2000년 한국정보처리학회 추계 학술발표논문집 제 7 권 제 2 호

개방형 프로그래머블 라우터를 위한 QoS 인터페이스 연구

이진수*, **, 홍선미*, 김태일*, 진성일**
*한국전자통신연구원, **충남대학교 컴퓨터과학과
e-mail : jslee@cs.cnu.ac.kr, smhong@etri.re.kr, tikim@etri.re.kr, sijin@cs.cnu.ac.kr

A Study on QoS Interface for open programmable Router

Jin-Su Lee*, **, Seon-Mi Hong*, Tae-Il Kim*, Sung-il Jin**
*Electronics and Telecommunications Research Institute,
**Dept. of Computer Science, Chung-Nam National University

요 약

근래 인터넷을 통한 대용량의 멀티미디어 데이터와 여러 종류의 사용자 서비스를 충족하기 위해서 네트워크에 대한 연구가 활발히 진행되어지고 있다. 따라서 본 논문에서는 차세대 개방형 네트워크에 적용할 수 있는 라우터를 위한 객체 지향적 Open Programmable Interface 기술을 연구함으로써 앞으로 IP Router 에 도입될 정책이나 구성의 동적 변경에 유연성 있게 대처할 수 있고, 향후 수요가 급증하게 될 다양한 대용량 멀티미디어 데이터와 여러 종류의 QoS 에도 능동적으로 대처할 수 있는 방안을 제시한다.

1. 서론

최근 들어 컴퓨터 통신의 확산과 함께 인터넷 사용의 전세계적인 보편화로 인하여 그 사용량이 급증함에 따라 이를 활용한 여러 다양한 서비스가 제공되고 있다. 또한 멀티미디어 데이터의 사용이 함께 증가함에 따라 서비스에 필요한 네트워크의 전송량 확대와 함께 인터넷 상의 여러 지점에 설치되어 있는 Router 의 데이터 처리 능력도 점점 증가되고 있는 추세이다. 인터넷의 급속한 성장은 Router 에게도 새로운 변화를 요구하게 되었는데, Backbone 망에서는 소수 링크상에서의 고속 Routing 이 필요하게 되었고, Enterprise 네트워크에서는 포트 당 가격이 저렴해야 하며,쉽게 설치할 수 있어야 한다. 또한 QoS 를 지원할 수 있는 Router 가 다양한 종류의 고속 포트를 지원할 수 있어야 하고, 중앙의 음성 스위치를 통과 할 수 있어야 한다. 이러한 발전 상황을 고려해서 네트워크도 상대적으로 다양한 프로토콜을 지원할 수 있는 Open programmable Interface 를 요구하게 되었는데, 현재 외국의 여러 표준기관이나 포룸(Forum)에서는 이러한 개방형 프로그래머블 네트워크나 멀티서비스 스위칭 시스템에 관한 여러 종류의 표준 안을 제안 하고 있다.

IEEE PIN(Programmable Interface for Networks)에서는 현재까지 qGSMP 프로토콜과 함께 BIB(Binding Information Base)의 표준 문서를 발표한 상태이며 앞으로 얼마 있지 않아 IP Internetworking 과 signaling Inter-working 에 관한 표준 문서를 발표할 예정이다. 이와 함께 네트워크 상의 스위치들을 하나로 묶어 관리하는 Binding Controller 를 두어 Source 로부터 Destination 까지의 데이터 전송을 책임지는 System Architecture 를 제시하고 있다.

MSF(Multiservice Switching Forum)는 다양한 통신 인프라(Infra) 구조의 서로 다른 서비스들을 수용하기 위해 조직된 포룸으로 최근에 두 가지 표준 안을 발표했는데 하나는 multi-service switching system 에 관한 Architecture 이고, 다른 하나는 VSI(Virtual Switch Interface)로 multi-service 제공에 대한 통합된 API implementation protocol 을 정의한다. MSF 에서 제시한 System Architecture 는 application service 에 대해 요구되

는 QoS 를 제공하는 구조를 보여주는데 네트워크상의 서비스는 서로 다른 Backbone 네트워크 상에서도 작동 가능하다.

Parlay 또한 telecommunication hardware 와 rich IT(Information Technology) application 사이에 API 에 관한 2 개의 official specification 을 제안했다. Parlay 에서 제시한 개념은 Backbone 인프라 구조와 수 많은 IT application service 의 두 계층사이에 내재 되어있는 API 가 타협 점을 가지고 문제를 해결해 나가는 모델을 가진다.

본 논문에서는 현재 가장 활발한 활동을 보이고 있는 IEEE PIN 1520 개방형 네트워크 모델을 바탕으로 하여 적용 가능한 다양한 서비스에 대한 인터페이스들을 정의하고 이를 IP 라우터에 적용하기 위한 절차를 제시한다. 논문의 구성은 1 장 서론을 이어서, 2 장에서는 IEEE PIN 1520 모델의 특성을 알아보고, 3 장에서는 현 IP 라우터가 개방형 네트워크에 접목되기 위해 필요한 인터페이스를 객체지향적으로 설계,정의한다. 그리고, 4 장에서는 원활한 QoS 제공을 위한 인터페이스를 제안하며, 마지막 5 장에서는 결론과 함께 향후 연구 방향에 대해 제시한다.

2. 개방형 네트워크 구조

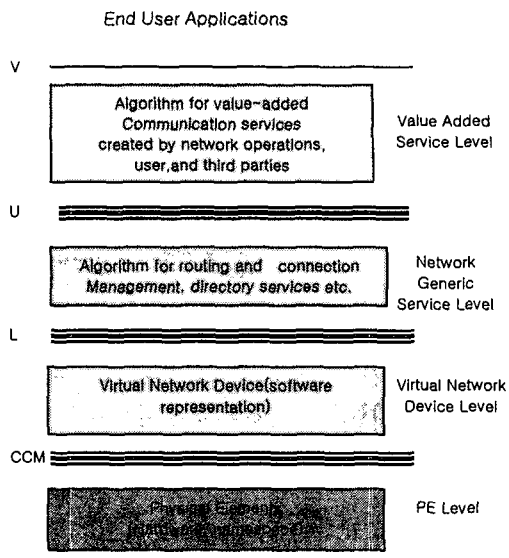
기존 인터넷의 정책은 다양한 사용자의 요구 사항과 대용량 데이터의 급증으로 인해 보다 유연한 서비스 제공을 필요로 하게 되었고, 이 요구를 바탕으로 차세대 네트워크 구조인 개방형 네트워크의 개념이 등장하였다. 개방형 네트워크는 다양한 서비스 형태의 수용과 서로 다른 네트워크 구성요소의 결합을 지원할 수 있어야 하는데, 이를 위해 네트워크 구성 요소들에 대해 하드웨어로부터 미들 웨어, 응용 서비스에 이르기 까지 프로그래밍 언어에 의한 제어가 필요하게 되며, 이를 위해서는 표준화된 API 들의 정의가 선행되어야 한다. 이 API 를 개방형 프로그래머블 인터페이스라고 말한다.

2.1 IEEE PIN 1520 모델

개방형 네트워크에 대한 표준을 마련하기 위한 연구는 IEEE, MSF, ISC, Parlay 등에서 활발히 진행중이다. 이 중 IEEE 는 IP 를 기반으로 하는 네트워크 구조를 바탕으로 연구가 시작되었고, MSF 는 ATM 을 바탕으로 연구를 시작하였다. 그러므로 본 논문에서는 IP 라우터를 지원하기 위한 인터페이스 설계를 목표로 하기 때문에 IEEE 의 PIN 1520 모델을 바탕으로 인터페이스를 제안한다.

그림[2-1]은 IEEE PIN 1520 개방형 네트워크의 제안된 모델을 보여준다.

IEEE 의 모델은 4 단계의 계층구조와 각각의 단계를 연결해주는 4 개의 인터페이스를 정의하고 있다. 이들은 하위레벨로부터 상위레벨 쪽으로 계층화되어 상속 받는 형태를 지닐 수 있으며, 실제 자원에 대한 추상화 작업은 이러한 상속을 바탕으로 이루어지게 된다.



[그림 2-1] IEEE PIN 1520 Reference Model

2.2 P1520 APIs

IEEE 에서 제안한 4 개의 인터페이스를 살펴보면, CCM(Connection Control and Management)인터페이스는 가장 하위의 물리적인 하드웨어와 리소스(Resource)들의 계층인 PE 계층과 이를 추상화 시킨 VNDL 계층을 연결해주는 역할을 하며, 표준화된 인터페이스 모델을 따르기 보다는 각각의 Vendor 들에 의해 개발되어져 특허화 되어가고 있으며, 엄밀히 말하자면 프로그래밍 인터페이스가 아니라 스위치와 외부 에이전트(Agent) 간에 하위레벨에서 상태 및 제어 정보를 교환하기 위한 프로토콜들의 집합이다. L(Lower)인터페이스는 물리적인 객체들이 추상화 되어있는 VNDL 계층과 이 추상화 되어진 객체들을 사용하여 네트워크를 구동할 수 있는 알고리즘들이 집합되어있는 NGSL 계층을 연결해주는 역할을 하고, 네트워크 자원상태에 대한 직접 액세스와 변경이 가능하게 하기 위한 API 를 정의하고 있다. U(Upper) 인터페이스는 네트워크 서비스를 포괄적으로 다루며 사용자의 연결에 대한 요청이 이 인터페이스를 통해서 이루어 지게 된다. 이 연결들은 point-to-point 나 point-to-multipoint 형태가 될 수 있으며, VPN 에서와 같이 임의의 그래프일 수 도 있다. 이 인터페이스의 장점은 연결 설정에 대한 요청을 그 설정 절차에 사용되는 알고리즘과는 관계없이 Parameter 화 할 수 있다는 일반성에서 찾을 수 있는데, 이러한 인터페이스와 구현과의 분리는 하나의 네트워크에서 서로 다른 다수의 연결 관리 방식의 공존을 가능하게 해준다. V(Value-added) 인터페이스는 사용자 소프트웨어를 위해 부가 서비스 형태의 풍부한 API 를 제공한다.

3. 자원과 흐름 추상화 인터페이스 모델

본 장에서는 개방형 네트워크에 적응할 수 있는 라우터 API 를 설계하기 위해서 라우터 기능의 가장 핵

심이 되는 라우팅 동작에 필요한 자원들을 추상화 시킬 수 있는 인터페이스를 설계 한다. 여기서 정의된 인터페이스를 바탕으로 특정서비스에 적용될 수 있는 인터페이스의 설계가 이루어진다. 이 모델은 실제 하드웨어 리소스들을 추상화 시키는 논리적 추상화 인터페이스와 하드웨어들간에 이동되는 흐름(Flow)에 대한 추상화를 지원하는 흐름 추상화 인터페이스로 구분된다. 이 단계의 인터페이스 구축은 사실상 각 Vendor 들에 종속적(dependent)이며, 실제 적으로 많은 회사들이 고유의 인터페이스에 대한 특허(special right)들을 보유한 상태이므로 실제 표준에 대한 구축 작업은 일반적인 자원(resource)들에 국한되게 된다.

3.1 논리적 추상화 인터페이스

```
class Resource_Abstract
{
private:
enum QUEUING {REDUCE,SHARING,ASSIGN,PRIORITY,FIFO};
enum DROP    {REDUCE,PRECEDENCE,CUT};
public:
Typedef unsigned long QUEUE_ID;
Typedef unsigned long QUEUE_PRECEDENCE;
Typedef unsigned long DROP_PRECEDENCE;
Typedef sequence<QUEUE_ID> QUEUES;
Typedef unsigned long INTERFACE_ID;
Typedef unsigned long IP_ADDRESS;
Const   unsigned short PORT;
Const   INTERFACE_ID  INTERFACE_ANY;
Struct QUEUE
{
   QUEUE_ID       q_id;
   unsigned long  q_size;
   QUEUING        q_choice;
   DROP           d_choice;
}
Struct L_INTERFACE
{
   INTERFACE_ID   l_id;
   QUEUES         out_queue;
}
Struct FLOW_POSITION
{
IP_ADDRESS     ip_add;
IP_ADDRESS     add_mask;
INTERFACE_ID   inteface;
}
Struct PORT_DEFINE
{
Unsigned short min_port;
```

[그림 3-1] 자원과 흐름에 대한 추상화 클래스의 예

논리적 추상화 인터페이스는 예를 들어 라우터 안에 있는 라인카드(Line card)등과 같은 물리적인 자원들을 상위레벨에서 논리적으로 사용하기 위한 추상화를 지원해주는 인터페이스이다. [그림 3-1]은 큐(Queue)에 대한 추상화를 지원하기 위한 클래스 정의와 데이터 형태의 선언을 보여준다.

그림에 제시된 클래스는 하나의 큐에 대한 속성들을 정의할 수 있으며, 각각의 큐는 5 가지의 속성을 가진 알고리즘을 선택할 수 있다.

REDUCE 는 traffic 을 감소시키기 위해 대역폭에 대한 감축을 지원할 수 있고, SHARING 은 높은 순위의 traffic 이 낮은 지연(delay)을 요구할 경우, 낮은 순위의 traffic 들 사이의 남아있는 대역폭 사용을 지원하며, ASSIGN 은 등급(class)을 가지고 traffic 을 구분하여 각각의 traffic 에 등급에 따른 대역폭할당을 지원한다. PRIORITY 는 높은 우선순위의 traffic 이 낮은 우선순위의 traffic 으로 인해 지연되는 것은 방지할 수 있고, FIFO 는 모든 traffic 에 대해 우선순위 없이 동일한 조건으로 서비스를 제공하게 된다. 위의 예는 큐에 대한 추상화를 보여 주고 있지만, IP 주소나 Line Card, 포트 등도 위와 같은 형태로 추상화 되어 질 수 있다.

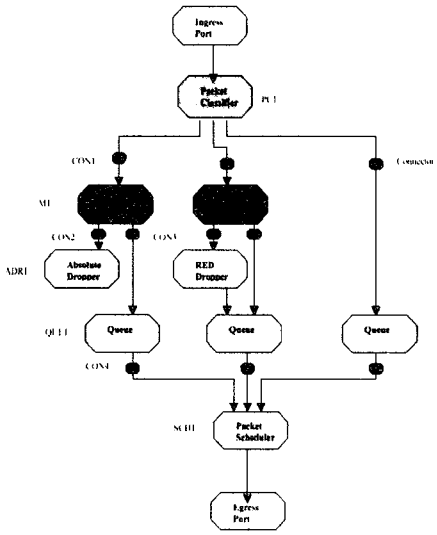
3.2 흐름 추상화 인터페이스

하나의 흐름(flow)은 근원지와 목적지, 그리고 프로토콜을 가질 수 있는데, 이러한 하나의 흐름에 대해 위의 그림과 같은 추상화 작업이 이루어 질 수 있으며, 이를 통해 상위 단계와의 통신이 가능하게 된다. FLOW_POSITION 은 라우터에 입력되어지는 하나의 흐름에 대해 근원지와 목적이 주소, 인터페이스를 정의하고 PORT_DEFINE 에서는 흐름에 할당될 포트에 대한 정의를 할 수 있다.

4. 개방형 라우터를 위한 차등 서비스 Interface 모델

본 장에서는 개방형 네트워크에서 QoS 를 제공하기 위한 방안으로 주로 제시되고 있는 2 개의 서비스모델 중, 차등 서비스(Differentiated Service)를 지원할 수 잇는 라우터 API 를 설계한다.

4.1 L 인터페이스 추상화



[그림 4-1] 라우터 안에서의 데이터 흐름도

위의 그림은 라우터 안의 흐름을 보여주고 있으며, Ingress 포트로 들어온 패킷은 Classifier 에 의해 그 등급이 분류되어 Meter 로 전달되고 Meter 에서는 이미 정해진 정책에 따라 해당 패킷에 대한 처리를 하고 Drop 되지 않은 패킷은 서로 다른 서비스를 하는 큐 중에 하나로 보내어 지어 Packet Scheduler 를 통해 Egress 포트로 나오게 된다. 위와 같은 동작은 인터페

이스에 대한 추상화가 선행되어져야 하는데, 추상화의 단계는 크게 BBB(base building block), RBB(resource building block), SBB(service-specific building block)으로 나누어 지며, 각각은 상위 block 에 대한 계승구조를 가진다. BBB 는 Action, Component, Condition, Target, PU(Processing Unit)으로 구분되어지고, RBB 는 BBB 를 바탕으로 네트워크 요소들의 기능들로 나타내어 질 수 있는데 Queue, Classifier, Profile, Forward, Compute checksum, Filter 등이 이에 해당하며, SBB 는 여러 응용들에 대한 미리 준비되어져야 할 요소들에 대한 추상화가 이루어 지는데 예를 들어, Diff-Serv 를 위해 하나의 DSProvision class 가 생성되어 질 수 있는데 이러한 class 들이 SBB 에 해당하게 된다.

4.2 L 인터페이스 설계

```
class Common
{
Public :
    typedef unsigned long     Data_Path_ID;
    typedef undigned long     RBB_ID;
    typedef sequence<RBB_ID>  RBB_ID_LIST;
    typedef unsigned long     PORT_ID;
    typedef unsigned long     IP_ADDRESS;
    Struct Rate_Profile
    {
          float         bandwidth;
          float         burst size;
          float       time_ave_window;
     }
}
class Data_Path
{
   public:
     attribute   Data_Path_ID       id;
     attribute   Port_IDin_port;
     attribute   Port_IDout_port;
     attribute   RBB_ID_LIST        rbb_list;
   private:
     void get_ingress_port();
     void set_ingress_port();
     void get_egress_port();
     void set_egress_port();
     void get_rbb_list();
     void set_rbb_list();
     void create_data_path(out RBB_ID id);
     void write_data_path(in RBB_ID id);
}
```

[그림 4-2] 공용 요소와 데이터 경로 클래스의 예

위의 추상화 단계를 거치면, 생성되어진 객체들에 대한 통신을 할 수 있는 인터페이스 구축이 필요하게 되는데 [그림 4-2]과 [그림 4-3]은 QoS 제공을 위한 Diff-Serv 의 인터페이스 설계를 보여준다. Data_path 는 RBB 의 순서를 정의하며 IP 장치에 대한 두개의 포트 간에 하나의 경로를 설정하게 되고, Packet Classifier 는 IP 패킷의 헤더 부분을 조사하여 패킷에 대한 분류를 하게 되며, Shaper 는 각각의 PHB(Per Hop Behavior)의 규정에 따라 전송률을 조절하여 보내주게 되고 Shaping 과정을 거쳐서 나오는 packet 에 대한 Profile 을 관리하게 된다. Dropper Interface 는 PHB 규정에 따라 현재 packet 의 가용여부를 판단하고 3 가지 정책을 가지고 Drop 되는 packet 의 관리를 담당하게 되며, Meter Interface 는 분류된 packet 의 전송률이 미리 정해진 규약에 맞는가를 검사하고 confirm 되는 packet 과 출력되는 Flow 의 Profile 을 관리하게 된다.

Shaper, Dropper, Meter 등도 [그림 4-3]과 같은 형태로 구현 되어질 수 있다.

```
class Connector
{
   public:
     attribute RBB_ID   in_rbb;
     attribute RBB_ID   out_rbb;
   private:
     void get_in_rbb();
     void set_in_rbb();
     void get_out_rbb();
     void set_in_rbb();
     void create_Connector(out RBB_ID id);
     void destroy_Connector(in RBB_ID id);
}
Class Packet_Classifier
{
   public:
     enum FILTER_TYPE     {IP, MAC, RSVP, FREE}
     attribute RBB_ID   id;
     attribute unsigned short       outputs;
     attribute FILTER_TYPE          type;
     attribute FILTER_LIST          rules;
     struct FILTER
     {
        FILTER_RULE                 rule;
        unsigned short  output;
     }
     struct MAC_FILTER_RULE
     {
        unsigned long   source_mac;
        unsignedlong                destination_mac
     }
   private:
     void get_outputs();
     void set_outputs();
     void get_type();
     void set_type();
     void get_rules();
     void set rules();
```

[그림 4-3] QoS 제공을 위한 클래스의 예

5. 결론

지금까지 개방형 네트워크에 적용 가능할 수 있는 라우터 인터페이스에 대해 기술하였다. 현재 세계적인 표준화 단체에서는 계속적인 인터페이스 표준화에 박차를 가하고 있으며, 본 연구는 향후 도래하게 될 개방형 네트워크 구조에 보다 유연한 대처와 기술접목에 있어 많은 도움을 줄 것으로 기대된다.

참고문헌

[1] M. Ranuparan, Jit Biswas, Wand Weiguo, "L+ Interface for routers that support differentiated Services," P1520/IP-012

[2] Jit Biswas, John Vicente, Michael Kounavis, Daniel Villela, Michah Lerner, "Proposal for IP L-Interface Architecture," P1520/IP-013

[3] IETF, "Definition of the Differentiated Services Field(DS Field) in the Ipv4 and Ipv6 Headers," RFC2474, December 1998

[4] IETF, "A Conceptual Model for Diffserv Routers," draft-diffserv-model-03.txt, May 2000