

# Decoupling Congestion Control from TCP for Multi-hop Wireless Networks: Semi-TCP\*

Shengming Jiang  
School of Electronic and  
Information Engineering  
South China University of  
Technology, China  
shmjiang@scut.edu.cn

Qin Zuo  
<sup>a</sup>Wireless Network Lab.  
South China University of  
Technology, China  
<sup>b</sup>ZTE Co. Ltd, China  
jessicazuo2005@163.com

Gang Wei  
School of Electronic and  
Information Engineering  
South China University of  
Technology, China  
ecgwei@scut.edu.cn

## ABSTRACT

TCP performs poorly in multihop wireless networks and even worse if end-to-end connectivity is often broken such as in challenged networks. Lots of research has been carried out but this problem has not been solved completely. Recently, hop-by-hop congestion control originally proposed for wired networks has been applied for multihop wireless networks to significantly improve performances. We think that (i) moving congestion control down to lower layers is essential to overcome TCP problems in multihop wireless networks and (ii) in this case, it is necessary to further decouple congestion control from TCP. Such TCP only retains reliability control and is called semi-TCP henceforth. Due to using hop-by-hop congestion control, the congestion control efficiency of semi-TCP will not rely on the availability of end-to-end connectivity, which makes semi-TCP more suitable than TCP for challenged networks. Besides performance improvement, semi-TCP may further reduce overall system complexity by removing redundant congestion control and using simple congestion control rather than TCP congestion window. This paper discusses such a semi-TCP using a hop-by-hop congestion control that only slightly modifies the RTS/CTS protocol used in the IEEE 802.11 DCF. Furthermore, a solution to a deadlock problem in the RTS/CTS-based hop-by-hop congestion control is also studied. The performance of this semi-TCP is investigated in comparison with TCP-NewReno via simulation in NS2.

## Categories and Subject Descriptors

C.2.2 [Network Protocols]: Protocol verification

## General Terms

Algorithms, Performance

\*This research was supported partially by The National Hi-Tech Research and Development Program of China (863) under Grant No. 2007AA01Z210.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHANTS'09, September 25, 2009, Beijing, China.

Copyright 2009 ACM 978-1-60558-741-7/09/09 ...\$10.00.

## Keywords

TCP, semi-TCP, hop-by-hop congestion control, multi-hop wireless networks and challenged networks.

## 1. INTRODUCTION

TCP is the most important transport protocol for the Internet. However many studies and experiments show that TCP performs poorly in wireless networks especially in multi-hop wireless networks as the number of hops in TCP connections increases [8, 14]. This is because TCP cannot allow a source node to learn congestion situation in wireless networks so that no proper action can be taken immediately to both ongoing and released congestions. The efficiency of the end-to-end based TCP also decreases dramatically as round-trip latency increases and it almost cannot work if end-to-end connectivity is broken (refer to Section 2.1 for more discussion), which may happen frequently in challenged networks. Moreover, some characteristics of wireless networks such as unreliable radio links, shared media and terminal mobility cause some networking functions (e.g., routing) to perform poorly, which further degrades TCP performance. Please refer to Section 2.1 for more details.

The above problems of TCP have been studied for more than one decade, resulting in many proposals. Although these proposals vary in designs and methods, similarly they all try to improve TCP's capability of judging congestion situation in networks by using more efficient mechanisms. Typical schemes include negative acknowledgement (NACK) [1, 25, 22], explicit congestion notification (ECN) [6, 18] and measurement using probing or monitoring mechanisms [24, 9]. More details can be found in [11, 16, 20, 12]. Although these proposals can improve TCP performance in wireless networks, they do not solve the problem completely. Furthermore, their performance still rely on the availability of end-to-end connectivity like TCP, which makes them unsuitable for challenged networks as discussed in Section 2.1.

Recently some proposals such as [19, 27, 3, 21, 26] apply hop-by-hop congestion control in multihop wireless networks to significantly improve performance. This control approach was originally proposed for wired networks since it can react fast to both on-going and released congestions for high performance. However, its implementation is complex since every node along a path (i.e., the source, routers or switches and destination) needs to be involved in congestion control. Therefore, it is seldom really deployed in wide-area networks. However, we think is an essential and nature solution of congestion control in shared media wireless net-

works especially for challenged network in which end-to-end connectivity is often unavailable as discussed in Section 2.2.

If hop-by-hop congestion control is implemented below the transport layer, the same function of TCP becomes redundant, and we can further decouple congestion control from TCP as discussed in Section 2.3. This TCP will only retain reliability control and is called semi-TCP henceforth. Besides performance improvement, semi-TCP also reduces overall complexity by removing redundant congestion control from the transport layer and using control simpler than TCP congestion window. Most importantly, due to hop-by-hop congestion control, its congestion control efficiency will not rely on the availability of end-to-end connectivity. This makes it more suitable for challenged networks than TCP and its variants since it is difficult to maintain end-to-end connectivity in such kind of networks.

This paper discusses such a semi-TCP using RTS/CTS based hop-by-hop congestion control in comparison with TCP-NewReno [7] through simulation in NS2. Here, the hop-by-hop congestion control only requires slight modification of the RTS/CTS handshake protocol used in the IEEE 802.11 DCF so that overall system complexity is not increased. One problem with such congestion control is a deadlock situation, in which two congested neighboring nodes always reject accepting transmission from each other so that the congestion therein can never be released. Therefore, a simply solution to this problem is also discussed.

The remainder of the paper is organized as follows. Section 2 describes semi-TCP in detail, while a semi-TCP using an RTS/CTS-based hop-by-hop congestion control is discussed in Section 3 and investigated by simulation in NS2 in Section 4. Finally, the paper is summarized in Section 5.

## 2. SEMI-TCP FOR MULTIHOP WIRELESS NETWORKS

This section first briefly summarizes the major problems of TCP (Section 2.1) before discussing hop-by-hop congestion (Section 2.2) and decoupling congestion control from TCP (Section 2.3) in multihop wireless networks.

### 2.1 Overview of TCP in wireless networks

TCP was originally designed for wired networks to control transmission reliability and congestion. The first function is realized through packet retransmission, with which, a connection is established between a pair of source and destination and the source will retransmit all packets that have not been acknowledged successfully. The second function is performed by a source through sliding its congestion window to adjust output traffic according to the congestion status determined according to the reception of acknowledgement packets (ACK). If congestion is detected, congestion window is shrunk; otherwise it keeps increasing up to some threshold. Thus, TCP simplifies intermediate networking units (e.g., routers) by only requiring sources and destinations to be involved in congestion and reliability control so that it can run successfully over various types of networks [5].

With the current design, a TCP source cannot react efficiently to congestion as discussed below. Firstly, once a congestion happens no matter where it is, the minimum reaction delay is one round-trip latency between the source and destination. This delay is the time interval from when a congestion happens to when the source's action arrives at

the congested node. It includes packet queuing delay, transmission time and propagation time and increases with path length. This is because a TCP source judges network congestion status according to the information relayed by the destination. Secondly, due to the same reason, it will take at least one round-trip latency for a source node to learn whether a congestion state is released, and more time will be used by this node to restore its normal transmission window due to the slow start mechanism adopted by TCP. In both cases, the network bandwidth is wasted.

When TCP is applied in wireless networks especially multi-hop wireless networks, many new issues arise as discussed below. Different from wired networks, the bandwidth in wireless networks is scarce so that any bandwidth wastage is undesirable. Another problem is that TCP cannot distinguish between packet losses caused by congestion or non-congestion factors such as poor channel quality or terminal mobility in wireless environments. This problem causes TCP source nodes to unnecessarily decrease transmission windows, resulting in low network throughput. This is because once a packet loss is detected, the source node has to shrink its transmission window.

If packet-level end-to-end connection is broken, all efforts for congestion control do not make sense since the congested node under consideration may become unreachable by the source or destination due to broken links caused by mobility. Mobility may also causes path change from an originally non-congested path to a congested one, which requires an immediate congestion control for the congested link. However, it is impossible for the end-to-end oriented congestion control of TCP especially with a large round-trip propagation delays. Unfortunately, the above problems caused by mobility often happens in mobile multi-hop wireless networks such as in challenged networks.

### 2.2 Hop-by-hop congestion control in wireless networks

It is a consensus that hop-by-hop congestion control is much more efficient than end-to-end one (e.g., TCP) since the former enables nodes to learn congestion situation accurately and take fast actions accordingly. Particularly, it can react quickly to what is happening over a link such as congestion or link brokage so that the unavailability of end-to-end connectivity will not affect its performance. Therefore, it is more suitable than TCP for mobile multi-hop wireless networks to handle the problems mentioned above. But hop-by-hop congestion control increases implementation complexity due to per-node involvement in congestion control. However, a radio link is a shared medium, which requires medium access control (MAC) for medium sharing. With MAC, each competing node needs to detect activities of or to interact with its neighboring nodes. Therefore, some information capture or exchange mechanisms between neighbors have been already implemented in wireless networks. In this case, it is relatively easy to implement hop-by-hop congestion control with piggyback mechanisms in multi-hop wireless networks without significant increase in system complexity.

Take the IEEE 802.11 DCF [13] as an example. Due to the hidden terminal problem, an RTS/CTS handshake protocol has been adopted and standardized. Basically, the RTS/CTS protocol requires a node to send an RTS packet first to the receiver, which will send back a CTS if it is clear to receive. It is not difficult to find that this RTS/CTS

exchange can be slightly modified by including congestion information so that hop-by-hop congestion control can be implemented. Actually, this is just the basic idea behind the hop-by-hop congestion control schemes discussed in [28, 15]. Now IEEE 802.11s under discussion also tries to consolidate such function into the new standard [3]. There are also some hop-by-hop congestion control schemes implemented at the data link layer such as [19], which changes MAC parameters such as CWmin and CWmax of IEEE 802.11e to carry congestion control information. In [21], an implicit hop-by-hop congestion control is discussed, by which the information on congestion status and control is obtained through observing transmission activities of its neighboring nodes rather than explicit information exchange.

There are also some cross-layer-2&3 designed hop-by-hop congestion control schemes such as [4, 27, 26]. The scheme proposed in [26] tries to distinguish between packet losses caused by radio link failure and congestion according to the information obtained from MAC and routing protocols. Both schemes in [4, 27] try to provide congestion control by combining hop-by-hop and end-to-end congestion controls. Particularly, an ABR-like explicit rate-based control is proposed in [4], while [27] develops and formulates a congestion control algorithm by considering MAC constrains in terms of channel access time in wireless ad hoc networks.

Note that both ATM ABR flow control and IntServ for IP QoS are per-flow based schemes aiming at better performance in terms of fairness and QoS granularity, respectively. However, it is shown that it is difficult to practically implement these schemes albeit they can perform better. These lessons are especially important for wireless networks since they are not powerful in processing and communication capability as well as energy supply. Therefore, here we advocate the hop-by-hop congestion control schemes that only need slight modification of the existing MAC sublayer for multihop wireless networks. This is because these scheme have been already implemented in the real network so that exploring them to further provide congestion control will not increase system complexity.

### 2.3 Decoupling congestion control from TCP

As discussed above, hop-by-hop congestion control is a promising and essential approach to overcome TCP problems in multihop wireless networks. Also some features of wireless network especially those based on IEEE 802.11 can be exploited to implement such control without a big increase in system complexity. Apart from the ad hoc transport protocol (ATP) [23] which only decouples congestion control from reliability control, here we suggest decoupling congestion control from TCP to further reduce processing burden and system complexity of wireless nodes by removing redundant congestion control function. Such TCP is called semi-TCP since it only retains the reliability control function. Since the efficiency of hop-by-hop congestion control does not rely on the availability of end-to-end connectivity, semi-TCP is more suitable for challenged networks than the end-to-end based TCP because it is difficult to maintain end-to-end connectivity in such kind of networks.

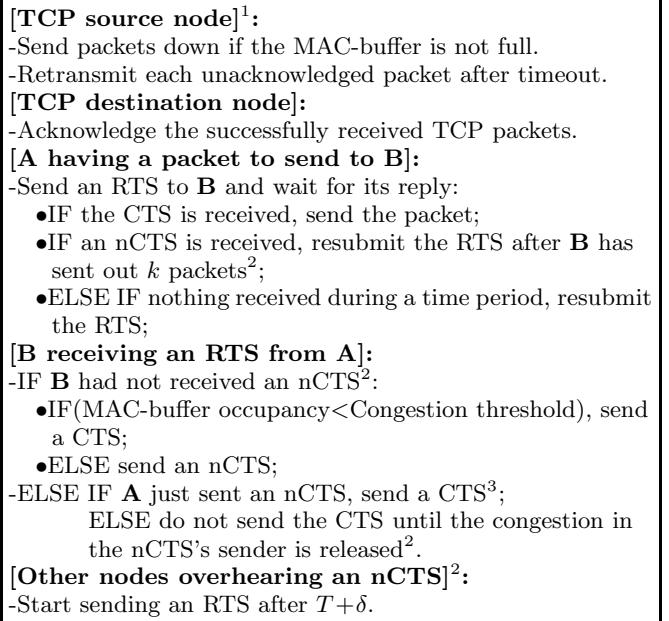
With semi-TCP, the transport layer will mainly focus on reliability control, and packets are passed down to lower layers according to the availability of the MAC buffer at source nodes. Thus, semi-TCP can avoid complex operations for congestion window, per-flow management and rate-

based congestion control. For implementation, decoupling congestion control from TCP can be either explicit or implicit. With explicit decoupling, TCP is re-implemented by removing congestion control function to reduce system complexity. Implicit decoupling tries to keep implemented TCP intact by adding a sub-layer above the MAC layer to animate TCP ACK behavior so that the implemented TCP can adjust its congestion control window according to congestion status reflected by the MAC layer at source nodes.

### 3. A SEMI-TCP USING AN RTS/CTS BASED HOP-BY-HOP CONGESTION CONTROL

The RTS/CTS protocol is slightly modified below to provide hop-by-hop congestion control for semi-TCP, and Fig. 1 gives an overview of using it for node **A** to send a packet to **B** while its detail is discussed in the following sections.

- A CTS can be sent only if no congestion occurs at the receiver; otherwise, a negative CTS (nCTS) is sent to the RTS's sender. Congestion status is measured by the occupancy of the MAC buffer.
- A TCP source node controls packet transmission according to the availability of its MAC buffer rather than the reception of TCP ACK packets.



Refer to Sections <sup>1</sup>3.1, <sup>2</sup>3.2.1, <sup>3</sup>3.2.2 for more details.

**Figure 1: Overview of nodal behaviors of semi-TCP**

#### 3.1 TCP packet transmission control

As illustrated in Fig. 1, semi-TCP no longer regulates packet sending time according to the congestion window. Instead, this time is determined by the availability of the MAC buffer as discussed below. Let  $L$  denote buffer size and  $x$  buffer space reservation. Then the general condition on buffer availability with  $x$  is simply expressed as

$$\eta(x) \leq L - x. \quad (1)$$

Unlike in wired networks, in multi-hop wireless networks such as ad-hoc networks, a node can be both a traffic source and relay node simultaneously. Therefore, some MAC buffer spaces ( $g$ ) need to be reserved for transient traffic (refer to Section 3.2.3). Then the condition for a TCP packet to be sent down to the MAC layer is just  $\eta(g)$  in this case.

If more than one TCP flow present at one node, an arbitrating scheme is needed to assure fairness among them. A simple scheme is to let each flow generate a random number once Condition (1) is held, and the flow with the largest number wins. Albeit its simplicity, this scheme may cause bandwidth wastage since the chance may be given to those flows actually suffering from congestion. Therefore, a more sophisticated scheme should give the opportunity to a flow with least presence in the buffer. It is also possible to combine these two schemes and the details should be addressed separately. Here the first scheme is adopted so that we can focus on how to decouple congestion control from TCP.

Another important function provided by TCP is the end-to-end transmission reliability control, by which each unacknowledged packet is retransmitted by the source once the retransmission timeout (RTO) is due. Only this part is kept intact in semi-TCP so that its implementation should be much simpler than that of the original TCP.

### 3.2 RTS/CTS based hop-by-hop congestion control

Here the RTS/CTS protocol originally used to handle the hidden-terminal problem is slightly modified to also provide control congestion between two neighboring nodes.

#### 3.2.1 RTS/CTS/nCTS protocol

As defined in the IEEE 802.11 DCF [13], once a node, say **A**, gets channel access, it needs to send an RTS to the receiver, say **B**, before sending a packet. Upon receiving this RTS, **B** sends a CTS to **A** if none of its neighboring nodes is transmitting (called ‘**primary condition**’); otherwise, **B** just does nothing. This protocol is modified below by integrating congestion control into it.

- If **B** had received an nCTS, it does not send a CTS until the congestion in the sender of this nCTS is released. However, if **A** is just the sender of this nCTS, **B** has to send a CTS to **A** to avoid the deadlock situation as discussed in Section 3.2.2.
- If **B** has not received an nCTS, the following consideration is carried out (refer to Fig. 1):
  - An **additional condition** imposed for **B** to send a CTS to **A** is that there is no congestion in **B**.
  - If the above primary condition is held but not the additional condition, **B** needs to send an nCTS to **A** to indicate that it is congested.
- Once **A** receives an nCTS from **B**, it needs to monitor the activities of **B**. The congestion in **B** is considered as “released” if **B** has transmitted out  $k$  packets. Only in this case, **A** can resubmit its RTS.
- The other nodes overhearing this nCTS can submit RTSs only **A** to give a priority to **A**. That is, if the congestion in **B** is released at time  $T$ , **A** can resubmit its RTS at  $T$  while the other nodes at  $T+\delta$ , where  $\delta > 0$  is a small random number.

- If **A** does not receive anything during a pre-defined time period, it needs to resubmit its RTS since the above situation may be caused by RTS collision.

**A** can transmit a data packet to **B** only if it receives the CTS from **B**, which is the same as the original RTS/CTS protocol. How to measure congestion and determine  $k$  is discussed in Section 3.2.3.

Note that once **A** has received or overheard an nCTS, it will not send anything until **B** has successfully sent out  $k$  packets so that the congestion therein can be released quickly. If, during the above period, **A** has received an RTS from another node, say **C**, it is not good for **A** to send anything for this RTS in order to allow **B** to release its congestion as soon as possible. However, with the IEEE 802.11 DCF, **C** will repeat sending the RTS for several times. This will not only waste wireless resources but also affect congestion release at **B** since **B**’s transmission may be interrupted by such RTS submissions. The original objective of such a repeated RTS submission is to check link availability. Here this design is enhanced to avoid the above weaknesses as explained below. If **C** cannot receive anything, it must be due to one of the following causes: (i) **A** (i.e., the RTS receiver) is out of the range of **C** or powered off; (ii) **A** has received an nCTS and (iii) RTS collisions. In both cases (i) and (ii), repeating RTS submission does not make sense, while case (iii) may not happen frequently due to the use of CSMA/CA before RTS transmission.

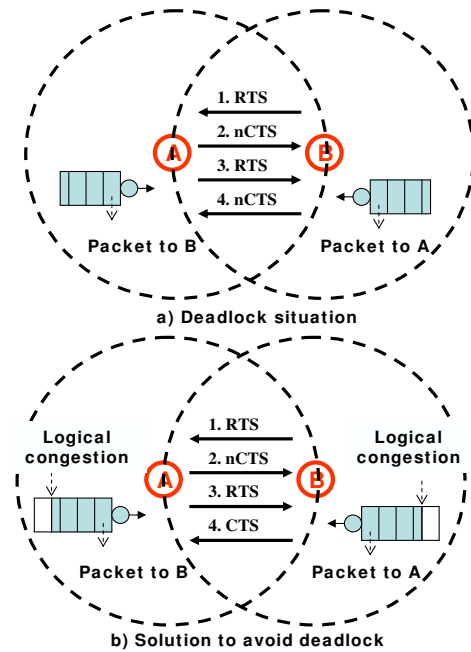


Figure 2: Deadlock situation and a solution

#### 3.2.2 Deadlock situation

If **A** returns an nCTS to the sender of an RTS **B**, it tells **B** that it is suffering from congestion and unable to receive any data. However, this will cause a deadlock situation as illustrated in Fig. 2(a). To release the congestion in **A**, **A** has to send out data in its buffer if packet dropping is undesirable. But if the receiver of the data at the head of

line (HOL) in **A**'s buffer is just **B**, which however is also congested with its HOL packet just destined to **A**. In this case, if **B** also returns an nCTS to **A** upon receiving an RTS from **A**, then the congestion in both **A** and **B** cannot be released unless dropping packets.

As illustrated in Fig. 2(b), this deadlock situation can be avoided by reserving  $n$  buffer spaces to receive packets from those congested nodes (e.g., **A** here). With this reservation, if an RTS is sent by a node that just sent out an nCTS (i.e., **A**), one of these reserved buffer spaces can be used to receive the packet from **A**. In this case, a CTS should be sent out by **B** as indicated in Figs. 1 and 2(b).

### 3.2.3 Settings of $g$ , $k$ , $m$ and $n$

The congestion status at the MAC layer is simply measured by the buffer availability subject to the thresholds set according to (1). Given a total buffer size of  $L$ , a congestion is said 'happening' if the occupancy is larger than  $L-n$  as illustrated in Fig. 3 since  $n$  buffer spaces are reserved to avoid the deadlock situation discussed in Section 3.2.2. Therefore, the condition for no congestion is simply  $\eta(n)$ .

Regarding the condition for TCP to send a packet down to the MAC layer as mentioned in Section 3.1,  $m$  buffer spaces need to be reserved to the active neighboring nodes as illustrated in Fig. 3. Considering  $n$  for the deadlock situation, we have to reserve  $g = m+n$  buffer spaces. Therefore a simple condition for TCP to send a packet is just  $\eta(m+n)$ . A sophisticated condition should consider the buffer occupancy by packets from different nodes for fair buffer sharing. This issue needs to be addressed in detail separately.

For  $m$  and  $n$ , they can be simply determined according to the number of active neighboring nodes ( $a$ ), which can be measured through detecting nodal activities. Assume each neighboring node be given  $\gamma$  buffer spaces (simply  $\gamma = 1$ ) for normal packet transmission, then  $m = a\gamma$ . For  $n$ , we can roughly reserve 1 buffer space for each active neighboring node, hence  $m = a$ . Then in this case,  $g = (\gamma + 1)a$ . Regarding  $k$ , its minimum value is 1, which means that congestion can be released after the congested node has sent out 1 packet. Consider the worst case in which a congested node has to transmit out  $n$  packets in order to avoid the deadlock situation to happen at its neighboring nodes,  $k = n$ , which should be the maximum.

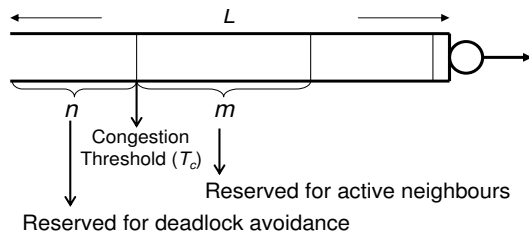


Figure 3: Threshold setting of MAC buffer

## 4. SIMULATION INVESTIGATION

Here we study the performance of semi-TCP in comparison with TCP-NewReno [7] through simulation study in NS2. To our best knowledge, decoupling congestion control from TCP and moving it to the MAC layer have not been reported so far. Here we only try to capture some fundamental performance characteristics of semi-TCP. Therefore,

similar to [8], we choose stationary and lowly mobile multi-hop wireless networks to simplify performance evaluation by avoiding consideration of other issues such as high mobility.

### 4.1 Simulation model

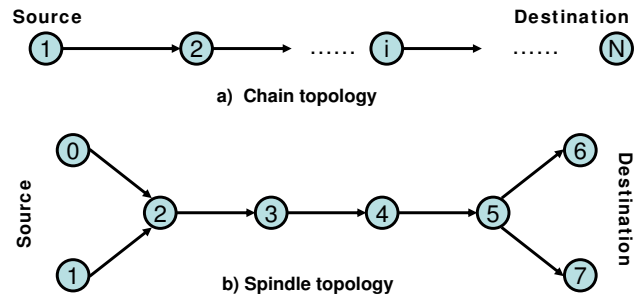


Figure 4: Simulated network topologies

The simulation adopts AODV [17] for routing and the IEEE 802.11 MAC with the enhanced RTS/CTS/nCTS protocol discussed in Section 3.2.1. Similar to [10, 8], string and spindle topologies illustrated in Fig. 4 are simulated. The successful radio transmission range is set to 250 meters while the interference range to 500 meters. A channel data rate of 11 Mbps is used for data transmission and a basic rate of 1 Mbps for control packets (e.g., RTS/CTS/nCTS). FTP is simulated with a packet size of 512 bytes. All simulations last more than 300 seconds to ensure at least 10,000 packets to go through the network. By default, congestion threshold ( $T_c$ ) is set to 10 and  $k=5$ . A sub-queue is provided to store control packets (e.g., TCP ACK, routing and ARP packets) and privileged over the sub-queue for data packets.

### 4.2 Numerical result discussion

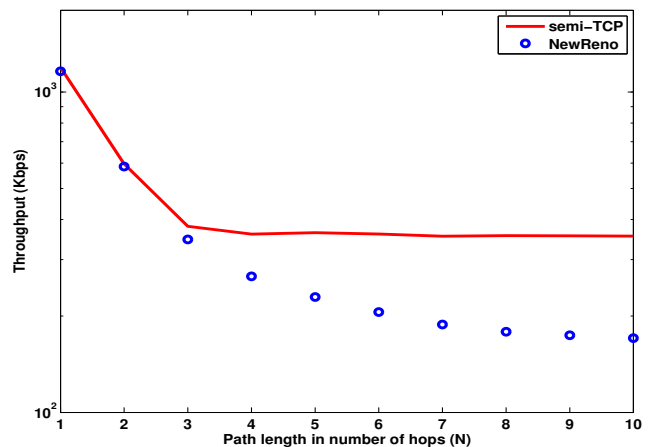
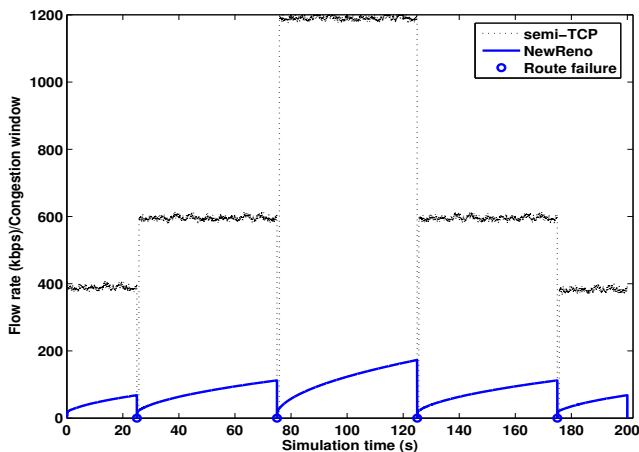


Figure 5: Throughput versus path lengths for the string topology

#### 4.2.1 Performance comparison

Fig. 5 compares the throughput given by TCP-NewReno and semi-TCP. We can find that semi-TCP can much outperform TCP-NewReno in most cases as the path length increases. In some cases, the improvement can hit 75%. For

both semi-TCP and TCP-NewReno, their throughput declines dramatically as path length increases initially. This is mainly due to the channel sharing limit imposed by the MAC-layer contention. As the number of hops increases, MAC-layer contention also increases. Particularly for a chain topology, this factor will bound the throughput of a node up to one-third of the channel rate [2]. Therefore, the throughput of the semi-TCP tends to be stable as path length exceeds 3 hops. However, the efficiency of TCP becomes lower as path length increases, the throughput of TCP-NewReno continues decreasing with the increase of path length.



**Figure 6: Adaption of semi-TCP rate and TCP-NewReno congestion window with a chain topology**

Now we look at the adaption of semi-TCP’s transmission rate and TCP-NewReno’s congestion window by using a chain topology consisting of 5 nodes with the one in the middle of the chain being the destination. The source node is moving toward the destination at a speed of 2  $m/s$ . As illustrated in Fig. 6, once a route failure happens, the congestion window of TCP-NewReno shrinks immediately and then recovers slowly. With the semi-TCP, its transmission rate is not affected by packet loss but the availability of the MAC buffer so that it can maximize wireless channel utilization. With the RTS/CTS-based hop-by-hop congestion control, a route failure does not impact passing packets down to the MAC layer if there are enough buffer spaces. However, a route failure will cause more packets to be queued in the MAC buffer, which will eventually affect transmission rate as illustrated here.

#### 4.2.2 Effect of parameter setting

As discussed in Section 3.2.3, there are some parameters to be decided for the MAC buffer. Among them,  $m$  and  $n$  can be simply decided according to the number of neighboring nodes and then  $g$  can be determined accordingly. For example, for the two topologies illustrated in Fig. 4, a node can only have 2 neighbors maximally. Therefore,  $m$ ,  $n$  and  $g$  can be easily determined. Here we mainly investigate how the following two parameters will affect the performance with a chain topology: the congestion threshold (i.e.,  $T_c$ , refer to Fig. 3) and the number of packets to be transmitted by a congested node in order to indicate a congestion is released (i.e.,  $k$ , refer to Section 3.2.1).

As illustrated in Fig. 7(a), the throughput increases when

$T_c$  is increased from 5 to 10. This is because in this case, the smaller  $T_c$ , the more misjudgement on congestion status may happen, i.e., a node is regarded as ‘congested’ even if it has many buffer spaces available. This misjudgement reduces throughput by unnecessary transmission reduction. However, if  $T_c$  is too large, there will also be more misjudgement on congestion status but in another direction, i.e., more congestions did happen, causing packet dropping and reducing throughput, as illustrated in the figure. Regarding the delay as illustrated in Fig. 7(b), it well follows the queueing theory, i.e., the larger the queue, the longer the delay, since the delay is measured only for the served packets.

Regarding  $k$ , its effect on performance is similar to  $T_c$ . As illustrated in Fig. 8(a), when  $k$  is increased from 1 to 5, the throughput also increases. However, when  $k$  is continuously increased, the throughput decreases but slowly. This is because, with  $k = 1$ , misjudgement on congestion release may happen frequently so that a node may try to transmit to a still congested node. However, if  $k$  is set too large, it will make nodes too cautious to react quickly to previously congested nodes with unnecessary waiting. Regarding the delay as illustrated in Fig. 8(b), it depends on congestion status. In the case of heavy congestion with  $k = 1$ , the delay is longer. However, in general there is no big difference in delays given by different  $k$  settings.

Fig. 9 shows the average queue length and average number of nCTSs sent for each packet for all nodes in the network against  $T_c$  and  $k$ . As illustrated in Fig. 9(a), both average queue length and the number of nCTS increase as  $T_c$  increases while decrease when  $k$  is increased as illustrated in Fig. 9(b). As  $T_c$  increases while  $k$  decreases, the number of packets waiting in the buffer increases.

## 5. REMARKS

Since hop-by-hop congestion control is much more efficient than the end-to-end control used by TCP, this paper suggests directly decoupling congestion control from TCP, which is called semi-TCP. Such semi-TCP is investigated by using an RTS/CTS-based hop-by-hop congestion control scheme, which only requires a slight modification of the original one implemented in the IEEE 802.11 DCF. The simulation studies in NS2 show that this semi-TCP can much outperform TCP-NewReno. Compared with many other proposals only modifying the original TCP, the semi-TCP may not be the best solution in terms of end-to-end interoperability with TCP due to the large popularity of TCP. However, since TCP has so many problems in mobile wireless networks while some characteristics of wireless networks favorable for hop-by-hop congestion control, it may be worth of exploiting such decoupling approach in order to achieve higher performance and lower implementation complexity in multi-hop wireless networks.

More studies are needed to further investigate semi-TCP for many issues such as (i) its impact to end-to-end behaviors of transport layer (e.g., fairness) especially when it co-exists with the original TCP, (ii) its effect to high-layer networking functions (e.g., packet routing and TCP acknowledgement schemes) and applications originally developed on top of TCP, (iii) its performance and robustness for more complex scenarios with complicated network topologies, more traffic flows and high mobility etc, particularly a deep performance investigation in challenged networks in comparison with other TCP schemes for mobile wireless networks (iv)

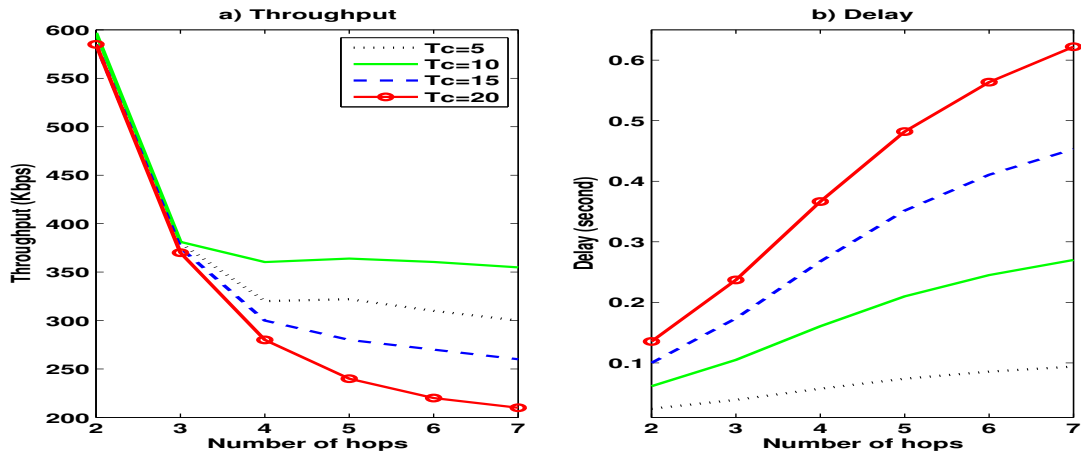


Figure 7: Performance of semi-TCP against congestion threshold ( $T_c$ ) and path lengths ( $k=5$ )

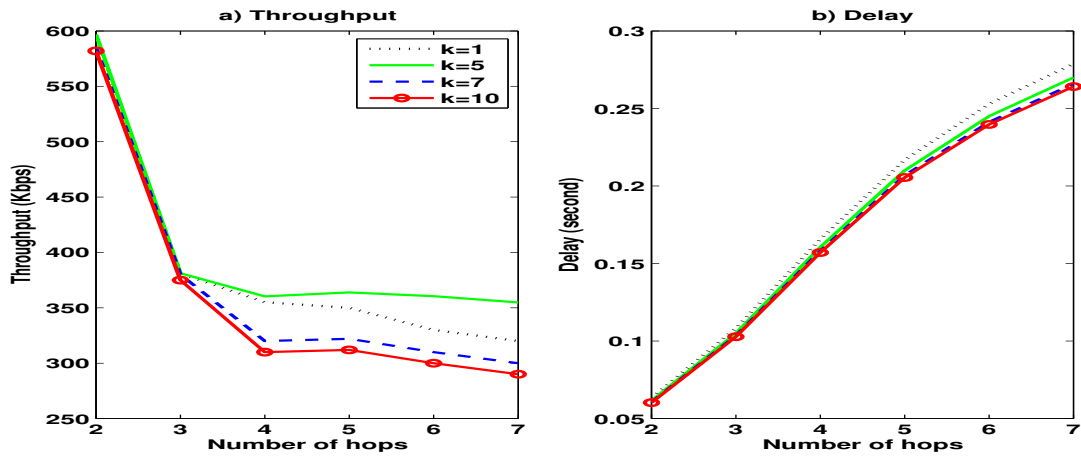


Figure 8: Performance of semi-TCP against  $k$  and path lengths ( $T_c=10$ )

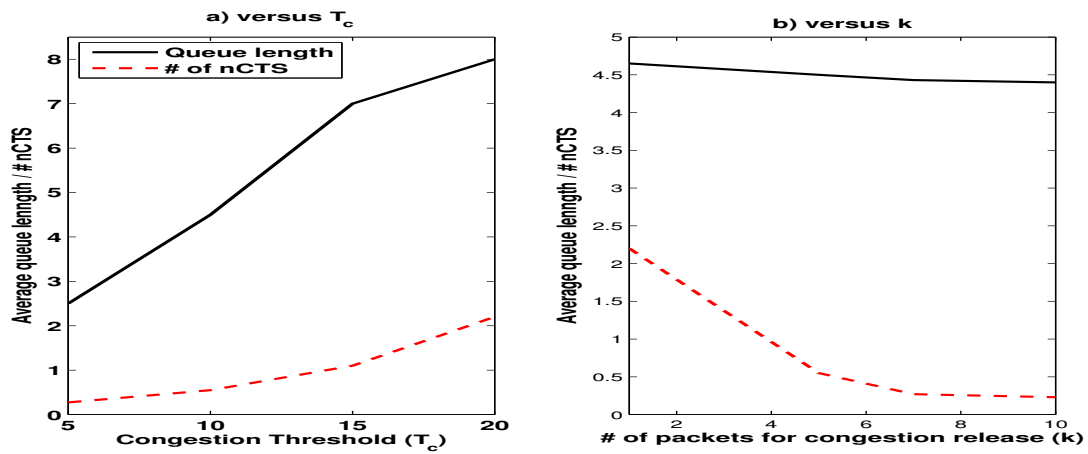


Figure 9: Average queue length and the number of nCTSs per data packet for semi-TCP

its performance against the setting of the parameters discussed in Section 3 and (v) its inter-operability with the original TCP as well as (vi) its implementation that can avoid changes in the implemented TCP layer.

## 6. REFERENCES

- [1] H. Balakrishnan, S. Sehan, and R. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks (WINET)*, vol. 1, no. 4, pp. 469–481, Nov. 1995.
- [2] D. Berger, Z. Ye, P. Sinha, S. Krishnamurthy, M. Faloutsos, and S. Tripathi, "Tcp-friendly medium access control for ad-hoc wireless networks: alleviating self-contention," in *Proc. IEEE Int. Conf. Mobile Ad-hoc & Sensor Systems*, Port Lauderdale, FL, USA, Oct. 2004, pp. 214–223.
- [3] J. Camp and E. Knightly, "The ieeec 802.11s extended service set mesh networking standard," <http://networks.rice.edu/papers/mesh80211s.pdf>, 2007.
- [4] K. Chen, K. Nahrstedt, and N. Vaidya, "The utility of explicit rate-based flow control in mobile ad hoc networks," in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*, vol. 3, Atlanta, Georgia, USA, Mar. 2004, pp. 1921–1926.
- [5] D. Clark, "The design philosophy of the darpa internet protocols," in *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, 1988, pp. 106–114.
- [6] S. Floyd, "Tcp and explicit congestion notification," *ACM SIGCOMM Comp. Commun. Reivew (CCR)*, vol. 24, no. 5, pp. 10–23, Oct. 1994.
- [7] S. Floyd and T. Henderson, "The new-reno modification to tcp's fast recovery algorithm," *IETF RFC 2582*, Apr. 1999.
- [8] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on tcp throughput and loss," *IEEE Trans. Mobile Computing*, vol. 4, no. 2.
- [9] M. Gerla, M. Sanadidi, R. Wang, A. Zanella, C. Casetti, and S. Mascolo, "Tcp westwood: congestion window control using bandwidth estimation," in *Proc. IEEE Global Tele. Conf. (GLOBECOM)*, vol. 3, San Antonio TX, USA, Nov. 2001, pp. 1698–1702.
- [10] M. Gerla, K. Tang, and R. Bagrodia, "Tcp performance in wireless multi-hop networks," in *Proc. IEEE WS. Mobile Computing Systems & App. (WMCSA)*, New Orleans, LA, USA, Feb. 1999, pp. 41–50.
- [11] A. Hanbali, E. Altman, and P. Nain, "A survey of tcp over ad hoc networks," *IEEE Commun. Surveys & Tutorials*, vol. 7, no. 3, pp. 22–36, 3rd Quarter, 2005.
- [12] M. C. I. Chlamtac and J. Liu, "Mobile ad hoc networking: Imperatives and challenges," *Ad Hoc Networks Journal (Elsevier)*, vol. 1, no. 1, pp. 13–64, Jul. 2003.
- [13] *Medium Access Control (MAC) sub layer and 3 Physical Layer Specifications*, IEEE Std 802.11-1997, <http://grouper.ieee.org/groups/802/11/main.html>.
- [14] V. Kawadia and P. R. Kumar, "Experimental investigations into tcp performance over wireless multihop networks," in *Proc 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, Aug. 2005, pp. 29–34.
- [15] J. Lee, M. Lee, R. Taori, and S. Lee, "Congestion control for wireless mesh networks," Jun. 2005, pending USA patent No. 60/686459.
- [16] K.-C. Leung and V. Li, "Transmission control protocol (tcp) in wireless networks: Issues, approaches, and challenges," *IEEE Commun. Surveys & Tutorials*, vol. 8, no. 4, pp. 64–79, 4th Quarter 2006.
- [17] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," *IETF RFC3561*, Jul. 2003.
- [18] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ecn) to ip," *IETF RFC 2481*, Jan. 1999.
- [19] B. Sadeghi, A. Yamdad, A. Fujiwara, and L. Yang, "A simple and efficient hop-by-hop congestion control protocol for wireless mesh networks," in *Proc. Annual Int. Wireless Internet Conf. (WICON)*, Boston, USA, Aug. 2006.
- [20] B. Sardar and D. Saha, "Survey of tcp enhancements for last-hop wireless networks," *IEEE Commun. Surveys & Tutorials*, vol. 8, no. 3, pp. 20–34, 3rd Quarter 2006.
- [21] B. Scheuermann, C. Locherta, and M. Mauve, "Implicit hop-by-hop congestion control in wireless multihop networks," *Ad Hoc Networks*, pp. 260–288, Apr. 2008.
- [22] F. Sun, V. Li, and S. Liew, "Design of snack mechanism for wireless tcp with new snoop," in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*, vol. 5, Atlanta, Georgia, USA, Mar. 2004, pp. 1046–1051.
- [23] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar, "Atp: A reliable transport protocol for ad hoc networks," *IEEE Trans. Mobile Computing*, vol. 4, no. 6, pp. 588–603, 2005.
- [24] V. Tsaoussidis and H. Badr, "Tcp-probing: Towards an error control scheme with energy and throughput performance gains," in *Proc. IEEE Int. Conf. Net. Protocols (ICNP)*, Osaka, Japan, Nov. 2000, pp. 12–21.
- [25] S. Vangala and M. Mehta, "The tcp sack-aware-snoop protocol for tcp over wireless networks," in *Proc. IEEE Veh. Tech. Conf. (VTC) - Fall*, vol. 4, Orlando, FL, USA, Oct. 2003, pp. 2624–2628.
- [26] X. Y. Wang and D. Perkins, "Cross-layer hop-by-hop congestion control in mobile ad hoc networks," in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*, Las Vegas, USA, Mar./Apr. 2008.
- [27] Y. Yi and S. Shakkottai, "Hop-by-hop congestion control over a wireless multi-hop network," *ACM/IEEE Trans. Networking*, vol. 15, no. 1, pp. 133–144, 2007.
- [28] H. Zhai, J. Wang, and Y. Fang, "Distributed packet scheduling for multihop flows in ad hoc networks," in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*, vol. 2, Atlanta, Georgia, USA, Mar. 2004, pp. 1081–1086.