모바일 애드 혹 네트워크에서의 TCP 최적화를 위한 수직, 수평 플로우 컨트롤

박영삼[0], 석용호, 최양희
서울대학교 멀티미디어 통신 연구실
{yspark[0], yhseok, yhchoi}@mmlab.snu.ac.kr

Vertical and Horizontal Flow Controls for TCP Optimization in the Mobile Ad Hoc Networks

Youngsam Park[0], Yongho Seok and Yanghee Choi

요 약

Mobile Ad Hoc Network(MANET)에서는, 각 모바일 노드들이 하나의 연결을 유지하는 단말 호스트로 동작함과 동시에 라우터로서도 동작한다. 따라서 유선 네트워크와는 달리 라우터 뿐만 아니라 단말 호스트에서의 네트워크 혼잡이 고려의 대상이 되어야 한다. 네트워크 계층 간에 동작하는 플로우 컨트롤 알고리즘을 사용한다면, 단말 호스트(플로우의 소스 혹은 목적 노드)에서의 패킷 드롭을 줄임으로써 TCP throughput을 증가시킬 수 있다. 일반적으로 MANET의 링크 레이어 프로토콜로는 IEEE 802.11 MAC 프로토콜이 많이 사용되지만, 이 프로토콜은 링크 레이어에서의 플로우 컨트롤을 제공하지는 않는다. 링크 레이어 플로우 컨트롤은 패킷이 전송된 직후에 버퍼 오버플로우로 인해 드롭되는 것을 방지할 수 있으며, 이는 결과적으로 대역폭의 낭비와 전력사용을 줄일 수 있다. 이 논문에서 2개의 플로우 컨트롤 알고리즘, 즉 수직 플로우 컨트롤과 수평 플로우 컨트롤을 제안한다.

1. 서론

MANET[1]은 모바일 인프라스트럭쳐를 통해서 네트워크를 형성하는 무선 모바일 노드들의 집합이다. 각 노드는 중앙의 AP(Access Point) 혹은 베이스 스테이션의 간섭 없이 자신의 통신 범위 안에 있는 다른 노드들과 직접 통신을 한다.

MANET의 또 다른 특징은 유선 네트워크 비해 상대적으로 낮은 throughput이다. MANET의 throughput을 높이기 위해서 멀티 홉 라우팅과 무선 채널 그리고 노드의 이동성 등을 고려한 트랜스포트 레이어와 링크 레이어에서의 성능 향상이 제안되어 왔다. 그리고 무선 채널의 대역폭이 증가함에 따라, 링크 레이어와 트랜스포트 레이어에서의 플로우 컨트롤의 중요성 또한 증가하게 되었다. 링크 레이어에서의 플로우 컨트롤을 통해서 MANET에서의 노드들은 네트워크 혼잡이 발생했을 때 생길 수 있는 패킷 드롭을 방지할 수 있다. 현재 가장 많이 사용되고 있는 링크 레이어 프로토콜인 IEEE 802.11에서는, RTS/CTS의 교환을 통해서 히든 터미널 문제와 신뢰성문제를 해결했다[2]. 그러나 802.3x에서와 같이 플로우 컨트롤 문제에 대해서는 고려된 바가 없다.

그 외에도, MANET은 단말 호스트가 패킷 포워딩에 참여하는 독특한 특성을 가진다. 특정 단말 호스트를 통해서 너무 많은 양의 트래픽이 포워딩 된다면, 그 단말 호스트 자신이 보내려고 하는 패킷의 일부가 버퍼 오버플로우로 인해서 드롭된다. 그러나 현재의 IEEE 802.11 MAC 프로토콜은 이러한 문제점 역시 고려하지 않고 있다.

이 논문은 다음과 같이 구성된다. 2절에서 MANET에서의 throughput을 증가시키기 위한 2가지 플로우 컨트롤 알고리즘을 제안한다. 3절에서는 NS-2를 이용한 시뮬레이션 결과를 통해서 제안된 플로우 컨트롤 알고리즘이 throughput을 증가시키고 파워 소모를 감소시킨다는 것을 보여준다.

2. TCP 최적화 알고리즘

2.1. 문제 정의

MANET에서는 각각의 단말 노드들이 2가지의 역할을 동시에 수행한다. 우선, 단말 노드들은 연결의 소스 혹은 목적 호스트에 해당하는 하나의 단말 호스트로서 작동하고, 동시에 다른 노드들간의 트래픽을 릴레이해주는 라우터로서의 역할도 수행한다. 이러한 특성으로 인해 MANET에서의 네트워크 혼잡은 유선 네트워크에서와는 다르게 다루어져야 한다. 네트워크 congestion은 연결 상의 중간 노드들의 버퍼를 포화시켜서 결국 패킷 드롭을 유발한다. 이 문제에 대처하기 위해서 TCP와 같은 혼잡 컨트롤 알고리즘이 사용된다. 그러나 무선 멀티 홉 네트워크에서는 각 단말 노드들이 단말 호스트로서의 역할과 라우터로서의 역할을 동시에 수행해야 하기 때문에, 유선 네트워크에서보다 더 자주 네트워크 혼잡이 발생하게 된다.

* 본 논문은 20003 년도 두뇌한국 21 과 국가지정연구실 프로젝트 지원을 받아 수행되었음

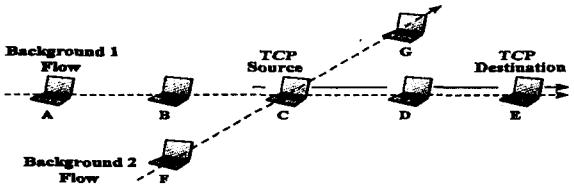

그림 1. 무선 멀티홉 네트워크 토폴로지

그림 1 은 MANET 의 간단한 토폴로지를 보여주고 있다. 노드 C 는 2 개의 크로스 트래픽을 릴레이 한다. 또한 이와 동시에 노드 C 는 노드 E 를 TCP 목적 노드로 가지는 TCP 소스 노드로서 동작한다. 이 토폴로지가 보여주듯이, 3 개의 플로우가 노드 C 에 집중되고 있다. 만약 상당한 수준의 크로스 트래픽이 생성되어 노드 C 의 인터페이스 큐(Interface Queue, IFQ)를 가득 차게 만든다면, 노드 C 에 의해서 생성된 TCP 패킷들은 IFQ 에서 패킷 드롭이 발생되게 된다. 이는 TCP 가 IFQ 의 상태를 확인하지 않고 단순히 congestion 윈도우의 크기에 따라 최대한 많은 패킷을 전송하려고 하기 때문이다. 이렇게 TCP 소스에서 일어나는 패킷 드롭과 재전송은 throughput 의 감소를 가져오는 오버헤드로 작용하게 된다. 또한 패킷의 재전송은 파워 소모를 증가시킨다.

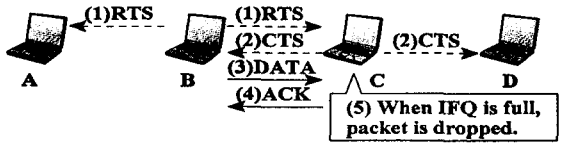

그림 2. RTS/CTS를 이용한 패킷 전송

그림 2 는 RTS/CTS 를 사용하는 IEEE 802.11 에서의 데이터 전송 과정을 보여준다. 노드 B 와 C 사이에 RTS/CTS 의 교환이 성공적으로 이루어졌다면, 노드 B 는 노드 C 를 향해 데이터 패킷을 전송한다. 이때 노드 C 가 네트워크 혼잡을 겪고 있다면, 이 패킷 전송 과정은 오버헤드로서 동작하게 된다. 만약 노드 C 의 IFQ 가 가득 차있다면, 노드 B 가 노드 C 로부터 CTS 를 받았다 하더라도, 노드 B 로부터 노드 C 를 향하는 패킷은 노드 C 에 의해 드롭된게 된다.

이러한 문제들을 해결하기 위해 2 개의 플로우 컨트롤 알고리즘을 제안하고자 한다. 하나는 수직 플로우 컨트롤로써 한 단말 노드의 계층 사이에서 동작하며, 다른 하나인 수평 플로우 컨트롤은 2 개의 이웃한 노드들 사이에서 동작한다.

2.2. Vertical Flow Control

MANET 에서는 유선 네트워크에서보다 더 빈번히 네트워크 혼잡이 발생한다. 단말 호스트는 종종 자신을 지나는 패킷들로 인해서 네트워크 혼잡을 겪게 되고, IFQ 가 가득 차 있을 때 TCP 가 패킷을 보내려 한다면, 결국 단말 호스트의 IFQ 에서 패킷 드롭이 일어나게 될 것이다. 이러한 패킷 드롭은 TCP 의 재전송을 요구하게 되고, 이는 다시 네트워크 오버헤드로서 작용한다. 이 문제에 대한 해결책으로 TCP 가 패킷을 보내고자 할 때에는 먼저 자신의 인터페이스의 생태를 확인해서 IFQ 가 받을 수 있을 만큼의 패킷에 대해서만 전송을 시도하는 보완책을 고안했다. 알고리즘 1 이 수직 플로우 컨트롤 알고리즘을 서술한다.

Algorithm 1 Vertical Flow Control

```
1: nPacket <- 0;
2: if IFQ.Length() < IFQ.Limit() then
3:   Maxburst <-min{Maxburst, IFQ.Limit()-IFQ.Length()};
4: else
5:   Maxburst <- 0;
6: end if
7: while Seq Number _ Highest ACK + cwnd do
8:   if Maxburst == nPacket then
9:   break;
10: end if
11: Send packet();
12: nPacket + +;
13: Seq Number + +;
14: end while
```

Maxburst 는 TCP 가 한번에 보낼 수 있는 패킷 개수의 최대값이며, 보통 무한대로 설정된다. 그러나 Maxburst 값을 적절하게 조절함으로서 패킷 드롭을 막을 수 있다. IFQ 의 현재 큐 길이가 최대 값보다 짧은 상태라면, Maxburst 는 두 값의 차이로 설정하고, IFQ 의 현재 길이가 최대 길이와 같거나 더 큰 상태라면, Maxburst 는 0 으로 설정되며 TCP 는 더 이상의 패킷 전송을 하지 않는다.

2.3. Horizontal Flow Control

패킷을 전송할 넥스트 홉의 IFQ 가 가득 찬 상태일 때의 패킷 전송을 막기 위해, 우리는 RTS/CTS 메시지를 다른 목적으로 사용한다. RTS 를 받은 노드는 IFQ 의 상태를 확인해서 가득 찬 상태라면 CTS 를 보내는 대신 다음의 두가지 방법중 하나를 택한다. 첫 번째 방법은 RTS 에 대한 CTS 메시지를 보내지 않는 것이다. 그러나 이 방법은 RTS 의 재전송을 최대 7 번까지 유발한다[3]. 이에 우리는 CTS 를 보내지 않는 대신, 넥스트 홉의 IFQ 가 가득 찼음을 알리는 flagged CTS 를 보내는 두 번째 방법을 제안하고자 한다. 이 CTS 를 받은 단말 노드는 넥스트 홉의 IFQ 가 가득 찼음을 알게 되고 패킷 전송을 연기하게 된다.

그러나 이 알고리즘을 모든 종류의 플로우에 대해서 적용시키는 것은 비효율적이다. Cumulative TCP ACK 이나 UDP 패킷의 경우 네트워크 혼잡에 상관없이 전송하는 것이 오히려 나을 수 있다[4]. 이상적으로는 패킷을 분류하고 이에 따라 Horizontal 플로우 컨트롤의 적용 여부를 결정해야 한다. 그러나 현실적으로 이는 불가능하기 때문에, 패킷을 크기에 따라 분류해서 RTS_Threshold 보다 작은 크기를 가지는 패킷에 대해서는 이 플로우 컨트롤을 적용하지 않도록 한다. 따라서 작은 크기를 가지는 TCP ACK 의 경우 플로우 컨트롤의 영향을 받지 않고 전송되며, 큰 크기를 가지는 UDP 패킷의 경우 플로우 컨트롤의 적용을 받게 된다. 작은 크기를 가지는 UDP 패킷의 경우는 플로우 컨트롤을 적용받지 않는데, 이는 크기가 작기 때문에 재전송으로 인한 오버헤드가 비교적 작기 때문이다.

3. 성능 평가

본 논문에 제안된 두 가지 플로우 컨트롤 알고리즘을 NS-2 를 이용해 시뮬레이션하고, 이를 통한 throughput 과 전력 소모에서의 이득을 측정하였다. 네트워크 토폴로지는 그림 1 의 토폴로지가 사용되었다. 1Mbps 의 CBR(Constant Bit Rate)를 가지는 Background 1 Traffic 이 노드 A 로부터 노드 E 로 전송되고, 노드 F 로부터 노드 G 로 전송되는 Background 2 Traffic 을 200Kbps 에서 600Kbps 까지 50Kbps 의 간격으로 증가시켰다. Background 2 Traffic 이 노드 C 를 지나기 때문에, 이를 통해 노드 C 에서의 네트워크 congestion 수준도 조절 할 수 있다.

그림 3 는 수직, 수평 플로우 컨트롤 알고리즘을 적용 여부에 따른 TCP 플로우의 throughput 을 보여준다. 중간 노드에서의 네트워크 congestion 이 심할수록 플로우 컨트롤 알고리즘을 통한 throughput 이득이 커짐을 알 수 있다.

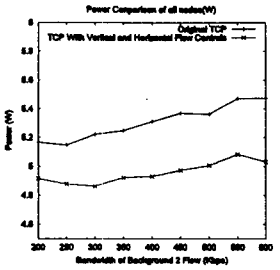

그림 3. Throughput

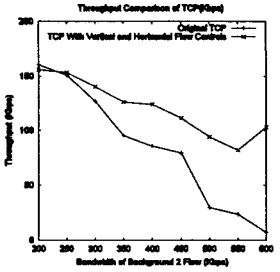

그림 4. Power 소모

그 4 는 네트워크 상의 모든 노드에 의한 전력 소모량을 나타낸다. 이로서 플로우 컨트롤 알고리즘을 사용할 경우 패킷 드롭을 줄이고 패킷의 재전송을 줄임으로써 전력 소모량을 줄일 수 있다는 것을 알 수 있다.

4. 결론

본 논문에서는 MANET 에서의 TCP 최적화를 위한 수직, 수평 플로우 컨트롤 알고리즘을 제안하였다. 수직 플로우 컨트롤의 핵심 기능은 TCP 가 IFQ 의 상태를 확인해서 불필요한 패킷 전송과 드롭을 막는데 있다. 그리고 수평 플로우 컨트롤 알고리즘은 하나의 단말 노드가 RTS/CTS 를 이용해 넥스트 홉의 IFQ 를 확인한 뒤, IFQ 가 가득 차있을 경우 패킷 전송을 연기하는 알고리즘이다. 마지막으로 제안된 플로우 컨트롤 알고리즘을 통해 더 높은 throughput 과 파워 소모의 감소 효과를 가져오는 것을 시뮬레이션을 통해 보였다.

Reference

[1] Charles E. Perkins, "Ad Hoc Networking," Addison Wesley, 2000.

[2] IEEE Computer Society, "802.11: Wireless LAN Medium Access Control (MAC) and

[3] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," to appear in IEEE INFOCOM 2003.

[4] E. Altman and T. Jimenez, "Improving TCP over multihop network using delayed ACK," to appear in MADNET 2003.