토픽 기반 메시지 시스템의 설계 및 구현

김경곤[0] 한인
(주)뱅크타운 부설기술연구소 시스템체계연구실
{kaddol[0] , ihan}@banktown.com

Design and Implementation of Topic-Based Message System

Kyung-Kon Kim[0] In Han
Dept. of R&D Center, Banktown Inc.

요 약

본 논문은 기업 및 개인간의 정보 교환을 지원하는 토픽 기반 메시지 시스템을 구현한다. 기존의 등록(Subscribe)/발표(Publish) 메시지 시스템과 동일한 기능을 수행하는 등록(Subscribe)/발표(Post) 메시지 시스템으로 메시지를 처리하는 서버인 메시지 처리기와 ActiveX 컨트롤을 사용한 클라이언트로 구성된다. 메시지 처리기는 운영체계에 의존하지 않고 독립적으로 작동하고, 클라이언트는 Win32용과 웹 기반으로 구현하여 특정 운영체계에 국한되지 않고 웹 기반을 포함한 다양한 환경에서 사용 가능하도록 가용성을 높였다.

1. 서 론

기존에 존재하던 메시지 시스템은 기업간 거래나 개인간의 정보교환을 위하여 분산 환경에서 소프트웨어 프로세스간 데이터 교환을 원활히 하는데 목적이 있다. 최근 인터넷 사용자 수가 급증함에 따라 메시지 시스템에 동시에 접근하는 클라이언트 수가 대규모화되고 있기 때문에 이를 지원하는 연결 기술의 필요성이 대두되고 있다. 또한, 인터넷의 확산 및 활용 증가와 함께 기업간, 개인 기업간의 정보 교환이 빈번하게 이루어지고 있으며 이러한 정보 교환을 지원하는 메시 지 시스템의 요구도 점점 증대되고 있다.

본 논문에서 제시하고 있는 토픽 기반 메시지 시스템은 이러한 요구사항을 만족하는 메시지 시스템으로서 기존의 등록/발표 방식과 동일한 기능을 수행하는 등록/발표 방식을 제공하고 있다. 토픽(Topic)이라는 가상 채널을 통해 메시지 처리기와 클라이언트간의 메시지 교환이 이루어진다.

2장에서 토픽 기반 메시지 시스템의 개념 및 기존의 메시지 시스템에 대한 관련 연구를 기술하고, 3장에서 본 논문에서 제안하는 토픽 기반 메시지 처리기에 대해서 설명한다. 4장에서 토픽 기반 메시지 시스템을 이용하여 구현한 예를 기술하고, 5장에서 결론 및 향후 과제에 대해서 논한다.

2. 관련 연구

토픽 기반 메시지 시스템은 조직간 실시간으로 정보를 공유하고 비즈니스의 처리를 원활하게 하는데 효율적인 시스템이다. 토픽 기반 메시지 시스템은 토픽이라는 가상 채널을 통해 메시지를 주고 받는다. 즉, 메시지 처리기는 다수의 토픽을 유지하고, 임의의 클라이언트는 자신이 수신하고자 하는 메시지와 관련이 있는 토픽에 등록하여 해당 토픽에 발표되는 모든 메시지들을 수신하게 된다. 하나의 컴퓨터는 하나 이상의 토픽에 등록과 발표를 할 수 있다. 특수한 경우로 하나의 노드는 토픽에 등록만을 하거나 발표만을 할 수 있도록 하는 경우도 있다. 다음 [그림 1]은 토픽 기반 등록/발표 메시지 시스템의 전체적인 개념도이다.

[그림 1]에서 발표자1이 메시지 서버로 메시지를 발표하면 토픽1에 등록한 등록자1과 등록자 2가 메시지를 수신하게 된다.

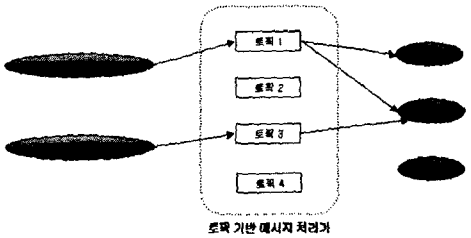

그림 1 토픽 기반 메시지 시스템 개념도

만약 발표자2가 메시지를 전송하게 될 경우는 토픽2에 등록을 한 등록자2만이 메시지를 수신하게 된다. 등록자3은 어떠한 토픽에도 등록을 하지 않았기 때문에 메시지 서버로부터 전송되는 어떠한 메시지도 수신할 수 없다. 이러한 특징으로 메시지 처리기와 클라이언트간의 대화식의 메시지 교환이 가능하다.

본 논문에서 제시하고 있는 토픽 기반 메시지 시스템과 비교를 해보기 위해 기존의 존재하던 메시지 시스템인 KnowNow[1,2], Kenamea[3,4], Elvin[5]에 대해 기술을 한다.

KnowNow(이하 KN)는 KN Event Router라는 메시지 서버와 KN Microserver라는 클라이언트 프로그램을 이용하여 대화식의 메시지 교환을 한다. KN Microserver에는 인스톨이 필요하지 않은 Javascript Microserver, Win32 및 IIS를 지원하는 Windows Microserver, C나 C++을 지원하는 C Microserver, 자바 환경에서 사용할 수 있는 Java Microserver, PDA에서 사용할 수 있는 Pocket PC Microserver가 있다. 기존의 웹 환경에서는 웹 페이지에 원하는 모든 정보를 모두 입력하여 HTTP 데이터로 전송하여 오류가 없다는 전체 데이터의 검증을 받은 후에 다음 페이지로 이동할 수 있도록 되어 있다. 하지만 KN에서는 하나의 입력필드라도 데이터가 오류가 발생하면 즉시 다시 입력하게끔 하여 불필요한 HTTP 통신의 수를 줄일 수 있을 뿐만 아니라 웹 페이지의 변화가 아닌 입력필드의 변화에 대응하게끔 되어 있어 실시간으로 처리될 수 있다. Win32 응용 프로그램과 웹 환경으로 개발된 프로그램과의 원활한 메시지

교환으로 웹 환경에 국한되지 않고 다양한 환경에서 사용할 수 있게 하였고, KN Javascript Microserver는 웹 서버와 HTTP 연결의 단절로 인한 여러 문제점들을 보완해 주면서 인스톨이 필요하지 않은 제로-인스톨(Zero-Install)의 특징을 가지고 있다. 또한, 메시지 교환시 서로 호환이 잘 되도록 하기 위해서 표준 데이터 포맷인 XML, SOAP 메시지 등과 같은 메시지를 사용하고 있다.

KN와 유사한 동작방식을 가지고 있는 Kenamea는 안전하고 확장성 있는 메시지 라우팅 서비스를 제공하는 메시지 서버인 Message Switch와 Connector라는 클라이언트간에 메시지 교환을 한다.

Elvin은 토픽 기반 메시지 시스템과는 달리 내용(Content) 기반 메시지 시스템에 해당한다. 즉, 주고 받는 메시지 내용에 있는 데이터를 메시지 서버가 체크하여 동일한 데이터라는 것이 확인되어진 클라이언트들에게만 메시지 교환이 이루어지도록 한다는 것을 의미한다. 메시지 서버인 Elvin Server와 클라이언트인 Elvin Client로 구성된다.

KN나 Kenamea는 안정적인 메시지 라우팅 서비스를 지원하지만 이미 패키지화 되어서 비용면에서 부담이 있고, Elvin의 경우 독특한 elvin이라는 프로토콜을 주로 지원하기 때문에 범용성에서는 다소 문제가 있다.

3. 토픽 기반 메시지 시스템

본 논문에서 제안하는 토픽 기반 메시지 시스템은 토픽이라는 가상채널을 가지고 메시지를 중계해 주는 메시지 처리기와 토픽에 등록하고 메시지를 발표하는 ActiveX 형태의 클라이언트들로 구성된다. 다음 [그림 2]는 토픽 기반 메시지 시스템의 구성도이다.

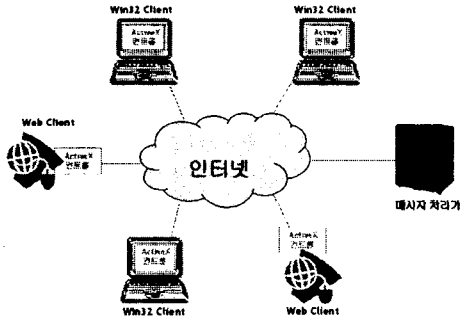

그림 2 토픽 기반 메시지 시스템 구성도

[그림 2]에서 메시지 처리기는 각 클라이언트들이 등록하고 발표를 가능하게 하는 가상채널인 토픽을 자료구조에 가지고 있으면서 메시지 교환이 가능하게 해준다. 클라이언트는 ActiveX를 포함하여 작성될 수 있는 웹 페이지 기반의 웹 클라이언트와 Win32 클라이언트가 있다. 클라이언트들은 모두 비동기(Asynchronous)통신을 하고 ActiveX 컨트롤을 통해 메시지를 송·수신한다. 이는 재사용 가능한 컴포넌트를 이용하여 응용 프로그램의 개발시간 단축과 프로그램 기능 및 품질향상에 기여한다. 또한, 토픽 기반 메시지 시스템의 네트워크 통신 프로토콜은 TCP/IP 프로토콜 상에서 이루어지고 믿을 만한 메시지 보장방법을 허용하기 위해서 소켓 통신과 쓰레드를 이용하도록 하였다.

토픽 기반 메시지 시스템에서 메시지 교환은 다음과 같은 절차에 의해 이루어진다. 클라이언트가 메시지 처리기에 접속하게 되면, 메시지 처리기에 하나의 토픽에 등록하고 해당 토픽에 전송하고자 하는 메시지를 발표하면 메시지 처리기는 이를 토픽에 등록된 모든 클라이언트들에게 전달하게 된다. 그 후에 해당 토픽으로 메시지 교환을 원하지 않을 경우에는 해당 토픽에서 해제(UnSubscribe)하면 된다. 다음 [그림 3]은 토픽 기반 메시지 교환 개략도이다.

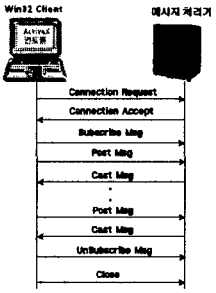

그림 3 토픽 기반 메시지 교환 개략도

[그림 3]을 통해 주요 연산은 4가지로 요약될 수 있다. 즉, 등록/발표/전달/해제(Subscribe/Post/Cast/UnSubscribe)이다. 등록이라 함은 클라이언트가 메시지를 교환할 가상채널인 토픽을 메시지 처리기에 등록해서 메시지 교환할 준비가 되었다는 것을 의미한다. 발표는 클라이언트가 같은 토픽에 등록된 모든 클라이언트로 메시지를 보내기 위해 메시지 처리기로 메시지를 보내는 것이고, 전달은 메시지 처리기가 클라이언트로부터 받은 메시지를 같은 토픽에 등록된 모든 클라이언트로 메시지를 보내주는 것이다. 해제는 특정 토픽에 등록된 클라이언트가 해당 토픽과의 발표/전달 동작을 멈춘다는 것을 의미한다. 이와 같은 동작을 통해 메시지 처리기와 클라이언트들간의 대화식의 메시지 교환이 일어나게 되는 것이다.

이와 같은 연산을 하기 위해서는 메시지 처리기와 클라이언트간의 전송 전문이 필요하게 된다. 4가지 동작의 전송 전문은 다음 [그림 4]와 같은 XML 형태를 가지게 된다.

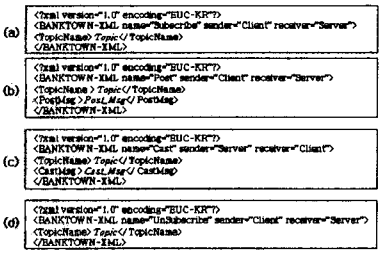

그림 4 토픽 기반 메시지 시스템 전송 전문

[그림4]의 전송 전문은 BANKTOWN-XML 엘리먼트의 name 속성 값에 따라 메시지 처리기에서 처리하는 4가지 주요 연산에 해당됨을 알 수 있다. TopicName 엘리먼트는 가상채널인 토픽에 대한 값이고, PostMsg/CastMsg는 발표/전달하는 실질

적인 데이터들이다. (a)는 Subscribe용 전문, (b)는 Post용, (c)는 Cast용, (d)는 UnSubscribe용 전문이다. 이와 같은 XML 전송전문을 주고 받으면서 메시지 처리기와 클라이언트간의 대화식의 메시지 교환이 일어난다.

4. 토픽 기반 메시지 시스템 구현

본 논문에서 제안하는 토픽 기반 메시지 시스템을 이용하여 인터넷의 많은 보급으로 실생활에 많이 이용되고 있는 채팅 시스템을 구현해 보았다. 토픽 기반 메시지 처리기는 파이썬(Python)으로, 클라이언트는 VC++로 구현된 ActiveX 컨트롤을 이용하여 VB와 자바 스크립트를 포함한 html로 구현해 보았다.

메시지 처리기에서 구현된 주요 메써드는 주요한 연산인 등록/발표/해제 등 세 가지로 구분될 수 있다. [그림 5]는 채팅 시스템의 각각의 연산의 메시지 교환 과정을 보여준다.

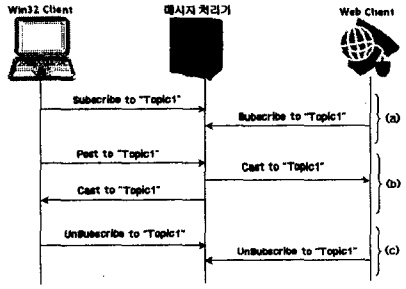

그림 5 채팅 시스템 메시지 교환

[그림 5]에서 (a)는 대화방 등록에 해당하는 부분이다. 채팅을 하려면 대화방이 존재하여 해당 대화방을 통해 이루어지는데, 클라이언트가 메시지 처리기에 등록 전문을 보내면 가상채널인 토픽값이 대화방으로 등록되어진다. (b)는 채팅을 서로 주고 받는 과정에 해당하는 부분이다. 한 클라이언트가 메시지 처리기에 채팅 내용을 보내면 메시지 처리기는 대화방에 등록된 모든 클라이언트로 채팅 내용을 보내주게 되어 실질적인 채팅이 이루어진다. (c)는 대화방 탈퇴에 해당하는 부분이다. 채팅을 더 이상 원하지 않을 때 해당 대화방에서 탈퇴하면 되는 것이다.

클라이언트에서 사용한 메써드와 이벤트는 [그림 6]과 같다. 여기서 EvpCli는 ActiveX 컨트롤의 이름이다.

```
[메써드]
EvpCli.Connect()
EvpCli.Subscribe RestMsg
EvpCli.Post txtMyMsg.Text
EvpCli.UnSubscribe
EvpCli.RequestMsg (strMsg)
EvpCli.Close()
[이벤트]
EvpCli_OnRecvXml(ByVal recvXml as string)
EvpCli_OnRecvMsg(ByVal recvMsg as string)
```

그림 6 클라이언트 메써드 및 이벤트

[그림 6]에서 메써드는 메시지 처리기의 주요 연산과 관계가 있는 것으로 연결/등록/발표/해제/종료를 클라이언트가 하기 위한 것이고, 이벤트는 메시지 교환과 직접적으로 관계가 있는 것이다. EvpCli_OnRecvMsg는 전달 전문에 해당하는 채팅 내용을 수신하는 이벤트이고, EvpCli_OnRecvXml는 보다 다양한 응용을 하기 위해 필요한 이벤트로, 본 채팅 시스템 뿐만 아니라 향후에 추가되어지는 전송전문을 처리하기 위한 것이다.

즉, 채팅 시스템을 요약해 보면 클라이언트가 대화방을 메시지 처리기에 등록하고 나서 해당 대화방으로 채팅 메시지를 발표하면 메시지 처리기는 등록된 모든 클라이언트로 채팅 메시지를 전달하게 된다. 채팅 프로그램의 실행결과는 [그림 7]과 같다.

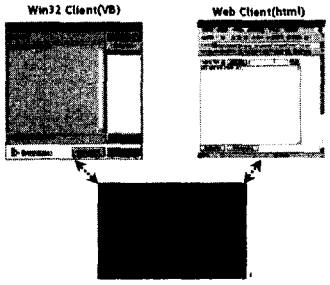

그림 7 채팅 프로그램 실행결과

[그림 7]에서 ActiveX 컨트롤을 사용한 여러 환경의 클라이언트와 메시지 처리기 사이에서 채팅이 원활하게 진행되는 것을 볼 수 있다.

5. 결론 및 향후 과제

본 논문에서는 조직간 실시간으로 정보를 공유하고 비즈니스의 처리를 원활하게 하고 대규모화되는 클라이언트를 처리할 수 있는 토픽 기반 메시지 시스템을 제안하고 구현하였다. 웹 환경에서만 국한되지 않고 win32용과 웹 클라이언트를 동시에 사용할 수 있도록 ActiveX 컨트롤을 이용하여 클라이언트를 구현하였다.

향후 과제는 토픽 기반 메시지 시스템에서 클라이언트의 제로-인스톨의 기능을 추가하고 모바일 클라이언트의 환경에서도 연동될 수 있도록 구현하는 것이다.

참고 문헌 및 사이트

[1] KnowNow In. ,"PRODUCT BRIEF:EVENT ROUTING IN KNOWNOW",http://www.knownow.com/products/whitepapers/KNBrief_EventRouting_Intro.pdf, April 2002

[2] KnowNow, "WHITEPAPER:KNOWNOW WEB NATIVE MESSAGING",http://www.knownow.com/products/whitepapers/KNWhitepaper_WebMessaging.pdf, April 2002

[3] Kenamea, "Products>Overview",http://www.kenamea.com//products_over.html, April 2002

[4] Kenamea,"Products>Features",http://www.kenamea.com/products_features.html, April 2002

[5] Elvin, "What is elvin", http://elvin.dstc.edu.au/intro/what.html, April 2002

[6] Mark Lutz. "Programming Python", O'REILLY, March 2001, pp. 526-537, 985-1019