On-Demand Routing의 성능향상을 위한 쿼리 수신 노드 선택

손동현° 김대희
{dhson, ninanoo}@dsys.korea.ac.kr

Selective Query Node for On-Demand Routing

Donghyoun Son, Daehee Kim
Department of Electronics Engineering, Korea University

Abstract

On-Demand routing protocol들은 Route를 셋업하기 위해 network-wide flooding scheme을 사용한다. 이 경우 많은 수의 overhead가 쿼리 패킷들이 생산되어 네트웍 퍼포먼스가 저하되는 결과를 초래한다. 우리는 이것을 일컬어 " Broadcast storm problem" 이라고 한다. 여기에 우리는 Selective Query Node라는 scheme을 제안하여 flooding시에 발생하는 broadcast storm 문제를 완화하려고 한다. SQN은 이웃노드의 topology를 고려하여 proactive한 라우팅 정보를 제공하며 세가지 제한요소를 두어 쿼리를 받은 이웃노드가 받은 쿼리를 중지할 수 있도록 하여 퍼포먼스 향상을 기할 수 있도록 한다.

1. Introduction

On-demand routing protocol들은 적은라우팅 오버헤드로 인하여 상당히 많이 사용되고 있다[1,2]. 그러나 온디멘드 라우팅 프로토콜은 Network-Wide Flooding을 사용하므로 많은 수의 쿼리패킷을 생성하여 전체적인 네트웍 퍼포먼스를 저하시키는 결과 를 가져온다. 범위가 넓은 네트웍의 경우는, 목적지를 찾기 위한 시간적인 소요가 무시할 수 없을 정도로 크며 또한 컨트롤 트래픽의 양도 상당하게 된다. 그러므로 과도한 컨트롤 트래픽을 제어하는 것이 꼭 필요하다.

이 논문은 Selective Query Node(이하 SQN)를 제안하는데 주된 사항은 쿼리를 받을 노드를 제한함과 동시에 선택하도록 하는 것을 포커스로 하고 있다. 이 논문에서 제안하는 프로시져는 많은 수의 쿼리 패킷을 줄이며 결과적으로 네트웍 퍼포먼스를 증가시키는 역할을 한다. 세가지 요소가 고려되어 동작하는데 첫 번째 이웃 노드의 개수, 두 번째 노드의 이동속도 그리고 마지막으로 노드 간의 거리가 그 세가지 요소이다. 또한 SQN 정보를 유지하기 위한 하나의 테이블도 요구되며 이 테이블은 주기적으로 업데이트 되도록 구성되어져 있다.

2. Selective Query Node

2.1 Neighboring Table

SQN은 AODV[1,3]에 그 바탕을 두고있으나 Flooding scheme에 있어서는 AODV와 다른 방식을 가진다.SQN은 과도한 쿼리패킷을 줄일수 있도록 디자인되었으며 이러한 목적을 달성하기 위해서 쿼리패킷을 받을 이웃노드들에 대한 정보가 필요하고 이것을 통해서 SQN은 쿼리패킷을 포워딩할지 폐기할지를 결정한다.

네이버링테이블은 Hello메시지에 의해 주기적으로 업데이트 되며 이 테이블을 사용하여 경로를 설정하고 링크복구를 수행하며 또한 링크가 끊어짐을 추적한다. 네이버링테이블은 다섯가지의 이웃노드들에 대한 정보를 가지는데 Ticks, Distance, Speed, Link delay, receiving Time이 그것이다.

2.2 Selective Query Node Mechanism

애드혹네트웍에서는 모든노드가 랜덤하게 분포되어있으며 그 각각의 노드들은 각자의 Mobility가 존재한다.

만약 이웃한 노드들이 쿼리송신 노드(송신노드)로부터 가까이 있다면 어느 가까이 있는 한 노드가 쿼리패킷을 받고나서 그 쿼리패킷을 또 인접한 송신노드로부터 가까이 있는 다른 노드로 재전송할 것이다. 물론 이 다른 노드도 소스노드로부터 가까이 있는 노드들 중에 하나이다. 이 경우 송신노드로부터 가까이 있는 노드들은 자신들이 받은 쿼리패킷을 한번이상 재전송하여 국지적인 Flooding이 발행하여 트래픽을 가중시킨다. 또다른 경우로는 이웃노드가 송신노드로부터 멀리 떨어져 무선전송범위의 임계지역에 있는 경우 이 이노드는 송신노드로부터 이미 피킷을 받았음에도 불구하고 이 노드보다 송신노드에 가까이 있는 노드로부터 이미 수신한 쿼리패킷을 또 받을 것이다. 이경우 패킷을 드랍하는 것은 가능하지만 재전송노드가 재전송하는 시점에 이미 트래픽은 발생하게되어 소모적인 트래픽이 존재하게 되는것이다. 물론 이 경우 네트웍 퍼포먼스에도 영향을 미치게 된다.

SQN은 Forwarding Threshold Zone(이하 FTZ)을 설정하기 위해 송신노드와 수신노드들간의 거리의 평균을 고려한다. FTZ을 구한 다음엔 송신노드는 FTZ 내에있는 노드들의 갯수와 FTZ 밖에 있는 노드들의 개수를 파악한다. 만약 FTZ 밖에 있는 노드의 숫자가 많다면 송신노드는 쿼리패킷을 재전송하고 그 반대의 경우 송신노드는 재전송할 쿼리패킷을 폐기한다.

Mobility도 또한 현재의 topology와 이웃노드들의 정보를 변화시킬수 있는요소이므로 고려대상이다. 그러므로 노드들의 이동속도가 빠를수록 더 빈번하게 이웃노드들의 정보를 교환해야할 필요성이 있다. SQN은 노드의 Mobility와 Hello 메시지 주기를 고려하여 FTZ 을 설정한다.

소스노드는 단순Flooding을 사용하여 이웃노드들에게 쿼리패킷을 전송한다. 소스노드가 단순 Flooding을 사용하는 이유는 방향성을 없애기위해서이다. 소스노드로 부타 쿼리패킷을 받은 이웃노드는 자기노드의 FTZ를 계산한다. 그 다음 단계로 송

신 노드로서 자신이 계산한 FTZ내에 있는 자신의 이웃노드들의 개수와 FTZ밖에 있는 노드들의 개수를 파악하여 FTZ안에 있는 노드의 개수가 많은 경우 코리패킷전송을 하지 않는다.

네이버링테이블을 통해서 송신노드는 자신의 이웃노드들과의 평균 거리(D_{ave})를 계산하여 가지고 있다. 그리고 이웃노드들의 평균 스피드(S_{ave})도 테이블내에 저장된다. 송신노드는 아래의 식에 의해 FTZ의 반경을 구할 수 있다.

$$R_{ftz} = D_{ave} + a(S_{ave} - S_{sn}) \qquad (1)$$

R_{ftz} : FTZ의 반경
D_{ave} : 송신노드와 이웃노드들간의 평균거리
a : Hello message 주기
S_{ave} : 이웃노드들의 평균 속도
S_{sn} : 송신노드의 속도

(1)식에 의해서 송신노드는 FTZ의 반경을 구하고 FTZ를 경계로 이웃노드들의 존재 개수를 파악한다.

$$E_{qd} = \sum_{k=0}^{n} \{(D_k - R_{ftz}) / | D_k - R_{ftz} |\} \qquad (2)$$

N : 전체 이웃노드의 개수
D_k : K번째 노드와 송신노드간의 거리
E_{qd} : 쿼리송신여부의 Estimatation

(2)식에 의해서 송신노드는 E_{qd} 를 구할 수 있으며 이값에 의해서 쿼리패킷을 제전송할지 폐기할지를 결정하는데 만약 이값이 0이거나 양수이면 FTZ밖에 있는 노드의 개수가 더 많다는 의미이기 때문에 재전송을 할것이고 그렇지 않은 경우 재전송을 포기한다.

만약 재전송이 이루어졌을 경우 송신노드는 쿼리패킷전송시에 R_{ftz} 를 포함하여 전송한다. 쿼리패킷을 받은 이웃노드들은 제일먼저 수신된 패킷네에서 R_{ftz} 확인하여 자신의 네이버링테이블내에 있는 송신노드로부터 자기와의 거리와 비교를 한다. 이때 R_{ftz} 보다 거리가 더 짧은 경우 수신노드의 위치가 FTZ내에 있던 것으로 생각하고 쿼리패킷을 수신하자마자 재전송을 포기하고 반대의 경우는 식 (1),(2)를 이용하여 재전송과정을 되풀이한다. 그림 2.1에 이에대한 순서도를 보여주고 있다.

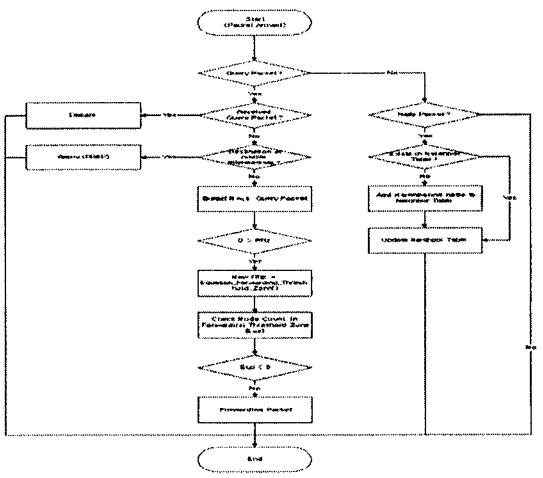


Fig 2.1 The flow of SQN scheme

그림 2.2에 SQN의 예를 들어 보았다. 송신노드 A는 자신의 FTZ를 구한 뒤 쿼리패킷을 전송한다. 이후 이웃노드들은 노드A와 자신과의 거리를 쿼리패킷내의 FTZ와 비교해서 작으면 수신한 쿼리 패킷을 폐기한다. 노드 B,C,D,E는 여기에 해당된다. 나머지 노드들은 같은 방식으로 쿼리패킷에 자신들의 FTZ를 넣어서 패킷을 재전송한다.

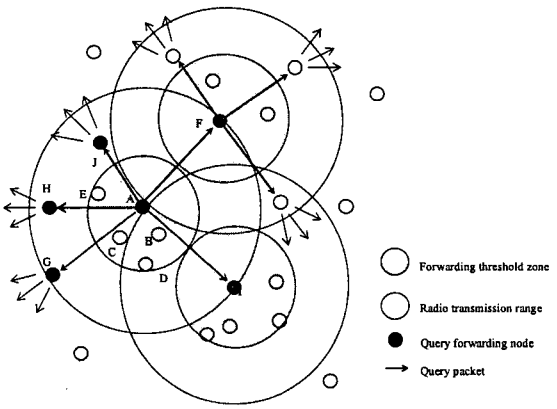
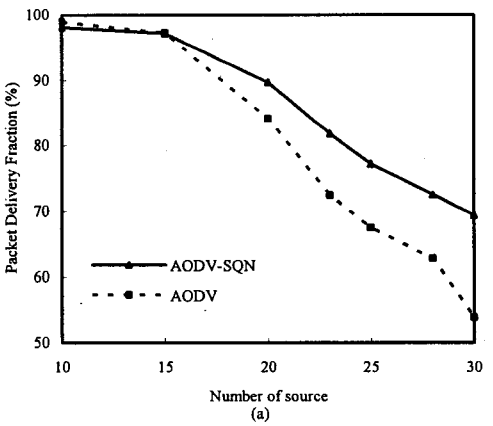


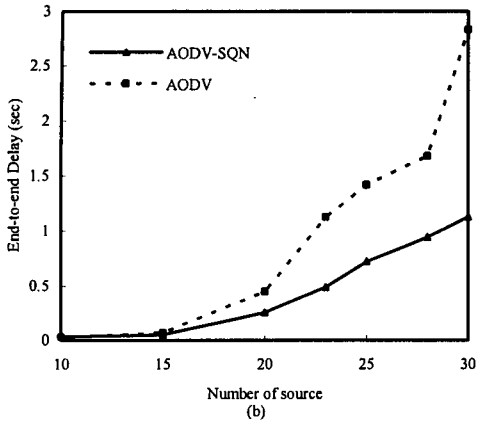
Fig 2.2 Example of Selective Query Node

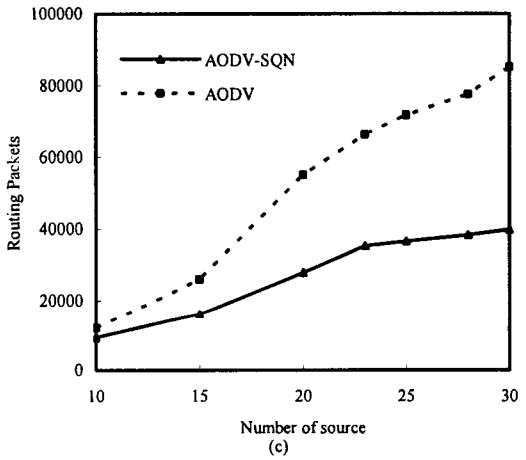
3. Performance Evaluation

아래 그림 3.1에 있는 그래프를 보면 SQN scheme은 패킷전송비율, 종간단 Delay, 라우팅패킷량 이라는 항목을 두고 볼 때 각각 현저한 성능향상을 가져왔다. 이러한 결과는 소스노드의 숫자가 많아짐에 따라서 더욱 현저하게 차이를 나타냈다.

그림 3.1에서 소스노드의 개수가 15개 이상되기전까지는 기존 AODV와 비교해서 큰 차이가 나지 않지만 15이상부터는 많은 차이를 나타낸다. 그림 3.1(C)에서보면 30소스인 경우 패킷전송 라우팅 패킷의 수가 54%정도 줄어듬을 보여준다. 줄어든 쿼리패킷을 수많큼 네트웍부하는 줄어들것으로 판단되며 여기서 알수 있는 것은 SQN은 네트웍 부하가 많아질수록 더 좋은 성능을 발휘함을 알수있다.







4. Conclusions

이 논문에서 우리는 Selective Query Node (SQN) scheme을 제시하였다. SQN은 broadcast storm problem을 해결하기 위해 생각되어졌으며 AODV를 기본으로 하여 Flooding시에 과도한 쿼리패킷을 억제하여 라우팅 오버헤드를 최소화하고 있다. 더나아가 SQN은 멀티플 액세스 간섭을 회피하여 라우팅설정시간을 단축시키고 네이버링테이블을 이용하여 로컬리커버리 시간을 단축시키는 효과를 가진다.

시뮬레이션 결과에서 보듯이 SQN은 과도한 쿼리패킷을 효과적으로 약 50% 감소시키고 이러한 감소율은 패킷전송비율을 10~18%향상시키는 결과를 가져왔다. 전체적으로 보면 SQN은 네트웍 퍼포먼스의 현저한 향상을 보여준다. 또한 고밀도의 네트웍에서 더 나은 성능 개선효과가 있다.

향후 계획은 더 다양한 네이버링테이블 내용을 가지고 좀더 복잡한 라우팅 scheme을 전개하려하며 각각의 다른 density의 네트웍에서의 성능개선효과를 검증하려한다.

5. References

[1] E.M. Royer and C.K. Toh, " A review of Current Routing Protocols for Ad-Hoc Wireless Network," *IEEE personal Communications*, vol.6 no.2, pp.46-55, April 1999 [2] Charles E. Perkins " Ad Hoc Networking," *Addison Wesley*, pp.173-219,2001.

[2] Charles E. Perkins " Ad Hoc Networking," *Addison Wesley*, pp.173-219,2001

[3] C.E. Perkins and E.M. Royer " Ad-Hoc On-demand Distance Vector Routing," *In Proceeding of the second IEEE workshop on Mobile Computer Systems and application*, pp90, Feb 1999.