Regular paper

Journal of the KIECS. pp. 303-308, vol. 11, no. 3, Mar. 31 2016, t. 71, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2016.11.3.303

전송률 보장을 위한 TCP 트래픽 제어에 관한 연구

이면섭*

A study on the Throughput Guarantee with TCP Traffic Control

Myun-Sub Lee*

요 약

최근 네트워크 기술이 급속도로 발전하고 인터넷 상에서 멀티미디어 서비스 같은 높은 대역폭을 요구하는 서비스들이 증가하면서 네트워크 트래픽이 급속히 증가하고 있다. 이러한 트래픽의 급증으로 인하여 네트워크 자원 부족 및 서비스의 품질 저하등의 문제점들이 발생하고 있으며, 이를 해결하기 위해 QoS(: Quality of Service)를 보장하기 위한 다양한 기법들이 연구되고 있다. 현재 QoS를 보장하기 위한 기법들 중 가장 대표적인 방식이 차등화 서비스(Differentiated Service)이다. 차등화 서비스에서는 AF(: Assured Forwarding) PHB(: Per Hop Behavior)를 정의하고 이를 통해 통계적으로 일정 수준 이상의 전송률(throughput)을 보장하고자 한다. 그러나, 인터넷 트래픽의 대부분을 차지하고 있는 TCP의 혼잡제어 방식은 개별 플로(flow)를 관리하지 않으면서 전송률을 보장하려고 하는 차등화 서비스의 방식과 근본적인 부조화를 보인다. 본 논문에서는 이러한 부조화 현상을 시뮬레이션을 통해 예시적으로 보여주고, 망에서 터미널의 TCP를 제어하여 이를 해결할 수 있는 방안을 제시하였다. 본 논문에서 제안한 방식은 TCP를 제어하기 위해 ACK에 포함되는 수신 윈도우 크기 정보를 조작하며, 이는 현재 사용중인 TCP 알고리즘의 수정을 필요로 하지 않는다.

ABSTRACT

Recently, as the rapid development of network technology and the increase of services required high bandwidth such as multimedia service, the network traffic dramatically increases. This massive increase of network traffic causes some problems such as the degradation of QoS and the lack of network resources and, to solve these problems, various research to guarantee QoS have been performing. Currently, The most representative method to guarantee the QoS is the DiffServ(: Differentiated Service). The DiffServ defines the AF(: Assured Forwarding) PHB(: Per Hop Behavior) and statistically ensures the throughput over the certain level of data rate. However, the TCP congestion control method that make up the majority of the Internet traffic is not fundamentally suitable to the DiffServ that guarantees the throughput without managing the individual flow. Therefore, in this paper, we present this mismatch through the simulation as an example and propose the solution by controlling the TCP of the terminal in the network. The proposed scheme utilizes the information of the reception window size included in the ACK frame and does not require any modification of the TCP algorithms currently in use.

키워드

Differentiated Service, Throughput, TCP, Traffic, Network
차등화 서비스, 전송률, TCP, 트래픽, 네트워크

* 인천대학교 정보기술대학 컴퓨터공학과
• 접 수 일 : 2016. 02. 01
• 수정완료일 : 2016. 03. 13
• 게재확정일 : 2016. 03. 24
• Received : Feb. 01, 2016, Revised : Mar. 13, 2016, Accepted : Mar. 24, 2016
• Author : Myun-Sub Lee
Dept. of Computer Science & Engineering, Incheon University
Email : nantian@inu.ac.kr

Ⅰ. 서 론

인터넷 기술이 발전하고 사용 인구가 늘어나면서 요구되는 서비스의 종류도 다양화되었지만, 현재의 인터넷에서는 사용자에게 최선형(best-effort) 서비스만 지원하고 있다. 그러나, 각 서비스들은 트래픽의 특성에 따라 서로 다른 QoS(: Quality of Service)를 보장받기를 원하는데, 이를 만족시키기 위해 IETF(: Internet Engineering Task Force)에서는 현재의 인터넷에서 QoS를 보장하기 위한 방법들을 제안하였는데, 그 중에서 대표적인 것이 Intserv(: Integrated Service)와 Diffserv(: Differentiated Service)이다. IntServ는 각 플로(flow)별로 요구하는 QoS 파라메터와 현재 상태를 관리하여, 도착하는 모든 패킷에 대해 적절한 처리를 해주기 때문에 비교적 오랜시간 유지되는 애플리케이션에는 의미가 있지만 짧은 기간 동안만 유지되는 애플리케이션에는 적합하지 않으며 사용자가 많아지고 망이 커지면 각

사용자 플로에 따라 비례하는 상태정보의 증가로 인하여 라우터에 과중한 처리를 요구하게 된다.

이러한 단점을 극복하기 위해 DiffServ가 등장하였다. DiffServ는 망과 사용자간에 플로가 형성될 때, 목표전송률이나 최고 목표전송률과 같은 내용의 서비스 수준 협약(SLA : Service Level Agreement)을 하고 이 협약을 이행하는 방법과 규칙으로 PHB(: Per-Hop Behavior)와 DSCP(: DiffServ Code Point)를 사용하여 망의 경계에서는 복잡한 기능을 처리하고 망의 내부에서는 경계에서 처리되어 이행할 규칙을 담은 DSCP를 바탕으로 PHB에 의해 서비스를 제공한다. 하지만, DiffServ에서는 모든 플로들을 몇 개의 클래스로 묶어 같은 클래스에 속하는 패킷들은 플로 구분 없이 똑같이 취급함으로써 이상적인 QoS 보장이 더 어려워졌다[1-2]. 대표적인 경우가 TCP 트래픽에 대해 AS(: Assured Service)로 평균 전송률을 보장하고자 하는 경우인데, 이는 TCP의 혼잡제어(congestion control) 알고리즘이 전송률을 보장하고자 하는 시스템에 맞지 않기 때문이다. 즉, TCP의 전송률은 송신 윈도우의 크기와 RTT(: Round Trip Time)에 의해 결정이 되고, 송신 윈도우의 크기는 주로 망의 혼잡 정도와 플로가 지나는 병목구간의 수에 의해 결정되므로, 전송률을 보장하기 위해서는 플로별로 이러한 요소들을 고려해 주지 않으면 어려운데, DiffServ의 망 내부에서는 플로를 구분하지 않으므로 이러한 요소들을 고려해 주는 것이 근본적으로 불가능한 것이다. TCP와 같이 윈도우 기반의 혼잡제어 알고리즘을 사용하는 트래픽에 대해서는 이런 문제가 불가피하게 발생할 수밖에 없는데, 그렇다고 해서 이 문제를 해결하기 위하여 TCP 알고리즘을 수정할 수는 없다.

이 문제는 한 클래스 전체에 할당된 대역폭을 각 플로에 공평하게 분배하는, 플로간의 공정성 문제로 볼 수도 있다. RED(: Random Early Detection)나, 이의 변형된 형태인 RIO(: RED In/Out), MRED(: Modified RED) 등을 사용하여 플로간의 공정성을 향상시킬 수 있다는 연구 결과가 있으나, 이 역시 전송률을 직접 제어하거나, 전송률을 결정하는 요소(RTT 또는 지나는 병목구간의 수)를 직접적으로 고려하는 것이 아니기 때문에 그 효과에 한계가 있다[3]. 4장의 시뮬레이션 결과에서 그러한 예를 보여주고 있다. 더욱이 이러한 방식을 사용하는 경우, 다양한 환경에서 최적의 파라메터를 찾아내는 일이 상당히 어려운 문제로 제기되고 있다.

본 논문에서 제안하는 방식은 RED나 그 변형 알고리즘과 같이 패킷 폐기를 통해 간접적으로 TCP를 조정하는 것이 아니라, 명시적인 방법으로 TCP 소스에서의 트래픽 발생량을 직접 제어하여 전송률을 조정한다. 즉, AS에 속하는 TCP 소스들은 계약된 전송률에 따라 정해진 양의 트래픽만을 발생하고, 망의 내부에서는 패킷손실이 발생하지 않도록 보장해 발생한 트래픽이 가능한 한 패킷 손실 없이 전송되도록 해 문제를 해결하였다. TCP 소스를 제어하는 것은 TCP 알고리즘을 직접적으로 수정하지 않고, 망의 leaf 노드에서 ACK의 TCP 헤더에 포함되는 가용 수신윈도우 크기 정보를 조작함으로써 구현할 수 있다[2].

2장에서 본 논문에서 제안하는 알고리즘을 자세히 설명하였고, 3장에서는 시뮬레이션 모델을 보였다. 4장에서 RIO 알고리즘을 이용한 시스템과 제안된 알고리즘을 이용한 시스템에서의 결과를 비교하고, 마지막 5장에서는 개선할 점과 앞으로의 연구방향을 소개하고 결론을 맺었다.

II. 제안하는 알고리즘

본 논문에서 제안하는 시스템의 구성은 그림 1과 같다. 그림 1에서 Routing Module에서는 일반적인 라우터에서 행하는 라우팅과 포워딩을 수행한다.

Throughput Meter(: TM)에서는 호스트로부터 망으로 흘러 들어가는 트래픽의 양을 측정하여, 이 양과 요구 전송률을 비교해 그 결과에 따라 TCP Controller(: TCPC)를 제어한다.

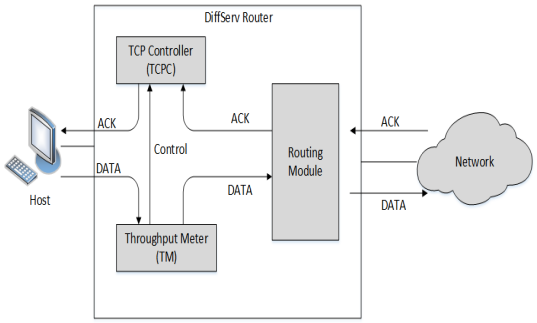

그림 1. 시스템 구성

Fig. 1 System architecture

TCPC에서는 Routing Module로부터 ACK를 받을 때마다 이로부터 RTT를 계산하고, 평균적인 RTT를 추정하여 필요한 송신 윈도우 크기를 계산한다. RTT 추정은 TCP의 RTT 추정 알고리즘과 같은 것을 사용하며, 이로부터 필요한 송신 윈도우 크기는 다음과 같이 계산된다.

$$RTT_{smoothed} = (1-\alpha)RTT_{smoothed} + \alpha RTT \quad (1)$$

$$S_{win} = R_{req} RTT_{smoothed} \quad (2)$$

위 식에서 RTT는 현재 받은 ACK로부터 계산된 RTT이고, $RTT_{smoothed}$가 실제 사용할 평균화된 RTT 값이다. R_{req}는 요구 전송률이고 S_{win}이 필요한 송신 윈도우 크기가 된다. α는 0에서 1 사이의 상수이다.

계산된 값은 마지막으로 TM의 제어를 받아 적절하게 보정된 후 TCP 헤더의 Window size 필드에 기록되어 호스트로 보내진다. 이때 헤더의 체크썸은 다시 계산되어야 한다. TM은 현재 실제로 전송하고 있는 속도와 요구 전송률을 비교해 현재 전송률이 크면 값을 감소시키고, 현재 전송률이 작으면 값을 증가시킨다. 두 값이 같으면 계산된 값을 바꾸지 않는다. 최종적으로 결정된 값이 원래 Window size 필드에 기록되어 있던 값보다 큰 경우는 현재 수신 단말에서 수용할 수 있는 속도보다 요구 전송률이 더 큰 경우이므로 원래의 값을 수정하면 안 된다. 그림 2에 이 과정이 정리되어 있다.

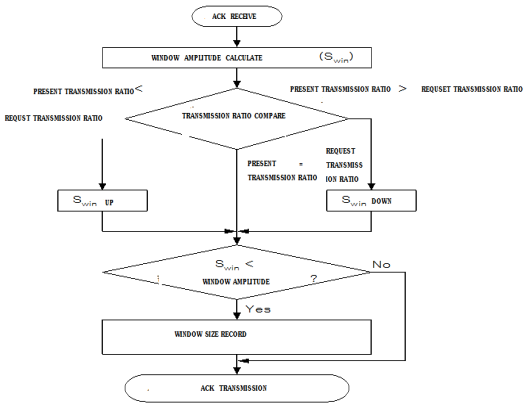

그림 2. TCPC 동작 알고리즘

Fig. 2 TCPC operation algorithm

그림 2를 참조하면, 처음 전송을 시작할 때 송신 TCP는 패킷 손실이 없는 한 혼잡제어 윈도우(congestion window) 크기를 꾸준히 증가시킨다. 실제 송신 윈도우의 크기는 혼잡제어 윈도우와 ACK에 기록된 수신 윈도우 크기 중 작은 것으로 결정되므로, 위의 알고리즘을 사용할 때, 실제 수신 단말의 처리 속도가 충분히 빠르고 망에서의 패킷 손실이 거의 없다면 송신 단말은 원하는 송신 윈도우 크기를 꾸준히 유지하게 된다. 패킷손실을 최소화하는 것은 망이 가지고 있는 사용 가능 자원의 양에 맞추어 접속 제어(admission control)가 이루어지고, 필요한 자원이 충분히 예약되었다는 가정하에 가능하다. 때때로 발생하는 패킷손실에 의한 전송률 저하는 TM의 제어에 의해 다시 회복되게 된다.

ACK의 수신 윈도우 크기 정보를 조작해 TCP 소스의 전송률을 제어하는 방법은 이미 다른 연구에서 사용된 바가 있으나 사용 목적에 있어서 본 논문에서와는 차이가 있다[6].

III. 시뮬레이션 모델

시뮬레이션에 사용한 토폴로지는 그림 3과 같이 backbone router에 leaf router가 연결되고, 거기에 다시 호스트가 연결된 일반적인 형태이다.

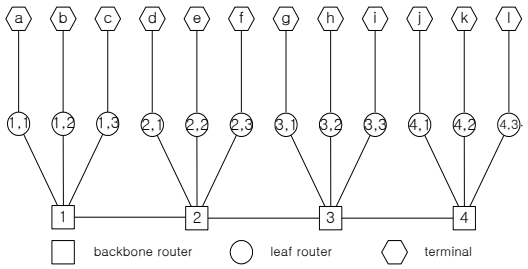

그림 3. 시뮬레이션에서 사용된 네트워크 토폴로지
Fig. 3 A network topology used in the simulation

AS 트래픽 전체에 대한 요구 대역폭이 모든 링크에 독점적으로 할당되고 다른 클래스와 독립적인 큐를 사용한다는 가정하에, PS(: Premium Service)와 BS(: Best effort Service) 트래픽은 시뮬레이션에서 고려에서 제외할 수 있다. 시뮬레이션에 사용된 플로들은 모두 같은 클래스에 속하는 것으로 하였다. 단순화를 위해 하나의 leaf router에는 하나의 호스트만을 연결하였다. 각 플로가 backbone 링크를 지나는 모양은 그림 4에 나타난 것과 같다. 그림에서 한 화살표가 하나의 플로를 나타낸다[1].

시뮬레이션은 NS2(ns-2.1b6)와 Murphys의 DiffServ patch를 이용하였고, 사용한 파라메터는 다음과 같다[5].

1) The Network Simulator - ns-2 http://www.isi.edu/nsnam/ns/ February 1999.

표 1. 시뮬레이션 파라메터
Table 1. Simulation parameters

Parameter	Value
window size of Max. TCP	20 packets
packets size	1000 Byte
transfer delay of link	10ms(All)
Link buffer	1200Kbps(All)
token rate of traffic conditioner	250Kbps
buffer size of each node	100 packets

표 2. RIO 파라메터
Table 2. RIO parameters

Parameter	Value
queue weight	0.002
Min_th for in profile packets	45
Max_th for in profile packets	90
Maxp for in profile packets	0.02
Min_th for out of profile packets	20
Max_th for out of profile packets	40
Maxp for out of profile packets	0.1

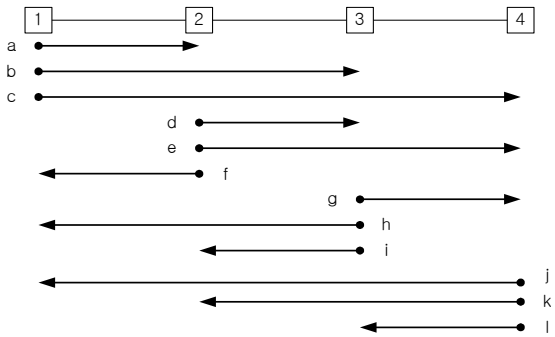

그림 4. backbone에서의 플로들의 흐름
Fig. 4 The data flow of flows in the backbone

위의 파라메터 중 트래픽 컨디셔너의 token rate가 보장하고자 하는 전송률이 된다. 트래픽 컨디셔너는 이 속도를 넘어가는 패킷에 대해서는 손실 우선 순위를 높게 표시한다[8-9]. 세 개의 병목 링크 중, 가장 많은 플로가 지나고 있는 가운데 병목 링크에 4개의 플로가 흐르므로, 클래스 전체의 요구 대역폭은 1000Kbps인데, 실제로는 1200Kbps의 대역폭이 예약되어 있으므로 20%의 추가 대역폭이 여분으로 예약된 것이다.

시뮬레이션 시간은 300인데, 처음 30초 동안은 측정을 하지 않았다.

Ⅳ. 시뮬레이션 결과

4.1 TCP 혼잡제어 알고리즘과 RIO를 결합한 모델에서의 결과 기존의 TCP 혼잡제어 알고리즘과 RIO를 결합한 모델에서의 시뮬레이션 결과는 그림 5와 같다. 그래프에 0으로 표시된 시간이 측정을 시작한 시간이다. 가독성을 위해 시간에 따른 전송률 대신, 수신 데이터량을 비트수로 나타내었다.

측정된 전송률을 정리하면 표 3과 같다. 3장에서 예상했던 것과 같은 결과가 나왔음을 알 수 있다. 필요 대역폭의 120% 이상의 대역폭을 예약했음에도 불구하고, 12개의 플로 중 6개의 플로가 요구 전송률을 보장받지 못하고 있고, 이 중 2개는 요구 전송률의 절반 정도의 전송률 밖에 서비스 받지 못하고 있다.

표 3. TCP 혼잡제어 알고리즘과 RIO를 결합한 모델에서의 플로 그룹별 전송률

Table 3. The flow group rates in the model that combines the TCP congestion control algorithm and RIO

Flow	Throughput(kbps)
a, f, g, l	641.6, 671.9, 635.7, 627.1
d, i	346.2, 338.3
b, e, h, k	223.5, 238.1, 221.6, 231.5
c, j	193.7, 186.5

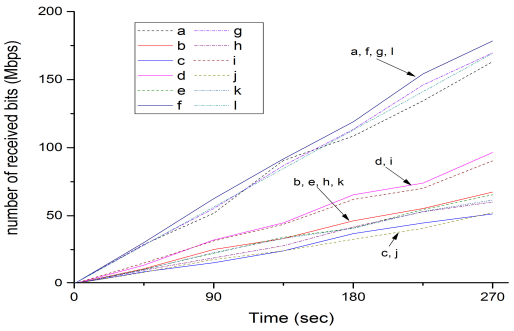

그림 5. TCP 혼잡제어 알고리즘과 RIO를 결합한 모델에서의 시뮬레이션 결과

Fig. 5 Simulation result in the model that combines the TCP congestion control algorithm and RIO

4.2 제안된 알고리즘을 사용한 모델의 결과

제안된 알고리즘을 사용한 모델에서의 시뮬레이션 결과를 그림 6과 표 4에 보였다.

표 4. 제안된 알고리즘을 사용한 모델에서의 전송률

Table 4. The throughput in the model that uses the proposed model

Flow	Throughput (kbps)	Flow	Throughput (kbps)
a	254.9	g	255.0
b	255.0	h	255.0
c	254.9	i	255.0
d	254.9	j	255.0
e	255.1	k	255.1
f	255.1	l	255.0

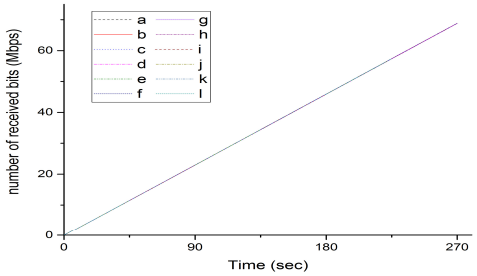

그림 8. 제안된 알고리즘을 사용한 모델에서의 시뮬레이션 결과

Fig. 6 Simulation result in the model that uses the proposed algorithm

그림 6을 참조하면, 플로별로 전송률의 차이가 거의 없고 모든 플로가 보장하고자 하는 250kbps 전송률을 아주 작은 오차로 만족시키고 있음을 알 수 있다.

Ⅴ. 결 론

이상에서 본 바와 같이 본 논문에서 제안한 방식은 요구 전송률을 만족시키기 위한 접속제어와 자원 예약이 되어 있다는 가정하에 요구되는 전송률을 완벽하게 만족시키고 있지만, 다음과 같은 두 가지 단점을 가지고 있다.

첫째, ACK 정보를 조작하려면 TCP 패킷 헤더를 라우터에서 열어본 후 필드를 수정하고 체크섬을 다시 계산해야 하는 추가적인 오버헤드가 발생한다. 이를 해결하기 위한 방법으로 ACK 정보를 조작하는 대신 ACK 패킷 자체를 shaping하는 방법을 생각할 수 있다. TCP 송신 윈도우가 가득 차면 ACK가 도착할 때마다 패킷이 전송되므로, ACK를 shaping함으로써 소스의 전송률을 조절할 수 있을 뿐만 아니라, 소스 트래픽 자체를 shaping하는 효과도 줄 수 있다. 따라서, 원래의 트래픽 컨디셔너의 필요성이 없어진다. 즉, 데이터 패킷이 망으로 유입되는 부분에 놓여 있던 컨디셔너를 ACK 패킷이 망을 빠져나가는 부분으로 옮겨 놓는 것만으로 충분하므로 추가적인 오버헤드가 없어진다.

둘째, 제안된 시스템은 망에 여분의 대역폭이 남아 있을 경우에도 이를 이용하지 않는다. 물론 PS와 AS 트래픽이 이용하고 남는 대역폭은 모두 BS 트래픽이 이용할 것이므로 반드시 남는 대역폭이 낭비된다고 말할 수 없으나, AS 트래픽에 비해 BS 트래픽이 훨씬 적은 경우에는 오히려 AS 보다 BS 트래픽이 더 나은 서비스를 받는 경우도 발생할 수 있을 것이다. 이를 해결하기 위해서는 망 상태를 감지해 이에 반응하도록 할 필요가 있다. 그 방법으로 실제 전송률이 요구 전송률보다 떨어지는 플로에 대해서만 여기서 제안된 방식을 적용하는 것을 생각해 볼 수 있다. 이렇게 함으로써 AS가 확률적으로 BS보다 나은 서비스를 받으면서, 최소한 요구 대역폭 이상을 전송률은 보장받을 수 있게 된다.

이 두 가지 문제에 대해서는 좀 더 연구가 필요할 것으로 생각된다.

감사의 글

이 논문은 인천대학교 2014년도 자체연구비 지원에 의하여 연구되었음

References

[1] K. Nichols and A. Kyle, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.“ RFC 2474, December 1998.

[2] S. Blake and A. Tveit, “An Archetecture for Differentiated Services.“ RFC 2475

[3] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” *IEEE/ACM Trans. On Networking*, vol.1 no.4, August, 1993, pp. 1209-1216.

[4] D. stiliadis and A. Varma, “Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithm,” IEEE/ACM Trans. Netw., vol6, no.5, Oct. 1988, pp. 611-624.

[5] S. Lee, “Video Flame Detection with Periodicity Analysis Based False Alarm Rejection,”*J. of the Korea Institute Of Electronic Communication Sciences*, vol. 6, no. 4, Aug., 2011, pp. 479-485.

[6] K. Kim, K. Ban, S. Heo, and E. Kim, “Design and Implementation of System for Sensing Data Collection in RFID/USN,” *J of the Korea Institute Of Electronic Communication Sciences*, vol. 5, no.2, April 2010, pp. 221-226.

[7] B. Kim, “Service Quality Criteria for Voice Services over a WiBro Network,” *J. of the Korea Institute Of Electronic Communication Sciences*, vol. 6, no.6, Dec., 2011, pp. 823-829.

[8] R. Cao and L. Yang, “The affecting factors in resource optimization for cooperative Communications: A case study,” *IEEE Trans. Wireless Communication*, vol. 11, no. 12, Dec. 2012, pp.4351- 361.

저자 소개

이면섭(Myun-Sub Lee)

1985년 국민대학교 전자공학과 졸업 (공학사)
1987년 인하대학교 대학원 전자공학 졸업(공학석사)
2005년 국민대학교 대학원 전자공학과 졸업(공학박사)
2016년 인천대학교 정보기술대학 컴퓨터공학 교수
※ 관심분야 : 이동통신시스템, 센서네트워크통신