고성능 PCE (Path Computation Element) 프로토콜 소프트웨어 구조

이원혁* · 김승해* · 김현철**

요 약

정보통신 환경의 급변과 더불어 정보 사회의 기반이 되는 네트워크도 다양한 기술의 발전과 더불어 기존의 고정된 형태에서 벗어나 대용량의 다양한 데이터를 유연하게 전송할 수 있는 능동 가변형으로 진화하고 있다. 더불어 다양한 사용자들의 요구사항을 반영하기 위해 차세대 전달망은 DWDM 전송 시스템과 OXC 로 재구성이 쉬운 동적인 망을 형성하고, 이후에 ROADM/PXC를 기반으로 하는 동적으로 망을 관리하고 제어할 수 있는 GMPLS 기술을 도입하여 망을 운용하는 형태로 발전하고 있다. 본 논문에서는 이러한 차세대 네트워크에서 경로계산을 위해 IETF 제안한 Path Computation Element (PCE) 프로토콜을 구현하기 위한 전체 S/W 구성 및 기능 블록들을 제안하였고, 상위 계층에서 PCE 프로토콜을 사용할 때 필요한 API를 제안하였다.

A Software Architecture for High-speed PCE (Path Computation Element) Protocol

Wonhyuk Lee* · Seunhae Kim* · Hyuncheol Kim**

ABSTRACT

With the rapidly changing information communication environment and development of technologies, the informati on networks are evolved from traditional fixed form to an active variable network that flexible large variety of data can be transferred. To reflect the needs of users, the next generation using DWDM (Dense Wavelength Division M ultiplexing) transmission system and OXC (Optical Cross Connect) form a dynamic network. After that GMPLS (Ge neralized Multi-Protocol Label Switching) can be introduced to dynamically manage and control the Reconfigurable Optical Add-drop Multiplexer (ROADM)/Photonic Cross Connect (PXC) based network. This paper propose a softw are architecture of Path Computation Element (PCE) protocol that has proposed by Internet Engineering Task Force (IETF) to path computation. The functional blocks and Application Programming Interface (API) of the PCE protoco l implementation are also presented.

Key words : PCE, PCEP, GMPLS, QoS, Path Computation

접수일(2013년 12월 13일), 수정일(1차: 2013년 12월 22일), 게재확정일(2013년 12월 23일)

* 한국과학기술정보연구원 (KISTI)
** 남서울대학교 컴퓨터학과

1. 서 론

멀티미디어 활용 경험의 증가, 보안 및 개인화의 요구 증대, 자유로운 이동성에 대한 욕구 증가와 같은 현상으로 촉발된 데이터 트래픽의 증가와 새로운 요구사항들의 증가는 음성이나 전용선 서비스를 주된 목적으로 하는 현재의 시간분할 다중화 (TDM: Time Division Multiplexing)을 기반 네트워크에서 좀 더 유연하고 동적인 구성이 가능한 광 네트워크로의 전환을 요구하고 있다. 이러한 광 네트워크는 데이터, 비디오, 그리고 음성을 전달할 수 있는 다수의 채널을 제공하는 인프라가 되었다 이처럼 다계층 광 네트워크는 기존 IP가 갖는 범용성 및 규모성과 WDM (Wavelength Division Multiplexing)의 풍부한 대역 제공 능력을 결합하여 폭증하는 인터넷 대역 요구를 경제적, 효과적으로 수용할 수 있게 하는 차세대 통신 네트워크라 할 수 있다 [1][2].

이처럼 다계층 네트워크에서는 여러 가지 제약사항 (constraints)을 고려한 경로계산이 필요하며 이를 위해 제안된 방식이 PCE (Path Computation Element) 기반 구조이다. PCE는 사용자의 요구사항에 따라 다른 종류의 경로(path)를 계산할 수 있기 때문에 트래픽 엔지니어링 기능을 제공할 수 있으며 애플리케이션이나 소프트웨어 형태로 구현될 수 있다. 모든 계층을 포함하는 네트워크 관점은 네트워크를 보다 효율적으로 전송할 수 있게 하고, 네트워크 장애가 발생하는 경우에 대해서도 보다 신속한 복구가 가능하게 한다. PCE 기반 네트워크 구조에서 경로계산을 요청하는 PCC (Path Computation Clients)와 경로계산을 수행하는 PCE 간의 통신은 PCE 프로토콜 (PCEP: PCE Protocol)을 이용한다 [3][4].

본 논문은 PCE 기반 네트워크에서 PCEP를 구현할 때 PCEP를 구성하는 전체 소프트웨어 구조와 각각의 기능 블록들의 구조와 기능을 제안하였다. 아울러 본 논문에서는 제안된 PCEP 소프트웨어를 상위 계층에서 제어할 수 있는 API를 제안하였다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 PCEP 전체 구조와 각각의 기능 블럭들에 대해서 기술하였다. 3장에서는 PCEP API의 종류와 더불어 세부 내용을 제안하였다. 마지막으로 4장에서는 본 논문의 결론에 대해서 기술하였다.

2. PCEP 소프트웨어 구조

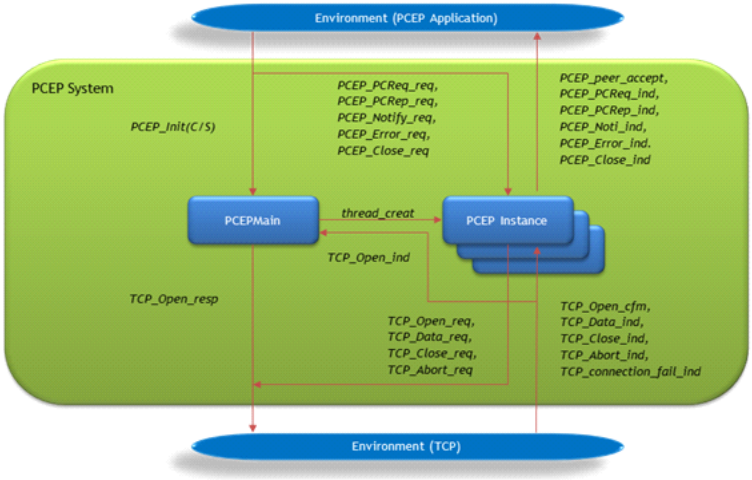
IETF에서 제안한 PCE 기반 구조에서 PCC와 PCE의 분리와 PCE 간의 협업 모델을 통해 다양한 장점을 제공하지만 이를 위해서는 PCC-PCE, 그리고 PCE-PCE 간 통신 프로토콜이 필요하다. 즉 PCC는 하나 이상의 TE LSP에 대한 경로계산 요청을 PCE로 전달하기 위해 PCEP를 사용한다. 아울러 PCE는 제약사항을 만족시키는 하나 이상의 계산된 경로를 PCC로 전달하기 위해 PCEP를 사용한다.

PCEP는 request/response 형태의 프로토콜이며 대표적 전달 프로토콜인 TCP를 기반으로 동작한다. 따라서 PCEP는 통신의 신뢰성(reliability)과 순차적인 전송(in-order delivery)을 지원한다. 또한 PCEP는 SIP (Session Initiation Protocol)과 유사하게 세션 (session)을 기반으로 하는 프로토콜 (session-based protocol) 이다. 즉 통신을 위해서 PCE와 PCC는 세션을 설정하여 서로의 파라미터를 협의하고 지원 능력 (capabilities)을 학습한다 [5][6][7].

PCEP S/W는 (그림 1)에서와 같이 PCC/PCE와 같은 상위 인터페이스와 하위로는 정규 TCP 인터페이스를 제공한다. (그림 1)에서와 같이 PCEP S/W에서는 로컬 PCC/PCE의 메시지를 원격 피어 (remote peer)로 전달하기 위해서 PCEP_Init(C/S), PCEP_PCReq_req, PCEP_PCRep_req, PCEP_Notify_req, PCEP_Error_req, PCEP_Close_req 등의 인터페이스를 제공하며 원격 피어(remote peer)로부터 전달되는 메시지를 상위 계층으로 전달하기 위해서 PCEP_peer_accept, PCEP_PCReq_ind, PCEP_PCRep_ ind, PCEP_Noti_ind, PCEP_Error_ind. PCEP_Close_ind과 같은 인터페이스를 제공한다. 그러나 이러한 인터페이스는 개념적인 것이며 실제 API 에서는 pcep_init, pcep_peer_send, pcep_peer_receive, pcep_peer_accept 와 같은 형태로 구현된다.

2.1 자료 구조

PCEP S/W는 PCEP 구현을 (그림 2), (그림 3)에서

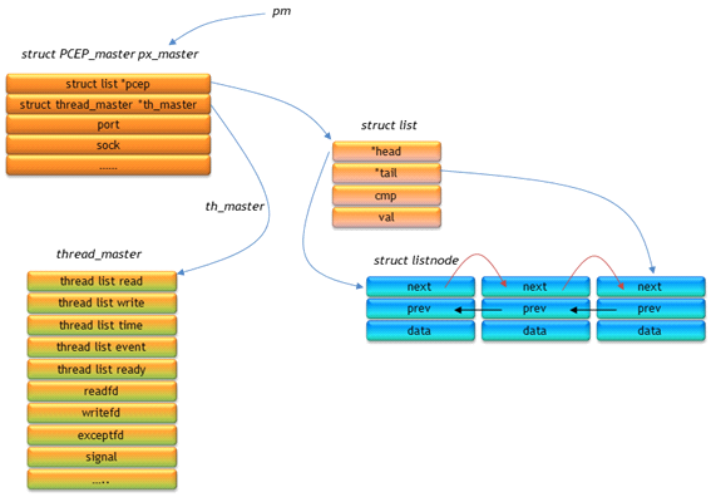


(그림 1) PCEP 구현 전체 구조

나타내고 있는 바와 같이 struct PCEP_master px_master, struct thread_master, struct list, struct listnode , struct pcep, struct peer, struct stream와 같은 여러 형태의 자료구조로 구성된다.

우선 struct PCEP_master px_master 는 (그림 2)에서와 같이 시스템 내의 모든 pcep instance를 관리하는 자료구조이고 pm(pointer master)은 시스템내의 px_master를 가리키는 포인터이다. struct PCEP_master px_master의 *pcep는 system 내의 PCEP instance를 관리하는 struct list 에 대한 포인터이며 해당 struct list는 struct listnode와 같은 형태로 시스템에 존재하는 모든 PCEP instance를 관리한다. 즉 본 PCEP implementation에서는 하나 이상의 PCEP instance가 동시에 구동되는 것을 지원할 수 있다.

이와 더불어 struct PCEP_master px_master 는 각각의 PCEP instance에서 사용되는 input, output, event, signal 등과 같은 모든 스레드(thread)를 관리하는 thread_master 에 대한 포인터 값을 아울러 저장하고 있다. 또한 struct PCEP_master px_master 는 PCEP에서 사용되는 포트(port) 번호, TCP에서 할



(그림 2) PCEP 구현 자료 구조 - 1

당된 소켓번호, 그리고 시스템 관리에 필요한 자료를 포함하고 있다.

thread_master는 PCEP에서 사용되는 모든 스레드와 관련된 자료를 담고 있다. 본 PCEP implementation에서는 TCP connection으로부터 데이터를 수신하기 위한 socket port 인 readfd, TCP connection으로 데이터를 전송하기 위한 writefd, exception 처리를 위한 exceptfd 등과 같은 입출력 포트를 지원하며 이러한 file descriptor 정보 모두를 thread_master에서 관리한다.

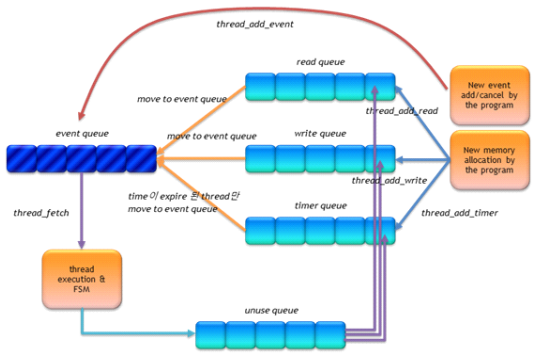
이러한 file descriptor로부터 데이터를 수신하거나 송신하고자 할 때 개개의 송수신 요청은 스레드형태로 구성되며, 이러한 스레드들은 thread_master에서 링크드 리스트 형태로 관리된다. 즉 외부로 데이터를 전송하고자 하는 모든 요청들은 thread list write 에서 관리한다. 또한 외부로부터 데이터를 수신하고자 하는 모든 요청들은 스레드 형태로 thread list read 에서 관리한다.

아울러 본 PCEP 구현에서는 타이머와 관련된 요청들을 관리하는 thread list timer, 그리고 다양한 event와 관련된 스레드를 관리하는 thread list event 를 지원한다.

이처럼 다양한 스레드들의 리스트를 처리하기 위해 본 PCEP 구현에서는 (그림 3)과 같은 thread queue manager와 scheduler를 구현하였다. (그림 3)에서와 같이 프로그램에서 데이터 수신을 요청하는 경우, 이러한 요청은 thread_add_read 함수를 통하여 read queue로 전달되고 저장된다. 비슷한 형태로 TCP connection을 통하여 외부로 데이터를 전달하고자 하는 요청은 thread_add_write 함수를 통하여 write queue로 전달되고 저장된다. 한편 타이머와 관련된 요청들은 thread_add_timer 함수를 통하여 타이머 큐로 전달되고 저장된다.

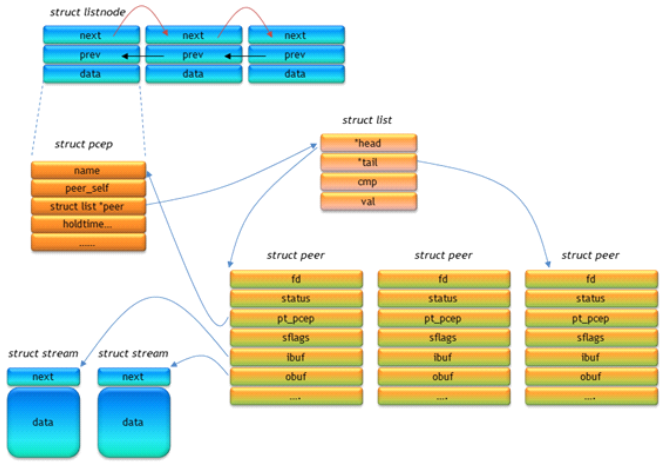
(그림 3)의 이벤트 큐는 thread fetch를 통하여 execution을 대기하고 있는 스레드를 관리하는 queue 이다. read queue와 write queue에 저장된 thread 들은 순차적으로 event queue로 전달되지만 타이머 큐에 있는 스레드들은 원하는 타이머가 만료(expire)된 경우에만 event queue로 전달된다. (그림 3)의 이벤트 큐는 thread fetch를 통하여 execution을 대기하고 있는 thread를 관리하는 queue 이다. read queue와 write queue에 저장된 thread 들은 순차적으로 이벤트 큐로 전달되지만 타이머 큐에 있는 thread 들은 원하는 timer가 만료된 경우에만 이벤트 큐로 전달된다. 또한 PCEP S/W에서는 19가지의 이벤트를 지원하며 thread_add_event를 통하여 event를 생성하는 경우, 이러한 이벤트들은 직접 이벤트 큐에 저장되며 신속한 처리를 보장 받는다.

이벤트 큐에 저장된 모든 스레드 들은 thread fetch를 통하여 event, timer, read, 그리고 write thread 순으로 처리된다. 즉 이벤트 큐에 이벤트 요청이 있는 경우 모든 이벤트를 먼저 처리하고 그 이후에 timer 관련 스레드들이 처리된다. (그림 3)에서와 같이 처리가 끝난 스레드들은 unused queue에 자신이 소유했던 공간들을 반납하며 이러한 공간들은 다시 필요한 곳으로 재배치된다. 만일 필요한 공간이 없는 경우에는 시스템으로부터 필요한 공간을 할당받아 해당 queue에 추가하게 된다.



(그림 3) PCEP 구현 쓰레드 큐와 스케쥴러

한편 (그림 4)에서와 같이 링크드 리스트 형태로 관리되는 pcep instance 들은 각각 struct pcep 를 이용하여 필요한 정보를 관리한다. struct pcep는 pcep instance의 이름과 해당 PCEP와 연결된 모든 peer 정보를 struc peer list 로 관리를 한다. 아울러 stuct pcep는 pcep start time 과 같은 pcep instance 와 관련된 기타 정보들을 담고 있다. 본 구현에서 가장 많은 정보를 담고 있는 struct peer는 (그림 4)에서와 같이 list 리스트형태로 관리되며 pcep instance와 통신을 수행하는 remote pcep instance를 나타내며 struct



(그림 4) PCEP 구현 자료구조 - 2

peer 에는 file descriptor, status, 그리고 input/output buffer 에 대한 내용이 저장된다. file descriptor는 local pcep와 통신하는 port를 나타내며 status는 PCEP session의 현재 상태를 나타낸다.

ibuf와 obuf는 각각 input buffer와 output buffer를 나타내며 struct peer 에서는 필요한 경우 시스템으로부터 메모리를 할당받아 ibuf, obuf 형태로 관리를 하며 전달이 끝난 이후에는 해당 버퍼를 시스템으로 반환한다. struct peer 의 pt_pcep는 peer로 하여금 자신이 연결된 local system의 PCEP를 알려준다.

3. PCEP API

PCEP API는 PCEP client와 server의 경우로 나뉘며 초기 시작 API를 제외한 나머지 API는 공통으로 사용된다. (표 1)은 PCEP API와 관련 내용을 기술하고 있다. (표 1)에서 나타내고 있는 바와 같이 하나의 host 에서 PCEP instance와 PCC/PCE 간의 통신은 메시지의 안전한 전송과 protocol FSM의 동작을 보장하기 위해 따로 추가적인 인터페이스를 사용하지 않고 pcep_peer_receive() 함수를 사용하여 전달한다. 예를 들어 Error 또는 Notification을 전달하기 위해 PCEP instance는 따로 함수를 구현하지 않고 PCEP instance와 local PCC/PCE 간의 통신 방법인 pcep_peer_receive(Error/Notify)를 사용한다.

4. 결 론

차세대 네트워크 표준의 재정과 ITU-T의 NGN 표준의 정립을 통해 전달망의 기능이 더욱 고도화되고, 기존 네트워크 구조의 점진적 진화보다는 혁신적인 변혁이 진행 되고 있다.

본 논문에서는 이러한 차세대 네트워크에서 경로계산을 위해 IETF 제안한 PCE 프로토콜을 구현하기 위한 전체 S/W 구성 및 FSM 구현을 위한 자료구조, 쓰레드와 스케줄러를 구현하기 위한 블록, 그리고 복수개의 피어와 통신하기 위한 통신 제어블럭들을 제안하였다. 또한 본 논문에서는 상위 계층에서 PCE 프로토콜을 사용할 때 필요한 API를 제안하였고, 각각의 세부 파라미터 및 간단한 사용법을 아울러 제안하였다.

<표 1> PCEP API

API	내용
pcep_pre_init	*pcep_pre_init)* 함수는 PCEP를 구동하기 위해 맨 처음 호출해야 하는 함수이다. PCEP와 관련된 모든 data structure의 초기화 및 구동을 담당하는 함수로 동일한 함수 이름을 사용하지만 client와 server 일 경우에 다르게 동작한다. struct pcep { char *name; struct peer *peer_self; struct list *peer;

	struct in_addr local_id;
	u_int32_t default_openwait;
	u_int32_t default_keepwait;
	u_int32_t default_keepalive;
	u_int32_t default_deadtimer;
	u_int32_t default_cleartimer;
	u_int32_t restart_time;
	u_int32_t stalepath_time;
	};
	
	● Client 인 경우
	
	struct peer *pcep_pre_init()
	
	client mode로 동작하는 경우 *pcep_pre_init()*은 struct peer 에 대한 pointer를 결과로 넘긴다. peer는 TCP connection을 통하여 연결된 remote PCEP의 모든 정보를 담고 있는 structure 로서 아래와 같이 pcepd.h 에 정의되어 있다.
	
	struct peer
	{
	struct pcep *pt_pcep;
	int lock;
	struct in_addr remote_id;
	struct in_addr local_id;
	struct stream *ibuf;
	struct stream_fifo *obuf;
	struct stream *work;
	int status;
	int ostatus;
	u_int32_t LocalOK;
	u_int32_t RemoteOK;
	u_int32_t OpenRetry;
	int ConnectRetry;
	u_int32_t arg;
	int fd;
	int ttl;
	char *desc;
	char *host;
	unsigned short port;
	union sockunion su;
	union sockunion *su_local;
	union sockunion *su_remote;
	u_int16_t sflags;
	u_int16_t cap;
	
	/* Timer values. */
	u_int32_t v_start;
	u_int32_t v_connect;
	u_int32_t v_openwait;
	u_int32_t v_keepwait;
	u_int32_t v_keepalive;
	u_int32_t v_deadtimer;
	u_int32_t v_cleartimer;
	
	/* Threads. */
	struct thread *t_read;
	struct thread *t_write;
	struct thread *t_start;
	struct thread *t_connect;
	struct thread *t_openwait;
	struct thread *t_keepwait;
	struct thread *t_keepalive;
	struct thread *t_deadtimer;
	struct thread *t_cleartimer;

	unsigned long packet_size; }; ● Server 인 경우 *int pcep_pre_init()* client 경우와 달리 server side에서 pcep_pre_init()은 다른 결과 값을 제공한다. server side에서는 TCP socket accept 이 발생하기 전까지는 peer 에 대한 정보를 구할 수 없고 pcep_pre_init() 에서는 필요한 structure 들을 초기화 하고 동일 호스트에서 다른 pcep server가 미리 존재하는지를 검사한다.
pcep_peer_send	remote peer 에게 메시지를 전달하기 위해 사용하는 함수이다. *pcep_peer_send()* 에서는 사용자로부터 수신한 msg에 PCEP Header를 추가하여 remote peer로 TCP를 기반으로 전달하는 기능을 수행한다. client side에서는 *pcep_pre_init*() 함수 이후에 바로 사용할 수 있다. *int pcep_peer_send (struct peer *peer, u_int32_t msg_type, u_char *msg, size_t msg_len);* ● peer : message를 전달하고자 하는 remote peer를 나타낸다. ● msg_type : 전달하고자 하는 PCEP message 종류를 나타내며 pcepd.h 에 다음과 같이 정의되어 있다. ● msg : 전달하고자 하는 데이터를 가리키는 포인터이다. msg는 PCEP 헤더를 포함하지 않는 순수 objects 들의 list 이다. ● msg_len : msg의 순수 길이를 나타낸다. objects 들의 길이를 나타내는 msg_len이 4의 배수가 아닌 경우에는 PCEP에서 4의 배수로 조정하여 원격 peer로 전달한다.
pcep_peer_accept	server side에서 TCP connction accept을 기다리고 있다가 remote peer로부터 TCP connection 요청이 전달된 경우, PCEP server는 해당 remote peer에 대한 정보를 만들고 local PCC/PCE로 *pcep_peer_accept()* 함수를 통하여 *remote peer*에 대한 정보를 알려준다. 즉 PCEP s/w가 상위 PCC/PCE로 어떤 remote peer 가 접속을 했는지를 알려주는 창구가 바로 *pcep_peer_accept()* 이다. *void pcep_peer_accept (struct peer *peer)*
pcep_peer_receive	*pcep_peer_send()*와 반대의 기능을 수행하는 함수로 remote peer 로부터 전달된 메시지를 Local PCC/PCE로 전달하는 기능을 수행하는 함수이다. 함수의 parameters 들은 *pcep_peer_send()*와 동일하다. *int pcep_peer_receive (struct peer *peer, u_int32_t msg_type, u_char *msg, size_t msg_len);* ● peer : message를 전달하고자 하는

	remote peer를 나타낸다. ● msg_type : 전달하고자 하는 PCEP message 종류를 나타내며 pcepd.h 에 다음과 같이 정의되어 있다. ● msg : 전달하고자 하는 데이터를 가르키는 포인터이다. msg는 PCEP 헤더를 포함하지 않는 순수 objects 들의 list 이다. ● msg_len : msg의 순수 길이를 나타낸다. objects 들의 길이를 나타내는 msg_len이 4의 배수가 아닌 경우에는 PCEP에서 4의 배수로 조정하여 remote peer로 전달한다.

참고문헌

[1] K. Kompella and Y. Rekhter, “Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE),” RFC 4206, Oct. 2005.

[2] A. Bonerjee, J. Drake, J. P. Lang, and B. Turner, “Generalized Multiprotocol Label Switching: an Overview of Routing and Management Enhancements,” IEEE Communications Magazine, Vol. 39, No. 1, pp. 144-150, Jan. 2001.

[3] K. Shiomoto, E. Oki, D. Shimazaki, and T. Miyamura, “Multilayer Traffic Engineering Experiments in MPLS/GMPLS Networks,” IEEE BROADNETS, Sep. 2007.

[4] Richard Douville, Jean-Louis Le Roux, Stefano Secci, "A Service Plane over the PCE Architecture for Automatic Multidomain Connection-Oriented Services", IEEE Communications Magazine, Vol. 46, No. 6, pp. 94-102, Jun. 2008.

[5] A. Farrel, J.-P. Vasseur, and J. Ash, “A Path Computation Element (PCE)-Based Architecture,” IETF RFC 4655, Aug. 2006.

[6] J. Ash and J. L. Le Roux, “PCE Communication Protocol Generic Requirements,” IETF RFC 4657, Sep. 2006.

[7] J. P. Vasseur, J. L. Roux, A. Ayyangar, E. Oki, A. Atlas, and A. Dolganow, “Path Computation Element (PCE) Communication Protocol (PCEP),” IETF RFC 5440, Mar. 2009.

[저 자 소 개]

이 원 혁 (Wonhyuk Lee)

2001년 2월 성균관대학교 학사
2003년 2월 성균관대학교 석사
2010년 8월 성균관대학교 박사
2003년 3월 ~ 현재
한국과학기술정보연구원
선임연구원

email : livezone@kisti.re.kr

김 승 해 (Seuunhae Kim)

1997년 2월 한남대학교 학사
2003년 2월 전북대학교 석사
2008년 2월 전북대학교 박사
1996년 ~ 현재
한국과학기술정보연구원
선임연구원
첨단연구망서비스실장

email : shkim@kisti.re.kr

김 현 철 (Hyuncheol Kim)

1990년 2월 성균관대학교 학사
1992년 2월 성균관대학교 석사
2005년 8월 성균관대학교 박사
2006년 9월 ~ 현재 남서울대학교
컴퓨터학과 교수

email : hckim@nsu.ac.kr