웹 서비스에서 효율적인 서비스 선택을 위한 QoS 협상 브로커 설계

이 수민, 박 제연, 송 영재
경희대학교 컴퓨터공학과
{leesm21c, jy_bak, yjsong}@khu.ac.kr

Design of QoS negotiation broker for efficient Service selection on Web Service

Su-Min Lee, Jea-Youn Park, Young-Jae Song
Dept of Computer Engineering, Kyung Hee University

요 약

이기종간 분산 환경에서 상호연동성을 고려한 웹 서비스가 각광받고 있다. 주요 웹 서비스 플레이어들이 선택한 SOAP, UDDI, WSDL과 같은 표준을 이용하여 전체 웹 서비스들 금융, 하이테크, 미디어, 엔터테인먼트 등 전체 웹 서비스들이 개발되고 있다. 대부분의 웹 서비스들이 표준을 확립해 감에 따라, 사용자 요구에 가장 적합한 QoS의 선택은 서비스 간 차별화의 요점이 될 것이다.

기존의 연구에서는 각 서비스에 추가적인 컴포넌트를 합성하여 서비스 사용자와 서비스 제공자끼리의 협상을 하는 방법과 서비스 사용자의 요구의 기준과 비중에 맞추어 랭킹을 계산하여 그 값을 사용자에게 돌려주어 수동적으로 선택할 수 있는 방법을 제시하였다.

하지만, 계속 추가되는 컴포넌트 합성은 서비스 양이 방대해질수록 시스템 복잡도는 증가하고, 그로 인해 서비스 시간지연 및 자원 낭비의 문제점이 있으며, 계산되어 랭크된 서비스들을 사용자가 선택하도록 하였을 경우 최상위에 랭크된 서비스가 사용자에 가장 적합한 지 알 수 없으며, 다양한 서비스 사용자의 요구에 만족하기 어려운 문제점을 가지고 있다.

본 논문에서는 QoS 협상 브로커를 제시하여 이러한 문제점들을 최소화하고 서비스 사용자가 요구하는 기능적인 서비스 측면을 유지하면서, 서비스 사용자의 요구조건에 효율적인 비 기능적 측면을 제공하는 서비스 제공자를 자동적으로 발견할 수 있게 한다.

1. 서 론

웹 서비스는 이기종간 분산 환경에서 빠른 이합집산을 가능하게 할 수 있는 가장 표준화되고 경제적인 수단이며, 다양한 사용자 및 제공자와의 연결성을 유지하는 중요한 매개체로서 그 활용은 보다 광범위해지고 있다. 웹 서비스의 확산에 따라 서비스 품질(QoS)은 서비스 제공자의 성공을 구분하는 중요한 요소가 되어가고 있다[1]. 웹 서비스의 QoS를 보장하는 것은 웹 서비스가 지닌 동적이고 예측불허의 본성 때문에 매우 중요한 과제이다.

서비스 지향 컴퓨팅(Service-Oriented Computing)의 패러다임은 빠른 속도로 웹 서비스에서 서비스 사용자의 기능적인 요구를 유지하고, QoS 기반 서비스를 효율적으로 제공하여야 한다.

이러한 서비스 사용자의 비 기능적 목적을 충족하는 서비스 선택을 위하여 기존 제안에서는 각 서비스에 추가적인 컴포넌트를 합성하여 서비스 사용자와 서비스 제공자끼리의 협상을 제시한다[2]. 하지만 서비스의 양이 방대해질수록 시스템의 복잡도가 증가하고, 그로 인한 자원 낭비, 서비스 시간 지연 등의 단점이 있다.

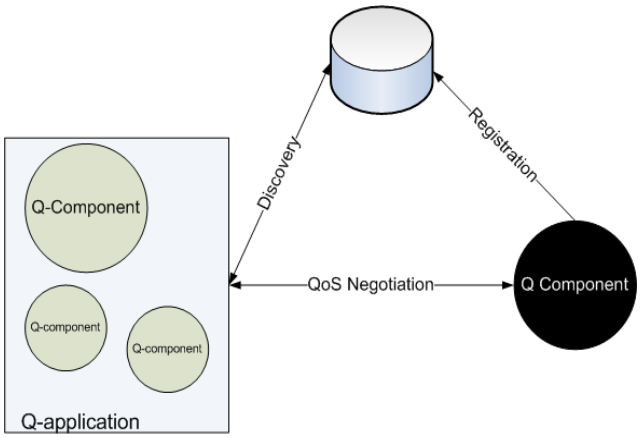
다른 제안에서는 등록된 서비스 제공자들을 대상으로 요구된 기능적인 면의 서비스를 제공하는지 여부를 가린 후, 서비스 사용자가 요구한 품질 기준과 비중에 맞추어 랭킹을 계산하여 그 값을 사용자에게 되돌려주어 최적의 서비스를 선택할 수 있도록 도움을 준다[3]. 하지만 랭킹이 가장 높은 서비스가 서비스 사용자에 가장 최적의 서비스일 보장은 없으며, 다양한 사용자의 요구에 적합하기 어렵다.

이러한 문제점을 해결하고자 본 논문은 사용자의 비 기능적인 요구에 충족하는, 가장 적합한 서비스를 자동적으로 검색할 수 있고, 시스템 자원 낭비를 최소화시킨 QoS 협상 브로커를 제시한다.

2. 관련 연구

2.1 QoS 기반 프레임워크

클라이언트/서버 시스템 내에서 서비스 사용자가 서버로부터 서비스를 요청할 때, 서버는 보통 이미 다른 요청에 대해 서비스하고 있다[2]. 만일 시스템이 과중한 요청을 받고 있는 경우라면 사용자는 가난한 응답시간과 처리량을 갖게 되기 때문에, 서비스 사용자는 보증 받을 수 있는 QoS 수준을 필요로 한다[4].



(그림 1) QoS 기반 어플리케이션

2.1.1 Q-컴포넌트의 주요 태스크

Q-컴포넌트는 서비스 레지스트리에 자신을 등록하고, QoS를 보증하는 동안 동시 발생하는 서비스 요청에 대해 서비스를 제공하며, 다른 컴포넌트들을 위해 만들어진 QoS 테이블을 유지한다. QoS 서비스 요청인 p는 Q-컴포넌트와의 세션 시작을 위해 사용되어진다.

2.1.2 계산되는 QoS 속성

- 응답시간 (Response time)

$$\Delta QoS_R = \frac{R_{\max} - R_{measured}}{R_{\max}}$$

- 처리량 (Throughput)

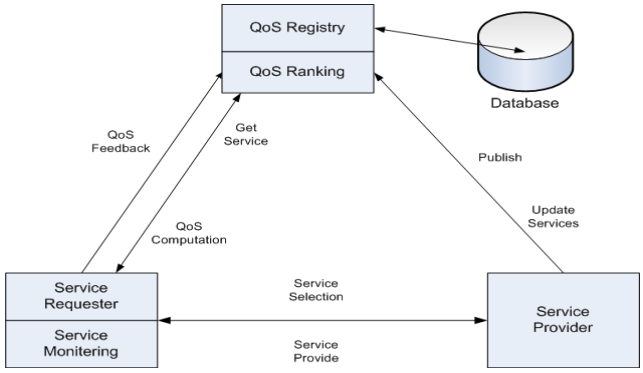
$$\Delta QoS_X = \frac{X_{measured} - X_{\min}}{X_{\min}}$$

- 거절될 확률 (Probability of reject)

$$\Delta QoS_P = \frac{P_{\max} - P_{measured}}{P_{\max}}$$

2.2 QoS 랭킹 아키텍처

QoS 랭킹 아키텍처는 비 기능적인 면의 속성들을 계산하여, 사용자가 원하는 기능적인 속성을 제공하는 서비스들의 랭킹을 부여하여 사용자가 원하는 최적의 서비스를 선택할 수 있도록 돕는다.



(그림 2) 웹서비스 랭킹 아키텍처

서비스 사용자와 제공자를 위한 확장 가능한 QoS 모델을 보안된 실행 사용자 피드백과 실행 모니터링을 통해, 공정하고 오픈된 계산을 달성한다. 이는 새로운 도메인 명세기준의 추가되어질 경우에도 계산 모델의 변경 없이 확장 가능하다[3].

2.2.1 고려되어야 할 품질 기준

- 실행가격 $q_{pr}(s)$

서비스 사용자가 하나의 웹서비스 s를 사용함으로써, 서비스 제공자에게 지급해야할 총 액수이다.

- 실행시간 $q_{du}(s)$

사용자 요청 시간부터 서비스를 받기까지의 시간으로 $q_{du}(s) = T_{process}(s) + T_{transmission}(s)$ 이다.

- 평판 $q_{rep}(s)$

서비스 s를 사용한 사용자에 의해 평가되어진 $q_{rep} = \frac{\sum_{i=1}^{n} Ri}{n}$ 이며, R_i는 마지막 사용자에 의해 평가되어진 평판이다.

- 트랜잭션 $q_{tx}(s)$

트랜잭션의 값은 0 이나 1로써, 1은 웹서비스가 트랜잭션을 제공한 것이고, 0은 서비스 사용자의 기능적인 요구를 충족할 수 없는 경우이다.

- 배상 비율 $q_{comp}(s)$

서비스 제공자가 서비스를 유지할 수 없거나, 요청을 완료할 수 없을 때 사용자에게 상환해야하는 실행 가격의 비율이다.

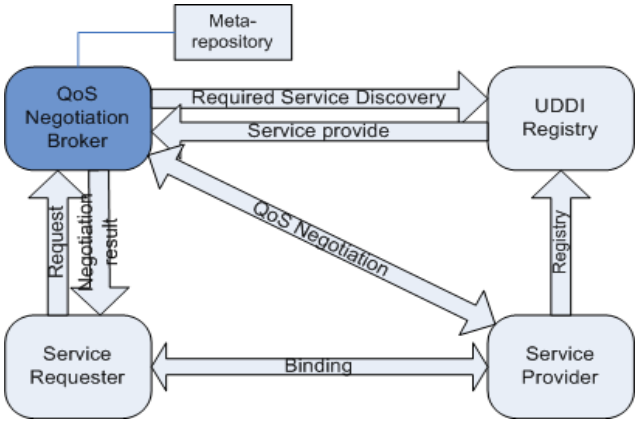
- 패널티 비율 $q_{pen}(s)$

서비스 사용자가 서비스를 계약을 위반하여 취소했을 시 적용되는 패널티 비율이다.

3. QoS 협상 브로커 설계

3.1 QoS 협상 브로커 아키텍처

QoS 협상 브로커는 서비스 사용자의 QoS 요구사항을 고려하여 최적의 웹 서비스를 선택할 수 있는 메커니즘을 제공한다.



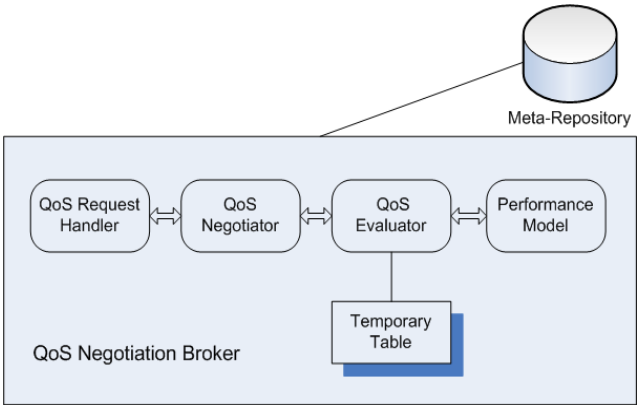
(그림 3) QoS 협상 브로커 아키텍처

QoS 협상 브로커는 UDDI에 등록된 서비스를 가지고 있을 수도, 가지고 있지 않을 수도 있다. 이는 이미 기존에 협상을 하여 사용자가 서비스를 선택한 경우, 그 서비스 제공자에 대한 정보가 QoS 협상 브로커 내에 있는 메타 저장소에 저장이 되어 있을 것이고, 협상된 내용이 없는 경우 UDDI 레지스트리에서 정보를 가져온다.

서비스 사용자는 요구된 QoS 조건에 맞는 서비스를 찾기 위하여 UDDI에 접속하여 검색하는 대신 QoS 협상 브로커에 접속한다. QoS 협상 브로커에 현재 이용 가능한 정보가 없을 시, UDDI에 등록되어진 서비스에 대한 정보를 검색한다. 이 결과로 전달되어진 WSDL 파일을 통해 새로운 QoS 협상 브로커에 리스트가 생성되어지고, 이 리스트를 통해 사용자를 위한 최적의 웹 서비스를 추출하게 된다. 이 추출된 서비스가 사용자의 요구조건에 맞아 사용자가 선택하게 되었을 때 제공된 서비스와 협상된 내용이 브로커 내의 메타 저장소에 저장된다.

3.2 QoS 협상 브로커

QoS 협상 브로커는 기존 논문에서 제시한 추가적인 QoS 컴포넌트를 통한 방법의 알고리즘을 보다 확장하여 적용하되, 각 제공되는 서비스마다 추가적인 컴포넌트를 부여하는 것이 아니라 하나의 브로커를 따로 두어 그 기능을 수행함으로써, 전체적인 시스템의 복잡도를 대폭 감소시켜 보다 빠르고 정확하게 수준 높은 서비스를 제공할 수 있다.



(그림 4) QoS 협상 브로커

3.2.1 QoS 요청 핸들러 (QoS Request Handler)

세션의 시작을 요청하고 협상 수행 후 세션의 종료를 알리고, 서비스 사용자에게 협상을 하기위한 요청이 인정, 협상된 값인 Count Offer 또는 거절되었는지에 대한 결과를 서비스 사용자에 알려주며, 종료 될 때 임시적인 저장장소인 테이블의 내용을 삭제하는 역할을 한다.

QoS 서비스 요청인 p는 Q-컴포넌트와의 세션 시작을 위해 사용되어진다.

$p=(r_{id}, S_{id}, N, R_{MAX}, X_{MIN})$

- r_{id} : 서비스 요청에 의해 생성된 요청의 ID
- S_{id} : 세션 내 실행되는 서비스의 ID
- N : 동시성 레벨의 최대 수
- R_{MAX} : 세션 내 최대 평균 응답시간
- X_{MIN} : 세션 내 최소 처리량

3.2.2 QoS 협상자 (QoS Negotiator)

동시 발생하는 요청 수준 N을 서비스가 충족할 수 없을 때, 그 값을 증가 또는 감소하여 평가자에 의해 평가하여, 재요청되는 Counter Offer를 생성한다.

Current and other requests are satisfied				Accept
Reason	**Remedy**	**Current**	**Others**	**Decision**
Only MAXR is Violated	Decrease N	OK	OK	Counter Offer
		MINX is violated or N=0	OK	Reject
Only MINX is Violated	Increase N	OK	OK	Counter Offer
		MAXR is violated	OK	Reject
MINX and MAXR are violated	Decreasing N reduces X and increasing N increases R. So, there is no solution			Reject

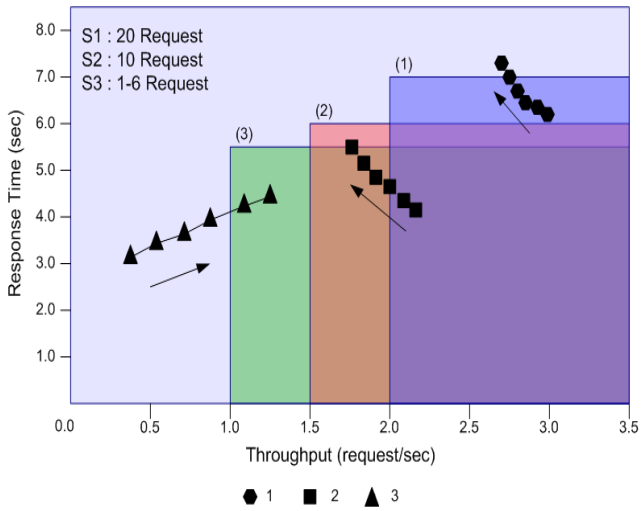
<표 1> 평가자에 의한 결정테이블

3.2.3 QoS 평가자 (QoS Evaluator)

QoS 협상자(Negotiator)에 의해 평가된 요청의 내용을 테이블에 저장하며, <표 1>의 결정테이블과 같이 QoS 목적에 맞는지 평가하게 된다. 이때 퍼포먼스 모델을 참조한다.

요청이 평가되는 매 순간, 퍼포먼스 모델을 참조하며 이 모델은 세션을 포함해서 요청이 평가되어지기 위한 응답시간, 처리량 같은 QoS 조건의 값을 예측하기위해 사용된다.

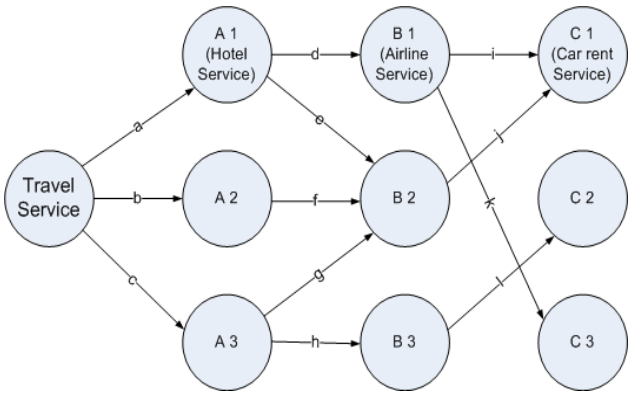
3.3 동시성 수준의 요청 수 협상



(그림 5) 동시성 수준의 최적 값 선택

(그림 5)는 3개의 서비스를 갖는 서비스 S에 대해 서비스 사용자가 요청을 하였을 경우, 서비스 S의 각 서비스의 QoS 범위를 모두 만족시키는 동시 발생되는 요청 N의 값을 협상하여 N의 수가 5일 때 모든 서비스를 만족시킬 수 있음을 보여준다.

3.4 QoS 협상 브로커의 예



(그림 6) 서비스 검색 경로

숙박, 항공편, 자동차대여 서비스를 포함하는 여행 사이트의 경우, 고객은 각각의 서비스를 따로 의뢰하는 것이 아니라, 여행 사이트를 통해서 모든 서비스들을 제공받을 수 있다. 어떤 고객의 관심은 가격에, 다른 고객은 품질 또는 시간에 중점을 두는 것과 같이 사용자의 요구는 다양하다.

QoS 협상 브로커는 한 여행 사이트가 가지는 복잡하게 얽힌 서비스들의 경로를 탐색, 사용자 요구에 가장 적합한 서비스 경로를 추출하게 된다. 또한 인기 있는 서비스 경로의 경우, request의 증가로 인한 지연으로 예약시간을 중시하는 고객의 요구를 만족하기 어렵게 되었을 때 이를 분석하여 현재 상황에 가장 적합한 경로를 협상하게 된다.

4. 결론 및 향후과제

논문에서 제시한 QoS 협상 브로커는 랭크된 서비스에 비해, 각각의 서비스 사용자에 맞는 최적의 서비스 QoS 수준을 자동적으로 찾을 수 있다. 일반적인 컴포넌트에 추가적 컴포넌트인 Q-컴포넌트를 추가하지 않고, 하나의 브로커가 대신하기 때문에, 시스템의 비중이 적어지고, 복잡도를 간소화할 수 있다.

또한 협상한 기록을 QoS 협상 브로커 내에 있는 메타저장소에 저장하여, 같은 요청이 들어올 경우 참조할 수 있게 하여 협상과정에서 발생하는 지연 등을 최소화하여 서비스 선택을 용이하게 한다.

향후 연구과제로는 QoS 협상 브로커 실행 모델의 실제적인 설계로 아직은 추상적인 개념을 완성도 있는 시스템으로의 구현이다.

참 고 문 헌

[1] Anbazhagan Mani, IBM Software Labs “For Web Service QoS, http://www-128.ibm.com/developerworks/kr/webservices/library/ws-quality.html, January 26, 2002.

[2] "A Framework for QoS-Aware Software Components," Daniel A. Menascé, Proc. 2004 ACM Workshop on Software and Performance, San Francisco, CA, January 14, 2004.

[3] "Quality of service: QoS computation and policing in dynamic web service selection", Yutu Liu, Anne H. Ngu, Liang Z. Zeng, May 2004, ACM