템으로 CORBA Event Service[9], JEDI[6]등이 있다.

그룹기반 방식의 대안으로 최근에 연구되고 있는 방식은

※ 본 연구는 2000년 한국학술진흥재단의 연구비에 의하여 연구되었음. (KRF-2000-EA0079)
† 종신회원 : 아주대학교 정보및컴퓨터공학부 교수
†† 정 회 원 : 아주대학교 정보및컴퓨터공학부 교수
논문접수 : 2001년 9월 3일, 심사완료 : 2001년 11월 23일

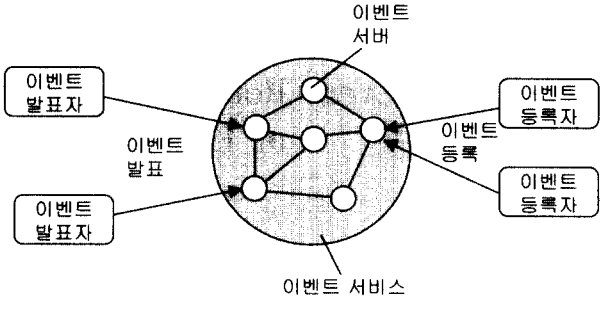
내용 기반 등록/발표(content-based subscribe/publish) 시스템이다. 이러한 시스템의 핵심 요소인 내용기반 이벤트 라우팅은 이벤트 수신자는 자신이 받고자 하는 이벤트의 등록정보를 이벤트 브로커(이벤트 서버)에게 전달하고 이벤트 브로커는 발표된 이벤트를 그 내용(content)을 주소로 사용하여 이벤트 수신자를 찾아 전달한다. 이 부류에 속하는 대표적인 시스템으로 Elvin[10], Keryx[8], Gryphon[1,2]과 SIENA[3-5]등이 있다. Elvin은 이벤트 브로커가 중앙집중형 서버에 존재하며, Keryx는 계층적인 네트워크로 이루어져 있다. Gryphon과 SIENA는 일반적인 형태의 peer-to-peer 네트워크에 기반을 두고 있다. 본 연구에서는 인터넷 구조에 적합한 peer-to-peer 아키텍처에 한정하여 논의하기로 한다. Gryphon과 SIENA에서의 이벤트 개념은 동일하나 이벤트를 등록하고 이벤트를 전달하는 라우팅 알고리즘에 있어서 차이를 나타내고 있다. SIENA는 이벤트를 등록할 때 이벤트 등록정보의 관계를 고려하여 등록 트래픽을 줄이고 이 관계 정보를 이벤트 서버에 유지함으로써 효율적으로 이벤트를 전송할 수 있는 방법을 제공한다는 점에서 Gryphon에 비해 우수하다. 자세한 비교는 4절을 참조하기 바란다.

그러나 SIENA의 라우팅 알고리즘은 두 가지 문제점을 가지고 있다. 그 중 하나는 이벤트가 다중경로를 통해 한번 이상 이벤트 등록자에게 전달될 수 있다는 점(다중경로 문제)이고 다른 하나는 등록 트래픽을 줄이기 위해 사용된 전략으로 인해 등록정보가 불완전하게 되는 경우(불완전 등록문제)이다. 본 논문에서는 SIENA의 이벤트 통지 서비스에 바탕을 두면서 SIENA의 두 가지 문제점을 개선하기 위한 방법을 제안한다. 본 논문에서 제안한 방법에서는 이벤트 등록을 위한 스패닝트리의 역경로 정보와 등록정보를 함께 이용하여 SIENA의 다중경로문제를 해결한다. 그리고 불완전등록문제를 유발하는 SIENA의 가장 포괄적인 등록(most general filter)원칙을 유보함으로써 불완전등록문제를 해결하고, 이벤트 필터의 병합 및 분리 연산을 제안하여 이벤트 전송 시에는 이 원칙을 유지할 수 있도록 하여 이벤트 전송 트래픽을 줄일 수 있도록 한다.

본 논문은 2장에서 SIENA의 이벤트 라우팅 알고리즘을 개략적으로 설명하고 문제점에 대하여 기술한다. 3장에서는 본 논문에서 제안하는 개선된 이벤트 라우팅 알고리즘이 소개되고 4장에서 관련 논문들과 비교한 후 5장에서 결론을 맺는다.

2. SIENA 이벤트 라우팅 알고리즘

SIENA에서 이벤트 통지 서비스의 일반적인 구조는 (그림 1)과 같다. 크게 두 가지 영역으로 나누어지는데 이벤트 서비스와 클라이언트이다. 클라이언트는 이벤트 발표자(event publisher)와 이벤트 등록자(event subscriber)로 나누어진다. 이벤트 등록자는 이벤트 서비스에 이벤트를 등록(subscription)하거나 해지(unsubscription) 또는 수신하고, 이벤트 발표자는 이벤트를 생성하여 이벤트 서비스에 이를 발표(publication)한다. 이벤트 서비스는 이벤트 서버들의 네트워크로 이루어지며 이벤트 등록이나 해지, 이벤트의 전달을 담당하는 이벤트 브로커이다. 이벤트 서비스에서 가장 중요한 부분은 이러한 이벤트 정보들에 대한 라우팅이다. 이 절에서는 SIENA의 내용기반 이벤트 라우팅 알고리즘에 대하여 개략적으로 설명하고 그 라우팅 알고리즘의 문제점을 지적한다. SIENA에 대한 보다 자세한 내용은 [3]을 참조하기 바란다.



(그림 1) 이벤트 통지 서비스의 일반적인 구조

2.1 SIENA 이벤트 및 이벤트 필터

SIENA에서의 이벤트(event)란 속성들의 집합으로 표현된다. 하나의 속성 α는 ($type_\alpha$, $name_\alpha$, $value_\alpha$)의 세 가지 요소로 이루어진다. 아래 속성들의 집합은 주식 값의 변화를 나타내는 이벤트의 예이다.

```
string exchange = KOSDAQ
string company = SEC
float previous = 170000
float change = 1000
```

이벤트를 등록하기 위한 데이터 구조를 이벤트 필터(event filter)라고 하는데 하나의 이벤트 필터는 속성조건의 집합으로 이루어져 있다. 각 속성조건 ϕ은 ($type_\phi$, $name_\phi$, $operator_\phi$, $value_\phi$)의 네 가지 요소로 이루어져 있다. 다음은 주식 값의 변화와 관련된 필터의 예이다.

```
string exchange = KOSDAQ
string company = SEC
float change > 100
```

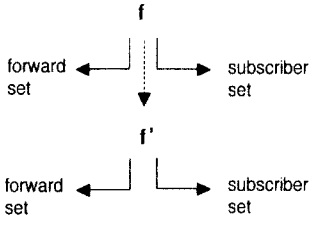
속성 α가 속성조건 ϕ를 만족시키기 위한 조건은 타입과 이름이 각각 일치하고 $operator_\phi(value_\alpha, value_\phi)$가 참이 될 때이다. 필터 f의 모든 속성조건에 대하여 만족되는 속성이 이벤트 n에 존재하면 그 때 n은 f에 "매치된다" 또는 f는 n을 "포함한다(cover)"고 말하고, 기호로는 $n < f$로

나타낸다. 위에서 예로든 주식의 필터는 주식 변화 이벤트를 포함하고 있다.

두개의 필터 f와 f'에도 포함관계(covering relation, <[1])가 존재한다. 다음 조건을 만족하면 f'가 f를 포함한다고 말한다.

$$f < f' \Leftrightarrow \forall n : n < f \Rightarrow n < f'$$

SIENA에서 이벤트를 등록하고 전달하기 위해 두 가지의 자료구조를 이용한다. 하나는 이벤트 등록자로부터 잠재적인 이벤트 발표자들에게 이벤트 등록을 전송하기 위한 스패닝트리(spanning tree, 이하 ST라 줄여 부른다)이다. 원등록자(original subscriber)[2]는 하나씩의 고유한 ST를 가지고 있다. 다른 하나는 필터의 등록과 이벤트의 전송에 모두 사용되는 자료구조로 각 서버가 유지하는 poset 구조이다. (그림 2)는 이벤트필터들의 poset 구조의 모습이다.



(그림 2) poset 구조($f' < f$)

각 서버는 주변 서버들이나 클라이언트들로(통틀어서 이웃이라고 한다)부터 자신에게 도달한 이벤트 필터를 poset 구조에 저장한다. 각 필터의 아래 오른쪽화살표 다음의 등록자집합(subscriber set)은 이 이벤트 필터를 전달한 직접 등록자(immediate subscriber)들의 집합을 의미한다. 이는 뒤에 이벤트가 전송되어 오면 원등록자에게 전달하기 위한 경로로 사용된다. 그리고 왼쪽 화살표의 앞에 있는 forward 집합은 이 이벤트 등록이 계속해서 전파될(또는 전파된) 이웃을 나타낸다.

여기서 이벤트 등록을 전달하는 방식은 가장 포괄적인 필터(most general filter) 원칙을 사용하는데 이는 필터의 포함관계를 이용하여 보다 포괄적인 등록 f가 전달되면 f에 포함되는 f'는 같은 이웃으로 전달되지 않아도 된다는 원리이다. 이런 방식은 등록정보 전달을 크게 줄여준다. 이러한 정보를 저장하는 poset 구조는 등록을 전달할 때 이러한 불필요한 정보의 전달을 차단하는 역할도 하지만 또한 발생하는 이벤트를 역순으로 원등록자에게 전달하기 위해서도 사용한다. 이벤트의 전달 역시 공통적인 라우팅을 하나로 합쳐서 전달한 후 가능하면 서로 다른 라우팅을 위한 복제(replication)를 뒤로 연기시킨다는 철학에 바탕을 두고 있다.

2.2 이벤트 등록

서버 X에 이웃 U로부터 이벤트 등록 f가 전달되었다고 가정하자(subscribe(U, f)). 서버 X에서는 두 가지의 작업을 해야한다. 하나는 전달된 이벤트 등록을 반영하기 위해 자신의 poset 구조를 수정해야 하고, 다른 하나는 이 이벤트 등록을 다시 이웃에게 전파해야 한다. 사실 이 두 가지는 상호 연관되어 있다. 우선 첫 번째 작업을 살펴보면, 크게 세 가지 경우로 나뉜다.

첫째, 이미 서버 U로부터 f를 포함하는 필터 f'가 등록되어 있으면 아무런 작업이 필요 없다.

둘째, f가 이미 존재하고 등록자집합에 U가 포함되어 있지 않다면 등록자집합에 U를 포함시킨다.

셋째, 위 두 가지의 경우가 아니라면 이 이벤트 등록은 포함관계에 따라 poset의 한 곳에 삽입된다. 루트 노드일 수도 있다.

위 세 가지경우의 작업이 끝나면 엄밀하게는 두 번째와 세 번째의 경우, poset구조에서 삽입된 f에 의해 포함되는 즉, f의 서브트리에서의 각 노드의 등록자정보에서 U를 제거한다. 왜냐하면 이미 U로부터 더 포괄적인 이벤트가 등록되었기 때문이다. 다음 작업은 이 이벤트 등록을 이웃에게 전파해야 한다. 그리고 이 이웃정보를 이벤트 등록 노드의 forward 집합에 기록해야 한다.

이벤트 등록 f가 전파되어야 할 이웃의 집합을 *forwards*(f)라고 하자.

$$forwards(f) = neighbors - NST(f) - \bigcup forwards(f') \quad (f' \in Ps \wedge f < f') \qquad (1)$$

$NST(f)$는 f의 원등록자를 루트로 하는 ST에서 노드 X의 자식노드가 아닌 인접한 노드들의 집합을 의미한다. 등록자로부터 잠재적인 발표자들로의 경로는 하나 이상 존재할 수 있기 때문에 하나의 경로로만 이벤트 등록이 전달되도록 하기 위해서는 각 노드(원등록자로 사용될)에 대하여 ST를 미리 계산해 놓아야 한다. 그리고 위 식의 마지막 항은 f를 포함하는 f'가 이미 전파되었다면 전파된 이웃은 f의 forward 집합에서 제외되어야 한다는 것을 의미한다. 이는 앞에서 언급한 SIENA 라우팅의 철학인 이벤트 등록정보의 전송량을 줄이기 위한 방법이다. 식 (1)에서 계산된 *forwards*(f)는 poset에서의 필터 f의 forward 집합에 추가된다.

마지막으로 f가 전파되면 poset에서 f에 포함되는 f'의 forward 집합에서 f의 forward 집합의 노드를 제거해야 한다. 그 이유는 더 포괄적인 필터가 이미 전달되었기 때문이다.

---

1) 의미의 혼동이 없기 때문에 이벤트와 필터사이의 포함관계와 같은 기호를 사용한다.

2) 하나의 이벤트 필터를 처음으로 등록하는 등록자를 원등록자라 부르고, 이 이벤트 필터를 중계하는 인접한 노드를 직접 등록자(immediate subscriber)라고 부른다.

2.3 이벤트 해지

서버 X가 이벤트 해지 메시지(unsubscribe(U, f))를 이웃 U로부터 받았다고 하자. X는 f에 의해 포함되는 모든 이벤트 필터의 등록자집합에서 U를 제거해야 한다. U가 제거된 후 poset은 크게 다음 두 가지의 경우로 변경된다.

하나는 U가 제거된 필터의 등록자집합이 공집합이 되는 경우이다. 이 경우 그 필터는 poset에서 제거되어야 한다. 다른 하나는 U가 제거되어도 하나 이상의 등록자가 남아 있는 경우이다. 두 경우 모두 U의 제거는 그 노드의 forward 집합에 변화를 일으키게된다. 즉 그 등록자의 제거는 그 등록자로 인해 전파된 이웃(*forwards*(f)) 중에서 다른 등록자(있다면)의 *forwards*(f)에 속하지 않는 이웃은 forward 집합에서 제거시켜야 한다. 이렇게 제거된 노드에게는 이벤트 해지메시지를 보내야 한다.

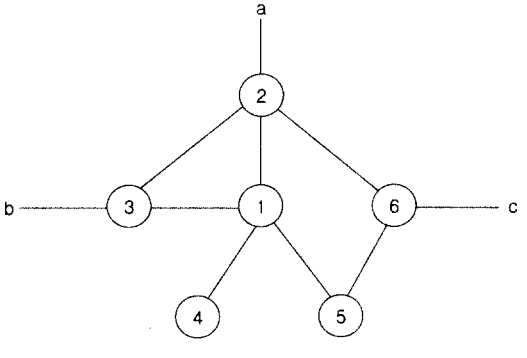
그리고 어떤 이벤트 등록의 forward 집합이 줄면 그 하위 노드의 forward 집합에 영향을 주게 된다. 이는 식 (1)에서의 마지막 항에 따른 것이다.

2.4 이벤트 전달

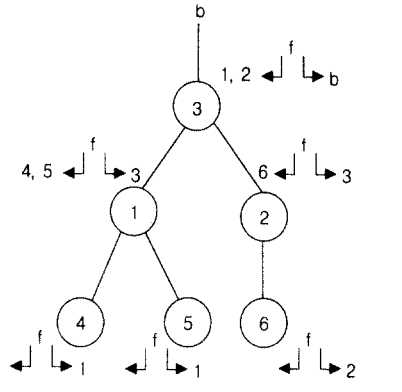
이벤트의 등록이 끝나서 각 서버마다 자신의 poset이 설정된 후 이벤트가 발생하면 이 poset정보들을 이용하여 적절한 등록자로 전달(notification)될 수 있다. 이벤트와 매치되는 필터의 집합을 찾은 후 그 필터의 등록자들에게 그 이벤트를 전달한다. 필터의 집합을 찾는 방법은 poset의 루트에서부터 출발하여 이벤트가 그 필터에 매치되는 지를 검사하여 매치되면 그 필터를 찾는 필터 집합에 넣은 후, 그 필터의 자식노드들을 마찬가지 방법으로 조사한다. 각 자식의 필터가 이벤트를 포함하면 그 필터를 다시 필터 집합에 넣고 그 필터의 자식들을 조사한다. 이 방식은 poset 그래프를 너비우선으로 탐색한다.

2.5 SIENA 라우팅 알고리즘의 문제점

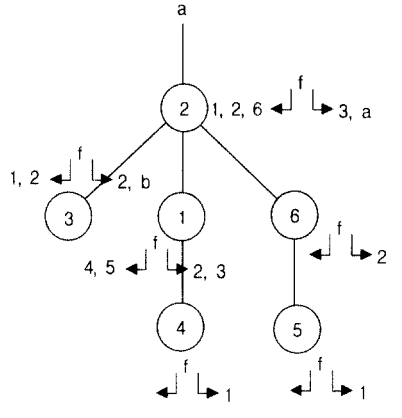
SIENA의 라우팅 알고리즘의 문제로는 크게 두 가지가 있다. 하나는 이벤트가 다중 경로를 통해 원등록자로 전달될 수 있는 다중경로문제(multi-path anomaly)이고, 다른 하나는 이벤트 등록이 ST를 통해 전달될 때 모든 노드에 전달되지 않아 발생하는 불완전 등록문제(incomplete subscription anomaly)이다. 이러한 문제들을 설명하기 위한 예제로 (그림 3)의 네트워크 구조를 사용한다. (그림 4)는 예제 네트워크에서의 각 클라이언트를 위한 ST와 *b*, *a*, *c*가 차례대로 f를 각각 등록한 후의 모습을 보여주고 있다.



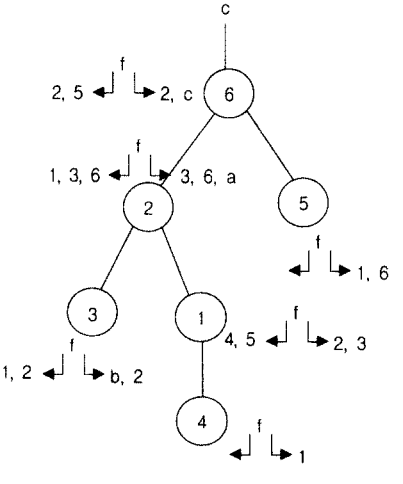
(그림 3) 예제 네트워크 구조



(a) *b*가 f를 등록



(b) *a*가 f를 등록



(c) *c*가 f를 등록

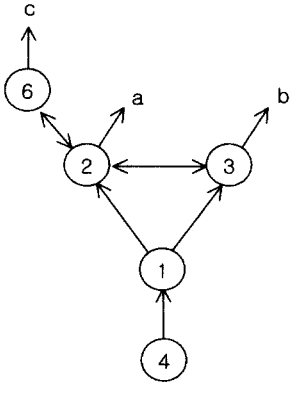
(그림 4) 이벤트 등록

2.5.1 다중 경로문제

다중 경로문제는 어떤 노드에서 발생한 이벤트가 여러 경로를 통해 하나의 등록자로 전달될 수 있다는 것이다. 예를 들어, 4에서 발생한 이벤트 $e(e < f)$는 아래의 여러 가지 경로를 타고 a에 전달될 수 있다.

$$4 \text{ -> } 1 \text{ -> } 2 \text{ -> } a$$
$$4 \text{ -> } 1 \text{ -> } 3 \text{ -> } 2 \text{ -> } a$$

다중 경로의 한가지 경우는 경로가 사이클을 이룰 수 있다는 것이다. 가령, 노드 4에서 이벤트가 발생하면 4 -> 1 -> 2 -> 3 -> 2- > 3 -> … 와 같은 경로가 가능하다. (그림 4)의 poset 구조를 이용하여 4에서 발생한 이벤트가 a, b, c에 전달될 수 있는 경로를 유향 그래프로 나타내보면 (그림 5)와 같다.



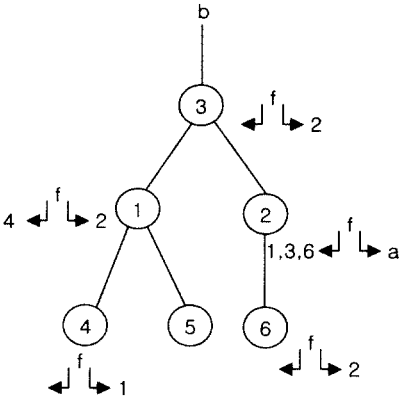
(그림 5) 노드 4를 위한 이벤트 전송경로 그래프

이 문제는 여러 등록자의 ST의 경로가 한 노드에서 서로 교차할 수 있고, 또 이 노드에서의 등록자집합은 전달된 이벤트가 어디에서 발생해서 어디로 가는지는 알 수 없고 단지 인접한 등록노드들만 알고 있기 때문에 발생한다. 가령, 노드 3에 전달된 이벤트가 4에서 b로 가는 이벤트인지 3으로부터 a를 향해 가는 이벤트인지를 구분할 수 없다는 것이다.

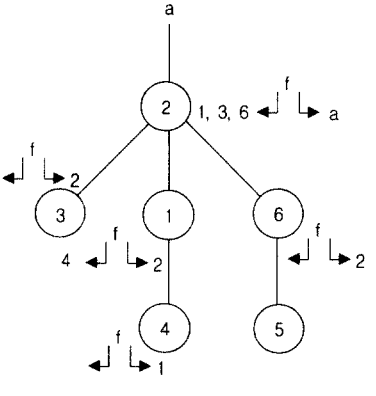
2.5.2 불완전 등록 문제

불완전 등록문제는 어떤 이벤트의 등록이 원등록자 ST의 모든 노드에 전달되지 않기 때문에 발생하는 문제이다. SIENA에서 등록정보를 전송하는 방식을 보면 식 (1)의 세 번째 항에 따라 전송하고자 하는 필터보다 더 포괄적인 필터가 이미 전송된 이웃에게는 그 필터를 전송하지 않는다. 따라서 두 개의 ST가 일부분에서 경로가 겹치면 하나의 ST에서 이벤트 f를 그 경로를 따라 전송하면 다른 ST에서는 그 경로를 따라서는 f를 전송하지 않게 된다. 클라이언트 a와 b의 ST를 고려해보자. 두 ST에서는 2-6과 1-4부분이 모두 겹치는 경로들이다. b가 f를 등록하여 노드 2에서 6으로 f가 전달되면, a가 f를 등록할 때 f가 2에 도착하면 b에 의해 이미 6에 전달되었기 때문에 6에는 전달되지 않는다. 또한 1과 4사이에서도 같은 현상이 일어난다. 그런데 전자의 경우에는 a가 등록한 f가 2를 통해 6에 전달되지 않기 때문에 6의 자식인 5에도 전달되지 않는다. 즉, 5에는 b의 ST만을 타고 f가 전달되어 있다.

이로 인해 발생하는 문제는 치명적인 문제로서 어떤 등록자가 등록을 해지 할 때, 다른 등록자의 등록에 대한 이벤트의 전송 경로가 없어질 수 있다는 것이다. 예를 들어, 앞에서와 같이 b와 a에서 순서대로 각각 f를 등록한 후 b에서 f를 해지한다고 가정하자. 해지한 후의 모습은 (그림 6)과 같다. (그림 6)에서 노드 5는 이벤트 f를 a에 전송할 전달 경로를 잃어버리게 된다. 그 이유는 a에서의 f의 등록은 앞선 b의 등록으로 인해 5까지 전달되지 못하고 2에서 차단되었으며, 이 정보가 b에 해지되었기 때문이다. 그러나 a가 먼저 f를 등록하면 이러한 현상이 발생하지 않는다. 즉, 불완전 등록문제는 이벤트 등록의 순서에 따라 달라질 수 있다.



a) b의 ST



b) a의 ST

(그림 6) b에서 f를 해지한 후의 모습

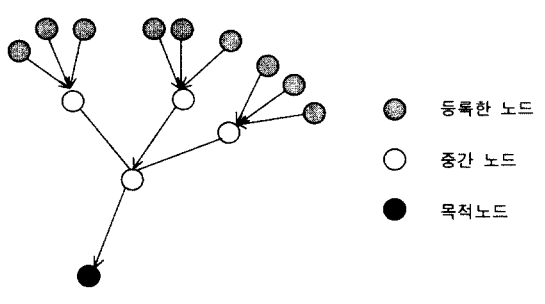
3. 등록 스패닝트리의 역경로를 이용한 이벤트 라우팅 알고리즘

3.1 이벤트 등록경로 및 이벤트 전달경로

다중 경로 문제를 해결하기 위해서는 이벤트가 발생하는

노드에서 그 이벤트를 등록한 각 등록자에 대한 유일한 경로를 생성해야 한다는 것이다. 이를 위해 본 논문에서는 SIENA의 이벤트 등록 알고리즘을 정확하게 원등록자의 ST를 따라 모든 노드에 전달하도록 수정한다. 그리고 이벤트의 전달은 정확하게 이 ST의 역경로를 타고 전달되도록 한다. 모든 등록자 ST의 각 등록 경로를 합치면 (그림 7)과 같은 역트리(reverted tree)가 된다(정확하게 말하면, 등록 경로의 합이 트리가 될 수 있는 등록자 ST들을 만들 수 있다(3.4절 및 부록 참조). 이벤트의 전달에 대한 자세한 내용은 3.4절에서 다룬다.

본 논문에서는 이벤트가 역트리의 경로만을 따라서 전달될 수 있도록 이벤트 라우팅 방법을 제안한다. SIENA의 이벤트 등록방법이 서로 다른 등록자로부터 발생한 등록들 사이의 포함관계에 따라 중복 등록을 피하기 위한 방법을 사용한데 비해, 본 논문에서 사용하는 방법에서는 이벤트 등록을 원등록자의 ST를 따라 모든 노드에 전달한다. 단, 같은 서버에 인접한 원등록자들로부터 발생한 등록에 대해서는 SIENA의 등록 전략을 그대로 사용한다. 그 이유는 이러한 클라이언트들은 같은 ST를 공유[3]하기 때문에 불완전 등록문제가 발생하지 않기 때문이다.



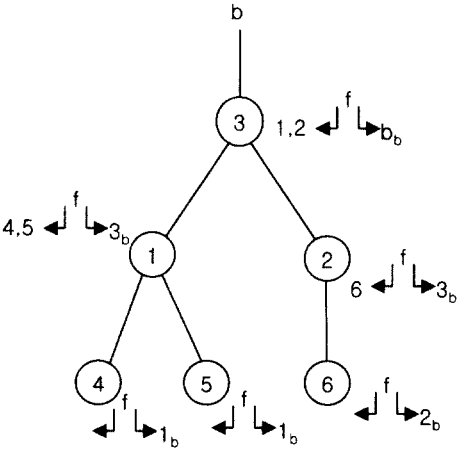
(그림 7) 이벤트의 등록 경로와 역트리

3.2 이벤트 병합 연산을 이용한 이벤트 등록

본 논문에서의 서버 X가 인접한 서버 U로부터 subscribe (U, f, a)를 받은 경우의 등록방법은 아래 알고리즘과 같다. 본 논문에서는 등록자 개념 외에 원등록자를 나타내는 기호를 등록자에 아래첨자로 붙인 확장등록자(tagged subscriber)를 사용한다. 예를 들어, 한 등록의 확장등록자가 2_a라면 2는 그 등록의 등록자를 의미하고 a는 원등록자를 의미한다. 확장등록자의 집합에서 같은 등록자를 사용하는 여러 개의 확장등록자가 있다면 첨자리스트를 가진 확장등록자로 약식으로 표현한다. 즉, 2_a, 2_b는 $2_{a,b}$로도 표현한다. 알고리즘에서 Ps는 노드 X의 poset을 나타낸다.

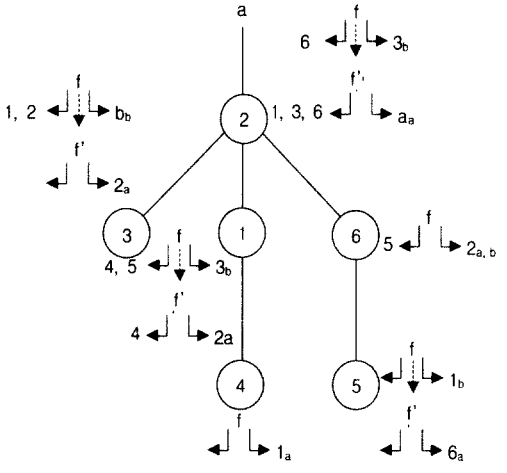
algorithm subscribe
input U : 등록자
f : 필터
a : f 의 원등록자
begin
(1) **if** $\exists f' \in$ Ps s.t. $f < f'$ and $U \in$ *subscribers*(f')
(2) **if** $U_a \in$ *tagged_subscribers*(f')
(3) **return**
(4) **else**
(5) U_a를 *tagged_subscribers*(f')에 추가한다. (merge)
(6) **elseif** $\exists f' \in$ Ps s.t. $f' = f$ and $U \notin$ *subscribers*(f')
(7) U_a를 *tagged_subscribers*(f')에 추가한다.
(8) **else**
(9) f의 등록정보를 포함관계에 따라 poset Ps의 정확한 위치에 삽입하고 U_a를 *tagged_subscribers*(f)에 추가한다.
(10) **endif**
(11) **if** $\exists f' \in$ Ps s.t. $f' < f$, $U_t \in$ *subscribers*(f')
(12) Ut를 *tagged_subscribers*(f')에서 제거하고 *tagged_subscribers*(f)에 넣는다. (merge)
(13) **if** *subscribers*(f') = $\varnothing$
(14) Ps로부터 f'를 제거한다.
(15) **endif**
(16) **endif**
(17) *children*(a, X)에 속하는 각 이웃노드에 subscribe(X, f, a)를 보낸다
end

SIENA와의 가장 큰 차이는 forward 집합을 구하는 방법이다. 위 알고리즘에서 노드 X에서의 f의 forward 집합은 a의 ST에서의 X의 자식 노드(*children*(a, X)라고 표기한다)들이다. SIENA와는 달리 기존에 노드 X를 통해 전달된 필터와의 포함관계에 상관없이 항상 a의 ST에서 X의 모든 자식노드들에 전달된다. 따라서 poset에서 각 필터의 forward 집합 정보는 중요하지 않다. 하지만 이해를 돕기 위해 계속사용하기로 한다. 또 한가지의 차이점은 알고리즘에서 보듯이 poset에서의 각 등록의 등록자에 원등록자를 추가한다는 점이다. 이 추가된 정보는 서로 다른 ST를 타고 전달된 등록을 구분하기 위해서이다. (그림 8)은 b 에서 f를 등록한 다음 a에서 f'를 등록한 후의 모습이다.



a) b 에서 f를 등록한 후

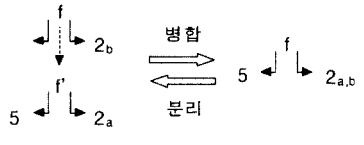
3) 클라이언트 a의 ST를 말할 때, 이벤트 서버 네트워크에서의 a가 연결된 서버의 ST를 간주한다.



b) a에서 f'를 등록한 후

(그림 8) f와 f'를 등록한 모습

SIENA에서는 b가 f를 등록할 때 b가 이미 노드 2에서 6으로 전파했기 때문에 a가 f를 등록할 때는 보내지 않는다. 따라서 노드 5에서는 노드1을 통해 전파된 등록정보를 이용한다(불완전한 등록문제 야기, 2.5절 참조). 그러나 본 논문에서는 각자의 ST를 따라 등록정보를 보내기 때문에 다른 등록자가 이미 보냈더라도 다시 보내게 된다. f가 이미 a의 ST 따라 경로 2 -> 6을 지나갔지만 f'도 경로 2 -> 6을 지나 노드 5까지 전달된다. 한가지 주목할 것은 하나의 노드(e. g. 노드 2)로부터 두개의 포함관계를 가진 등록이 전달된 경우(가령, 노드 6) (그림 9)에서와 같이 더 포괄적인 필터로 병합(merge)된다(알고리즘 5번, 12번 라인 참조). 이것은 f에 포함되는 어떤 이벤트가 노드 6에 전달되면 항상 노드 2로 전달될 것이기 때문이다. 뒤에 f의 해지 시 다시 분리(split)된 후 f만 해지되어야 한다(3.3 해지방법 참조).



(그림 9) 병합 및 분리 연산

수정된 등록방법은 하나의 등록이 자신의 ST를 따라 모든 노드에 전달되었기 때문에 등록정보의 전송량이 SIENA보다 증가한다. 하지만 병합과 분리 연산을 통해 포함관계에 있는 이벤트의 전달경로는 여전히 SIENA에서처럼 통합되기 때문에 이벤트의 전송량은 증가하지 않는다. 다행히도 일반적으로 이벤트의 등록에 비해 이벤트의 전달이 빈도수가 크기 때문에 이벤트 전송량의 증가는 SIENA에서 발생한 두 가지 문제에 비하면 큰 결함이 아닐 것이다.

3.3 필터 분리연산을 이용한 이벤트 해지

기본적으로 해지 방법은 SIENA에 비해 매우 쉽다. 왜냐하면 SIENA에서는 한 필터에서 어떤 등록자의 삭제로 인해 그 필터의 forward 집합이 변경되고 이는 다시 추가적인 등록 및 해지를 연쇄적으로 발생시킨다. 그러나 본 논문의 방식에서는 하나의 등록이 다른 ST에 병합연산을 제외하고는 다른 ST에 영향을 받지 않기 때문에 해지 또한 다른 ST의 영향을 거의 받지 않는다. 기본적으로 원등록자로부터 각 노드까지 ST를 따라 그 해지정보를 전달하면 된다. 하지만 효율적인 이벤트 전송을 위해 병합된 poset 노드들의 분리연산을 고려해야 한다.

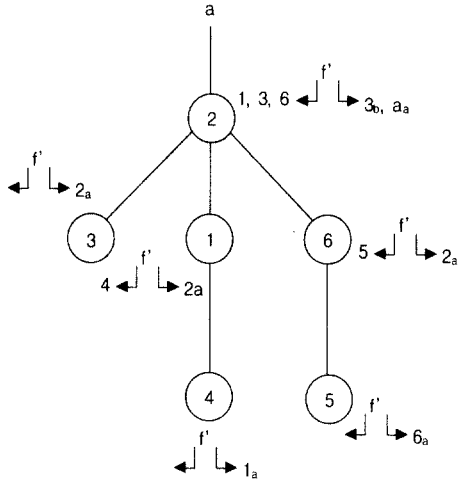
서버 X가 이웃노드 U로부터 unsubscribe(U, g, a)를 받았다고 하자. g에 의해 포함되면서 U_a를 등록자 집합에 포함하는 모든 f에 대하여 다음 세 가지 경우로 나누어 연산을 수행한다.

i) $f' < f$를 만족하는 가장 포괄적인 필터 $f' \in Ps$에 대하여, *children*(a, X)과 *children*(b, X)의 교집합이 공집합이 아닌 그런 b가 f'의 원등록자 중에 존재한다면, 그 교집합의 각 원소 노드 W에 split(f, f', X, b)을 전달한다. 그리고나서 X의 poset에서 f의 등록자 집합에서 U_a를 제거한다.

ii) $f < f'$를 만족하는 가장 포괄적인 필터 $f' \in Ps$에 대하여, *children*(a, X)과 *children*(b, X)의 교집합이 공집합이 아닌 그런 b가 f'의 원등록자 중에 존재한다면, 그 교집합의 각 원소 노드 W에 split(f', f, X, b)을 전달한다. 그리고나서 X의 poset에서 f의 등록자 집합에서 U_a를 제거한다.

iii) i) - ii)의 경우가 아니라면 X의 poset에서 f의 등록자 집합에서 U_a를 제거한다.

모든 경우에서 U_a를 제거한 후 등록자 집합이 ∅라면 필터 f를 poset에서 제거한다. 그리고 a의 ST에서의 자식 노드로 unsubscribe (X, g, a)를 보낸다.

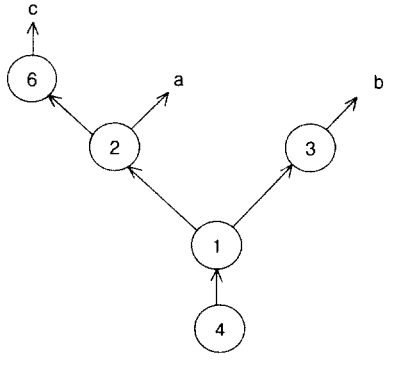


(그림 10) (그림 8) (b)에서 b가 f를 해지한 후의 모습

split(*f*, *f'*, *X*, *b*)를 받는 노드는 자신의 poset에서 *f*로부터 *f'*를 분리한다. *f'*의 등록자정보는 X_b가 되고 그 위치는 필터들의 포함관계에 의해 결정된다. (그림 10)은 (그림 8)(b)에서 *b* 가 *f*를 해지한 후의 모습이다. 노드 2가 unsubscribe(*f*, *3*, *b*)를 받으면 조건 i)에 해당하는 경우로써 poset에 a_a로부터 전달된 $f'(f' < f)$가 존재하고 *children*(*a*, *2*)과 *children*(*b*, *2*)의 교집합이 {6}이다. 따라서 노드 6에 split (*f*, *f'*, *2*, *b*) 메시지를 보낸다. 노드 6이 이 메시지를 받으면 poset의 *f* 노드를 (그림 9)에서 병합하기 전처럼 분리한다.

3.4 역경로 트리를 이용한 이벤트 전달

본 논문에서는 이벤트가 발표자로부터 등록자에게 전달되는 경로는 등록아 전달된 경로의 역경로이다. 따라서 문제는 이 역경로를 어떻게 계산하는 가이다. 본 논문에서는 두 가지 정보를 이용하여 이 역경로를 산출한다. 하나는 poset의 필터에 저장된 등록자 정보이고, 다른 하나는 이벤트 발표자를 위한 ST트리이다. 이벤트 발표자를 노드 *U*라 하고, $\Sigma = \{S_1, S_2, \cdots, S_n\}$을 각 노드를 위한 등록자 ST의 집합이라고 하자. 각 트리(S_i)에서의 루트로부터 U로의 등록 경로(P_{iu})는 정확하게 하나씩 존재한다. S_i를 잘 설계하면 이들 경로의 집합 $\{P_{1u}, P_{2u}, \cdots, P_{nu}\}$은 노드 *U*를 루트로 하는 ST가 되며, 이는 U를 위한 이벤트 전송 ST로 활용한다. 부록 A에 등록자 ST를 만드는 한가지 방법이 첨부되어 있다. 제안된 방법을 (그림 4)에 적용하면 노드 4를 위한 이벤트 전송 ST는 (그림 11)과 같다.



(그림 11) 노드 4의 이벤트 전송 ST

모든 클라이언트의 이벤트 등록 ST가 미리 계산될 수 있기 때문에 각 노드를 위한 이벤트 전송 ST도 미리 계산될 수 있다. 따라서 각 노드는 모든 다른 노드(이벤트 발표자)로부터 모든 다른 노드(필터 등록자)로 가는 경로정보를 유지하고 있다. 하나의 노드에서 경로정보는 이벤트 발표자를 위한 ST에서의 자식 노드의 집합으로 표현할 수 있다. 이벤트 발표자 노드를 $P_1, P_2, \cdots, P_n$이라고 하면 노드 *X*가 유지해야 하는 경로정보는 *children*(P_i, *X*) (i = 1, ⋯, n)이다. 예를 들어, (그림 3)에서 노드 2의 경로정보는 다음과 같다.

children(*1*, *2*) = {*a*, *6*}, *children*(*2*, *2*) = {*a*, *3*, *6*},
children(*3*, *2*) = {*a*, *6*}
children(*4*, *2*) = {*a*, *6*}, *children*(*5*, *2*) = {*a*},
children(*6*, *2*) = {*a*, *3*}

예를 들어, 이벤트가 노드 4에서 이벤트가 발생하여 노드 2에 도착했다면 전파될 수 있는 다음 노드는 {*a*, *6*}이 된다.

노드 *P*에서 발생한 이벤트 *e*가 어떤 노드 *X*에 도착했을 때 실제로 이동해야 할 다음 노드의 집합 *next*(*P*, *e*, *X*)는 *X*에서의 경로정보와 *e*가 매치되는 필터의 등록자 집합으로부터 쉽게 계산할 수 있다. 노드 *X*의 poset에서 *e*를 포함하는 모든 필터들의 등록자 집합을 *subscribers*(*X*, *e*)이라 하면 *next*(*P*, *e*, *X*)는 다음과 같다.

$$next(P, e, X) = children(P, X) \cap subscribers(X, e) \quad (2)$$

(그림 8)과 같이 *b*와 *a*에서 *f*와 *f'*를 등록한 후, 노드 4가 생성한 이벤트 *e*가 *f'*에 포함된다고 가정하면 이 이벤트가 전송되는 경로는 다음 두 가지가 된다.

4 -> 1 -> 2 -> a
4 -> 1 -> 3 -> b

SIENA에서와는 달리 2에 도착한 *e*가 갈 수 있는 길은 *a* ({*a*, *6*} ∩ {*a*, *3*} = {*a*}) 뿐이다.

4. 분 석

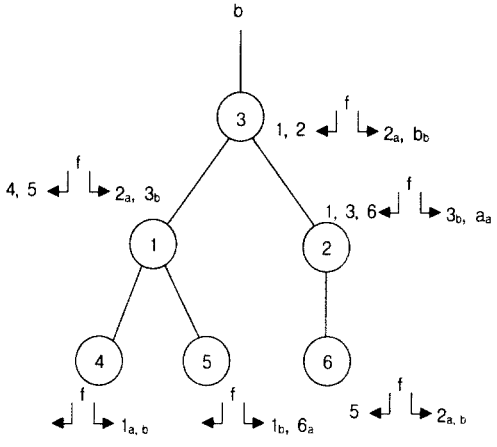
4.1 다중경로문제의 해결

이벤트 등록 알고리즘에 따르면 이벤트 등록이 전달되어 온 이웃 노드를 poset에 기억하고 있기 때문에 이 이웃노드를 통해 이벤트를 역으로 전달하면 반드시 모든 등록자에게 적어도 한번은 전달된다. 즉, 이벤트 등록이 전달되어온 역경로를 포함해서 하나 이상의 경로를 통해 이벤트가 하나의 등록자에게 전달된다. 이는 식 (2)의 *subscribers*(*X*, *e*)에 반영된다. 그리고 이벤트 발표자를 위한 ST는 그 발표자로부터 모든 클라이언트에게 정확하게 하나씩의 이벤트 전송 경로를 제공한다. 이는 식 (2)의 *children*(*P*, *X*)에 반영된다. 이벤트 등록이 전달된 경로의 역경로의 집합이 이벤트 발표자를 위한 ST이므로 이 두 항의 교집합은 정확하게 이벤트 등록이 전송되어온 역경로의 집합으로 이벤트 발표자를 루트로 하는 하나의 트리가 된다. 이 트리는 이벤트 발표자를 위한 ST의 부분집합이다. 따라서 식 (2)에 의한 이벤트의 전송은 하나씩의 경로를 통해 각 이벤트 등록자에게 전달된다.

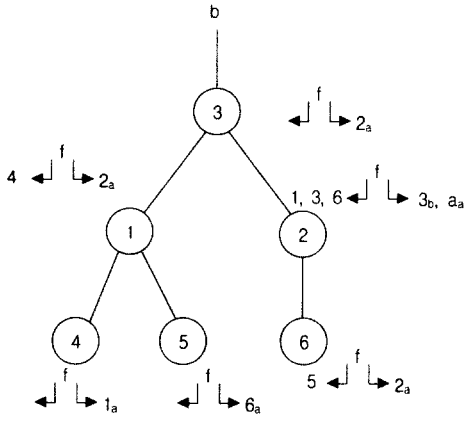
4.2 불완전 등록문제 해결

불완전 등록문제는 이벤트 등록이 효과적인 이벤트 등록을 위해 다른 이벤트 등록에 의해 차단되어 전달되지 않음으로써 생긴 문제이다. 본 논문에서는 하나의 클라이언트에서 제기된 이벤트등록은 자신의 ST를 타고 모든 노드에

전달되기 때문에 이러한 문제가 발생하지 않는다. 앞에서 든 예를 다시 고려해보자. (그림 12)의 (a)는 b와 a가 각각 f를 등록한 후의 poset 구조이며 (b)는 b가 f를 해지한 후의 모습이다. 노드 5에서 a에 의한 등록정보가 유지되어 있는 것을 알 수 있다.



(a) b, a가 f를 각각 등록한 후의 모습



(b) b가 f를 해지한 후의 모습

(그림 12) 불완전 등록문제의 해결

4.3 관련 작업과의 비교

두개의 대표적인 관련 작업인 SIENA와 Gryphon과 등록, 해지, 이벤트 전달, 세 가지 연산을 라우팅 경로 알고리즘과 전송량의 관점에서 비교한다(<표 1> 참조). SIENA는 포함관계를 기반으로 하는 가장 포괄적 필터(most general filter)의 등록원칙을 이용하여 이벤트 등록의 전송량과 이벤트의 전송량을 크게 줄인다. 또한 가장 포괄적 필터의 등록은 각 노드가 유지하는 등록정보의 양을 크게 줄인다. 따라서 이벤트를 전달할 때 인접 등록자를 찾기 위한 poset에서의 필터와의 매칭 연산이 크게 줄어든다. 그러나 앞에서 설명한 바와 같이 두 가지 문제를 안고 있다. 본 논문에서는 SIENA의 가장 포괄적 필터의 이점을 그대로 유지하면서 SIENA의 문제점을 해결한다. 한가지 단점은 같은 등록자 ST를 이용하지 않는 경우 등록정보를 ST를 따라 모든 노드에 브로드캐스팅 하기 때문에 등록정보의 전송량에 있어서 SIENA에 비해 늘어난다. 하지만 이벤트 전달을 위해 사용되는 등록정보의 유지는 포함관계를 이용하여 관리하기 때문에 이벤트 전송량은 SIENA와 기본적으로 같다.

〈표 1〉 관련작업과의 비교

	SIENA	개선안	Gryphon
자료구조	• ST 및 poset	• ST 및 poset	• ST 및 매칭트리
이벤트 등록	• ST에 기반 • 포함관계이용	• ST에 기반 • 부분적으로 포함관계이용	• ST에 기반 • 전 노드에 전송
이벤트 전달	• poset에 기반 전달	• 역경로 ST와 poset이용	• 매칭트리 이용
이벤트 해지	• ST와 poset 구조 이용	• ST와 poset 구조 이용	• 언급 없음
주요 장점	• 등록정보 전송량 적음 • 각 노드가 유지하는 등록정보 적음	• 이벤트 전달경로 계산 용이 • 각 노드가 유지하는 등록정보 적음	• 효율적인 매칭 알고리즘(국지적)
주요 단점	• 다중 경로문제 • 불완전 등록문제	• 등록정보 전송량 SIENA에 비해 많음	• 등록정보 전송량 많음 • 각 노드가 유지하는 등록정보 많음 • 매칭트리 갱신복잡

Gryphon은 등록정보의 저장을 위해 보다 정교한 매칭트리(matching tree)를 가지고 이를 이벤트의 전달시에 라우팅 경로의 계산(link matching algorithm)에 이용한다[2]. 매칭트리는 poset 구조에 비해 매칭되는 필터를 찾는데 국지적으로 효율적인 방법을 제공하나 이벤트 등록의 포함관계를 고려하지 않고 각 서버가 동일한 매칭트리를 가지고 있기 때문에 각 노드가 유지해야 하는 등록정보의 양이 매우 크다. 따라서 전체적으로는 라우팅 경로를 찾기 위해 poset구조에 비해 훨씬 많은 양의 트리를 찾아야 하기 때문에 비효율적이다. 또한 이벤트등록과 해지시 마다 매칭트리를 다시 계산하여 링크정보를 계산해야 한다는 단점을 가지고 있다. 그러나 매칭트리의 장점은 poset과 상호보완적이기 때문에 poset에 결합하여 사용할 수 있을 것으로 보인다.

또 한가지 중요한 요소는 이벤트의 전송량이다. 세 가지 방법에서 사용하는 이벤트 전달은 기본적으로 이벤트 등록의 역경로를 이용한다는 점에서 동일하다. 그러나 SIENA에서 다중경로문제로 인하여 이벤트의 전송량이 불필요하게 증가한다.

5. 결 론

이 논문에서는 대표적인 이벤트 통지시스템인 SIENA의 이벤트 라우팅 알고리즘의 문제점을 분석하고 해결책을 개발하였다. 본 논문에서 제안한 방법은 기본적으로 SIENA가 제안한 포함관계에 의한 가장 포괄적 필터등록의 원칙에 바탕을 두고 있다. SIENA 알고리즘이 가지고 있는 두 가지 문제점을 분석하고 그 해결책을 제시하였다. 본 논문에서 제시한 알고리즘은 등록정보의 전송량 부분에서는 다

소 증가하나 라우팅 경로를 찾는 알고리즘의 복잡도면에서나 이벤트의 전송량에서는 SIENA와 거의 동일하다.

앞으로 해야할 일은 제시한 알고리즘을 시뮬레이션 환경에서 등록의 유사성, 유사한 등록의 지역성 등 다양한 특성을 고려하여 분석하는 일이다. 본 논문에서는 현재 네트워크 위상에 관계없이 최소깊이의 클라이언트의 ST를 가정하고 있는데 네트워크의 위상에 따른 클라이언트의 ST를 어떻게 만드는 것이 좋은가에 대한 분석이 필요하다. 마지막으로 poset에서의 필터와 이벤트와의 효율적인 매칭을 위한 방법에 대한 연구이다. 한가지 고려할 수 있는 방법은 Gryphon의 매칭트리와 poset구조를 결합하여 두 가지의 장점을 동시에 얻는 것이다.

참 고 문 헌

[1] M. K. Aguilera, R. E. Strom, D. C. Struman, M. Astley, T. D. Chandra, "Matching Events in a Content-Based Subscription System," *Proceedings of 18th ACM Symposium on Principles of Distributed Computing(PODC'99)*, Atlanta, GA, pp.53-61, May, 1999.

[2] G. Banavar, T. Chandra, B. Mukherjee, J. nagarajaro, "An Efficient Multicast Protocol for Content Based Publish-Subscribe Systems," *Proceedings of 19th IEEE International Conference on Distributed Computing Systems(ICDCS'99)*, Austin, TX, pp.262-272, May, 1999.

[3] A. Carzaniga, D.S. Rosenblum, and A. L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," *ACM Transactions on Computer Systems*, 19(3), pp.332-383, Aug. 2001.

[4] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Achieving Expressiveness and Scalability in an Internet-Scale Event Notification Service," *Proceedings of 19th ACM Symposium on Principles of Distributed Computing (PODC 2000)*, Portland OR July, 2000.

[5] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Interfaces and Algorithms for a Wide-Area Event Notification Service," *Technical Report CU-CS-888-99*, Department of Computer Science, University of Colorado, October, 1999. (revised May 2000).

[6] G. Cugola, E. Di Nitto, and A. Fuggetta. Exploiting an Event-based Infrastructure to Develop Complex Distributed Systems. *Proceedings of the 20th International Conference On Software Engineering (ICSE98)*, Kyoto, Japan, Apr. 1998.

[7] M. Hapner, et.al. *Java Message Service*, Version 1.0.2 Sun Microsystems, Nov. 1999.

[8] M. Wray, R. Hawkes, "Distributed Virtual Environments and VRML : an Event-based Architecture," *Computer Networks and ISDN Systems 1-7*, 30, pp.43-51, 1998.

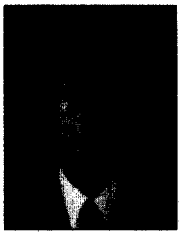
[9] Object Management Group, "Notification Service," *Technical Report*, Nov. 1998.

[10] B. Segall and D. Arnold. Elvin has left the building : A publish/subscribe notification service with quenching. *Proceedings of AUUG97*, Brisbane, Queensland, Australia, Sept. 1997.

부록 A.

보조정리 1. 노드의 집합 $V=\{v_1, v_2, \cdots v_n\}$와 에지의 집합 E를 가지는 그래프 $G=(V, E)$와 각 노드를 위한 ST의 집합 $\Sigma=\{S_1, S_2, \cdots, S_n\}$가 있다. 트리 S_i는 노드 v_i를 루트로 하는 최소 깊이의 G의 ST라고 가정하자. 각 v_i(i = 1, 2, ···, n)에서 임의의 $v_k \in V$로의 경로 P_{ik}는 ST S_i에 하나씩 존재한다. 이들 경로의 집합 P_{ik}(i = 1, 2, ···, n)은 이들 모든 경로들에 속하는 노드의 집합 V_k와 에지의 집합 E_k으로 이루어진 그래프 $G_k=(V_k, E_k)$의 최소깊이를 가진ST(v_k가 루트인)가 되는 Σ가 존재한다.

증명) 이 문제는 두 가지 문제로 나누어진다. 하나는 P_{ik}의 집합이 트리를 구성하는 가이고, 다른 하나는 그것이 최소깊이가 되는 지에 관한 문제이다. 우선 전자의 경우부터 고려해보자. P_{ik}의 집합이 트리가 되지 않는 경우는 v_k에서 두개의 노드 v_i, v_j로의 경로상에 사이클이 존재하는 경우이다. 사이클을 이루는 노드 중에 v_k로 연결되는 노드를 p라하고 v_i와 v_j로 연결되는 접점을 q라고 할 때, p와 q사이의 경로가 두 개 존재하고 이는 각각 P_{ik}와 P_{jk} 경로상에 있다. 그런데 두 경로의 길이가 다르다면 각 노드에서 v_k로의 각 경로가 최소라는 가정에 모순이 된다. 만약 p, q사이의 두 경로의 길이가 같다면 P_{ik}와 P_{jk}가 p, q사이에서 같은 경로를 선택하도록 하면 된다. 두 번째 문제는 P_{ik}(i = 1, 2, ···, n)이 v_i와 v_k사이에서 최소의 길이를 가지므로 바로 증명된다. □



류 기 열

e-mail : kryu@madang.ajou.ac.kr

1985년 서울대학교 전자계산기공학과 (공학사)

1987년 한국과학기술원 전산학과(공학석사)

1992년 한국과학기술원 전산학과(공학박사)

1992년~1993년 한국과학기술원 전산학과 연구원

1993년~1994년 일본 토쿄대학 정보과학과 객원연구원

1994년~현재 아주대학교 정보및컴퓨터공학부 부교수

관심분야 : 객체지향시스템, 컴포넌트기술, 메시징 시스템

이 정 태

e-mail : jungtae@madang.ajou.ac.kr

1979년 서울대학교 농과대학(학사)

1981년 서울대학교 계산통계학과(이학석사)

1988년 서울대학교 계산통계학과(이학박사)

1981년~1983년 울산대학교 전자계산학과 전임강사

1983년~1988년 아주대학과 전자계산학과 전임강사

1988년~현재 아주대학교 정보및컴퓨터공학부 부교수

관심분야 : 객체지향시스템, 컴포넌트기술, 프로그래밍언어