# IPv6 상에서 QoS 지원을 위한 실시간 처리용 DCP 프로토콜

정회원 조 인 휘*

# The Device Configuration Protocol with Real-Time Processing for QoS Support over IPv6

Inwhee Joe* ***Regular Member***

요 약

최근에는 인터넷 기반의 통신 방식이 원격 제어와 데이터 수집 분야에서도 널리 채택되어지고 있다. 현재의 인터넷에서 대부분의 네트워크 애플리케이션들은 Client-Server 모델로 개발되고 있다.

본 논문에서는 인터넷상에서 Client와 Server의 역할을 교환하여, 다양한 산업용 디바이스를 위해 단일화된 디바이스 인터페이스를 제공해 주는 DCP(Device Configuration Protocol) 프로토콜을 제안한다. 제안된 프로토콜은 TCP/IP 표준 프로토콜 상에서 돌아가는 응용 계층 프로토콜로 동작한다. 또한 DCP 프로토콜을 확장하여 실시간 처리 기능을 추가하고, IPv6의 QoS(Quality of Service) 지원기능인 FlowLabel과 연동하여 실시간 처리가 사용자 수준에서 가능하도록 하였다. 시뮬레이션 수행 결과 실시간 처리가 요구되는 패킷과 일반적인 패킷을 비교하였을 때, 지연시간 측면에서 보면 실시간 처리가 요구되는 패킷은 처리율과 무관하게 마감 시간 전에 처리됨을 알 수 있다.

Key Words : DCP, Real-Time, IPv6, QoS, Application Protocol

ABSTRACT

Recently, the Internet-based communication method has been adopted as an open networking solution in the field of remote control and data acquisition. In the current Internet, most networking applications are developed according to the client-server approach.

In this paper, we propose an innovative Device Configuration Protocol (DCP) that exchanges the traditional role between client and server to provide a uniform device interface over the Internet for various field devices. The proposed protocol is implemented as an application-level protocol running on top of the standard TCP/IP protocols. Also, the DCP protocol is extended with real-time processing to work with the FlowLabel of IPv6 for QoS (Quality of Service) support. The simulation results show that the real-time packets can be processed prior to the given deadline regardless of throughput, as compared to the normal packets.

## Ⅰ. 서 론

인터넷은 새 천년에 들어와서 그간의 텔레커뮤니케이션 기반의 통신 기반구조를 바꾸는 새로운 기반구조로서의 위치를 확보하기 위해 기술적 발전을 거듭했다. 인터넷의 네트워킹 기술은 IETF의 표준화 활동을 통해 새로운 통신 기반으로 인터넷을 발전시키기 위한 방향을 정립하여 진화되고 있다. 이

* 한양대학교 정보통신학부 이동네트워크 연구실(iwjoe@hanyang.ac.kr)
논문번호 : KICS2005-07-273, 접수일자 : 2005년 7월 6일
※본 연구는 한국학술진흥재단 젊은과학자연구(D00174) 지원으로 수행되었습니다.

러한 기술 중 가장 핵심적인 변화를 줄 수 있는 기술은 IPv6 QoS(Quality of Service) 관련 기술 등으로 요약될 수 있다.

현재의 인터넷은 인터네트워킹 기술로서 IP 버전 4를 기반으로 하고 있으나 폭발적인 수요 증가로 인해 주소 공간 부족과 다양한 멀티미디어 응용에 따른 서비스 품질 개선 요구 등과 같은 문제에 직면해 있다. 이 가운데 주소 공간 문제는 현재 인터넷의 IPv4를 자연스럽게 수용하면서 IPv6으로 진화해나감으로써 해결할 수 있을 것이다. 또한 인터넷에서 QoS(Quality of Service) 보장은 Intserv WG과 Diffserv WG에서 정의하고 있는 종합서비스모델과 차등화서비스모델을 수용함으로써 가능할 것이다. Intserv의 종합서비스모델은 RSVP를 이용하여 기존의 최선형서비스 이외에 보장형서비스와 부하제어형 서비스를 제공한다. 또한, DiffServ에서는 종합서비스모델보다 단순화되고 실현이 용이한 차등화서비스 모델을 제안하고 표준화 작업을 계속 진행중이다.

본 연구의 목적은 우선 기존의 인터넷의 산업제어 시스템(PLC, DCS) 및 응용 소프트웨어 전반에 걸쳐 통합적 서버의 기능을 수행할 수 있는 프로토콜을 제안한다. 또한, 이 프로토콜을 확장 개선하여 실시간 처리방식의 스케쥴링 알고리즘을 추가하고 IPv6에서 제공되는 QoS(Quality of Service) 기능인 FlowLabel에서 RSVP를 이용한 자원예약 기능과의 연동을 통해 긴급한 실시간 처리 기능이 요구되는 데이터를 상황에 맞게 처리한다. 특히, 디바이스의 소형화에 의한 비용절감 효과와 인터페이스 단일화에 의한 효율적인 디바이스 제어 및 관리뿐만 아니라, 차후 IPv6 환경으로의 전환 방향의 제시와 더 효율적인 QoS(Quality of Service)의 보장에 의한 통합 서비스 서버환경을 구축하여 신속한 실시간 처리로 인해 보다 나은 서비스의 질적 향상을 기대하는 것이다.

IT(Information Technology)산업의 발전이 우리 생활에 가져온 변화는 일일이 헤아릴 수 없을 정도로 많으며, 산업 전반에 걸쳐 사용되고 있는 필드 디바이스들과 응용 소프트웨어에 있어서도 예외는 아니다.

최근의 추세는 이러한 필드 디바이스들을 원격 접속하여 제어할 수 있는 인터넷 기반의 인터페이스 연구가 활발히 진행되고 있으나, 표준화된 인터페이스가 없으므로 개발 업체들 마다 다양한 방식의 독립적인 인터페이스가 사용되어지고 있고, 기본적으로 디바이스와 서버가 결합되어 있는 방식으로, 이를 사용자(관리자)가 클라이언트 프로그램으로 네트워크를 통해 접근하여 제어하는 방식이 대부분이다.

이러한 방식의 가장 큰 단점은 디바이스가 서버와 결합되어 있기 때문에 디바이스의 크기 증가로 인한 비용 문제와 현재의 추세인 기기의 소형화 및 경량화에 있어서도 역하며, nPNP(Networking Plug and Play)의 지원 불가, 서버 확장의 어려움 등이 있다. 이런 문제들을 해결하고자, 인터넷을 기반으로 하여, TCP/IP 표준 프로토콜 상에서 돌아가는 응용 프로토콜로서, 특히 응용계층에서 Client와 Server의 역할을 교환하여, 다양한 필드디바이스들의 인터페이스를 단일화하기 위해 DCP(Device Configuration Protocol) 프로토콜을 제안한다.

그러므로 본 논문에서는 산업제어 시스템(PLC, DCS) 및 응용 소프트웨어 전반에 걸쳐 통합적 서버의 기능을 수행하기 위해, DCP(Device Configuration Protocol) 프로토콜을 기반으로 한 실시간 처리 기능이 적용된 실시간 Server와 차세대 인터넷 환경의 대안으로 제시된 IPv6의 QoS(Quality of Service) 기능과의 연동으로 다양한 실시간 다중처리가 가능한 보다 확장 개선된 DCP 프로토콜을 제안한다.

본 논문의 구성은 다음과 같다. 제2장에서는 DCP 프로토콜의 개요 및 패킷 구조를, 제3장에서는 DCP프로토콜에 적용된 실시간 처리 알고리즘과, IPv6의 QoS(Quality of Service) 기능을 연동한 확장된 DCP 프로토콜을, 제4장에서는 시뮬레이션을 수행한 결과에 대해 논의를 한다. 마지막으로 제5장에서는 결론을 맺도록 한다.

## Ⅱ. DCP 프로토콜

### 2.1 Device Configuration Protocol 개요

Device Configuration Protocol(DCP)은 TCP/IP를 기반으로 등록된 디바이스를 인식하고, 디바이스에 인터넷상의 존재하는 서버를 알려줌으로써 연결시키는 중계자 역할을 하는 프로토콜이다.

그림 1은 다양한 종류(Mobile Device, Wireless LAN Device, Field Device)의 디바이스들 중에서 인터넷상의 보안 시스템을 위한 DMR(Digital Multiplex Recorder) Service와 Access Controller Service의 테스트 환경을 구축한 전체적인 구성도이다. 기존의 필드 기기 인터페이스와는 반대로 디바이스가 Client가 되고, 관리자의 PC가 Server가 되는 방식으로 디바이스와 Server가 분리되어 있다. 그러므로

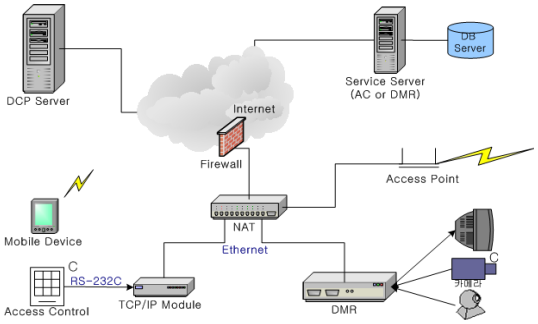


그림 1. DCP Service 인터페이스의 구성

DCP Server가 필요하며, DCP Server에 등록된 디바이스는 먼저 DCP Server로 접속후, 다시 DCP Server가 알려준 주소의 Service Server로 접속하게 되어 Service Server와 디바이스(Client)간의 커넥션이 확정되어 Service가 가능해 진다. 이러한 DCP 인터페이스를 사용하여 서비스를 구현하게 될 경우 디바이스와 Server의 분리로 인하여 디바이스의 소형화 및 경량화, 서버이전 및 확장의 용이성과 비용절감의 효과, nPNP(Networking Plug and Play)의 지원, 동시에 여러 디바이스들이 서버에 접속되어 통합적 관리가 가능함으로서 인터페이스의 단일화로 보다 편하게 종류별 디바이스 제어가 가능하다는 장점이 있다.

그림 2는 DCP 동작 절차를 나타내고 있다. 기존의 필드 기기 인터페이스와는 반대로 디바이스가 Client가 되고, 관리자의 PC가 Server가 되는 방식으로 디바이스와 Server가 분리되어 있다. 그러므로 인터넷상에 존재하는 서버를 찾아가기 위해 DCP Server가 필요하며, DCP Server에 등록된 디바이스는 먼저 DCP Server에 접속하게 되며, 다시 DCP Server가 알려준 주소의 Service Server로 접속하게 되어 Service Server와 디바이스(Client)간의 커넥션이 확정되어 Service가 가능해 지며, 중간에서 DCP Server와 디바이스(Client), Service Server와 디바이스(Client) 간의 제어에 DCP 프로토콜이 사용되어 진다.

실제로 연결이 되는 과정을 살펴보면, 그림 2에서 (1)서비스를 원하는 Client는 DCP packet을 이용하여 DCP Server에게 Service Server의 주소를 요청하면, DCP Server는 인증된 Client인지 확인후, (2)Service Server의 이더넷 주소를 DCP pecket에 실어 Client에 전송하며, (3)주소를 받은 Client는 Service Server로 접속하게 되어 커넥션이 완료된다. 또한 (3)′동시에 다른 디바이스 역시 똑같은 방식

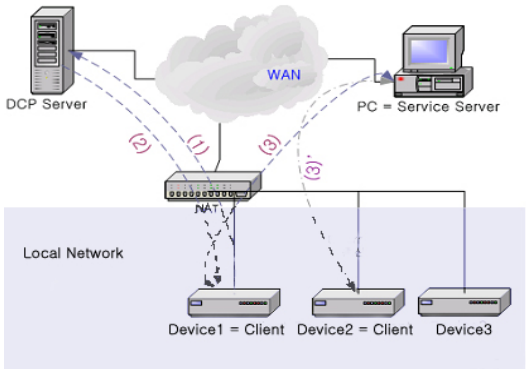


그림 2. DCP 동작 절차

으로 동일한 Service Server에 접속이 가능해 기존 방식의 문제점중 하나인 다중 디바이스 원격 접속 및 제어가 가능해지기 때문에 필드기기의 인터페이스를 단일화 할 수 있게 된다.

이러한 DCP 인터페이스를 사용하여 서비스를 구현하게 될 경우 여러 가지 다양한 장점이 있다.

디바이스와 Server의 분리로 인하여 디바이스의 소형화 및 경량화가 가능하게 되어 디바이스의 제조 단가의 절감으로 인한 타 업체와의 가격 경쟁력에서의 유리함, 서버 이전 및 확장의 용이성, 실시간으로 디바이스를 DCP Server에서 인식하므로 기존의 인터페이스에서는 불가능했던 nPNP(Networking Plug and Play)가 지원되게 되어 어떤 네트워크상에서라도 필요할 때 연결하여 디바이스의 서비스가 가능하며, 서버의 확장에 따른 비용의 절감 등이 가능하다.

무엇보다도 동시에 여러 다양한 디바이스들이 서버에 접속되어 통합적 관리가 가능하게 됨으로서, 보다 편리하게 종류별 디바이스 제어가 가능하며, 그로인해 단일화된 인터페이스를 적용할 수 있다는 장점이 있다.

### 2.2 DCP 패킷 구조

DCP packet은 TCP/IP 프로토콜을 기반으로 확장된 형태로 구성되어 있으며, 디바이스(Client)의 인증 및 Service Server의 주소 요청 및 응답에 사용된다.

* Class Number
10000 = Star 100R
10001 = Keeper M1
10002 = Netcam
10003 = MDU (ATI)

10004 = Evalution
10005 = ND (Network Display)
10006 = DMR (4 ch DVR)
10007 = DTR (1 ch DVR)
10008 = 8052 TCP/IP Evalution
10009 = 16x3 DBC

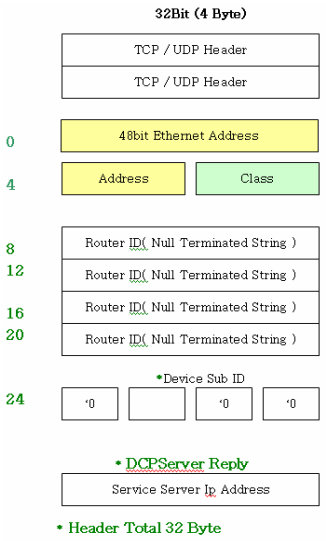


그림 3. DCP 패킷 구조

그림 3은 DCP packet의 구조를 보여주고 있다. 전체 packet의 크기는 32 Byte로 구성되어 있으며, 헤더는 TCP/UDP 헤더는 수정 없이 사용된다. 0 Byte에서 7 Byte는 인증을 위한 디바이스의 이더넷 주소와 각 디바이스의 다양한 서비스의 지원을 위한 종류별 구분 코드에 해당하는 Class 코드(5 Bit)가 포함된다.

8 Byte에서 24 Byte사이는 예전에 사용되어지던 디바이스들의 라우터 표시 관련 기능의 호환성을 지원하기 위해 포함되어 있으나, 현재의 디바이스들의 방식에서는 사용되지 않는다.

마지막으로 DCP Server가 새로운 디바이스로부터 서비스 요청을 받았을때 인증 후 Service Server의 주소를 알려주기 위한 DCP Server Reply에 사용되는 "Service Server IP Adress" 필드가 포함되어 있다.

아래는 디바이스중의 하나인 DMR 서비스에서 실제 Client의 인증을 위해 사용되는 DCP packet이다.

* Client Identification Field

[+0..+5] : Client Identification Number (MAC address사용)
[+6,+7] : Class (항상 10006)
[+8..+23] : Router ID (Null terminated string)
[+24] : unused (0)
[+25] : Device Sub-Id (1..255)
[+26,+26] : unused (0)
[+27..+31] : Service Server IP Address

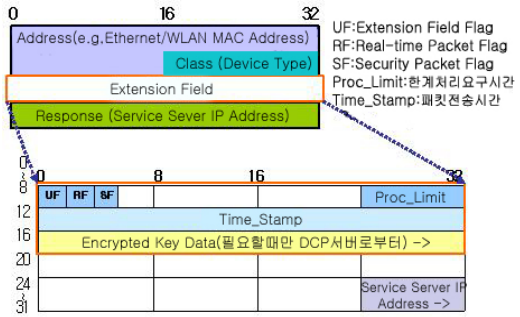


그림 4. DCP 패킷 구조와 확장 필드

## Ⅲ. 실시간 처리용 DCP 프로토콜

### 3.1 DCP 패킷 구조 확장

그림 4는 DCP 패킷 구조에 실시간 처리와 보안을 위한 실시간 처리 관련 필드들이 추가되어 있는 확장된 DCP 패킷 구조를 나타내고 있다.

기존의 DCP 인터페이스 환경과의 호환성을 고려하여, 현재 사용되어지지 않는 잉여 필드인 8 byte에서 24 byte를 활용하였다. Use Flag는 확장필드의 사용여부, Real_Time Flag는 실시간처리 패킷의 사용여부, Security Flag는 암호화 패킷 사용여부, Proc_Limit는 한계 처리요구시간, Time_Stamp는 데이터 전송시의 시간, Encrypted Key는 데이터의 보호를 위한 암호화된 키를 의미한다. 위에서 보았듯이 개선된 DCP 패킷 구조는 실시간 처리기능과 기타 보안기능 등이 추가되었다.

- Use_Flag (UF) : 1 Bit 크기로 개선된 확장 필드 사용 여부를 나타낸다. 기존의 라우터 ID를 사용하는 장비들과의 호환성을 위해 존재하는 플래그이다. TRUE로 세트시 확장필드를 사용한다는 것이며, FALSE 세트시 확장 전 Router ID 기능을 지원하는 기존의 DCP 모드이다.
- Real_Time_Flag (RF) : 1 Bit 크기로 Rea-time 기능의 사용 여부를 나타낸다.(FALSE 세트시

일반적인 데이터 처리 방식) 다중클라이언트에서의 데이터를 동시에 처리할 때 우선순위가 높은 실시간 처리가 요구되는 데이터의 처리를 원할때, 이 필드를 세트하며 그에 따른 Proc_Limit과 Time_Stamp를 포함하여 서버로 보내주며, 서버에서는 이 필드를 확인함으로써 실시간 처리가 요구되는 데이터 여부를 확인할 수 있다.

- Security_Flag (SF) : 1 Bit 크기로 기밀 데이터의 전송시 보안(암호화) 기능 사용할 때 설정한다.
- Proc_Limit : 1 Byte 크기로 RT_Flag 세트시 클라이언트에서 요구하는 데이터 처리요구 한계 시간을 나타낸다. (msec 또는 sec 단위로 필요한 단위로 설정하여 사용).
- Time_Stamp : 4 Byte(INT 형) 크기로 RT_Flag 세트시 클라이언트에서 데이터의 전송을 시작한 시간을 나타낸다. (예: 10시25분40초 => 102540) 기본 조건으로 서버와 클라이언트의 Time 동기화가 필수적으로 선행되어야 한다.
- Encrypted Key data : S_Flag 세트시 암호 및 해독시에 사용되는 KEY 값의 전송이 필요시만 사용되는 필드이다. 암호화된 Key 데이터를 보낸다. (데이터 크기에 따라 가변적)

### 3.2 실시간 처리 알고리즘

그림 5는 실시간적 다중처리가 가능한 개선된 DCP에 적용된 실시간 처리 알고리즘이다.

실시간 처리가 요구되는 데이터는 패킷의 전송시점의 Time_Stamp와 처리 우선순위에 따른 Proc_Limit 값을 포함하여 전송하게 됨으로서 서버에서는 처리 우선순위에 따라 RT_PacketBuffer에 저장되어 최대한 요구시간인 Deadline을 넘어서지 않도록 보장된다. 이는 산업제어 기기나 응용 소프트웨어에서 사용시 긴급 상황이나 실시간적 처리가 요구되는 데이터의 경우에 효율적인 처리로 인한 서비스의 질을 향상 시킬 수 있으며, 다중 클라이언트의 처리시 우선순위에 따른 처리가 가능함으로 효과적인 통합서버의 구성이 가능하다.

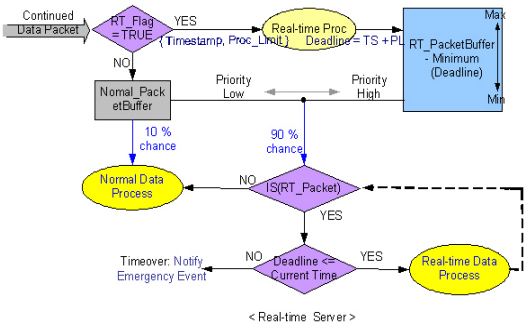


그림 5. 실시간 처리 알고리즘

위 알고리즘은 콘트롤 메시지의 전송부분에 관한 처리를 제외한 데이터 패킷의 처리부분의 실시간 스케쥴링 기법을 나타내고 있다. 다양한 종류의 클라이언트에서의 처리할 데이터 패킷은 항상 수신되며, 이때 Real_Time Flag의 세트 여부에 따라서 실시간 패킷과 일반 패킷으로 나누어 처리하게 된다. 실시간 패킷일 경우 Time_Stamp와 Proc_Limit를 가져와서 두 값을 더하여 Deadline 시점을 구한 뒤 QoS Buffer Scheduler에 따라 더 빠른 처리가 요구되는 데이터 순으로 RT_PacketBuffer에 저장하게 된다. 실제 한 데이터 패킷이 실행되고 있을 때에도 데이터는 항상 수신되므로, 대기중인 리얼타임 데이터들과의 우선순위 비교를 통해 실시간적으로 RT_PacketBuffer는 우선순위에 따라 업데이트 된다. 실시간 데이터 처리부분에서는 항상 RT_Packet Buffer에 데이터의 존재 유무를 확인하며, 만약 RT_PacketBuffer가 비어있다면 비로소 일반 버퍼에 대기하고 있는 일반 데이터가 처리된다. 그러나 만약 Client에서 실시간 처리 기능이 요구되는 데이터들이 매우 많아질 경우 일반 데이터들이 계속적인 기다림으로 인하여 처리가 되지 못한 채 무한히 대기해야 하는 문제가 발생할 수 있다. 이런 문제가 발생할 경우에 대비하여 RT_PacketBuffer와 Normal_PacketBuffer 간의 처리율을 9:1 또는 8:2 정도의 비율로 일반 데이터에도 기회를 제공함으로써 이 문제를 해결 할 수 있다.

실시간 데이터를 처리하기 전 현재의 시간과 비교하여 Deadline 사간의 초과 여부를 확인하고, 만약 초과되었다면, 긴급 메시지 통지 등 그에 따른 적절한 처리를 해준다. 시간의 초과되지 않은 경우 정상적으로 처리되며, 처리된 후 다시 실시간 처리에서 RT_Packet의 존재 여부를 검사하게 된다. 그림 5는 실시간 처리 기능을 적용하여 데이터를 처리할 때의 우선순위에 따른 스케쥴링의 동작 예를 보여준다. A는 실시간 데이터로 period = 2, execution time = 1, priority = 1 (Proc_Limit = 2 sec)일 경우를 가정하며 첫 패킷수신 시작시점은 0초를 의미한다. B는 period = 3, execution time = 1, priority = 2 (Proc_Limit = 1 sec)일 경우를 가

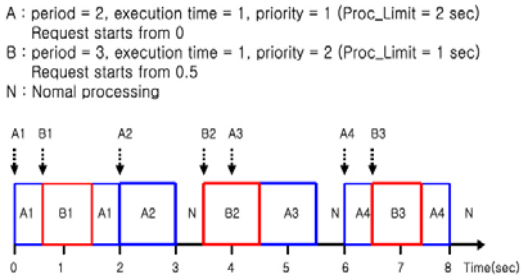


그림 6. 우선순위 스케쥴링 예제

정하며 첫 번째 패킷 수신 시작시점은 0.5초를 의미한다. N은 실시간 데이터의 처리 시간외의 일반 데이터를 처리해 줄 수 있는 시간이다. 0초에서 A1이 수신되었을 경우 우선권을 A1에게 주고 0.5초간 처리 후, 우선순위가 더 높은 B1이 수신되므로 A1은 일단 멈추고 B1을 먼저 처리해준 후 처리가 완료되면 다시 A1에게 우선권을 넘겨 A1이 마저 처리된다. 그리고 A와 B의 데이터 처리가 요구 되지 않는 잉여 시간들은 일반 데이터의 처리를 위해 할당된다.

그림 6에서는 단적인 예를 보여주기 위해 처리시간을 1초로 가정하였으나, 실제 처리시간은 1초 미만이므로 훨씬 더 유동적인 데이터의 실시간 처리가 가능하며, 더불어 처리 효율의 극대화를 위해 잉여 시간은 일반 데이터의 처리에 할당되어 실시간 처리와 더불어 QoS를 보장할 수 있다.

### 3.3 IPv6 QoS 기능과의 연동

현재의 인터넷은 IPv4를 기반으로 하고 있으나 폭발적인 수요 증가로 인해 주소 공간 부족과 다양한 멀티미디어 응용에 따른 서비스 품질 개선 요구 등과 같은 문제에 직면해 있다.

주소 공간 문제는 현재 인터넷의 IPv4를 자연스럽게 수용하면서 IPv6으로 진화해나감으로써 해결할 수 있을 것이며, 인터넷에서 QoS 보장은 Intserv WG과 Diffserv WG에서 정의하고 있는 종합 서비스모델(Flowlabel)과 차등화 서비스모델(Traffic class)을 수용함으로써 가능할 것이다. Intserv의 종합서비스모델은 RSVP를 이용하여 기존의 최선형서비스 이외에 보장형 서비스와 부하제어형 서비스를 제공하는 IPv6의 Flowlabel 필드를 활용하는 것이다.

FlowLabel에서 이용되는 RSVP 프로토콜은 네트워크의 종단간의 경로를 형성함으로써 네트워크 자원을 예약한다. 전송측에서는 필요한 대역폭을 명시한 PATH 메시지를 전송하여 QoS의 레벨을 요청하고, 수신측에서는 전송측에 RESV 메세지를 전송하여 경로에 대한 자원예약을 알린다. RESV 메시지를 통하여 수신측과 전송측 경로의 중간에 있는 (RSVP가 가능한)디바이스들은 네트워크 자원을 사용할 권한과 대역폭을 결정하게 되고, 자신의 네트워크 정책과 사용 가능한 대역폭을 만족한다면 RESV 메시지를 전송측에 전달하게 된다. 전송측이 RESV 메시지를 수신하게 되면 자원예약이 완료되어 경로에 따라 QoS 데이터가 전송될 수 있다.

IPv6의 Flowlabel을 통해 RSVP를 사용하여 원하는 자원을 예약하여 적절하게 사용할 수 있게 된다면, 실시간 처리가 요구되는 데이터를 훨씬 신속하게 처리할 수 있으므로, Flowlabel에서 제공되는 자원예약 기능과 실시간 처리용 DCP 프로토콜의 실시간 Server와의 연동으로 인한 상호 보완 효과로 보다 나은 QoS를 보장 할 수 있으므로 서비스의 질적 향상의 측면에서 큰 의미가 있다.

## Ⅳ. 시뮬레이션

### 4.1 DCP 프로토콜의 실시간 처리 기능 평가

우선 DCP 프로토콜의 실시간 처리 기능을 평가하기 위해 Java 언어를 사용하여 시뮬레이터를 제작하였다.

기본적인 구성으로 실시간 처리 알고리즘을 적용된 간단한 실시간 서버와, 항상 일반 패킷을 전송해주는 일반 클라이언트들과 실시간 처리용 패킷 및 일반 패킷을 전송할 수 있는 실시간 처리용 클라이언트로 구성되어 처리시간을 구할 수 있게 되어있다.

실제 네트워크 상황과 최대한 유사하게 하기 위해 일반 패킷은 일반 클라이언트를 통해 항상 일정하게 전송되고, 실시간 처리용 클라이언트는 패킷량을 점차 증가시키면서 일반 패킷과 실시간 처리용 패킷으로 나누어 서버로 전송한 후, 서버에서 패킷 처리 후까지의 응답시간을 비교 측정이 가능하도록 시뮬레이션 하였다.

실시간 서버를 실행한 후 실시간 처리용 클라이언트에서는 실제 테스트 환경과 유사하게 하기 위하여 항상 지속적인 일반 패킷을 전송하며, 동시에 처리가 요구되는 패킷량을 실시간 패킷과 일반 패킷으로 나누어 전송한다.

더 정확한 비교를 위해서 패킷량을 점차적으로 증가 시키며, 서버에서의 처리 종료 후 완료 메시지를 수신할 때까지의 처리시간을 ms 단위로 측정하였다.

그림 7은 전송된 패킷량에 따른 실시간 처리용

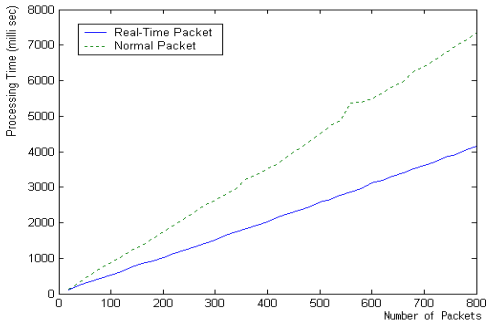


그림 7. 전송된 패킷수에 따른 처리시간 비교

패킷과 일반 패킷의 시뮬레이션 결과를 시각적으로 그래프를 그려 나타낸 것이다.

패킷의 수, 즉 패킷량이 증가 될수록 실시간 처리용 패킷과 일반 패킷에서의 처리 시간 차이는 점점 커짐을 알 수 있으며, 적은 량의 패킷을 전송할 때 보다 큰 데이터 단위로 전송될 때 그 차이는 더 크게 나타남을 보여 준다. 그러므로 여러 클라이언트를 관리해야하는 산업 제어 시스템이나 응용 서버 애플리케이션이 QoS를 보장해주는 실시간적 처리를 위해서는 실시간 처리 기능의 적용이 필수적이라 할 수 있다.

### 4.2 IPv6 상에서의 실시간 처리용 DCP 프로토콜 평가

#### 4.2.1 시뮬레이션 모델

본 논문의 실시간 처리와 IPv6의 Flowlabel의 연동과 수치 측정에서 사용되는 시뮬레이터는 ns-2를 근간으로 하여 Java 언어를 사용하여 제작하였다.

그림 8은 IPv6 QoS 기능인 Flowlabel Field를 사용하여 RSVP 프로토콜로 자원을 예약하면, Destination Node에서 실시간 처리가 적용된 실시간 Server로 IPv6 환경과의 연동을 통해 실시간 처리가 가능한지, 성능을 시뮬레이션하기 위한 전체 네트워크 구성도를 보여준다.

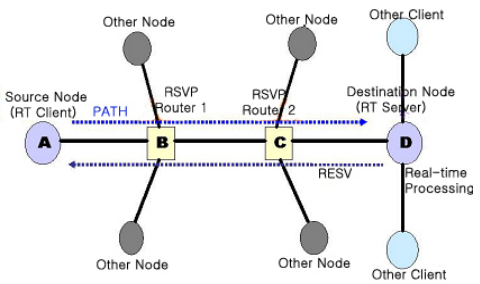


그림 8. 시뮬레이션 네트워크 구성도

표 1. 시뮬레이션 Parameter

| Simulation Parameter | Parameter Value |
|---|---|
| Packet Type | TCP |
| Packet Size | 1 Kbyte |
| Bandwidth | 1.8 Mb, 2 Mb |
| Transfer Rate | 1 Mbps |
| Delay | 1 ms |
| Offered Load Traffic | 120 %, 160 % (excessive) |

기본적인 구성은 Source Node, Destination Node, 자원 예약 기능을 갖춘 중간 Router1, 2 및 다른 Other Node들로 구성된다.

시뮬레이션 parameter는 표 1과 같으며, 대역폭은 중간 Router 1, 2의 링크에 따라 각각 다르게 설정되었고, 일반 패킷과의 비교를 위하여 대역폭을 초과한 과부하 트래픽을 인가하였다.

#### 4.2.2 시뮬레이션 결과

IPv6의 Flowlabel을 통해 RSVP 프로토콜로 자원을 예약한 후 실시간 Server에서의 실시간 처리에 따른 시간을 일반적인 패킷의 경우와 비교하여 측정하였다. 그림 9는 패킷 유형에 따른 처리 지연시간을 측정한 결과이며, 그림 10은 전송된 패킷수의 증가에 따른 처리시간을 측정하여 나타낸 시뮬레이션 결과이다.

그림 9는 과도한 트래픽 상태를 가정하였을 때 Proc_Limit(처리 요구 시간)가 각각 1초, 2초인 실시간 패킷과 일반 패킷의 Throughput에 따른 처리 지연시간을 비교하고 있으며, 일반 패킷의 경우에 지연시간이 점차 증가되었으나, 실시간 패킷일 경우 요구된 처리시간 이내에 처리가 되었음을 알 수 있다. 즉, 실시간 패킷에 대하여 네트워크 관점에서는 IPv6 QoS 기능을 통해 사전에 필요한 자원이 예약되어 있고, 또한 응용 계층에서는 서버내의 실시간 처리 알고리즘을 통해 마감시간 전에 실시간 패킷을 일반 패킷 보다 우선적으로 처리하기 때문에, 요구되는 처리시간을 만족시켜 줄 수 있다.

그림 10에서는 초과된 트래픽량과 패킷수의 증가에 따른 처리시간을 보여준다. 일반 패킷의 경우는 처리시간이 급격하게 증가됨을 알 수 있으며, 반면에 IPv6의 Flowlabel 기능과 실시간 처리 알고리즘을 동시에 적용하여 사용하였을 경우 트래픽량과 패킷수의 증가에 따라 점진적으로 조금씩 처리시간이 증가됨을 알 수 있다.

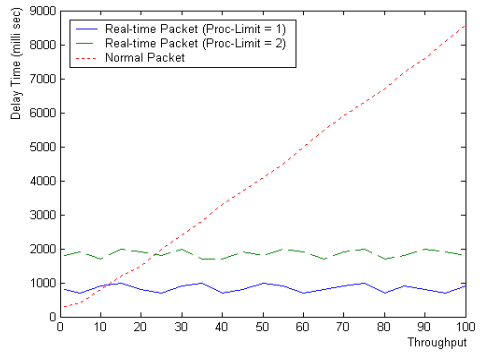


그림 9. 패킷 유형에 따른 처리 지연시간

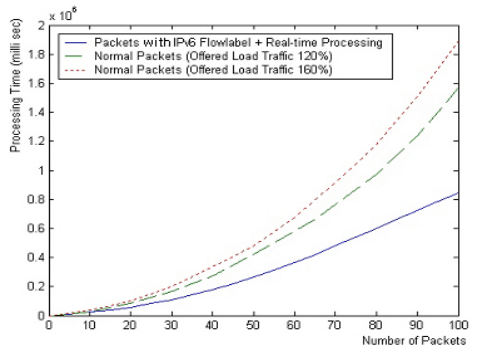


그림 10. 트래픽량과 전송된 패킷수에 따른 처리시간

그러므로 실시간 처리만 적용되었을 때보다 실시간 처리와 종단간의 자원 예약을 함께 사용했을 때 QoS의 보장에 있어서 훨씬 더 효과적임을 알 수 있다. 즉, 응용 계층의 실시간 처리 알고리즘만 가지고는 어느 정도의 QoS 개선 효과는 가능하지만, 궁극적으로 사용자 관점에서의 종단간 QoS 보장은 네트워크에서 자원 예약을 통해 QoS 보장을 위한 기반을 먼저 마련하고, 그 위에 응용 계층의 실시간 처리 알고리즘을 같이 사용할 때 실현 가능함을 알 수 있다.

요약하자면, 위 결과와 같이 IPv6의 Flowlabel 기능과 실시간 처리 기법을 동시에 연동하여 사용함으로써 트래픽의 집중도와 상관없이 항상 일반적인 패킷들보다 월등히 신속한 처리가 가능함을 알 수 있다.

## V. 결 론

본 논문에서는 TCP/IP 표준 프로토콜 상에서 돌아가는 응용 프로토콜로서, 인터넷상에서 다양한 산업용 디바이스(Mobile Device, Wireless LAN Device, Field Device)를 위해, Client와 Server의 역할을 교환함으로써 단일화된 디바이스 인터페이스를 제공해 주는 DCP 프로토콜을 제안하였다.

또한, DCP 프로토콜을 확장하여 실시간 처리 기능의 적용과 IPv6의 QoS(Quality of Service) 지원 기능인 FlowLabel을 연동하여 실시간 처리로 QoS (Quality of Service)를 보장 가능하도록 하였다.

이러한 실시간 처리가 보장되는 DCP 프로토콜을 산업 현장에 적용하게 된다면, 인터페이스의 단일화로 디바이스 독립적인 통합 서버구성이 가능하며, 또한 디바이스의 소형화에 의한 비용절감 효과와 네트워크 상에서 보다 효율적으로 각종 디바이스들의 제어 및 관리가 가능하다. 더욱이, 앞으로는 IPv6 환경으로의 전환시 IPv6의 QoS 기능과의 연동방안을 제시함으로써 보다 나은 실시간 처리로 인한 서비스의 질적 향상이 기대된다.

## 참 고 문 헌

[1] S. Deering, R. Hinden, "*Internet Protocol Version 6 (IPv6) Specification*", IETF RFC 2460, December 1998

[2] X. Tang, J. Tang, G.-B. Huang, C.-K. Siew, "*QoS Provisioning using IPv6 Flow Label in the Internet*", Proceedings of the Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing and the Fourth Pacific Rim Conference on Multimedia, Vol. 2, pp. 15-18, December 2003

[3] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "*RTP: A Transport Protocol for Real-Time Applications*", IETF RFC 1889, January 1996

[4] B. Lloyd, M. Susnik, "*Web Embedded Field Devices*", Conference Record of the 2002 Annual Pulp and Paper Industry, pp. 199-202, June 2002

[5] "*PROFIBUS 서비스 체계의 구축과 전망*", 한국자동화표준시스템연구조합 소식지, 제 26호, April 2001

[6] J. Rajahalme, A. Conta, B. Carpenter, S. Deering, "*IPv6 Flow Label Specification*", IETF Draft (draft-ietf-ipv6-flow-label-09.txt), December 2003

[7] J. Garms, D. Somerfield, "*Processional Java Security*", 정보문화사 출판, 2001

[8] A. Jones, J. Ohlund, "*Network Programming for Windows 2E*", 정보문화사 출판, 2003

[9] J. Postel, J. Reynolds, "*File Transfer Protocol (FTP)*", *IETF RFC 959*, October 1985

[10] R. Simon, C. Diedrich, M. Riedl, M. Thron, "*Field Device Integration*", Proceedings of IEEE International Symposium on Industrial Electronics ISIE, Vol. 1, pp. 150-155, June 2001

[11] E.-B. Fgee, W.J. Phillips, W. Robertson, S.C. Sivakumar, "*Implementing QoS Capabilities in IPv6 Networks and Comparison with MPLS and RSVP*", IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Vol. 2, pp. 4-7, May 2003

조 인 휘 (Inwhee Joe) 정회원

1983년 2월 한양대학교 전자공학과 졸업

1985년 2월 한양대학교 전자공학과 공학석사

1994년 12월 미국 University of Arizona, Electrical and Computer Engineering, M.S.

1998년 9월 미국 Georgia Tech, Electrical and Computer Engineering, Ph.D.

1992년 12월 (주) 데이콤 종합연구소 선임연구원

2000년 6월 미국 Oak Ridge 국립연구소, Researcher

2002년 8월 미국 Bellcore Lab (Telcordia), Research Scientist

2002년 9월~현재 한양대학교 정보통신학부 조교수

<관심분야> Mobile Internet, Cellular System and PCS, Wireless ATM, Mobile Ad-Hoc Networks, Multimedia Networking, Performance Evaluation