분산 이질 환경에서의 포인트 브로커의 설계 및 구현*

유병석°, 강미연, 김형식
충남대학교 컴퓨터과학과
{adoregnu, herbt, hskim}@cs.cnu.ac.kr

Design and Implementation of a Point Broker on Distributed Heterogeneous Environment

Byeongseok Yoo, Miyun Kang, Hyong-shik Kim
Department of Computer Science
Chungnam Nation University

요약

본 논문에서는 분산 이질 환경에서도 적용가능한 포인트 공유 모델을 제안하고 포인트 교환 기능을 갖는 포인트 브로커를 설계, 구현하였다. 머천트 서버와 브로커간 뿐만 아니라 브로커와 브로커간에도 동일한 방법으로 포인트 공유 기능을 지원하도록 함으로써 분산이질 환경에서도 별도의 조작이 불필요하다. 제안된 포인트 브로커는 SOAP 프로토콜에 기반한 웹서비스를 이용하여 구현되었고 교환율 관리, 포인트 정산 등의 기능도 제공한다.

1 서론

최근들어 각 기업들은 다른 기업들과 포인트 서비스를 공유함으로써 자사의 고객을 확보하기 위해 노력하고 있다. 이러한 포인트 서비스는 꾸준히 자사의 서비스를 이용한 고객에게 포인트를 부여함으로써 일정량의 포인트가 축적되었을 경우 선물을 주는 것을 모태로 한다. 소비자 입장에서는 당연히 공짜 개념과 같은 이 포인트를 얻기 위해 동일한 질과 가격의 제품이 있을 때 포인트 서비스를 시행하고있는 기업의 물건을 구입하게 될 것이다. 그로써 포인트 서비스를 도입한 기업은 자연스럽게 충성도가 높은 우수고객을 보할 수 있을뿐만 아니라 매출도 증대되는 이득을 얻게 될것이라 예상된다. 고객들은 별다른 노력없이 특정기업의 상품을 구매함으로써 현금과 동일한 가치를 지니는 포인트를 축적하게 되며 이를 가지고 쇼핑과 같은 제 2의 소비활동을 할 수 있게 되는 것이다.

소비자들은 한 기업에서만 상품을 구매하는 것이 아니라 여러 기업에서 상품을 구매하는 반면, 각 기업마다 제공되는 포인트 서비스에서의 포인트 적립율이 크지 않아 실제 포인트의 사용율은 적은 편이다. 따라서 분산되어 있는 포인트를 공유할 수 있도록 허용함으로써 소비자들의 만족도를 높이고 고객유인 효과를 증대하기 위한 방법들이 제안되어 사용되고 있다.

2 포인트 공유 모델 분석

2.1 중앙 집중형

하나의 포인트 공유 시스템만이 존재하며, 사용자 신상 정보나 누적 포인트 정보는 포인트 공유 시스템에 중앙 집중적으로 저장된다. 각 머천트 서버들은 어떠한 사용자 정보도 가지고 있지 않으며 유일하게 사용자를 인증하는 방법은 포인트 공유 시스템에서 발급한 카드에대한 일련번호이다. 그림 1과 같은 구조로 되어 있으며 대표적인 예로 (주)SK에서 운영하는 OKCacheBag[1]이 존재 한다.

2.2 분산형

주로 온라인 회원사 서버들이 포인트 공유 시스템을 이용하여 포인트를 공유하는 방법으로 사용자 신상정보와 포인트 정보는 각 머천트 서버에 저장되며 필요에 의해 포인트 공유 시스템에도 저장될 수 있다. 물론 사용자는 각각의 머천트 서버의 포인트를 통합하기 위해서 각 머천트서버에 회원 가입을 해야 하며 또한 각각에서 포인트를 적립해야 한다.

포인트 공유 시스템은 각 머천트 서버에 누적된 포인트를 교환하여주거나, 포인트 공유 시스템 자체의 포인트로 교환시켜 자체내의 쇼핑몰이나 유료 컨텐츠를 이용할 수 있도록 하고 있다. 현존하는 대표적인 예로서 포인트뱅킹[2]이나 포인트파크[3] 등을 들 수 있다.

* 본 논문은 한국과학재단 지정 지역협력연구센터(RRC)인 충남대학교 소프트웨어 연구센터의 지원으로 수행된 과제의 결과임

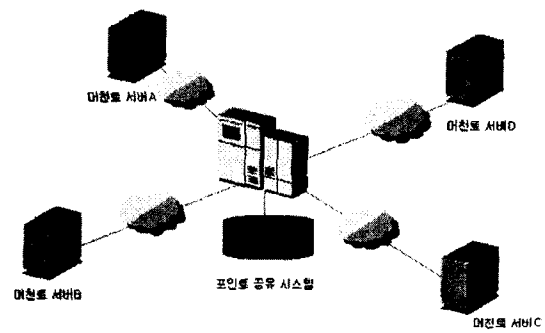

그림 1: 중앙 집중형 포인트 공유

2.3 각 포인트 공유 모델의 비교

위에서 정리한 모델을 정리하여 표로 나타내면 표 1과 같다.

종류	중앙집중형	분산형
사용자 정보	포인트 공유 시스템	각 머천트서버, 포인트 공유 시스템
포인트 정보	포인트 공유 시스템	각 머천트서버, 포인트 공유 시스템
교환율	존재하지 않음	포인트 공유 시스템
단위	통합 포인트	머천트 서버 마다 다양
예	OKCachBag	PointBanking, PointPark

표 1: 현존하는 포인트 공유 모델 분석

2.4 현재 포인트 공유 시스템의 문제점

각 머천트 서버와 포인트 공유 시스템의 문제점은 각 기업에서 제공하는 포인트의 적립률이 다르고 각 기업의 포인트의 절대적인 가치가 서로 다르다는 데 있다. 또한 대부분 기업의 포인트 시스템이 서로 상이한 환경에서 구현되어 있기 때문에 부가적인 노력이 필요하다.

중앙 집중형 포인트 공유 시스템은 사용자 정보가 한곳에 집중되어 공유시스템에 문제가 생기면 모든 회원사가 포인트 시스템을 사용할 수 없다. 분산형 포인트 공유 시스템은 이러한 경우는 방지할 수 있지만, 사용자 정보와 포인트 정보가 여전히 브로커에 저장되어 있어 브로커의 포인트는 사용할 수 없는 상황이 발생할 수 있다. 또한 사용자들은 자신이 가입한 회원사 이외에 포인트 브로커

에도 가입해야만 다른 회원사의 포인트를 공유할 수 있는 문제를 발생시킨다.

3 포인트 브로커의 설계

3.1 용어의 정의

- 포인트 : 머천트서버 별로 고객을 위해 제공하는 일정액의 마일리지
- 머천트 서버 : 자사 포인트 공유를 위해 제휴 과계에 있는 기업들
- 포인트 브로커 : 각 머천트 서버에서 제공하는 포인트를 중계하기 위한 매개자

3.2 포인트 공유 시스템의 구조

포인트 공유 시스템의 구성 요소는 포인트를 전달해주기 위한 브로커 시스템과, 포인트의 입출금에 대한 메소드를 제공하는 머천트 서버측의 웹서비스이다.

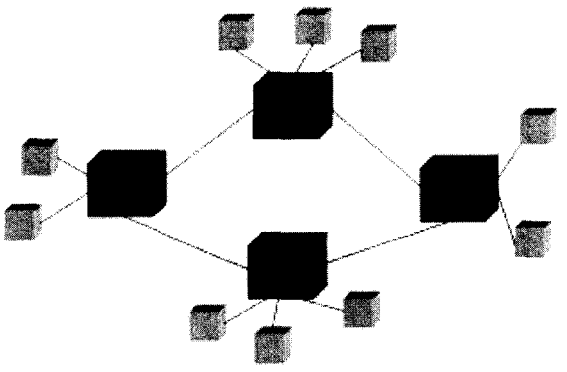

그림 2: 포인트 브로커 시스템의 구조

그림 2에서 보는 바와 같이 B_A, B_B, B_C, B_D등의 브로커는 머천트 서버 이외에 타 브로커와도 포인트를 교환할 수 있으며, 이 때 타 브로커는 일반적인 머천트 서버처럼 취급된다. 머천트 서버는 자신과 포인트를 교환하는 브로커 이외의 브로커의 존재를 알 필요가 없다. 그렇지만 브로커를 통해 다른 브로커에 속해있는 머천트 서버와 포인트를 공유 할 수 있게된다.

3.3 포인트 교환 메카니즘

3.3.1 역할 분담

2절에서 살펴본 포인트 공유 시스템들은 사용자와 포인트 정보가 포인트 브로커에도 저장되어 있어 회원사와 브로커 간에 독립성을 제공하지 못하였다. 그래서 브로커의 문제는 모든 머천트 서버에 영향을 미치게 되며, 브로커와 브로커로의 확장도 용이하지 않다. 이와 같은 문제점들을 해결하기 위한 본 논문에서 제안하는 포인트 브로커의 특징은 크게 두가지로 나눌 수 있다.

- 회원들과 포인트 정보는 머천트 서버에만 저장된다.
- 브로커는 포인트를 전달하는 역할만 한다.

따라서 머천트 서버의 회원들은 포인트 공유 브로커에 다시 가입할 필요가 없다. 포인트 공유 브로커에 가입되어 있지 않은 머천트 서버도 기존의 포인트 서비스에 포인트 공유 브로커와 통신을 위한 간단한 웹서비스를 추가함으로서 쉽게 포인트 공유 브로커에 통합될 수 있다. 브로커가 포인트 정보를 가지고 있지 않으므로 각 회원사는 브로커에 문제가 있어도 정상적인 포인트 서비스가 가능하다.

3.3.2 머천트 서버간 포인트 교환 방법

2.4절에서 살펴본 바와 같이 포인트의 절대적인 가치는 머천트 서버마다 다르다. 그래서 머천트 서버 사이에서 포인트가 교환될 때 각 머천트 서버에 맞는 적절한 단위로 변환(결제)되어야 한다. 이를 위해 포인트 브로커는 각 머천트 서버의 포인트 교환율 정보를 유지하게 된다. 교환율은 머천트 서버에서의 포인와 브로커 사이의 비율이며 브로커에서 머천트 서버로 교환될 때에는 역수가 적용된다.

일반적인 경우에는 브로커를 통해 교환될 때 수수료를 공제한다. 본 논문에서는 설명의 편의를 위해 수수료에 대한 적용은 생략하기로 한다.

M_A와 M_B라는 머천트 서버가 같은 브로커에 있고, 브로커에서 교환율이 각각 0.9, 1.5일 경우, 어떻게 포인트 교환이 일어나는지 살펴 보자.

- M_A에서 M_B로 포인트 100을 보낼 경우
  100 포인트는 브로커에서 M_A의 교환율 0.9가 적용되어 100×0.9 포인트로 변환된다. 브로커에서 M_B로 교환될 때 $(100 \times 0.9) \times \frac{1}{1.5}$ 만큼의 포인트가 M_B로 보내지게 된다.
- M_A에서 M_B의 포인트 100을 가져올 경우
  우선 M_B의 포인트 100을 포인트 브로커로 가져온다. 이때 M_B에 교환율 1.5를 곱하여 100×1.5가 된다. 이 포인트는 브로커에서 M_A로 이동하는 것이므로 M_A의 교환율의 역수를 곱하여 $100 \times 1.5 \times \frac{1}{0.9}$ 만큼의 포인트가 M_A로 보내지게 된다.

3.3.3 브로커를 경유한 포인트 교환 방법

브로커와 브로커사이에도 교환율 정보가 있어야 한다. 예를 들자면 한국에 있는 브로커에 속해있는 머천트 서버가 미국의 브로커에 속해있는 머천트 서버로 포인트를 보내는 경우가 이에 해당한다. 이 경우 한국의 브로커와 미국의 브로커 사이에는 포인트의 가치가 다르게 취급될 수 있으므로, 포인트 교환이 발생될 때 브로커를 일종의 머천트 서버로 간주하여 교환과정을 단순화 할 수 있다. 다음 예를 보자.

M_A라는 머천트 서버는는 B_A라는 브로커에 속해 있고, M_C라는 머천트 서버는 B_C라는 브로커에 속해 있다고 하자. M_A의 B_A에서의 교환율은 0.9이며, M_C의 B_C에서의 교환율은 1.5라 하자. B_A에서 B_C로의 교환율은 0.8이며, B_C에서 B_A로의 교환율은 1.2라 하자. B_A에서 B_C로의 교환율과 B_C에서 B_A로의 교환율의 곱은 1이하여야 한다.

- M_A에서 M_C로 포인트 100을 보낼 경우
  우선 M_A의 포인트는 B_A에서 M_A의 교환율을 적용받아 100×0.9으로 바뀐후 B_A에서 B_C로의 교환율을 적용받아 $100 \times 0.9 \times 0.8$만큼의 포인트가 보내진다. B_C에서는 M_C의 교환율을 적용하여 $100 \times 0.9 \times 0.8 \times \frac{1}{1.5}$ 만큼의 포인트가 M_C로 보내진다.
- M_A에서 M_C의 포인트 100을 가져 오는 경우
  단일 브로커일 때와 마찬가지로 M_C에서 100을 가져오면 B_C에서의 통합 포인트는 100×1.5가 된다. B_A로 갈때에는 B_A에서 B_C로의 교환율은 0.8이므로, 그 역수를 곱한 $100 \times 1.5 \times \frac{1}{0.8}$이 B_A의 통합 포인트가 되며 M_A에서 B_A로 갈때의 교환율은 0.9이므로 최종적으로 M_A는 $(100 \times 1.5 \times \frac{1}{0.8} \times \frac{1}{0.9})$ 만큼의 포인트를 받게 된다.

위에서의 설명처럼 브로커를 경유한 포인트교환도 머천트 서버간의 교환과 동일한 형태로 동작한다.

3.4 API

3.4.1 포인트 브로커

포인트를 머천트서버에서 다른 머천트 서버로 이전하기 위한 인터페이스를 제공해야 한다. 주요 API는 다음과 같다.

- int PointQuery(cid,uid,upasswd)
  머천트 서버(cid)에 속한 사용자(uid)의 포인트를 조회
- int PointDeposit(fcid,fuid,tcid,tuid,amount)
  머천트 서버(fid)의 회원(fuid)이 다른 머천트 서버(tcid)의 회원(tuid)으로 포인트(amount) 이전
- int PointWithdraw(fcid,fuid,fupasswd,tcid,tuid,amount)
  머천트 서버(fcid)의 회원(fuid)에게서 다른 머천트 서버(tcid)의 회원(tuid)으로 포인트(amount)를 가져옴
- string[][] GetCompanyInfo()
  브로커에 속해있는 머천트 서버의 정보를 수집하여야 할 경우에 사용된다. 2차원 배열 형태로 반환하며, 행의 개수는 관계된 브로커의 수이며 열은 브로커에 포함된 머천드 서버들의 목록이다.

3.4.2 머천트 서버

머천트서버에서는 브로커가 포인트를 입출금하기위한 인터페이스를 제공해야 한다. 주요 API는 다음과 같다.

- int PointQuery(uid,upasswd)
  회원(uid)의 포인트를 조회한다.
- int PointTake(fcid,fuid,tuid,tupasswd,amount)
  회원(tuid)에게서 포인트(amount)를 가져온다. 포인트 출금을 호출한 머천트 서버(fcid)와 회원(fuid)의 아이디는 로깅을 위한 목적으로 전달된다.
- int PointPut(fcid,fuid,tuid,amount)
  회원(tuid)에게 포인트(amount)를 이전한다. fcid, fuid는 PointTake와 마찬가지 목적으로 사용된다.

3.5 데이터 흐름

실제로 어떻게 포인트가 전달 되는지 그림을 통해서 살펴보자. 먼저 머천트 서버는 포인트를 입금할 머천트 서버를 지정해야 한다. 사용자가 다수의 머천트 서버들 중에 선택할 경우에는 GetCompanyInfo가 브로커에 연결된 머천트 서버를 포함한 모든 머천트 서버 혹은 조건을 만족시키는 일부 머천트 서버들에 관한 정보를 제공할 수 있다.

포인트를 입금할 머천트 서버가 결정되면 포인트 브로커의 PointDeposit 매소드를 통해 포인트가 입금되고 나머지 과정은 브로커에 의하여 머천트 서버에 투명한 방법으로 처리된다. 브로커는 머천트 서버를 대신하여 입금될 머천트 서버의 위치에 따라 PointPut을 이용하여 포인트를 목적 머천트 서버에 직접 입금할 수도 있고, 다른 브로커에게 위임하기 위하여 PointDeposit을 호출할 수도 있다. 출금과정도 입금의 과정과 크게 다르지 않고, PointDeposit 대신에 PointWithdraw, PointPut 대신에 PointTake를 호출한다.

그림 3과 같은 형태로 포인트가 이동하게 된다. 여기서 주목할 것은 M_A에서 M_C로 포인트가 이동하는 경우나 M_A에서 M_B로 포인트가 이동하는 경우나 M_A의 입장에서는 크게 다르지 않다는 점이다. 머천트 서버의 입장에서는 B_B가 보이지 않기 때문에 B_A가 M_A와의 중개 기능도 처리 하는 것처럼 보인다.

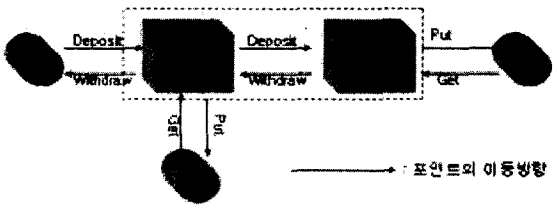

그림 3: 멀티 브로커 환경에서 포인트 이전

4 포인트 브로커의 구현

4.1 웹 서비스

포인트 서비스를 하는 머천트 서버들은 서로 다른 환경에서 구현되어 있을 수 있다. 이들 머천트 서버와 포인트 공유 브로커 사이의 통합을 위해서는 서로 다른 환경 사이에서도 적용 가능한 인터페이스가 제공되어야 한다.

웹 서비스는 플랫폼이나 언어에 상관없이 자유롭게 개발자가 좋아하는 기술을 선택할 수 있도록 해준다. 각 서비스에서 제공되는 것은 인터페이스일 뿐이며, 클라이언트 들은 HTTP 등의 프로토콜 위에 있는 SOAP을 통해 서버에 접속할 수 있게된다.

SOAP은 HTTP, SMTP 등의 프로토콜로 보내질 수 있다. 대부분의 방화벽은 인터넷을 통한 HTTP, SMTP 접속을 허용하므로 웹 서비스는 방화벽에서도 잘 작동한다. 또한 SOAP 메시지는 XML로 이루어진 텍스트 정보이므로 이러한 프로토콜 이외에 다른 프로토콜을 사용하더라도 통신이 가능하다.

4.2 인증 메카니즘

포인트 브로커와 머천트 서버사이에서 포인트를 교환할 때, 머천트 서버가 확실히 브로커의 회원이며 믿을만한가를 검증해야 한다. 이를 위해 머천트 서버가 포인트 브로커에 회원으로 등록할 때 머천트 서버 고유의 아디와 암호를 제공한다. 머천트 서버의 아이디와 암호는 SOAP 메시지의 헤더부분에 포함된다.

본 논문에서 웹 서비스는 HTTP 프로토콜을 이용하므로 요청에 대한 응답이 끝나면 연결이 끊어지게 된다. 따라서 매 요청마다 신뢰할 수 있는 머천트 서버에서 왔는지 검사해야 한다.

4.3 교환율 관리

3.3절에서 살펴본 바와 같이 포인트를 교환하기 위해서는 교환률이 필요하다. 교환율을 적용하는 방법에는 두가지가 있다. 첫째는 머천트 서버가 브로커에 가입된 다른 머천트 서버와의 교환율을 모두 보유하는 방법이다. 두번째는 브로커에서 통용되는 통합 포인트를 사용하는 방법으로, 머천트 서버는 다른 머천트 서버와의 교환율을 알 필요가 없고 브로커에서만 각 머천트 서버의 교환율을 보유하고 있으면 된다.

첫번째 방법의 경우 이미 운영되고 있는 머천트 서버에 다른 머천트 서버에 대한 교환율 정보를 추가해야 하며, 새로운 머천트 서버가 추가 될 때마다 그 데이터를 갱신해야 한다. 그러나 두번째 방법의 경우, 각 머천트 서버와 브로커의 교환율만 브로커에 저장되므로 머천트 서버에 교환율에 관한 정보를 추가 할 필요가 없다. 또한 새로운 머천트 서버가 추가되더라도 다른 머천트 서버들이 그 사실을 알지 못해도 포인트를 교환하는데 전혀 문제가 없다.

본 논문에서는 3.3절에서의 설명과 같이 두번째 방법을 사용하여, 각 머천트 서버의 포인트는 브로커에서 통합 포인트로 변경되어 다른 브로커로 이동하는 방법을 사용하였다.

4.4 포인트 정산

회원입장에서는 단순히 포인트를 한 머천트 서버에서 다른 머천트 서버로 옮겼을 뿐이지만 머천트 서버의 입장에서 보았을 때는 포인트에 대한 의무가 아무런 댓가없이 다른 머천트 서버로 옮겨간 것이된다. 따라서 각 머천트 서버사이에서는 이동한 포인트만큼의 가치에 대한 정산이 되어야 한다.

이를 위해 브로커는 머천트 서버별로 정산을 위한 정보를 유지해야 한다. 브로커는 각 머천트 서버별로 이루어진 포인토의 양을 표2 와 같은 2차원 형태의 배열을 유지하여 후에 정산에 필요한 정보를 머천트 서버에게 제공한다.

	M_A	M_B	M_C	...	B
M_A		50	-20	...	-60
M_B	-50		-30	...	40
M_C	20	30		...	70
...	...	...	...	...	...
B	60	-40	-70	...	...

표 2: 머천트 서버간 포인트 정산 내역

M_A, M_B, M_C 등은 머천트 서버를 나타 내며 숫자들은 그들 간에 교환된 포인트를 나타낸다. B는 브로커 자신을 표시하며, 다른 브로커와 연결될 때에는 해당 브로커를 포함해야 한다.

5 결론

최근 전자 상거래의 급속한 성장으로 각 전자상거래 사이트들은 경쟁적으로 포인트 서비스를 제공하고 있다. 본 논문에서는 이렇게 분산되어 있는 포인트를 공유하기 위한 포인트 브로커의 프로토타입을 제시하였다. 제시된 포인트 공유 브로커는 머천트 서버에 관한 최소한의 정보를 이용하여 사용자간 포인트 교환기능을 제공하며 서로 다른 환경에서의 머천트 서버도 동일한 방법으로 동작하도록 하였다. 또한 간단한 구조로 되어 있어 현재 서비스의 방식을 크게 바꾸지 않고 쉽게 적용할 수 있다.

References

[1] http://okcashbag.com.

[2] http://pointbanking.co.kr.

[3] http://pointpark.co.kr.

[4] 강미연, 개방 분산 환경에서의 포인트 공유 브로커의 설계 및 구현, 충남대학교 석사학위 논문, 2002년 12월.