Journal of the Korea Academia-Industrial cooperation Society
Vol. 14, No. 7 pp. 3468-3473, 2013

http://dx.doi.org/10.5762/KAIS.2013.14.7.3468

변형된 Dijkstra 알고리즘을 활용한 오버레이 멀티캐스트 QoS 향상 기법 연구

이형옥[1*], 남지승[1], 박준석[2]
[1]전남대학교 전자컴퓨터공학부, [2]한국전자통신연구원

A Study on QoS Improvement for Overlay Multicast Using Modified Dijkstra Algorithm

Hyung-Ok Lee[1*], Ji-Seung Nam[1] and Jun-Seok Park[2]
[1]**Division of Electronics Computer Engineering, Chonnam National University**
[2]**Electronics and Telecommunications Research Institute**

요 약 본 화상 회의, 인터넷 방송 등의 실시간 응용 시스템을 위해 오버레이 멀티캐스트 트리가 충족 시켜야 하는 조건은 크게 두 가지이다. 첫 번째는 호스트, 즉 트리 상의 노드의 차수가 적절한 상한이어야 한다는 것이고 두 번째 조건은, 멀티캐스트 트리의 지름, 즉 트리 상의 경로 거리로 볼 때 가장 먼 두 사용자 간의 거리가 작아야 한다는 것이다. 각 노드의 차수가 상한을 가지고 있을 때 지름이 최소인 트리를 구성하는 문제는 NP-complete 문제로 알려져 있다. 본 논문에서는 Dijkstra 알고리즘의 Cost값을 트리를 구성하는 노드들의 가용대역폭, Delay를 체크한 후, 제안하는 Score Function을 적용하여 계산한다. 그리고 그 값을 Dijkstra 알고리즘에 적용하여 트리를 구성하고 시뮬레이션을 통해 성능평가를 하였다.

Abstract Conditions that overlay multicast tree must satisfy for the real-time application system of a video-conference, an internet broadcasting is two things. First, the degree of nodes in a tree must be proper value. Second, the diameter of the multicast tree, distance between longest two users should be short. If the path between two users in the tree is long, the delay time in data transmission between two users great. So, it is not suitable to the application system such as video-conferences. In this paper, the cost of the dijkstra algorithm calculate with proposed score-function through checking the extra bandwidth, the delay and the requested bandwidth. It is composed the tree through the dijkstra algorithm.

Key Words : Degree of node, Dijkstra algorithm, Overlay multicast

1. 서론

인터넷 방송, 화상 회의 등의 집단 간의 통신을 위해서는 효율적이고 확장 가능한 멀티캐스트 메커니즘이 필요하다. 수년 전 까지 IP 멀티캐스트가 이를 위한 적절한 메커니즘으로 여겨져 왔다. 그러나 IP 멀티캐스트는 라우터의 구현, 혼잡 제어와 신뢰성 있는 전송에서 여러 가지 문제점에 맞닥뜨리게 되었으며, 그에 대한 대안으로 오버레이 멀티캐스트(Overlay Multicast) 방법이 제안되었다.

IP 멀티캐스트에서 패킷의 복제와 전송을 라우터가 담당하는 반면, 오버레이 멀티캐스트에서는 단말의 사용자가 패킷의 복제와 전송을 담당하게 된다. 즉 단말 사용자들로 이루어진 응용 계층에서 멀티캐스트 트리를 구성하여 그 트리의 경로를 통해 패킷을 전송한다. 각 사용자는

본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학IT연구센터육성 지원사업의 연구결과로 수행되었음" (NIPA-2013- H0301-13-1006)
*Corresponding Author : Hyung-Ok Lee(Chonnam Univ.)
Tel: +82-62-530-0422 email: narcis99@nate.com
Received April 5, 2013 Revised May 27, 2013 Accepted July 11, 2013

패킷을 받은 후에 트리 상의 다른 이웃 단말 사용자에게 패킷을 전송하게 된다. IP 멀티캐스트는 구현을 위해 라우터의 개선이 필요한 것에 반해, 오버레이 멀티캐스트는 각 사용자의 응용 프로그램만 갖추어지면 구현될 수 있다는 장점이 있다. 반면 오버레이 멀티캐스트는 IP 멀티캐스트에 비해 지연시간이나 대역폭 사용의 측면에 있어서 비효율적이라는 단점이 있다. 오버레이 멀티캐스트에서는 최대한 이런 비효율성을 줄이는 멀티캐스트 트리를 구성하는 것이 중요한 목표이고, 실시간 방송 서비스에서 사용자들에게 QoS(Quality of Service)를 보장하기 위해서 반드시 극복해야 할 문제점인 것이다.

본 논문에서는 각 호스트들의 제한된 자원과 네트워크 환경을 고려하여 실시간 방송 서비스에 적합한 오버레이 멀티캐스트 트리를 구성하는 알고리즘을 제안하고자 한다. 이 알고리즘은 각 노드들의 가용대역폭과 Delay를 제안하는 Score Function에 적용한 후, Dijkstra 알고리즘을 사용하여 최적의 트리를 구성하게 된다.

2. 본론

2.1 제약을 둔 라우팅 구조하의 트리구성 알고리즘

오버레이 멀티캐스트는 네트워크의 구성에 영향을 받지 않으며 네트워크를 전개해 나감에 있어서도 가속이 붙는다. 이러한 사실은 오버레이 멀티캐스트의 경우 모든 패킷들을 유니캐스트 패킷처럼 전송하기 때문이다. 종단 시스템(End System)에 의해 네트워크의 “Stateless" 상태를 포함하기 때문에 작은 수의 그룹만을 관리하면 되므로 그룹의 추가 시 그룹을 관리하기 위한 복잡한 프로세싱을 수행할 수 있다[1]. 다시 말하면 라우팅의 오버레드가 호스트들에게 분담되어서 수행되어 진다. 또한 에러제어, 흐름제어, 적체제어 등의 상위 계층에서 요구되어지는 사항을 쉽게 간략화 할 수 있다[2]. 라우터들에게 에러제어나 흐름제어, 적체제어를 할 수 있는 기능을 호스트의 응용계층에서 지원을 하여 가능하게 할 수 있다[3]. 오버레이 멀티캐스트는 효과적으로 멀티캐스트의 단점을 극복하였지만 항상 일정한 장소에 고정되어 서비스를 제공하고 있던 라우터의 기능을 일반 사용자 컴퓨터에 옮기다 보니 멀티캐스트 트리 구성 시 관리자가 예측할 수 없는 노드들의 빈번한 이탈과 서로 다른 시스템 성능에 따른 전송 지연(End-to-End Delay)과 같은 새로운 문제가 발생하게 되었다[4]. 이러한 문제점들을 해결하기 위해 여러 제약을 둔 오버레이 트리구성 알고리즘들이 제안되어 왔다.

가. WSP(Widest Shortest Path) [5]

WSP 알고리즘은 가능한 경로들 중 hop-count가 가장 작은 경로를 선택하는 방법이다. 만약 그런 경로가 여러 개 존재하면, 그중 가장 큰 대역폭을 가진 경로를 선택하게 된다. 그리고 대역폭이 같은 경로가 있으면 랜덤하게 선택하게 된다.

나. SWP(Shortest Widest Path) [6]

SWP 알고리즘은 가능한 경로들 중 대역폭이 가장 큰 경로를 선택하는 알고리즘이다. 같은 대역폭을 가진 경로들이 있을 경우에는 그 중 가장 작은 hop-count를 가진 경로를 선택한다. 또한 hop-count가 같은 경우에는 랜덤하게 선택한다.

다. SDP(Shortest Distance Path) [7]

SDP 알고리즘은 가장 작은 Distance를 가지는 경로를 선택한다. 이 Distance를 구하는 함수는 다음과 같다.

주어진 경로 $p = \{i_1, \ldots, i_k\}$이고, 링크 i_j의 가용대역폭이 R_{i_j}이라고 할 때, Distance를 구하는 공식은 $dist(p) = \sum_{j=1}^{k} \frac{1}{R_{i_j}}$ 이 된다.

2.2 Score Function

Dijkstra 알고리즘은 가중치가 있는 그래프의 최단 경로를 구하는 알고리즘이다. 출발 정점에서 시작하여 현재의 정점까지의 값과 인접한 정점의 가중치 합이 가장 작은 정점을 다음 정점으로 선택하고 그 경로를 최단 경로에 포함 시킨다.

이 과정을 모든 정점이 선택될 때까지 반복한다. 그리고 시작점에 연결되어 있는 정점 사이의 거리를 구해서 최소값을 갖는 정점에 표시한다. 표시를 해둔 정점에 연결되어있는 각 정점까지의 거리를 구하고, 이 때 계산된 정점(표시되어 있지 않은) 사이의 거리 중에서 최소값을 갖는 정점에 표시한다. 이 과정을 모든 정점에 표시할 때까지 반복하면, 각 정점에서 얻을 수 있는 값이 곧 시작점에서의 최단거리를 뜻하게 된다.

오버레이 멀티캐스트에 Dijkstra 알고리즘을 사용하면 노드간의 최단거리, 즉 최적의 Path를 얻을 수 있다. 그러나 Dijkstra 알고리즘은 노드들의 out-degree를 고려하지 않기 때문에 각 호스트들의 대역폭 문제를 해결할 수 없고, 그로 인해 네트워크 트래픽이 증가하게 된다[8]. 이는

실시간 방송 서비스에서 반드시 극복해야하는 문제점인 사용자들에 대한 QoS(Quality of Service)를 보장할 수 없게 한다. 본 논문에서는 이 문제를 해결하기 위해, 각 노드들의 대역폭과 노드들 간의 delay를 이용한 Score-function을 제안 한다.

전체 네트워크를 그래프로 $G(N, E)$로 표현하였을 때, N는 각각의 호스트를 나타내고 E는 호스트들 간의 링크를 나타낸다.

한 노드 i에서의 가능한 Out-degree는 다음과 같이 표현할 수 있다.

$$ {}_{ava}D_i = \left\lfloor \frac{\gamma \bullet {}_{\max}BW_i - {}_{use}BW_i}{{}_{req}BW} \right\rfloor \quad (0 < \gamma \leq 1) $$

- ${}_{ava}D_i$ = 노드 i의 Out-degree
- ${}_{\max}BW_i$ = 노드 i의 최대 대역폭
- ${}_{use}BW_i$ = 현재 사용 중인 대역폭
- ${}_{req}BW$ = 노드 i로 요청되는 대역폭

그리고 γ 값은 QoS를 위해서 정해놓은 값으로써 한 노드의 최대 대역폭을 조정할 수 있는 변수이다.

한 호스트의 Delay는 다음과 같이 정의하였다.

$$ RTT_i = \sum_{j \in f_i} RTT_{ij} \quad (f_i = i\text{노드와 인접한 노드}) $$

- RTT_{ij} = 노드i와 노드j사이의 RTT값
- RTT_i = 노드i와 인접한 모든 노드들 간의 RTT값의 합

여기서 노드 i 는 인접하고 있는 모든 노드들과의 RTT 값을 구하고 합한 값을 저장한다.

위의 두 식을 사용하여 다음 Score-function을 유도하였고, 이는 노드의 대역폭과 Delay를 고려하여 라우팅을 가능케 한다.

$$ Func(i) = \alpha\left(\left\lfloor \frac{\gamma \bullet {}_{\max}BW_i - {}_{use}BW_i}{{}_{req}BW} \right\rfloor \right) + (1-\alpha)\left(\frac{1}{RTT_i}\right) \quad (0 < \alpha \leq 1) $$

$Func(i)$의 값이 높을수록 가용대역폭이 많고, 딜레이가 적은 호스트를 뜻한다. 이 값은 각 링크의 가중치 값이 되고, Dijkstra 알고리즘을 통해 트리를 구성할 수 있다. 여기서 네트워크 상황과 응용분야에 따라 중요시되는 것이 대역폭일수도 Delay일 수도 있기 때문에 각 상황에 맞게 α값을 조정해야 할 필요가 있다.

2.3 Modified Dijkstra 알고리즘

다음은 Modified Dijkstra 알고리즘의 Pseudo Code이다.

```
Dijkstra (G, w, s)
    for (V[G]안의 모든 정점 v)
        d[v] = ∞
        previous[v] = Null
        d[s] = 0
        OutDegree = i (i는 임의의 상수)
    Q안에 모든 정점을 입력

    While (Q =!∅)
        u를 Q에서 빼낸다.
        (u = Q안에 있는 값 중 가장 작은 값)
        for (u에 인접한 정점 v를 선택, v는 Q
        안에 있는 정점으로 c(u, v)값이 작은
        순으로 선택)
            if (OutDegree = 0)
                d[v] = ∞, v를 Q안으로 넣는다.
                OutDegree = i
            else if ( d[v] > d[u] + c(u, v) )
                d[v] = d[u] + c(u, v)
                previous[v] = u
                OutDegree --
```

```
S = 모든 정점들의 집합
Q = 우선순위 큐
OutDegree = 제한하는 차수
c(u, v) = 정점 u, v사이의 Cost
d[v] = 정점 v의 Label, s = source
previous[v] = source에서 정점 v까지의
              Shortest Path에서 v 이전의 정점
```

2.4 성능평가 및 결과분석

2.4.1 시뮬레이션 환경

본 논문에서는 제안된 모델의 성능을 평가하기 위해

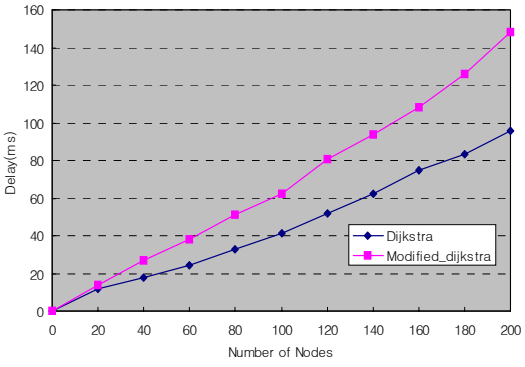
GT-ITM을 이용해서 Flat-random Graph 방식으로 토폴로지를 생성하였으며 총 노드의 수는 200개로 구성하였다. 각 노드들의 최대 대역폭은 약 100Mbps로 설정 하였으며, 노드들 간의 링크 확률은 10~60%로 변화시켜 보았다. 링크 대역폭은 10~20Mbps 로 주었으며, 이는 각 노드의 Out-degree에 할당된 대역폭의 합이 최대 대역폭인 100Mbps를 넘지 않도록 고려한 것이다.

제안한 모델에서 한 개의 오버레이 호스트가 가지는 최대 Out-degree 값은 100Mbps의 네트워크 환경에서 최소 25Mbps 정도의 대역폭이 필요한 HD급 영상을 서비스 한다고 가정하였고, 이를 모델링하기 위해서 Network Similator2를 이용하였다.

구성된 네트워크 토폴로지를 기초로 하여 노드의 수를 증가시켜 가면서 전송지연과 패킷 손실률, 링크혼잡도 등을 측정하였다. 또한, 제안한 알고리즘과의 성능비교를 위해 관련 연구에서 살펴보았던 Dijkstra, SWP, WSP, SDP 알고리즘을 동일 환경 하에서 테스트 하였다

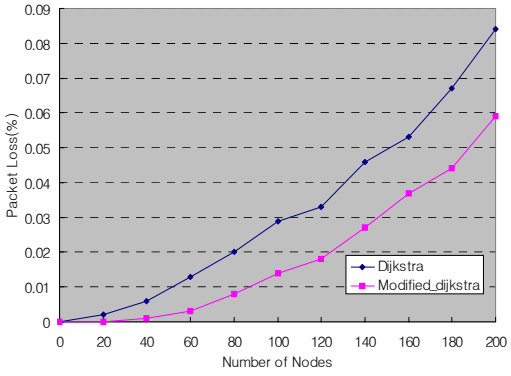
2.4.2 시뮬레이션 결과분석

첫 번째 시뮬레이션에서는 트리를 구성하는 노드의 수를 20개부터 200개까지 증가시켜가며 Dijkstra 알고리즘과 수정된 Dijkstra 알고리즘으로 구성된 트리의 평균 Delay값을 측정하였다.



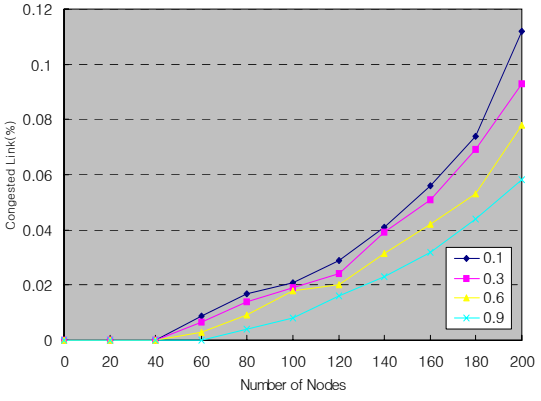
[Fig. 1] Average Delay vs. Number of Host

Fig. 1의 시뮬레이션 결과를 보면 총 딜레이의 합은 Dijkstra가 변형된 Dijkstra보다 작다는 것을 알 수 있다. Delay 뿐만 아니라 대역폭까지 고려한 제안 알고리즘의 성능이 낮게 나오는 것은 당연하다. 그러나 이 시뮬레이션에는 Dijkstra 알고리즘이 월등이 높을 것을 생각되는 Queuing Delay가 포함되어 있지 않기 때문에 정확한 결과라고 보기에는 힘들다.



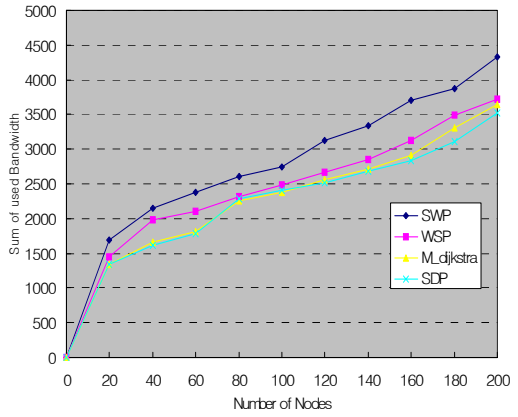
[Fig. 2] Packet Loss vs. Number of Host

Fig. 2의 시뮬레이션 결과를 보면 제안하는 알고리즘의 패킷 손실율이 Dijkstra 알고리즘의 손실율 보다 낮게 나타나는 것을 볼 수 있다. 이는 Score function에서 대역폭을 고려했기 때문에 패킷의 손실이 줄어든 것이다.



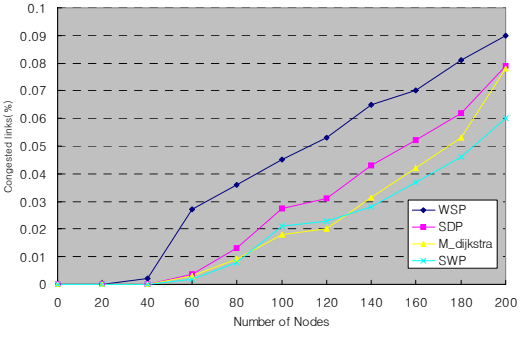
[Fig. 3] Congested Link vs. Value of α

Fig. 3은 Score function의 α값의 변화에 따른 혼잡 링크율을 볼 수 있다. α값이 작을수록 대역폭보다 Delay를 더 고려한 것이기 때문에 혼잡링크가 더 많은 것을 볼 수 있다. 또한 노드의 수가 40개 이하일 때는 혼잡링크가 거의 없는 것도 알 수 있다.



[Fig. 4] Total used Bandwidth vs. Number of Host

Fig. 4는 SWP, SDP, WSP 그리고 제안된 수정된 Dijkstra알고리즘이 호스트 수의 증가에 따라 사용하는 대역폭의 합을 보여준다. 최단 경로로써 대역폭이 가장 큰 경로를 선택하는 SWP 알고리즘이 가장 많은 대역폭을 사용함으로써 효율성이 떨어지는 것을 알 수 있다. 제안한 알고리즘은 대역폭의 역수를 가중치로 사용하는 SDP 알고리즘과 대역폭의 사용이 비슷한 것을 볼 수 있다.



[Fig. 5] Congested Link vs. Number of Host

마지막으로 Fig. 5는 호스트 수가 증가함에 따라 혼잡 링크율의 변화를 보여준다. 시뮬레이션 결과를 통해 최단 경로 선택에 있어서 최소 hop을 갖는 경로를 선택하는 WSP가 혼잡 링크가 가장 많은 것을 알 수 있다. 또한, 대역폭을 경로 선택에 있어서 최우선으로 두는 SWP가 혼잡 링크율이 가장 낮은 것을 볼 수 있다. 그리고 수정된 Dijkstra 알고리즘은 SWP와 비슷한 혼잡 링크율을 갖는 것을 알 수 있다.

3. 결론

본 논문에서는 각 호스트들의 제한된 자원과 네트워크 환경을 고려하여 오버레이 멀티캐스트 트리를 구성하는 알고리즘을 제안하였다. 이 알고리즘은 각 노드들의 가용 대역폭과 Delay를 Score-function에 적용하여 Cost값을 구한 후, 변형된 Dijkstra 알고리즘을 사용하여 최적의 트리를 구성하게 된다. 이로 인해 실시간 방송 서비스에서 극복해야 할 가장 큰 문제점인 QoS(Quality of Service)를 높일 수 있었고, 오버레이 멀티캐스트가 가지는 단점인 지연시간과 대역폭 사용의 측면에 있어서의 비효율성을 줄일 수 있었다[9].

모의실험을 통해서 제안 알고리즘이 트래픽 분배에서 Dijkstra 알고리즘보다 좋은 성능을 가진다는 것을 확인할 수 있었다. 그리고 다중 제약을 가지는 기존의 트리구성 알고리즘들과 비교 했을 때 대역폭의 효율성과 혼잡 링크 제어에 좋은 성능을 보이고 있음을 알 수 있다.

향후연구로는 제안 알고리즘은 고정된 토폴로지에만 적용이 용이하기 때문에, 트리의 확장성을 위해서 노드의 Join과 Leave정책을 수립하는 것을 고려할 수 있다.

References

[1] Y.Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling Conferencing Applications on the internet using on Overlay Multicast Architecture", ACM SIGCOMM'01, Aug. 2001.
DOI: http://dx.doi.org/10.1145/964723.383064

[2] L. Mathy, R. Canonico, and D. Hutchison, "An Overlay Tree Building Control Protocol" NGC 2001, November. 2001.
DOI: http://dx.doi.org/10.1007/3-540-45546-9_6

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast" ACM SIGCOMM '02, Aug. 2002.
DOI: http://dx.doi.org/10.1145/633025.633045

[4] V. Roca and A. El-sayed, "A host-based Multicast(hbm) Solution for Group Communications", 1st IEEE International Conference on Networking, Colmar, France, July. 2001.
DOI: http://dx.doi.org/10.1007/3-540-47728-4_60

[5] R. Guerin, Ariel Orda, and D. Williams, "QoS Routing Mechanism and OSPF Extension", Proc, of 2nd Global Internet Miniconference (Joint with Globecom'97), Nov. 1997.

DOI: http://dx.doi.org/10.1109/GLOCOM.1997.644603
[6] Q. Ma and P. Steenkiste, "On Path Selection for Traffic with Bandwidth Guarantees", Proc. of IEEE Int'l Conf. on Network Protocol, Oct. 1997.
DOI: http://dx.doi.org/10.1109/ICNP.1997.643714
[7] B. Davie and Y. Rekhter, "MPLS Technology and Applications", Morgan Kaufmann Publishers, 2000.
[8] Parra, O.J.S. and Rubio, G.L., "Dijkstra Algorithm Based Reliability Model", 2012 IEEE/ACM 16th International Symposium on, Nov, 2012.
DOI: http://dx.doi.org/10.1109/DS-RT.2012.31
[9] Montessoro, P.L and Wieser, S., "An efficient and scalable engine for large scale multimedia overlay networks", 2012 IEEE International workshop Technical Committee, June, 2012.
DOI: http://dx.doi.org/10.1109/CQR.2012.6267109

이 형 옥(Hyung-Ok Lee) **[정회원]**

- 2008년 2월 : 전남대학교 컴퓨텅정보통신공학과 (공학석사)
- 2008년 3월 ~ 현재 : 전남대학교 전자컴퓨터공학부 박사과정

<관심분야>
컴퓨터 네트워크, 통신 프로토콜

남 지 승(Ji-Seung Nam) **[정회원]**

- 1992년 2월 : Univ. of Arizona, 전자공학과 (공학박사)
- 1992년 1월 ~ 1995년 12월 : 한국전자통신연구원 선임연구원
- 2001년 2월 ~ 현재 : 전남대학교 인터넷창업보육 센터장
- 1995년 2월 ~ 현재 : 전남대학교 컴퓨터공학과 교수

<관심분야>
통신 프로토콜, 인터넷 실시간 서비스, 라우팅

박 준 석(Jun-Seok Park) **[정회원]**

- 1999년 2월 : KAIST 전산학과 (공학석사)
- 2006년 2월 : 인하대학교 컴퓨터정보공학 (공학박사)
- 1987년 1월 ~ 현재 : 한국전자통신연구원 스마트인터페이스 연구팀 팀장

<관심분야>
햅틱 인터페이스, 3D 제스처 인식, 휴먼-컴퓨터상호작용