ORB 브리지의 부하 균등을 위한 효율적인 알고리즘

김영균°, 김경하, 김영학, 오길호
금오공과대학교 컴퓨터공학부

An Efficient Algorithm for Balancing the Load of ORB Bridges

Young Gyun Kim, Kyoung Ha Kim, Young Hak Kim, Gil Ho Oh
School of Computer Eng., Kum-Oh National University of Technology

요 약

대 규모의 CORBA 시스템을 구축할 때 서로 다른 프로토콜을 갖는 여러 개의 ORB 도메인이 존재한다. 이러한 서로 다른 ORB 도메인들 간의 통신은 ORB 브리지를 통하여 수행된다. 따라서 전체 시스템에서 각 ORB 도메인 간의 브리지의 수는 가능하면 적어야 하고 각 브리지에 가중되는 부하 역시 적어야 한다. 본 논문에서는 이러한 문제를 그래프 개념으로 모델링하고 기본적인 그래프 연산들을 이용하여, 브리지의 수를 줄이고 각 브리지에 가중되는 부하를 균등하게 분할하여 전체 시스템을 구성하는 효율적인 알고리즘을 제안한다.

1. 서론

대규모의 CORBA(Common Object Request Broker Architecture) 시스템의 구축 시에 같은 프로토콜 혹은 서로 다른 프로토콜을 사용하는 여러 개의 ORB(Object Request Broker) 도메인들이 존재한다. 이러한 서로 다른 도메인들은 브리지 개념을 사용하여 네트워크상에서 연결되며, 브리지는 ORB 도메인의 프로토콜이 같은가 혹은 다른가에 따라 전 브리지(full-bridge)와 반 브리지(half-bridge)로 나누어 진다[1,2,3].

여기서 전 브리지는 같은 프로토콜을 사용하는 두 개의 도메인을 직접 연결해주는 역할을 하고, 반 브리지는 서로 다른 프로토콜을 사용하는 두 개의 도메인을 연결해주는 역할을 한다. 분산 환경에서 각 ORB 도메인 간의 통신 시에 불필요한 브리지의 경유는 전체 시스템의 성능 저하를 초래할 수 있다. 또한 많은 서로 다른 프로토콜을 갖는 ORB 도메인을 한 그룹으로 연결하는 것은 심한 병목 현상의 원인이 될 수 있다. 따라서 대규모의 CORBA 시스템 설계 시에 가능한 한 ORB 브리지의 수를 최소화하여야 하고, 각 ORB 브리지에 부가되는 부하 역시 줄어져야 한다. 현재 이러한 문제를 효율적으로 해결하기 위한 방법론들이 활발히 연구되고 있지 않다.

본 논문에서는 먼저 이러한 문제를 효율적으로 해결하기 위해서 같은 ORB 도메인들을 그룹별로 분할하고 각 그룹에 할당된 도메인들을 임계(threshold) 값에 따라 다시 분할하는 정책을 사용한다. 다음에 이와 같이 분할된 그룹들을 다시 임계값에 따라 중간 브리지 개념을 사용하여 연결하는 아이디어를 사용한다. 이러한 정책의 구현을 위해 본 논문에서는 위에서 기술한 문제를 그래프 개념을 도입하여 모델링하고, 기본적인 그래프 연산 들을 이용하여 효율적으로 도메인을 그룹화 하고 브리지를 구성하는 알고리즘을 설계한다.

2. ORB 브리지

ORB 브리지는 ORB 도메인 사이에서 상호 운용이 가능하도록 하는 역할을 하는 모듈을 의미한다. CORBA 에서는 ORB 도메인을 구성하기 위해서 인라인(in-line) 브리지와 요구 레벨(request-level) 브리지가 존재한다[4,5].

인라인 브리지는 ORB 간의 브리지를 구현하는 가장 직접적인 방법으로, ORB 내부에 직접 전송 모듈을 추가하여 같은 ORB 에 있는 것처럼 ORB 간의 직접적인 통신을 지원한다. 요청 레벨 브리지는 ORB 에서 제공하는 인터페이스를 사용하여 ORB 도메인 간의 통신이 이루어 진다.

클라이언트는 스텁 혹은 DII(Dynamic Invocation Interface)를 사용하여 다른 ORB 상의 구현 객체가 마치 클라이언트의 ORB 상에 있는 객체처럼 호출할 수 있다. 이 호출은 클라이언트의 DSI 를 이용하여 양쪽 ORB 사이의 브리지를 통해 전달되고, 서버 측의 ORB 는 이 요청을 DII 를 통해 전달 받고 처리한다. 요구-레벨 브리지는 다음과 같은 과정을 통해 이루어진다.

1) 클라이언트의 요청은 ORB 의 프록시 객체에 전달 된다.
2) 클라이언트의 프록시는 이 요청을 서버 ORB 에서 사용될 수 있는 구조의 내용으로 변환한다.
3) 프록시 객체는 서버 객체 상의 요청된 연산을 호출한다.
4) 연산의 결과는 임의의 타입으로 ORB 간 서로 보완적인 라우트를 통해 클라이언트에게 전달 된다.

요청 레벨 브리지는 DCOM(Distributed Component Object Model)과 같은 다른 분산 미들웨어와 상호 연동하기 위한 구조이다. 그림 1 은 이러한 원리를 도시한 예를 보여준다.

CORBA 표준에서는 ORB 도메인을 두 가지 방법에 의해서 구성한다. 하나는 직접 브리지 방식이고, 다른 하나는 중간 브리지 혹은 백본 브리지 방법이다. 직접 브리지 방법은 프로토콜이 같은가 혹은 다른가에 따라 각각 전 브리지 혹은 반 브리지에 의해서 ORB 도메인과 도메인을 직접 연결 한다. 반면에 중간 브리지 방법은 도메인들 간에 중간 혹은 백본 브리지를 위치시켜 연결부를 단순화 시킨다. 너무 많은 도메인을 백본 브리지로 연결한 경우 부하의 균형을 상실할 수 있다. 따라서

두 브리지의 적절한 구성에 의해서 전체 시스템이 구성되어야 한다.

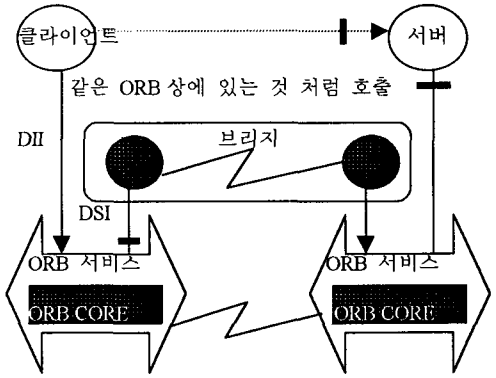

그림 1. 요구 레벨 브리지

그림 2 에서 ORB {A,C,E,F}, {B,D}는 각각 같은 ORB 프로토콜을 사용하므로 전 브리지에 의해서 연결되었으며, 나머지는 다른 프로토콜을 사용하므로 반 브리지에 의해 연결되었다. 그림 3 은 서로 다른 프로토콜들을 한 개의 중간 브리지 방법으로 연결한 예를 보여준다.

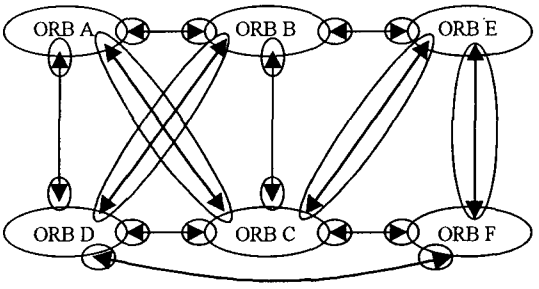

그림 2. 직접 브리지에 의한 연결

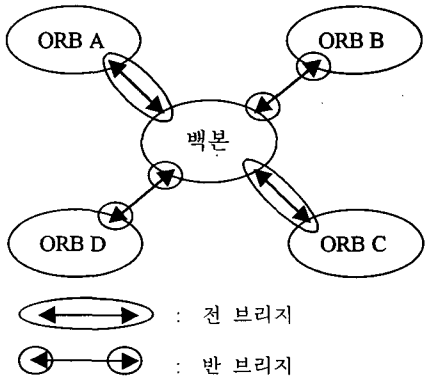

그림 3. 중간 브리지에 의한 연결

그림 2 와 같은 직접 브리지에 의한 연결은 너무 많은 브리지 들이 존재한다. 이와 같은 구성의 경우 각 ORB 도메인 간에 직접 연결된 브리지에 의해서 빠른 통신을 기대할 수 있지만 전체 시스템 구성을 위해 너무 많은 비용을 초래한다.

반면에 그림 3 과 같은 구성은 모든 ORB 도메인이 백본 ORB 에 의해서 연결되어 불필요한 통신 오버헤드가 발생할 수 있다. 예로써 ORB A 와 ORB C 는 같은 프로토콜을 사용함에도 불구하고 백본 ORB 를 경유한다. 또한 한 개의 백본 ORB 가 모든 ORB 를 지원하므로 해서 발생하는 통신 오버헤드 문제가 야기될 수 있다. 본 논문에서는 이러한 문제 요소를 관찰하여 가능한 브리지의 수를 줄이고, 또한 백본 ORB 에서 발생하는 부하의 균형을 해결하기 위한 방법을 제안하고자 한다.

3. 알고리즘 설계

본 논문에서 사용된 기본적인 아이디어는 다음과 같다. 먼저 같은 ORB 도메인들을 그룹별로 분할하고 각 그룹에 포함된 도메인들을 백본 ORB 를 사용하여 구성한다. 부하 균형 문제를 해결하기 위해서 만일 각 그룹에 포함된 도메인들의 수가 임계 값을 초가할 경우는 다시 각 그룹에 포함된 도메인들의 수를 임계 값으로 나누어 서브 그룹으로 분할한다. 서브 그룹 역시 같은 방법으로 백본 브리지를 사용하여 구성한다. 사실 이러한 경우 독립적인 프로토콜을 갖는 ORB 도메인은 한 개의 도메인으로 구성된 그룹으로 분할된다. 따라서 한 개의 도메인 만을 갖는 그룹들은 임계 값의 수 만큼 다시 재 그룹화 하여 백본을 구성하는 방법을 사용한다.

이러한 기본적인 아이디어의 구현을 위해 우리는 위의 문제를 그래프 개념을 도입하여 모델링하고, 그래프에 관련된 다수의 기본적인 연산들을 이용하여 문제를 해결한다. 먼저 그래프 G=(V, E)를 다음과 같이 정의한다. 여기서 V 는 ORB 도메인들의 집합으로 정의하며 후에 설명을 단순화 하기 위해서 이를 노드라는 용어로 표현한다. E 는 전 브리지와 반 브리지로 구성된 에지의 집합으로 정의한다.

본 논문에서는 전체 시스템을 단순히 그림 2 와 같이 직접 브리지에 의해서 연결된 구성을 입력으로 하여 여기서 발생된 부하와 브리지 수를 최소화하는 알고리즘을 설계한다. 또한 부하의 균등 분배를 위해서 임계 값을 δ로 정의한다. 이러한 임계 값은 분산환경에 따라 다를 수 있지만 여기서는 특정한 값을 고정하지 않고 일반적인 상황을 고려한다. 후에 우리는 계속되는 연구과제로 시뮬레이션을 통하여 이러한 임계 값의 적절한 수준을 찾을 예정이다. 지금까지 설명한 내용을 개략적으로 정리한 알고리즘은 아래와 같다. 각 단계에 대한 상세한 과정은 후에 설명한다.

[알고리즘 : ORB_MIN]
입력 : 직접 브리지에 의해 연결된 G=(V,E)
출력 : 브리지의 수와 부하의 균형을 고려한 ORB 도메인의 연결

1. {E – {반 브리지로 구성된 에지}}을 E'으로 설정하고, E'을 갖는 그래프를 G'=(V, E')으로 정의한다.
2. G'으로부터 연결 요소(connected component)를 찾는다. 각 연결 요소를 $p_1, p_2, \ldots, p_k$로 정의한다. 여기서 k 는 전체 연결요소의 수를 의미하고, 각 p_i는 같은 프로토콜을 갖는 ORB 도메인들로 구성된 그룹을 의미한다.
3. $p_1, p_2, \ldots, p_k$로부터 $|p_i| = 1$ 인 모든 p_i들을 하나의 그룹으로 모은다. 새로 생성된 분할을 $s_1, s_2, \ldots, s_{k'}$로 정의한다.
4. 다음을 수행하여 각 그룹 s_i를 임계 값 δ에 따라 서브 그룹으로 분할한다.
   for (i =1 to k')

```
{
  if (|s_i| > δ) {
    s_i를 |s_i| / δ개의 서브 그룹으로 분할하고, 각
    서브 그룹을 s_i(1), s_i(2), ..., s_i(|s_i| / δ)로 정의한다;
    각 서브 그룹 s_i(t)에 포함된 노드들을 하나의
    백본 브리지로 연결한다; }
  else {
    그룹 s_i에 포함된 노드들을 백본 브리지로
    연결한다; }
```

5. 단계 4에서 백본 브리지에 의해서 연결된 모든 서브 분할들을 $r_1, r_2, \ldots, r_{k''}$로 정의한다. 여기서 k''은 단계 4에서 백본 브리지로 연결된 모든 서브 그룹의 수를 의미한다.
6. 단계 3과 유사한 방법을 사용하여 $r_1, r_2, \ldots, r_{k''}$의 서브 그룹들을 k''/δ 개로 재 그룹하여 각 그룹을 백본 브리지로 연결한다. 이러한 과정을 δ보다 더 적은 그룹들의 모임이 있을 때까지 반복한다.

```
end
```

알고리즘 분석. 알고리즘 ORB_MIN에서의 입력은 직접 브리지에 의해 연결된 그래프 G=(V,E)를 고려하였다. 우리의 알고리즘은 이러한 G로 부터 브리지의 수와 각 브리지에 대한 부하를 고려하여 구성한 분할 그래프를 출력한다. 알고리즘에서 기술된 모든 나누기 연산 결과는 위쪽으로 반올림된다고 가정한다. 먼저 단계 1에서는 같은 프로토콜을 갖는 ORB 도메인들을 찾기 위한 방법으로 전 브리지와 반 브리지로 구성된 에지의 집합 E로부터 반 브리지로 구성된 에지를 제거하고 나머지 에지들을 E'에 삽입한다. 그러면 E'은 전 브리지를 갖는 에지 만으로 구성되고 이러한 그래프를 G'으로 정의한다.

단계 2에서는 G'으로부터 연결 요소 $p_1, p_2, \ldots, p_k$ 들을 찾는다. 여기서 각 p_i에 포함된 ORB 도메인들은 같은 프로토콜을 가지며 p_i는 하나의 그룹으로 정의된다. 그래프에서 각 연결 요소와 전체 연결 요소의 수(k)를 찾는 문제는 DFS(Depth First Search) 알고리즘을 이용하면 쉽게 해결될 수 있다[6]. 단계 3에서는 각 p_i에 포함된 ORB 도메인이 한 개만 있는 경우를 고려한다. 이러한 경우는 단계 1에서 반 브리지를 제거함으로 해서 발생될 수 있다. 따라서 그러한 p_i는 한 개의 독립된 프로토콜 만을 가지고 있으므로 p_i에서 백본 브리지의 사용은 의미가 없기 때문에, $|p_i| = 1$ 인 모든 그룹을 하나의 그룹으로 재구성해야 한다. 재구성된 그룹들을 $s_1, s_2, \ldots, s_{k'}$으로 정의한다. 여기서 k'<k 가 된다.

단계 3과 4에서는 한 개의 백본 브리지에 너무 많은 ORB 도메인들이 연결되어 부하 균형 문제가 발생하기 때문에, 이러한 문제의 해결을 위해서 임계 값 δ를 고려한다. 즉 분할된 한 그룹에 속한 ORB 도메인의 수가 δ보다 클 경우는 각 s_i를 $|s_i|/\delta$ 개의 서브 그룹으로 분할하여 각 서브 그룹에 포함된 ORB 도메인들을 백본 브리지로 연결한다. 그렇지 않은 경우는 각 s_i에 포함된 ORB 도메인들을 바로 백본 브리지로 연결한다. 그러면 모든 그룹 혹은 다시 서브 그룹으로 분할된 그룹들은 독립적인 하나의 네트워크를 형성한다. 그러나 전체 시스템이 구성되기 위해서는 이러한 독립적인 네트워크들은 하나의 연결된 그래프 형태로 구성되어야 한다.

따라서 단계 5와 6에서는 독립적 형태로 구성된 네트워크들을 하나의 네트워크로 구성하기 위한 절차를 보여준다. 여기서 각 백본 브리지에 연결되는 ORB 도메인들의 수는 임계 값 δ를 넘으면 안되기 때문에, 독립된 네트워크를 하나의 네트워크로 구성할 때 역시 단계 3과 유사한 절차가 고려되어야 한다. 먼저 $r_1, r_2, \ldots, r_{k''}$로 정의된 독립된 네트워크로부터 k''이 δ보다 적을 경우는 이들을 바로 백본 브리지에 의해서 연결하면 된다. 그러나 k''이 δ보다 클 경우는 한 개의 백본 브리지로 연결이 불가 하므로 그룹 $r_1, r_2, \ldots, r_{k''}$을 k''/δ 개의 그룹을 다시 형성한다. 즉 δ 개의 그룹들을 하나의 상위 그룹으로 형성하고 이러한 상위 그룹의 수가 k''/δ 개가 됨을 의미한다. 그러므로 이러한 형태의 분할은 전체 상위 그룹의 분할 수가 δ보다 적을 때까지 진행되며 각 그룹들은 같은 절차에 의해서 백본 브리지로 연결된다. 그림 4는 δ=2 인 경우를 가정하여 그림 2에 주어진 구성을 알고리즘을 적용하여 생성한 결과를 보여준다. 원래 {A,C,E,F}는 같은 프로토콜을 가지므로 한 개의 분할로 구성되었으나, 임계 값 δ에 의해서 두 그룹으로 다시 분할되었다. 또한 {B,D}는 또 다른 그룹으로 분할 되었으며 이러한 분할된 그룹들이 다시 연결되어 전체 시스템이 구성되어야 한다.

본 논문에서 ORB 도메인의 집합의 수는 |V|이기 때문에 우리의 알고리즘에서 단계 1은 O(|E|), 단계 3, 4, 5는 각각 O(|V|) 시간 복잡도를 갖는다. 다음에 단계 2의 경우는 입력이 연결 리스트를 갖는 자료구조로 주어질 경우 [6]에서의 알고리즘을 이용하면 O(|V|+|E|) 시간 복잡도에 해결될 수 있다. 단계 6은 k''/δ 번 반복되고 k''은 |V|보다 적기 때문에 O(|V|) 시간 복잡도를 갖는다. 그러므로 알고리즘 ORB_MIN의 전체 시간 복잡도는 O(|V|+|E|)이 된다.

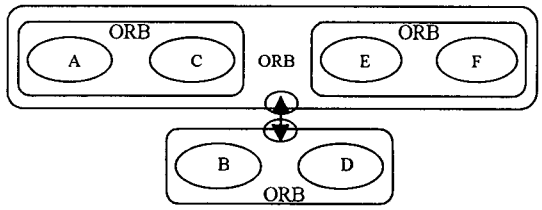

그림 4. δ=2 인 경우 그림 2를 재구성한 예

4. 결론

본 논문에서는 대규모의 CORBA 시스템을 구축할 때 서로 다른 프로토콜을 갖는 여러 개의 ORB 도메인들을 브리지의 수와 부하 문제를 고려하여 효율적으로 구성하는 알고리즘을 제안하였다. 본 논문에서 부하를 위해 고려한 임계 값은 일반적인 경우 이나 실제 분산 환경에서는 다른 값이 기대될 수 있다. 따라서 이러한 임계 값의 결정은 앞으로 연구과제이며 시뮬레이션을 통해서 예측될 수 있으리라 기대된다.

참고 문헌

[1] Building inter-ORB bridges(CORBA V2.2), Available from http://www.omg.org/, 1998.
[2] ORB interoperability architecture(CORBA V2.2), Available from http://www.omg.org/, 1998.
[3] Interoperability overview(CORBA V2.2), Available from http://www.omg.org/, 1998.
[4] D. C. Schmidt and S. Vinoski, "Object interconnection," *SIGS C++ Report magazine*, 1997.
[5] Object Management Group, The common object request broker : architecture and specification, 2.2ed, 1998.
[6] E. Horowitz and S. Sahni, Fundamentals of computer algorithm, *Computer Science Press*, 1978.