정책 기반 네트워크 관리에 관한 연구

김영하*, 육동철*, 최길영**, 이규호**, 박승섭*
*부경대학교
** ETRI (전자통신연구원)
e-mail : ipakp@hanmail.net

A Study on Policy Based Network Management

Young-Ha Kim*, Dong-Cheol Yuk*, Seung-Seob Park*
Kil-Young Choi**, Kyu-Ho Lee**
*Pukyong National University, **ETRI

요약

인터넷 환경에서 멀티캐스팅 서비스를 제공하기 위한 응용프로그램과 양질의 서비스를 받고자 하는 사용자들의 요구를 만족시키기 위해 네트워크 관리에 관한 정책이 필요하며, 정책시스템은 네트워크에서 QoS를 가능하게 하는 요소이다.

본 논문에서는 리눅스 기반 환경에서 정책을 적용할 수 있는 정책서버에 해당하는 PDP와 클라이언트인 PEP구현, 디렉토리 서비스 프로토콜인 LDAP 사용 환경 구축에 초점을 두어 설계 및 구현하였다.

1. 서론

네트워크 환경은 현재 가장 빨리 발전하는 분야 중 하나이다. 서비스의 요구에 따라 다양한 문제를 가지고 있다. 이러한 문제를 해결하기 위해서는 규칙이 필요하며 다양한 수용을 결정하기 위해 정책(Policy)이 필요하다. 정책 시스템은 네트워크에서 QoS(Quality of Service)를 가능하게 하는 필수 요소이다.

IETF(Internet Engineering Task Force)에서 표준화한 종합서비스 모델에서는 서로 다른 트래픽 클래스들이 특정 링크의 대역폭을 공유할 때, 네트워크 관리자는 COPS(Common Open Policy Service)와 LDAP(Lightweight Directory Access Protocol) 프로토콜을 이용하여 대역폭 공유와 우선 순위를 제어할 수 있는 기능을 제공하고 있다[1][2]. 실시간 서비스를 보장하기 위해서는 자원 예약 프로토콜과 효율적인 트래픽 제어기능이 포함된 정책 기반 컴퓨터 및 라우터가 구현되어야 한다.

2. 관련연구

본 장에서는 정책기반네트워크의 기본 개념과COPS프로토콜의 기본과 동작에 대해서 서술한다.

본 논문은 2001년도 ETRI의 과제수행으로 연구된것임.

2.1 정책 기반 네트웍

인터넷의 폭발적인 성장과 함께 미래의 네트워크에서 필수적으로 지원되어야 할 것이 QoS이다. 인터넷 환경은 현재 새롭게 등장하는 실시간 멀티미디어 그리고 멀티캐스팅 서비스를 제공하기 위한 응용프로그램들을 지원하기에는 적당하지 않다. 또한 양질의 서비스를 받고자 하는 사용자들의 요구를 만족시키기 어렵다. 각각의 사용자들의 요구에 맞게 차별화 된 서비스를 제공하는 것이 DiffServ(Differentiated Service)이다.

네트워크에서 관리자에 의한 QoS 보장은 사용자들이 질 좋은 서비스를 받을 수 있으므로 이러한 서비스를 위해서는 인증이 필요하다. 이를 정책으로 규정하여 반영 할 수 있다. 정책은 단순한 의미에서 특정 조건(Condition)이 있을 때, 동작(Action)으로 발생되는 하나 이상의 규칙이다. 정책은 때때로 다른 여러 규칙으로 구성될 수 있다. 정책은 정책을 포함하는 것이다.

2.1.1 정책의 구조와 기능

IETF의 RAP(Resource Allocation Protocol) 작업그룹은 RSVP(Resource Reservation Protocol)와 이를 가능케한 IntServ(Integrated Service)에 안정된 정책 제어 모델 설립을 추진하였다[3][4][5].

정책 기반 프레임워크은 크게 두 개의 구성으로

정책서버인 PEP(Policy EnforcementPoint)와 클라이언트로 PDP(Policy Decision Point)로 나뉘어져 있다. PEP는 직접적인 동작을 집행하고, PDP는 법의 근거를 바탕으로 결정을 내린다는 의미이다. PDP는 정책 저장소, 인증 서버 또는 다른 엔티티에서 정책에 대한 정보를 갱신 및 수정을 할 수 있다.

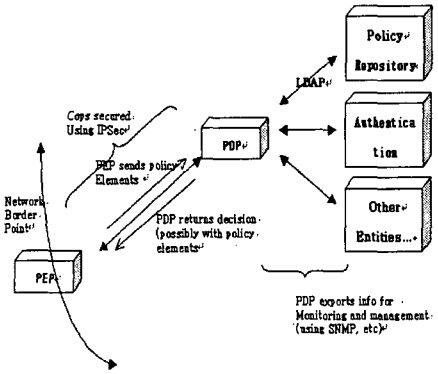

그림 1 정책 기반 프레임워크

그림1은 PEP, PDP, 정책 저장소 등의 세분화는 물리적 분리가 아닌 논리적 세분화 기능을 기반으로 하고 있다. PEP는 PDP의 수행이 불가능할 때, 그 이전의 백업용으로 내부적으로 정책 결정 수행하는 LPDP(Local PDP)가 포함 될 수도 있다. 그리고 보안상 PEP는 PDP의 마지막 결정을 참조하는데, 이 요구를 위해 request 메시지를 지속적으로 전송한다. 한다. PEP와 PDP간의 3개의 정책 시스템 기능을 소개한다.

- Decision-making

PEP로부터 정책 반영과 해석, 정책 충돌 검색, 정책 결정의 요구, 정책 조건 등을 포함하며, 외부 엔트리로부터 정책의 갱신, 요청을 PEP에 송신한다.

- *Enforcement*

PDP의 현재 네트웍 조건의 관련된 정책을 결정에 따라 PEP에 적용되는 것을 말한다.

- Metering

정책의 만족성, 클라이언트가 불공정한 네트웍 서비스를 받는지에 대한 능동적 또는 수동적인 검증이다.

2.2 COPS의 기본과 동작

그림2는 일반적인 기본 모델로 COPS는 PEP와 PDP 사이의 정책 정보를 교환하기 위해 사용하며, 임의의 LPDP는 PDP가 없을 때 local policy의 결정들을 내리기 위한 장치로 사용된다.

COPS는 PEP와 PDP사이의 정책 정보를 교환하기 위해 사용된다. PEP는 지속적인 TCP 연결로 request 메시지를 보내고, PDP로부터 decision 메시지를 수신한다. request 메시지가 PDP에 도착되어 decision 메시지를 수신하면 *configuration data*를 PEP에 성공적으로 설치를 할 수 있고, 설치 승인은 report 메시지를 통해 PDP에 전달된다. 오류 검출은 KA(keep-alive)메시지를 통해 연결되어 있다는 것을 PEP와 PDP가 계속 확인 할 수 있다. 오류가 검출될 때는, PEP와 PDP에 연결을 다시 설정하거나, 다른 PDP와 연결을 다시 시도한다.

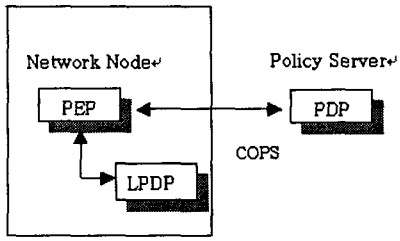

그림 2 COPS 기본 모델

2.2.1 COPS 통신 방법

COPS는 PEP와 PDP 사이에서 하나의 영구적인 TCP 연결을 사용한다. 하나의 PEP가 다중 클라이언트 타입을 지원할 수 있다면, PEP는 TCP연결을 통해 PDP로 특정한 클라이언트 타입을 명세하는 다중 Client-Open 메시지를 전송할 수 있다. PEP는 여러 개의 PDP를 가질 수 있다는 뜻이다.

만약, 클라이언트 타입이 거부되었다면 PDP는 다른 PDP 주소에 대한 PEP를 재 설정하여야 한다. TCP연결이 종료·손실되었을 때 request/decision 메시지 교환과 관련 있는 상태로 정리를 해야 되고, 타임아웃 조건에 의해 연결이 해제되었을 때, PEP는 명령 실패를 나타내는 에러코드를 <Error> 객체에 포함시켜 PEP에 client-close메시지를 전달해야 한다.

2.2.2 메시지 내용

메시지가 전송될 때 무결성 객체가 포함되어 있다면, 이는 메시지의 마지막 객체이다. 무결성 객체를 가지지 않은 메시지를 수신하게 되면 메시지를 수신하는 쪽은 에러코드가 명시된 client-close 메시지를 전달해야한다.

- Request(REQ) PEP→PDP

stateful 핸들이 새로운 요구를 설정하게 되면, 요구에 대한 수정은 이전에 설치된 핸들을 지정하는 REQ 메시지를 사용하여 만들 수 있다. PDP는 핸들과 교환된 정보를 조사하고 주어진 클라이언트타입에 대한 연결을 위하여 Decision을 사용한다. PEP는 로컬상태가 변화할 때면 PDP에 자신의 상태를 알려야 한다.

- Decision(DEC) PDP→PEP

PDP는 REQ의 응답으로 DEC메시지를 전달한다. 같은 클라이언트 핸들과 Context 객체와 Decision 플래그 타입 객체와 관련된 여러개의 decision 객체를 PEP에 전송한다. 프로토콜 에러가 있다면, error 객체를 대신에 전달한다.

- Report State(RPT) PEP→PDP

PDP의 decision를 수행하는 성공과 실패를 PEP가 PDP에 전달 사용되어진다. 상태변화와 관련된 계산치를 보고한다. 임의의 ClientSI(Client Specific Information)와 report의 형태로 지정되어진 Report-Type를 Client-Type 하나 당 추가적으로 정보를 제공한다.

- Client-Open(OPN) PEP→PDP

PEP에 의해서 PDP에 연결 요청시 사용되어진다. PEP에 지원되어지는 클라이언트 타입들에 의해서도 사용되어진다. 동일한 클라이언트타입을 위한 다중 client-open 메시지도 허용한다.

- Client-Close(CC) PEP→PDP, PDP→PEP

특별한 클라이언트의 타입이 더 이상 지원되지 않는다는 것을 PDP 혹은 PEP에 통보하기 위해 생성시킬 수 있다.

- Keep-Alive(KA) PEP→PDP, PDP→PEP

연결을 위해 수신된 모든 CAT메시지에서 명시된 모든 KA타이머의 최소 값으로 정의된 주기 내의 PEP에 의하여 전송되어져야 한다. KA메시지는 최소한의 KA타이머간격을 1/4 혹은 3/4사이로 임의적으로 생성시켜야 한다.

- Synchronize sate Complete(SSC) PEP→PDP

PDP가 동기화 단계의 요구를 PEP로 보낸 후 PEP가 PDP로 보낸다. PEP는 동기화를 끝낸다. 이것은 모든 클라이언트의 상태가 성공적으로 다시 요구되었을 때 PDP가 알 수 있도록 하기 위해 유용한 것이며, PEP와 PDP는 동기화가 완전하게 연결된다.

3. 정책 기반 네트워크의 구현

본 장에서는 자원 예약을 위한 RSVP데몬을 대신할 Service System과 PEP클라이언트, 정책서버와 연결되는 LDAP서버 구현을 위해 여러 대의 퍼스널 컴퓨터에 리눅스(와우리눅스7.1)를 사용해서 작업하였다.

3.1 Service System의 구성

서비스 시스템구성은 필요한 값만 준다고 가정할 때, 값만 주고 받을 수 있는 프로세스를 생성하여, 콘솔상에서 값을 입력하는 방식으로 구현하였다.

그림3은 메시지 신호 처리 흐름도를 나타내고 있으며, 메시지 프로세싱 방식을 사용해서 PepAgent와 시스템 서비스간의 값을 공유해서 두 프로세스간 통신을 할 수 있다.

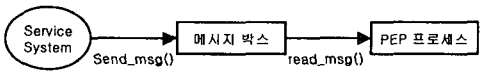

그림 3 메시지 신호 처리 흐름도

메시지 큐를 다루기 위해 클래스 MSG Class를 정의하였고, send_msg, read_msg 메서드는 qid를 식별해서 사용하였다. 메시지들은 차례대로 큐로 보내지고 큐에서 여러 가지 다른 방법으로 이 메시지들이 조회된다. 각각의 메시지 큐는 IPC 확인자(identifier)에 의해 유일하게 확인되어진다.

그림 4는 두 개의 메시지가 메시지 큐에 저장된 결과를 나타낸 것이며, 그림 5는 두 개의 메시지가 메시지 큐에 저장된 결과를 나타내고 있다. PepAgent는 read_msg()메서드를 사용해서 메시지 큐를 검사하여 메시지가 있을 경우 정책 결정을 위해 PdpAgen와 동작을 수행한다.

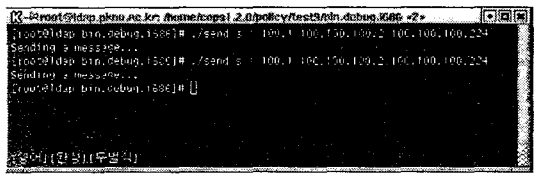

그림 4 두 개의 메시지 전송

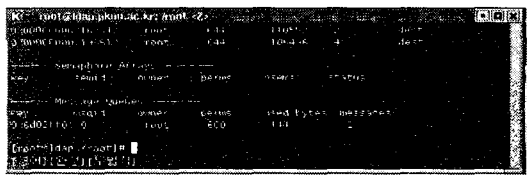

그림 5 메시지 큐에서의 두 개의 메시지 확인

3.2 LDAP 디렉토리 서버 구성

LDAP는 표준화된 고수준의 디렉토리 서비스 프로토콜이다. 모든 디렉토리서비스(액티브 디렉토리, 넷스케이프 디렉토리 서버와 NIS등)은 LDAP와 교신이 가능하다.

본 논문에서는 Openldap을 사용해서 LDAP프로토콜을 구성하였고, 정책 결정을 위해 LDAP 환경파일을 구성하여 slapd데몬을 실행하였다. 데이터파일이 저장되는 위치는 /var/lib/ldap이며, 에러나 수정을 할시에는 /var/lib/ldap 파일들을 수정하든지,저장경로를 새로 설정해야 한다. Suffix행은 LDAP서버에게 서비스할 LDAP 영역의 일부를 알려주고, 각 데이터베이스를 설명한 부분마다 하나씩 명시되어야 한다.Rootdn과 rootpw행은 서버에 대한 암호를 제공하며 서버의 관리자 권한을 부여한다. 암호는 모두 텍스트 또는 MD5형식으로 사용된다.

3.3 PEP클라이언트 구성

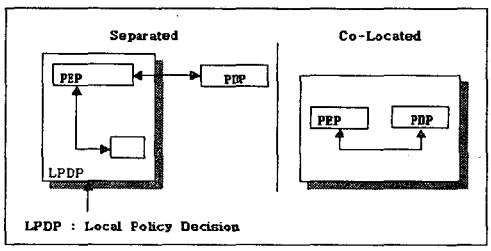

그림 6 Enforcement and Decision Point Scenarios

PEP는 항상 네트워크 노드 내부에 위치해야 하는 반면에 PDP는 어느 곳에서든지 위치할 수 있다. 그림 6은 PEP는 네트웍 노드의 컴포넌트이며, PDP

는 분리된 정책서버 안에 있는 엔티티이다. 한 개의 시스템에서 서버와 클라이언트간 통신이 가능하다.

그림 7은 COPS객체의 메시지 결합과 전송 수신 상태 관계를 나타내는 그림으로, 각 메시지의 통신 중에 KA메시지는 클라이언트와 서버간 연결이 설정되면 지속적으로 통신을 하며, KA메시지 통신을 위해서 스레드 방식으로 처리하였다.

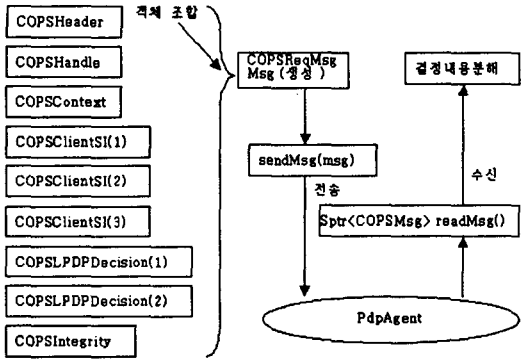

그림 7 PepAgent 프로세스에서 Request메시지를 PdpAgent로 전송할 때 동작 흐름도

3.4 PDP 서버 구성

PDP는 정책 결정을 위해 PEP와 COPS 프로토콜을 사용해서 통신한다. PEP와 통신할때는 소켓 방식으로 통신하며, PDP프로세스가 활성화되면, PEP에서 connect요청이 올 때까지 listen 상태에 들어간다. 이때 PEP의 요청 메시지가 함께 도착된다.

그림 8은 PepAgent에서 REQ메시지가 왔을 때의 동작에 대한 흐름을 나타내고 있는 그림이다. PepAgent는 PdpAgent에 정책을 요청하고 PdpAgent는 LDAP서버에 질의해서 결과를 PepAgent 전달한다. LDAP서버에 접근할 수 있는 것이 COPS 메시지 프로토콜의 REQ메시지이다. PdpAgent 프로세스는 MSG 클래스의 메시지 박스의 내용이 REQ메시지의 SI정보에 포함되어 전송하며, 이 정보를 토대로 질의를 형성한 후에 LDAP서버에 질의한다.

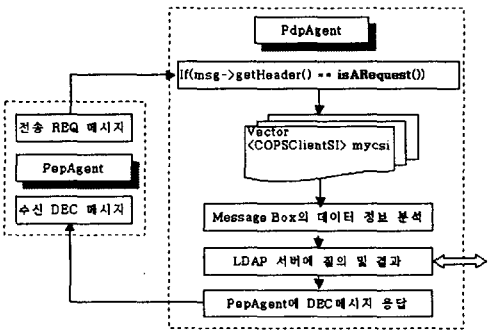

그림 8 REQ 메시지와 DEC 메시지의 처리 흐름도

4. 결론

현재 인터넷 기술 발전의 중심이 되는 IETF를 주축으로 기존의 인터넷에 다양한 멀티미디어 트래픽을 보다 효율적으로 수용하기 위한 연구가 진행되고 있다.

본 논문은 리눅스 기반 환경에서의 정책을 적용할 수 있는 COPS의 구현과 정책의 서버에 해당하는 PDP와 클라이언트인 PEP 구현, 디렉토리 서비스 프로토콜인 LDAP 사용 환경 구축 및 활용 방안, 정책 스키마 구축, 디렉토리 서버와 사용자간의 인터페이스 구현하였다. 본 정책시스템의 구축으로 QoS를 보장함으로써 사용자들에게 양질의 서비스를 제공 받을수 있을 것으로 사료된다.

5. 참고문헌

[1] Boyle, J., Cohen, R., Durham, D., Herzog, S., Raja, R. and A. Sastry, "The COPS (Common Open Policy Service)Protocol", RFC 2748 <http://community.roxen.com/developers/ idocs/rfc/rfc2748.html>, January 2000.

[2] Yeong, W., Howes, T., and S. Kille, "Lightweight Directory Access Protocol", March 1995.

[3] Braden, R., Zhang, L., Berson, S., Herzog, S. and S. Jamin, Resource ReSerVation Protocol (RSVP) Version 1 - Functional specification", RFC2205<http://community.roxen.com/developers/idocs/rfc/rfc2205.html>, September 1997.

[4] IntServ Working Group and [RFC 2211, 2212], http://www.ietf.org/html.charters/intserv-charter.html

[5] Yavatkar, R., Pendarakis, D. and R. Guerin, "A Framework policy Based Admission Control", RFC 2753, January 2000.