무선 메쉬 네트워크에서의 인터도메인 부하분산 기법

최형규° 한승재
연세대학교 컴퓨터과학과
{hgchoi°, sjhan}@cs.yonsei.ac.kr

Inter-Domain Load Balancing Algorithm for Wireless Mesh Networks

Hyoung-Gyu Choi° Seung-Jae Han
Department of Computer Science, Yonsei University

요 약

본 연구는 무선 메쉬 네트워크에서 균형적 네트워크 부하분산 휴리스틱을 제안한다. 다수의 무선 라우터가 다중홉으로 구성되어있는 무선 메쉬 네트워크는 유선망과 연결되어 있는 gateway 역할을 하는 무선 라우터 쪽으로 과부하가 걸리기 쉽다. 이 같은 상황은 추가적으로 발생하는 서비스 요구사항을 충족시켜줄 수 없을 확률이 높아 전체 네트워크 성능저하를 야기시킨다. 본 연구는 다수의 도메인으로 구성된 무선 메쉬 네트워크에서 gateway 무선 라우터로 몰리는 네트워크 부하의 균형을 달성하는 부하분산 (Load Balancing) 기법을 소개하고자 한다.

1. 서 론

무선 메쉬 네트워크(Wireless Mesh Networks)는 다수의 무선 라우터(Wireless Router)를 그물망형태로 연결시켜 기존의 유선 인프라를 기반으로 광대역 무선 통신을 가능케 하는 최근 주목 받고 있는 무선통신 기술이다. 무선 메쉬 네트워크는 데이터 트래픽의 발생 주체인 mobile client와 트래픽을 목적지로 relay해주는 mesh 라우터, 마지막으로 유선망과 연결되어 망 내부의 gateway 역할을 수행하는 sink 라우터로 구성된다. 무선 메쉬 네트워크는 망 내부의 커뮤니티 통신 보다는 인터넷 엑세스 통신이 대다수를 이루기 때문에 다량의 트래픽이 소수의 sink 라우터쪽으로 몰리기 쉽다. 이 같은 상황은 sink 라우터마다 걸리는 네트워크 부하간의 불균형을 초래하며 과부하가 걸린 sink 라우터의 서브넷은 더 이상의 통신 서비스 요청을 처리하지 못해 이를 drop한다. 이 결과 sink 라우터간의 부하 불균형 현상은 전체 네트워크의 성능저하를 유발시킨다. 따라서 무선 메쉬 네트워크에서의 sink 라우터에 몰리는 트래픽의 부하분산 기법은 네트워크 성능과 형평성 문제에 있어서 중요한 문제해결 방법이다.

최근 무선 메쉬 네트워크에서의 부하분산 기법이 다양하게 연구 되고 있다. 먼저 연구 [1]는 multiple sink 라우터 환경에서의 부하분산을 달성하는 clustering 기법을 제안하였다. 이 연구는 각 sink 라우터를 중심으로 형성되는 cluster간의 부하균형을 맞추어 cluster간의 부하분산을 위해 이루어지는 handover overhead를 최소화 하는 것에 초점을 맞추었다. 하지만 [1]는 clustering에 있어서의 metric으로 sink 라우터와의 hop수만을 감안하여 load에 민감한 clustering을 수행하지 못하였다. 이 외에 많은 연구들([2], [3], [4])이 무선 메쉬 네트워크에서의 최적화 부하분산 기법을 제안하였지만 이들은 복잡한 multi-path routing을 사용하였다. 연구 [5]에서 멀티 홉 네트워크 부하분산에서 single path 라우팅이 multi-path 라우팅과 비교했을 때 성능이 떨어지지 않는다는 검증 결과가 나왔기 때문에 본 연구는 다중 sink 라우터 환경에서의 single path 라우팅을 이용한 무선 메쉬 네트워크의 부하분산 기법을 소개한다. 제안 알고리즘은 네트워크를 sink 라우터를 중심으로 네트워크 load와 홉 수를 감안한 clustering을 통하여 다수의 도메인으로 나누고 이들간의 적절한 부하분산을 수행한다.

본 논문의 구성은 다음과 같다. 2절에서는 알고리즘 설명을 위한 가정 및 네트워크 모델을 소개하고 3절은 부하분산 알고리즘을 설명한다. 4절에서는 모의 실험을 통해 기존 알고리즘과의 성능비교 후 5절에서 결론을 맺는다.

2. 가정 및 네트워크 모델 설명

본 연구에서 가정하고 있는 기본적인 네트워크 topology는 grid 스타일의 다중 sink 라우터 환경이다. 우리는 알고리즘의 표현의 용이성과 이해를 돕고자 grid topology를 사용하였다. 물론 임의로 라우터를 배치한 네트워크로의 확장도 용이하다. 우리는 라우팅 문제만을 다루고 있기 때문에 본 연구에서는 잘 정의 된 multi-channel multi-radio interface를 지원하는 MAC 프로토콜이 있다고 가정한다. 따라서 라우터 간의 전파간섭(interference)이나 매질경쟁(medium contention)으로 인한 손실은 고려치 않는다. 우리는 single path 라우팅을 다루고 있으므로 각 라우터는 오직 한 가지의 path로 sink 라우터와 연결될 수 있다. 전체 네트워크는

n개의 sink 라우터를 기점으로 형성되어있는 n개의 네트워크 도메인으로 구성된다. 각각의 도메인은 sink 라우터를 root로 하는 tree를 형성한다. 이러한 도메인을 형성하는 과정을 clustering이라고 할 것이다. 이 clustering을 위해서 네트워크에 동기화 된 타이머가 있다고 가정한다. 그리고 특정 라우터가 부하분산을 위해 해당 네트워크 도메인을 바꾸는 동작을 defection이라 할 것이다.

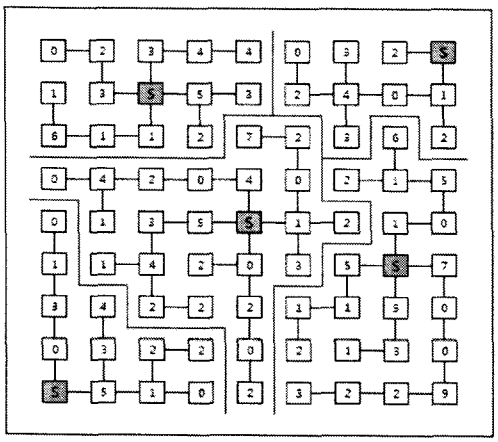

그림 1. Network Topology의 예

그림 1은 본 연구에서 가정하고 있는 전형적인 topology의 예이다. (n=5) 사각형은 mesh 라우터(이하 라우터)를 의미하고 S는 sink 라우터를 의미한다. 사각형 안의 숫자는 라우터의 weight를 의미하고 이는 라우터에 요청하는 mobile client의 request의 수를 의미한다(한 mobile client가 weight 1의 request를 발행한다고 가정). mobile client는 시간에 따라 움직일 수 있으므로 각 라우터의 weight도 서비스를 제공하는 mobile client의 개수에 따라 변할 수 있다. 우리는 각각의 라우터에서 해당 라우터를 포함하는 sub-tree의 모든 라우터 weight의 합을 cumulative load라고 할 것이다. 또한 우리는 각 sink 라우터의 인근 라우터를 top sub node라고 하고 이들간의 link를 top sub link라고 할 것이다. 또한 top sub node의 cumulative load를 top sub load라고 할 것이다. 우리는 각 link가 수용할 수 있는 최대의 대역폭을 link capacity라 할 것이고 이 top sub link capacity 중에 top sub load가 차지하는 비율을 네트워크 이용률(network utilization)이라고 할 것이다. 마지막으로 우리는 수식의 표기를 위해 i 번째 도메인에 있는 sink 라우터와 k 홉 떨어져 있는 라우터 중에 왼쪽으로부터 시계 반대방향으로 n 번째 떨어져 있는 라우터를 $R_i^{(k,n)}$ 으로 표기할 것이고 R은 표현에 따라 여러 문자로 대치될 수 있다.

3. Inter-Domain Load Balancing Algorithm

제안한 알고리즘(이하 IDLB)은 크게 2개의 부분으로 구성된다. 첫째는 각 sink 라우터를 중심으로 clustering을 통하여 sink 라우터 개수와 일치하는 네트워크 도메인을 형성하는 단계(Load Adaptive Clustering, 이하 LAC)이고 둘째는 이로 인해 형성된 cluster 간의 부하를 분산하는 단계(Outer Domain Load Balancing, 이하 ODLB)이다.

3.1 Load Adaptive Clustering

LAC는 기존의 홉 수만을 고려하던 clustering과는 다른 sink 라우터까지의 cumulative load와 홉 수를 고려하기 때문에 트래픽이 많은 지역에서는 느린 clustering을 적은 지역에서는 빠른 clustering을 수행하여 다수의 sink 라우터로 몰리는 트래픽을 균형적으로 분산시킬 수 있는 알고리즘이다. 한 번 clustering 완료 후에 추가적 네트워크 도메인 간의 부하분산은 handover를 일으키게 되고 이로 발생하는 오버헤드는 네트워크 성능을 저하시킬 수 있으므로 LAC를 통하여 각 네트워크 도메인 사이의 트래픽을 효율적으로 분산시켜 이 handover를 최소화 하는 것은 매우 중요하다. 이를 위해 LAC은 각 네트워크 도메인의 load에 따라 clustering 타이밍을 조절하며 clustering을 수행한다. 따라서 우리는 각 라우터에 동기화 된 타이머가 있다고 가정한다. 각 라우터는 알고리즘에 의하여 스케쥴 된 자신의 clustering time에 주변 cluster와 연결되지 않은 라우터들과 연결을 맺는다. 우리는 이를 위해 각 라우터마다 두 가지의 타이머를 운영할 것이며 clustering time 스케쥴링에 필요한 공식은 다음과 같다.

$$T_i^{(k,n)} = t + m \cdot k \cdot U_i^{(k,n)} \qquad (1)$$

$$B_i^{(k,n)} = t + m \cdot k \cdot U_i^{(k,n)} \cdot p \qquad (2)$$

(1)은 clustering time을 계산하기 위한 공식이고 (2)은 back-off time을 위한 공식이다. clustering time은 라우터가 clustering을 수행하기 위해 필요한 시간이고 back-off time은 clustering을 수행하던 라우터에서 overflow가 발생하여 해당 clustering을 지연시키기 위해 필요한 시간이다. t는 현재시각이고 m은 해당 라우터가 속해있는 path의 top sub link capacity이다. k는 해당 라우터와 라우터가 속해있는 도메인의 sink 라우터 사이의 홉 수를 나타내고 U는 라우터가 속해있는 path의 네트워크 이용률을 의미한다. 마지막으로 공식 (2)에서 p는 overflow에 대한 penalty 값으로 본 연구에서는 2를 사용한다. (1)에서 확인할 수 있듯이 라우터마다 clustering을 수행하기 위해 sink 라우터까지의 홉 수와 네트워크 이용률을 통한 load의 곱을 이용하여 다음 clustering time을 계산하므로 LAC는 네트워크 load에 효율적으로 적응하며 clustering을 수행할 수 있다. 이해를 돕기 위해 그림

2을 예로 들어 LAC의 전체 수행과정을 설명한다. 설명에 앞서 clustering을 위해 각 라우터는 다음과 같은 네 가지 상태(pendency, expansion, overflow, settlement) 중 반드시 한 가지 상태를 갖는다. pendency는 아직

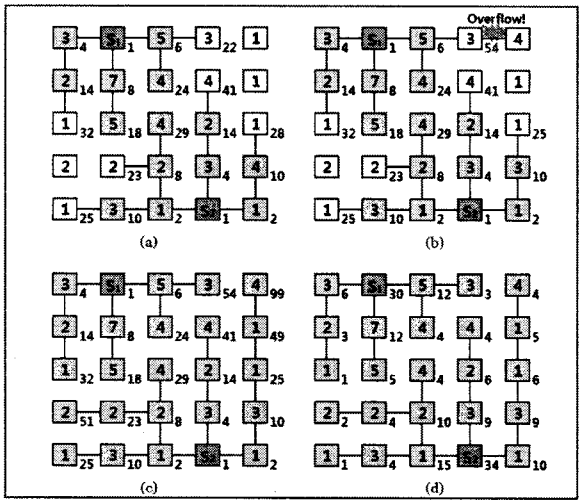

그림 2. LAC 예

라우터가 어떤 네트워크 도메인에도 속하지 않았다는 것을 의미한다. expansion은 clustering을 위해 clustering time을 기다리고 있음을 의미한다. overflow는 clustering을 위해 새로이 추가된 라우터의 weight이 해당 path의top sub link capacity를 넘어 overflow가 발생하여 clustering을 중단하고 back-off time을 할당받고 다음 clustering을 대기하고 있음을 의미한다. settlement는 더 이상 clustering을 수행할 수 없음을 의미한다. 그림 2의 각 라우터를 쉽게 표현하기 위해 우리는 좌측 상단을 원점으로 하는 r(x,y) 표현법을 사용할 것이다. 그림에서 사각형 안쪽의 숫자는 각 라우터의 weight를 나타내며 S는 sink 라우터를 의미한다. (a), (b), (c)에서 사각형 우측하단의 숫자는 각 라우터의 clustering time을 의미하고 (d)에서는 cumulative load를 의미한다. 그림에서 각 link 의 capacity는 15이다. 최초에 모든 라우터는 pendency 상태를 갖고 각자의 타이머를 0으로 설정한다. 알고리즘이 시작하면 sink 라우터는 각자의 clustering time을 1로 설정하고 상태를 expansion으로 변경한다. 타이머는 값을 1씩 증가하며 상태가 expansion인 라우터가 자신의 clustering time과 타이머의 시각이 일치할 경우 해당 라우터는 clustering을 시작한다. clustering을 수행하는 라우터는 주변 라우터들의 weight와 자신의 top sub load의 합이 top sub link capacity를 넘지 않는다면 해당 라우터와 연결을 맺는다. 이때 주변 라우터들은 자신의 상태를 expansion으로 바꾸고 clustering time을 설정한다. 더 이상 주변에 clustering을 수행할 라우터가 없는 라우터는 자신의 상태를 settlement으로 바꾸고 clustering을 종료한다. 그림 2-(a)은 타이머가 14일 때를 나타낸다. 그림에서 음영으로 되어있는 사각형은 상태가 settlement을 의미한다. 그림 2-(b)는 22초에 r(3,0) 라우터가 clustering을 수행하려다 overflow 상황을 맞이하여 자신의 상태를 overflow으로 바꾸고 back-off time을 54로 설정한 상태를 나타낸다. r(3,0) 라우터의 현재 top sub load가 12인 상황에서 r(4,0)의 weight이 4이므로 top sub link capacity 15를 초과하여 overflow가 발생하였다. 그림 2-(c)에서 확인할 수 있듯이 back-off time으로 인해 r(3,0)은 clustering이 지연되었고 clustering time이 49인 r(4,1)에 의해 r(4,0)는 S_2 네트워크 도메인에 속하게 된다. 그림 2-(d)는 모든 clustering이 완료된 모습을 보여주고 있으며 S_1과 S_2를 중심으로 한 네트워크 도메인의 load가 각각 30, 34가 됨을 알 수 있다.

3.2 Outer Domain Load Balancing

ODLB는 LAC로 달성하지 못한 네트워크 도메인간의 부하분산을 수행하는 알고리즘이다. ODLB는 LAC가 종료된 후에 특정 top sub link가 link threshold를 (본 논문에서는 link capacity의 90%로 설정하였다) 초과한 경우 동작한다. ODLB 역시 이해의 편의를 위해 그림 3와 함께 설명한다. 그림 3는 그림 2의 네트워크 설정을 따른다.

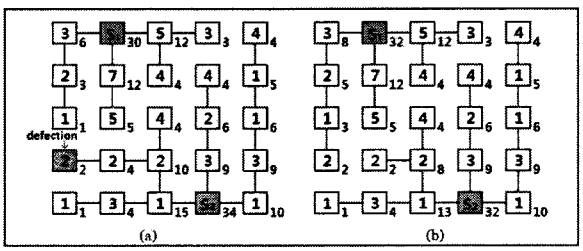

그림 3. ODLB 예

그림 3-(a)는 그림 2-(d)의 모습을 나타낸다. LAC는 완료되었지만 S_2의 r(2,4)의 link가 link threshold 13.5를 넘어섰으므로 ODLB가 동작한다. ODLB는 먼저 overloaded link의sub-tree에서 인근 네트워크 도메인이 맞닿아 있는 라우터들을 선별해서 cumulative load가 가장 적은 순서로 정렬을 한다. 그리고 난 후 정렬된 라우터를 하나씩 overloaded link가 link threshold 이하의 값이 될 때까지 defection을 수행한다. 이 때 defection이 되는 상대 도메인의 top sub load 역시 link threshold를 넘는 경우에는 defection이 일어나지 않는다. 그림 3-(a)에서는 r(0,3), r(1,3), r(2,2)가 r(1,4)의 sub-tree에 속한 S_1과 맞닿아 있는 라우터들이고 각각의 cumulative load 가 적은 순서로 defection을 수행하면 r(0,3)이 S_1으로 defection되며 r(2,4)의 top sub load가

13으로 떨어져 ODLB의 수행이 종료된다. ODLB의 수행종료 후 S_1과 S_2의 load는 각각 30, 34에서 32, 32로 변경되어 두 도메인이 더욱 향상된 트래픽 균형에 이름을 확인할 수 있다.

4. 모의실험

본 실험에서 우리는 연구 [1] (Inter-Domain Partitioning Algorithm, 이후 IDP)와 성능비교평가를 통해 알고리즘의 우수성을 검증한다. IDP는 멀티 sink 라우터 환경에서 홉 수를 metric으로 하여 clustering을 수행하고 우리 연구의 ODLB와 흡사한 inter domain 부하분산을 수행하기에 비교대상으로 선정하였다. 성능평가를 위해 metric은 drop count, defection count, average path length가 사용되었다. drop count는 추가적으로 발생하는 사용자 request를 네트워크가 처리하지 못하고 drop 시키는 횟수를 의미하고 defection count는 도메인간에 부하분산을 위해 행하여진 defection의 횟수이고 average path length는 네트워크의 모든 path의 길이에 평균을 낸 값이다. 실험은 두 가지 형태로 구분하였다.

4.1 Static case

먼저 우리는 모든 라우터의 weight이 고정되어 있는 static 환경에서 실험하였다. 네트워크 topology는 그림 1과 같다.(9x9 grid topology에sink 라우터 5개) 모든 link capacity는 20으로 하였고 link threshold는 18로 설정하였다. 각 라우터의 weight는 정해진 최대값 이내에서 무작위로 설정하였다. 그림 4은 static case 실험의 결과를 나타낸다. .가로축은 라우터가 가질 수 있는 최대 weight을 나타내고 세로축은 drop count를 나타낸다. 두 알고리즘 모두 최대 weight 5까지는 request drop이 일어나지 않았으며 6부터 drop이 발생하였으나 IDLB가 IDP보다 9%로 적은 drop count를 보이는 것을 알 수 있다. 이는 IDLB가 더 효율적으로 네트워크 트래픽을 분산시켜 더 많은 user request를 수용할 수 있음을 보여준다.

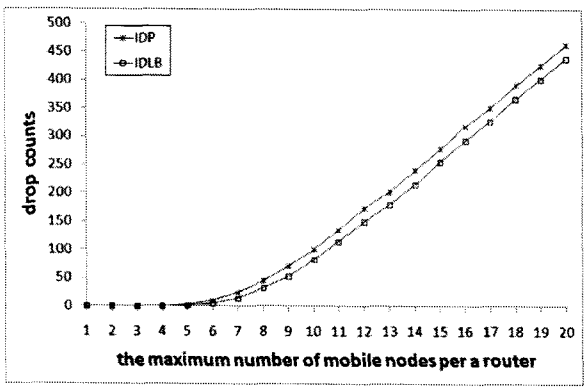

그림 4. Static case에서의 drop count 비교

우리는 두 알고리즘을 defection count와 average path length에서도 성능평가를 수행하였다. defection count는 handover를 일으키는 횟수이므로 이 값이 적어야 더 좋은 처리율을 가지는 네트워크라는 것을 검증할 수 있다. average path length 역시도 sink 라우터까지의 거리를 의미하므로 이 값이 적을수록 적은 delay time을 가질 수 있다는 것을 검증할 수 있다.

표 1. Static case에서의 Defection count와 average path length 비교

	Defection Count	Average Path Length
IDP	26.4	2.45
IDLB	20..8	2.33

표 1에서 우리는 IDLB가 IDP보다 적은 defection count와 average path length를 보이고 있음을 확인 할 수 있다. 따라서 IDLB는 좀 더 적은 handover와 delay-time을 보장할 수 있다.

4.2 Dynamic case

static case는 특정 상황에서의 성능을 평가할 수 있지만 이로는 mobile client들이 움직일 때 각 라우터의 weight 변화에 따른 네트워크 성능평가는 확인 할 수 없는 한계점이 있다. 따라서 우리는 dynamic case의 실험을 수행하였다. 네트워크 안의 mobile client에 [6]의 mobility pattern을 주어 동적으로 라우터의 weight를 변경하여 그 성능을 비교하였다. 결과는 그림 5와 같다.

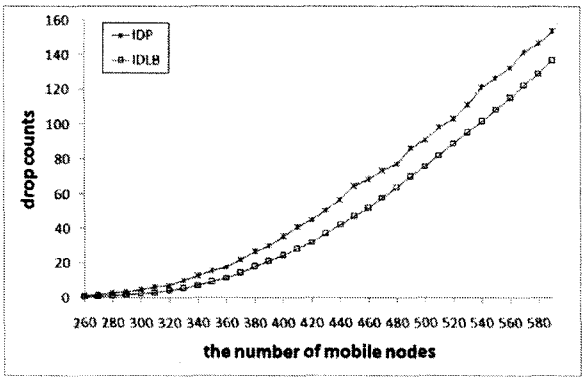

그림 5. Dynamic case에서의 drop count 비교

가로축은 네트워크에 있는 mobile client의 개수를 의미하고 세로축은 이 mobile node들이 100초 동안 움직였을 때의 발생한 drop count를 의미한다. 그래프에서 알 수 있듯이 우리는 IDLB가 IDP보다 19%의 우수성을 보여주고 있음을 확인할 수 있다. 이 결과는 오히려 IDLB가 라우터의 트래픽 load가 동적으로 변하는 환경에 더욱 효과적으로 대응하여 높은 트래픽 분산 처리율을 달성함을 보여준다.

5. 결론

본 연구는 다중 sink 라우터 환경에서의single path 라우팅 부하분산기법을 소개하였다. 제안한 알고리즘은 홉 수 뿐만 아니라 sink 라우터로 몰리는 load도 감안하면서 네트워크 도메인을 형성하였고 홉 수 만을 감안한 부하분산기법에 비해 동적환경에서 20%에 가까운 성능 향상을 달성하였다.

향후 연구에서는 네트워크 도메인간의 부하분산 뿐만 아니라 single 도메인 안에서의 부하분산 기법을 제안할 것이며 멀티 홉 네트워크에서의 interference model을 통한 부하분산 기법도 제안 할 것이다.

참 고 문 헌

[1] B. Xie, Y. Yu, A. Kumar, and D. P. Agrawal, *Load Balancing and Inter-Domain Mobility for Wireless Mesh Networks,* in Proc. IEEE GlobeCom, 2006.

[2] V. Mohit, D. Qi, U. Shambhu, and A. Vishal, *A New Paradigm for Load Balancing in Wireless Mesh Networks,* CSE Dept. TR 2006-11, State University of New York at Buffalo, 2006.

[3] M. Alicherry, R. Bhatia, and E. Li, *Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Network,* in Proc. ACM Mobicom, 2005.

[4] A. Raniwala and T. Chiueh, *Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network,* in Proc. IEEE INFOCOM, 2005.

[5] Y. Ganjali and A. Keshavarzian, *Load Balancing in Ad Hoc Networks: Single-Path Routing vs. Multi-Path Routing,* in Proc. IEEE INFOCOM, 2004.

[6] C. Bettstetter, *Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks,* in Proc. ACM MSWiM, 2001.