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

Yongho Seok, Youngsam Park and Yanghee Choi
Seoul National University,
Seoul, Korea
{yhseok, yspark, yhchoi}@mmlab.snu.ac.kr

*Abstract*— **In mobile ad hoc network (MANET), each end-node operates as an end-host and as a router simultaneously. Thus differently from wired-network, the occurrence of network congestion in end-host leaves much room for consideration. In the case of MANET, we can improve TCP throughput by using flow control algorithm that operates between layers because that algorithm reduces packet drops which caused by network congestion in end-hosts (source or destination). In MANET, usually IEEE 802.11 MAC protocol is used in a link layer transmission of two neighbored nodes for reliability and hidden terminal problem. But this protocol does not offer flow control mechanism in link layer. Link layer flow control can prevent packet drops caused by buffer overflow that can occur just after packet transmission. And in result, this prevention of packet drop can reduce the waste of bandwidth and energy which are limited in wireless environment. In this paper, we propose two flow control algorithms called, respectively, vertical flow control and horizon flow control. We show improved TCP throughput and reduced power consumption in end-nodes through the use of proposed two flow control algorithms in performance evaluation.**

## I. Introduction

A mobile ad hoc network(MANET) [1] is a set of wireless mobile nodes forming a dynamic autonomous network through a fully mobile infrastructure. This network is independent of any fixed infrastructure or centralized administration. A Node communicates directly with other nodes within its wireless communication range without the intervention of centralized access points or base stations.

Another feature of MANET is relatively lower throughput compared to wired network. In order to improve the throughput of MANET, several enhancements on transport and link layer considering the multi-hop routing, wireless channel and node mobility have been proposed. And recently, as the bandwidth of wireless channel is increased like in IEEE 802.11a (up to 54Mbps) and 802.11b (up to 11Mbps), the importance of flow control in the link and transport layer is also increased. Through flow control mechanism in link layer, nodes in MANET can prevent packet drop when network congestion arises. IEEE 802.11 [2], currently the most popular link layer(or MAC) protocol used in wireless network, provides the solution for both hidden terminal problem and reliability problem through the RTS-CTS-DATA-ACK exchanges. But it never carefully considers flow control issue such as 802.3x.

This work was supported in part by the Brain Korea 21 project of Ministry of Education and in part by the National Research Laboratory project of Ministry of Science and Technology, 2003, Korea.

Besides, MANET has a unique characteristic in that an end-host participates in the process of packet forwarding (i.e., relaying packets from one node to other node). When too much traffic is forwarded through an end-host, some of its own packets are dropped due to the buffer overflow. But current IEEE 802.11 MAC protocol does not consider this problem too.

In this paper, we propose two flow control algorithms to improve the throughput of MANET in Section III. Through simulation results using the NS-2 simulator, we show that proposed flow control algorithms are able to increase the throughput and also decease the power consumption in Section IV.

## II. Related Work

We briefly review the IEEE 802.11 Distributed Coordinated Function(DCF) [2]. As described in [2], a transmitting mobile station must first sense an idle channel for a time period of Distributed InterFrame Spacing(DIFS) after which it generates a random backoff timer chosen uniformly from the range $[0, w-1]$, where $w$ is referred to as the contention window. At the first transmission attempt, $w$ is set to $CW_{min}$(minimum contention window). After the backoff timer reaches 0, the mobile station transmits a short request-to-send(RTS) message. If successfully received, the receiving mobile station responds with a clear-to-send(CTS) message. Any other mobile stations which hear either the RTS or CTS packet uses the data packet length information to update its Network Allocation Vector(NAV) containing the information of the period during which the channel will remain busy. Thus, all mobile stations including hidden node can defer transmission suitably to avoid collision. Finally, a binary exponential backoff scheme is used such that after each unsuccessful transmission, the value of $w$ is doubled, up to the maximum value $CW_{max} = 2^m CW_{min}$, where $m$ is the number of unsuccessful transmission attempts.

The IEEE 802.11 frame format is shown in Figure 1. It shows the format of the MAC frame used for sending unicast data packets in an ad-hoc network. The frame control field carries frame identification information, such

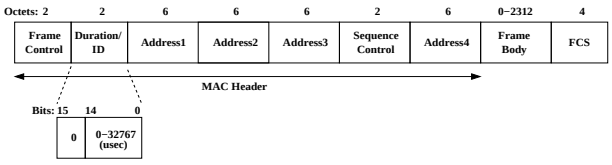


Fig. 1. General MAC frame format

as the type of frame (e.g. management, control or data), as well as protocol version information and control flags; the duration field contains the time remaining (in us) until the end of the packet transfer; the Address 3 is the unique network identifier; sequence control is a sequence number used to detect duplicate frames; and FCS is the frame check sequence. The duration field contributes the key role to our NCA; the period during transmission between given source and destination makes possible for the neighbor nodes to turn their power off resulting adaptive energy saving.

## III. Proposed TCP Optimization Algorithm

### A. Problem Definition

In MANET, each end-node performs 2 functions simultaneously. First, it operates as an end-host like the source or the destination of connection and second, it operates as a router which relays traffic between other sources and destinations. Because of this feature, network congestion in MANET must be handled differently from in wired network. Usually, network congestion arises when aggregated traffic exceeds the capacity of specific link. Network congestion saturates buffer of intermediate node, and causes packet drop. To cope with this problem, congestion control algorithm like TCP is used in each end-host. However, in wireless multi hop network, network congestion in end-host can arise more frequently than in wired network, since an end-host can act as a router simultaneously. Existing TCP does not consider this kind of network congestion seriously.

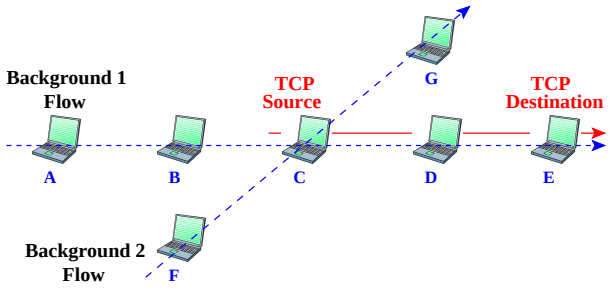


Fig. 2. Simple topology of mobile ad hoc network

Fig. 2 shows simple topology of mobile ad hoc network. In Fig. 2, node C operates as a router which relays cross traffics from node A to node E and from node F to node G. Simultaneously, node C operates as a TCP source which sends traffic to TCP destination, node E. As this topology shows, 3 flows of traffic are concentrated at node C. If a large quantity of cross traffic is generated and thus makes the IFQ of node C full, TCP packets generated by node C will be dropped and retransmitted after timeout. Because TCP does not check the state of the IFQ, it generates as many packets as the size of congestion window. Such packet drop and retransmission occurring in a TCP source node is just meaningless, and even could be an overhead which causes reduction of throughput. Also, this kind of packet drop and retransmission may increase power consumption.

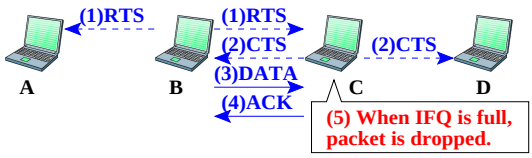


Fig. 3. Packet transmission using RTS/CTS

Fig. 3 shows the process of data transmission in the original IEEE 802.11 which uses RTS/CTS exchange. If node B wants to send packet to other node, that is node C in Fig. 3, it sends RTS message first. Then node C receives RTS message if it is within communication range of node B, and if it is not communicating with other node, it sends CTS message to node B. On receiving CTS message from node C, node B sends data packet to node C, and node C sends ACK packet to node B. But when node C suffers from network congestion, this process of packet transmission causes some inefficiency. If the IFQ of node C is full, even though node B receives CTS from node C, data packet from node B to node C will be dropped. This kind of packet drop could be another waste of bandwidth and power energy.

To solve these problems, we propose two flow control algorithms: One is the vertical flow control that operates between layers in an end-node, and the other one is the horizontal flow control that operates between two neighbored-nodes.

### B. Vertical Flow Control

According to protocol layering architecture, TCP does not consider lower layers. Typically, TCP tries to send as many packets as possible irrespective of the state of the IFQ. But as referred before, in MANET, network congestion arises more frequently than in wired network. Thus the IFQ of an end-host suffers from overloaded packets frequently, which may be transmitted from higher layer of that node or from other node. If TCP tries to send packets when its the IFQ is full, packet drop will occur at the IFQ of the end-host. These packet drops happen rarely in wired network for its high bandwidth, however in ad-hoc network, these could be a serious problem as mentioned before. As a solution for these problem, we modified existing TCP to check the state of the IFQ. Whenever TCP attempts to send packets, it would send only as many packets as the IFQ can hold by checking the state of the IFQ. Algorithm. 1 describes the scheme.

*nPacket* is the number of packets sent from TCP in one try of packet transmission. *the IFQ.Length()* and *the*

**Algorithm 1** Vertical Flow Control

```
1:  nPacket ← 0;
2:  if theIFQ.Length() < theIFQ.Limit() then
3:    Maxburst ← min{Maxburst, theIFQ.Limit() − theIFQ.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
```

*IFQ.Limit()* are functions which return current length of the IFQ and maximum length of the IFQ. *Maxburst* is the maximum number of packets TCP can send at a time, and usually is set to infinity in most TCP. *seq_number* means last sequence number of packet sent by TCP, and *highest_ack* is last sequence number of ACK packet received by TCP.

In Algorithm 1, we tried to prevent packet drops by controlling *Maxburst* adequately. When TCP tries to send packets, it check the state of the IFQ first. If current length of the IFQ is shorter than the maximum (or limit) length of the IFQ, *Maxburst* is set to the difference of those two values (Algorithm 1 line 3). The difference of the maximum length of the IFQ and current length of the IFQ means available length of the IFQ, and by setting *Maxburst* to available length of the IFQ, TCP cannot send packets more than the IFQ can accept. And if current length of the IFQ is not smaller than the maximum length of the IFQ, *Maxburst* is set to 0, which means that TCP does not send a packet since the IFQ is already full (Algorithm 1 line 5). After setting the value of *Maxburst*, TCP starts to send packets. The maximum number of packets that can be sent by TCP is the smaller number of *Maxburst* and *cwnd* (Algorithm 1 line 7 - 14).

### C. Horizontal Flow Control

To prevent sending packets when the IFQ of next-node is full, we use RTS/CTS control message on different purpose. When an end-node wants to send a packet to next-node, it sends RTS first. In original IEEE 802.11, if next-node is free to communicate with other node, it receives RTS message and sends CTS message to inform that it is free to communicate. We modified this process of exchanging RTS-CTS messages. On receiving RTS message from previous-node, next-node checks the state of its own the IFQ. After checking the state of the IFQ, if the IFQ has available space, next-node sends CTS message just like as original IEEE 802.11. When the IFQ is full, we can choose one of following two methods. First method is not sending CTS in response of RTS to prevent previous node from sending data packets. This simple method can prevent packet drop that may occur in the IFQ of next-node, but this causes retransmission of RTS up to 7 times and followed by link layer packet drop [3]. This may results in worse effect on TCP throughput and power consumption. So we choose the other method: Sending flagged CTS indicating that next-node's IFQ is full. On receiving flagged CTS, previous node which wants to send data packet recognizes that the IFQ of next-node is full. Then previous node postpones sending data packets until the next time when TCP tries to send packets. This scheme showed in Fig. 4.

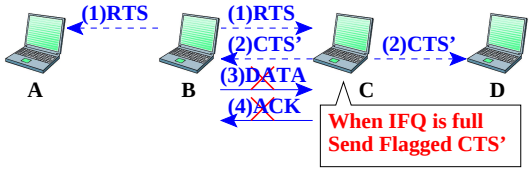


Fig. 4. Horizontal flow control for TCP optimization

This flow control algorithm can diminish predictable packet drops, however it is undesirable to apply this algorithm to all types of packets. For example, reliable transmission of TCP ACK may degrade the throughput rather than unreliable transmission of TCP ACK since it is cumulative ACK [4]. Also UDP designed for unreliable transmission can be an another case like that. It would be better to transmit delay-sensitive UDP packets of short length without flow control even if some of them are dropped in a congested receiver. So applying this algorithm only to TCP data packet with reasonable size is rather better than applying all types of packets.

Ideally, we should classify the packets according to their types and determine whether horizontal flow control algorithm will be applied or not. But in real, it is impossible to look into all packets flowed through a node due to limited resources. Thus instead of ideal method, we approximate the ideal approach by classifying the packets according to their length. Approximating the idal approach means that this algorithm will be applied only to packets whose length are longer than *RTS_Threshold*. TCP ACK that has relatively small length by nature will not be controlled by this algorithm thus it will be transmitted without considering network congestion. On the other hand, TCP packets that have sufficient length will be controlled so that the waste of bandwidth and power could be minimized. Although TCP packets that have small length are not controlled, it is trivial since retransmission of small packets results in relatively little waste of bandwidth and power. By applying this algorithm only to packets whose length are longer than *RTS_Threshold*, we can avoid unnecessary flow control for small packets like TCP ACK and delay-sensitive UDP packets with relatively small size. The scheme above is described in Algorithm 2.

When an end-node wants to send data packet to next-node, it check the size of that packet first. (Algorithm 2 SendPacket() line 2). If the size of packet to be sent is smaller than *RTS_Threshold*, then an end-node sends packet immediately without RTS-CTS exchanging. (Algo-

**Algorithm 2** Horizontal Flow Control

```
1:  SendPacket() {
2:  if PacketSize > RTS_Threshold then
3:    SendRTS();
4:    if ReceivedNormalCTS() then
5:      SendDataPacket() ;
6:    else if RecedvedFlaggedCTS() then
7:      HoldPacket();
8:    end if
9:  else
10:   SendDataPacket();
11: end if
12: }

1:  ReceiveRTSPacket() {
2:  if theIFQ.Length() < theIFQ.Limit() then
3:    SendNormalCTS();
4:  else
5:    SendFlaggedCTS();
6:  end if
7:  }
```

rithm 2 SendPacket() line 10). And if the size of packet is larger than *RTS_Threshold*, then an end-node sends RTS message to next-node and waits for CTS message. (Algorithm 2 SendPacket() line 3) When next-node receives RTS message, it checks the state of its the IFQ. If the IFQ has available space for another packets, next-node send *NormalCTS* and if not, it sends *flaggedCTS* to inform its the IFQ is full. (Algorithm 2 ReveiveRTSPacket() line 2-5) And finally, an end-node which sent RTS receives CTS message from next-node. If CTS from next-node is *NormalCTS* it sends data packet and if not, it postpones sending data packet. (Algorithm 2 SendPacket() line 4-7)

## IV. Performance Evaluation

We simulated the proposed flow control algorithms ,called vertical and horizontal flow control algorithms, to measure throughput and power consumption gains over original TCP IEEE 802.11 MAC protocol. Simulation was conducted on simple topology of mobile ad hoc network, Fig. 2 using $ns-2$ *simulator*. In simulation, the data rate of Background 1 Traffic from node A to node E was fixed to 1Mbps CBR. And the data rate of Background 2 Traffic from node F to node G was increased from 200Kbps to 600Kbps by 50Kbps. Since Background 2 Traffic passes through node C, the level of network congestion in node C can be controlled by varying the data rate of Background 2 Traffic.

Fig. 5 shows throughput of TCP flow from node C to node E either when vertical and horizontal flow control algorithms are applied or not. As the bandwidth of Background 2 Traffic is increased, throughput of TCP with vertical and horizontal flow controls as well as throughput of original TCP is decreased too. But throughput of TCP with vertical and horizontal flow controls decreases with relatively lower rate. From this graph, we can see that with the higher level of congestion, the more throughput gain could be obtained through our flow control algorithms. It is because vertical and horizontal flow control algorithm diminishes packet drops which could arises by predictable network congestions.

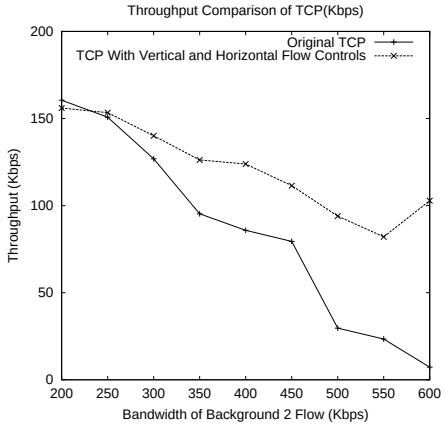


Fig. 5. Comparison of the TCP throughput between original TCP and TCP with Vertical and Horizontal flow control

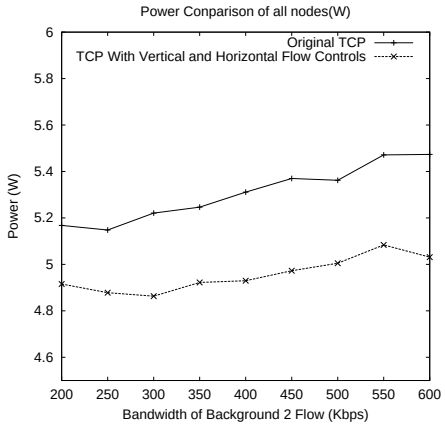


Fig. 6. Comparison of the overall power consumption between original TCP and TCP with Vertical and Horizontal flow control

Fig. 6 shows overall power consumption of simple topology of mobile ad hoc network, Fig. 2. Because TCP is designed for reliable transmission, the dropping of TCP packet causes end-to-end packet retransmission, and in result this wastes power energy which is a critical resource in wireless network like MANET. Proposed vertical and horizontal flow control algorithms prevent predictable packet drop followed by packet retransmission. In MANET, reduction of power consumption has as important meaning as throughput enhancement does. This result shows that we can reduce power consumption by using proposed flow control algorithms since they reduce packet drops and retransmissions.

## V. Conclusion

In this paper, we have presented *vertical and horizontal flow control algorithms* for TCP optimization in the mobile ad hoc network(MANET). The key function of vertical flow control is to prevent TCP from sending packet meaninglessly by checking the state of the IFQ. This prevent pre-

dictable packet drop in its own node. And the key function of horizontal flow control is to prevent an end-node from sending packet when the IFQ of next-node is full by checking the state of next-node's IFQ. In this purpose, we modified RTS/CTS control messages. We evaluated *vertical and horizontal flow control algorithms* using $ns-2$ *simulator*. The evaluation results show we can achieve better throughput and less power consumption through our proposed algorithms.

## References


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

[2] IEEE Computer Society, "802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," June 1997.

[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.

[5] Datasheet for ORiNOCO 11 Mbit/s Network Interface Cards, 2001. ftp://ftp.orinocowireless.com/pub/docs/ORINOCO/.