EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Decoupling congestion control from TCP (semi-TCP) for multi-hop wireless networks

Yegui Cai[1], Shengming Jiang[2], Quansheng Guan[3] and F Richard Yu[1*]

Abstract

Although many problems for transmission control protocol (TCP) in multi-hop wireless networks have been studied with many proposals in the literature, they are not solved completely yet. Different from the existing proposals to mitigate the limitation of TCP in multi-hop wireless networks, we propose a framework of semi-TCP which decouples two functionalities of traditional TCP, i.e., congestion control and reliability control, in order to get rid of the constraint of TCP's congestion window on performance enhancement. Specifically, we employ hop-by-hop congestion control which is more efficient than its end-to-end counterpart since the control efficiency of the later relies on the availability of end-to-end connectivity which is difficult to sustain in wireless networks. We implement hop-by-hop congestion control via intra-node and inter-node congestion control, and propose a distributed hop-by-hop congestion control algorithm based on the widely used request-to-send/clear-to-send protocol. Such a semi-TCP retains the reliability control in original TCP. Extensive simulations based on network simulator-2 show the promising performance of semi-TCP over traditional schemes.

1 Introduction

Many studies show that transmission control protocol (TCP) cannot perform well in multi-hop *ad hoc* networks [1-3]. This is because TCP cannot allow the source node to quickly learn the exact congestion situation in wireless networks so that no proper action can be taken immediately to both ongoing and released congestions. Moreover, some characteristics of wireless networks, such as unreliable radio links, shared media, and terminal mobility, cause some networking functions such as routing to perform poorly, which further degrades TCP performance.

TCP in wireless networks has been studied for more than one decade with many proposals published in the literature. Although these proposals vary in designs, they share a similarity in effort to improve TCP's capability of judging congestion status in networks using more efficient mechanisms. Typical schemes include negative acknowledgement [4-7], explicit congestion notification [8], and measurement using probing or monitoring mechanisms [9,10]. Extensive surveys on TCP in wireless networks can be found in [1,2,11-13].

Although these proposals can improve TCP's performance in wireless networks, they do not solve its problems completely. Recently, some proposals such as [14-17] apply hop-by-hop congestion control in multi-hop wireless networks to improve performance significantly. This approach was originally proposed for high performance in wired networks since it can react fast to both ongoing and released congestions, but its implementation is complex since every node along a path needs to be involved in congestion control so that it is rare to be really deployed in wide-area networks. However, we think that such congestion control is a solution of TCP's many problems in shared-media wireless networks. And as what our study in the following shows, some features of wireless networks can be exploited to implement such a control without a big increase in complexity.

To this end, we propose a novel framework termed *semi-TCP* [18]. Different from existing works, semi-TCP jointly considers the efficiency of congestion control and the functionalities of a transmission control protocol. An approach to tackle TCP's disability to obtain the exact knowledge of network congestion is to use hop-by-hop congestion control instead of end-to-end congestion control. If a hop-by-hop congestion control is implemented below the transport layer, the same function of

*Correspondence: richard_yu@carleton.ca
[1]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada
Full list of author information is available at the end of the article

TCP becomes redundant. In this case, semi-TCP suggests decoupling congestion control from TCP and moving it down to lower layers, and only its reliability control function is retained. With semi-TCP, the TCP congestion window is no longer used to regulate packet sending rate so that the throughput will not be limited by the round-trip time and the performance can be further improved. Furthermore, with the hop-by-hop congestion control, the congestion control efficiency will not rely on the availability of path connectivity.

The most related work with semi-TCP is the *ad hoc* transport protocol (ATP), an 'antithesis of TCP' [19]. ATP has the following characteristics: (1) A rate-based congestion control is used to replace the window-based congestion control of TCP; (2) A selective acknowledgment (ACK) is used to replace the accumulative ACK of TCP; (3) For congestion control, the intermediate nodes in the network provide congestion information which is consolidated at the ATP receiver and is fed back to the ATP sender. However, the congestion control in ATP still employs end-to-end congestion control on the transport layer. Such approach requires the coordination of the network and data link layers to provide the information required by the per-flow rate-based congestion control carried out by the source. This may complicate its implementation.

The contributions of this paper are as follows:

- We study the concrete protocol design issues in the framework of semi-TCP. Even the idea of hop-by-hop control can trace back to the age to ATM, there are still not very extensive literature in this line of research especially when the two major functionalities of TCP are decoupled.
- An important problem in hop-by-hop congestion control is its complexity since it requires the network to involve in the congestion control procedure. Here, we implement the hop-by-hop congestion control scheme based on widely used request-to-send/clear-to-send (RTS/CTS) protocol. Our scheme is simple to realize since only media access control (MAC) layer is involved and we take advantage of the broadcasting nature of wireless media to ease hop-by-hop congestion control.
- In the protocol design for hop-by-hop congestion control, we observe the deadlock problem which prevents congestion situations to be efficiently released. An algorithm is then proposed to fully address the deadlock problem.
- From extensive simulations on our platform developed based on network simulator-2 (NS-2), we find out significant performance gain of semi-TCP comparing with the state-of-the-art such as TCP-adaptive pacing (AP) [20].

The remainder of the paper is organized as follows. Section 2 discusses in detail the major reasons for decoupling congestion control from TCP, and a semi-TCP scheme using RTS/CTS-based hop-by-hop congestion control is described in Section 3. A simulation investigation in NS-2 is provided in Section 4. Finally, the paper is summarized in Section 5.

2 Motivations for congestion control decoupling

In general, TCP faces the following problems: the effect of the round-trip time (RTT) and ACK, misjudgment on non-congestive losses, and slow reaction to congestion. This section discusses the necessity of decoupling congestion control from TCP in multi-hop wireless networks.

2.1 Constraint of congestion window

Since TCP adopts a congestion control window to control the amount of output traffic based on the reception of ACKs returned by the destination node, the round-trip time (RTT) between the source and destination nodes significantly affects TCP's throughput. This RTT mainly includes packet queuing delay, transmission time, and propagation delay, and increases with path lengths. According to [21], in the wired-line network, the maximum sending rate for a TCP connection over a single cycle of the steady-state model is

$$\lambda_m \leq \frac{1.5l}{R}\sqrt{\frac{2}{3p}}, \tag{1}$$

where R is the minimum round-trip time, p is the steady-state link drop rate, and l is the maximum TCP segment size.

For a TCP connection consisting of n links, each of which has a round-trip time of τ with a dropping rate of p for packets, R can be estimated by $n\tau$, and the throughput is bounded by

$$\chi \leq \frac{l}{n\tau}\sqrt{\frac{3}{2p}}(1-p)^n, \tag{2}$$

which shows that the throughput decreases rapidly with n especially with small p, as illustrated in Figure 1. This is because in this case, n is the dominant factor that affects the performance. This phenomenon has been observed empirically as reported in the literature such as [22].

2.2 Effect of RTT and ACK

The reliability control in TCP is realized through segment retransmission, i.e., a TCP source retransmits each transmitted packet that has not been acknowledged successfully. The congestion control is performed by a source through throttling output traffic following its congestion window, which is determined according to the congestion status inferred through the reception of ACK segments. In

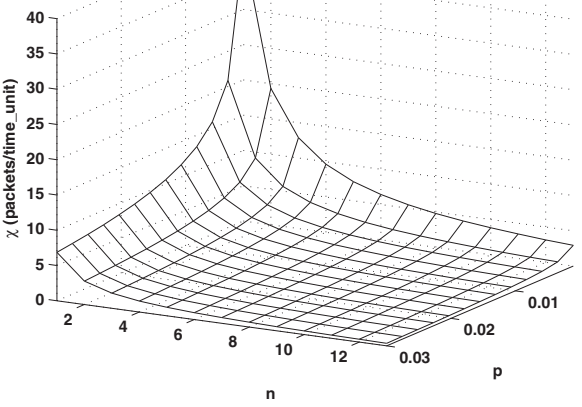

Figure 1 Maximum throughput of TCP ($\frac{l}{\tau} = 1$) versus p and n.

the case of congestion, the congestion window is shrunk; otherwise, it is increased for every received ACK and almost doubled every RTT.

Although the size of an ACK packet is much smaller than that of a data segment, it needs to go through the same MAC contention procedure as for a data segment to access the wireless channel. Therefore, reducing the number of ACK transmitted along the reverse route can lessen the MAC contention on the forward route since the wireless channel is a shared media. Therefore, delaying ACK transmission to make one ACK to acknowledge more segments can significantly improve TCP performance [23,24]. However, reducing the number of ACK to be sent by the destination will slow down the growth of the congestion window at the TCP source, thus decreasing the network throughput.

2.3 Misjudgment on congestion status

In wireless networks especially multi-hop *ad hoc* networks, many new issues arise for TCP. One problem is that TCP cannot distinguish between congestive losses and other losses caused by channel unreliability and terminal mobility. This problem causes the TCP source to unnecessarily decrease its congestion control window once a retransmission timeout (RTO) occurs, resulting in low network throughput. Furthermore, lost and delayed ACKs in the reverse route may also cause the source not to receive ACKs in time, which is also regarded as congestive losses on the forward route by the source node.

Another type of misjudgment is that even if the initial judgment on congestion status in a route is correct, this judgment may become invalid due to instant changes in the route. This change may be caused by either a routing protocol that cannot underpin a path or terminal mobility which changes radio links frequently. If the original route is changed, all effort for congestion control along this route no longer makes sense since the original congested node may not be part of the current route due to mobility, which may also cause a congested link to become part of an original congestion-free route.

The above problems are mainly due to the fact that TCP cannot have enough information on the network status for congestion control. Therefore, decoupling the congestion control from TCP and moving down this function to lower layers can avoid these problems since the lower layers can know immediately what happens in the network.

2.4 Slow reaction to congestions

A TCP source cannot react quickly to a congestion especially with a large RTT as discussed below. Once congestion occurs no matter where it is, the minimum reaction time, which is the time interval between when a congestion occurs to when the source's reaction arrives at the congested node, is at least one RTT. This is because the TCP source infers the network congestion status according to the reception of ACKs fed back by the destination. Due to the same reason, it will take at least one RTT for the source to learn whether a congestion state is released, and more time will be taken by the TCP source to restore its normal congestion window due to the slow start and congestion avoidance mechanisms adopted by TCP. In both cases, the network bandwidth is wasted. Different from wired networks, the bandwidth in wireless networks is scarce so that any bandwidth waste is undesirable.

If congestion control is carried out by the data link layer, the upstream node can take an immediate action on the congestion occurring at its downstream node by a link round-trip delay, which is much shorter than an end-to-end RTT.

2.5 Hop-by-hop congestion control in wireless networks

Although the hop-by-hop congestion control is more efficient than the end-to-end control and suitable for multi-hop mobile *ad hoc* networks (MANETs), its implementation complexity is high due to per-node involvement in congestion control. However with the MAC implemented in shared-media wireless networks, each node needs to detect activities of other nodes and even to interact with each other. Therefore, some mechanisms for information capture and exchange between neighbors have been already implemented in wireless networks. In this case, it is relatively easy to implement a hop-by-hop congestion control with piggyback mechanisms without a big increase in implementation complexity.

Take the IEEE 802.11 DCF [25] as an example. To address 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 first to the receiver, who 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 for hop-by-hop congestion control. Actually, this is just the basic idea behind the hop-by-hop congestion control discussed in [26].

There are also some other hop-by-hop congestion control schemes at the data link layer such as [14], which changes MAC parameters such as CWmin and CWmax of IEEE 802.11e to carry congestion control information. In [16], 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. Some cross-layer-designed hop-by-hop congestion control schemes have been also investigated [15,17,27]. All these hop-by-hop congestion control schemes have been investigated without decoupling congestion control from TCP, thought it makes the congestion control in TCP redundant.

3 A semi-TCP based on RTS/CTS protocol

Motivated by the previous section, this section discusses the protocol design of semi-TCP in IEEE 802.11 multi-hop wireless networks. The two aspects of the scheme are intra-node and inter-node congestion control. In *intra-node congestion control*, the upper layer in a wireless node limits the delivery of data packets to the lower layer according to the congestion situation in the lower queue. In *inter-node congestion control*, the neighboring nodes cooperate to release the congestion. We also spot out and discuss the deadlock situation in hop-by-hop congestion control. An effective scheme is proposed to solve this problem accordingly.

The following notations are used in the discussion.

- L: buffer capacity.
- $\aleph$: queue length, i.e., the occupancy of the buffer.
- T_c: general congestion threshold ($T_c \leq L$).
- m: buffer spaces reserved for transient traffic.
- k: the number of packets the congested node transmits before it is considered not congested.
- g: buffer spaces reserved to avoid the deadlock situation discussed in Section 3.3.

3.1 Intra-node congestion control

As mentioned earlier, semi-TCP does not use the congestion window of TCP to control the number of segments injected into the network. Instead, the number of segments to be transmitted is determined by the congestion status of the buffer at the lower layer, particularly the MAC sub-layer, as illustrated in Figure 2. In general, the buffer is regarded free of congestion if the following condition is satisfied:

$$\aleph < T_c. \tag{3}$$

Here, the congestion is a logical status rather than a physical congestion state in which the whole buffer has been occupied.

In a multi-hop MANET, a node can be both a traffic source and a router simultaneously. Under heavy traffic load, the source traffic of a node may dominate its buffer, causing transient traffic from other nodes to have less or even no opportunity to use the buffer. Therefore, some buffer spaces need to be reserved for transient traffic, which is admitted if

$$\aleph < T_c + m. \tag{4}$$

Another important function of TCP is the end-to-end reliability control, through which each unacknowledged segment is retransmitted by the source node once the RTO is due, until the segment is positively acknowledged. Only this part is kept in semi-TCP. Note that with TCP, duplicate ACKs are sent by the destination for the fast retransmission of out-of-order segments and the fast recovery of congestion widow. Since with semi-TCP, the congestion window is no longer used so that no duplicate ACKs are required to send in order to reduce the traffic load on the reverse route to further improve the performance.

3.2 Inter-node congestion control based on RTS/CTS

With inter-node congestion control, the congestion situation in the region will be implicitly fed back to the source node such that the sending rate of the source node will be throttled. Even there are many performance issues in the IEEE 802.11 when it is applied in multi-hop wireless networks, IEEE 802.11-like protocols are deployed in many testbeds [28]. So here, we implement the inter-node congestion control scheme based on IEEE 802.11 RTS/CTS protocol. We first introduce two subtypes of RTS and CTS to carry the congestion information: request-to-send-congested (RTSC) indicates that the sender of this RTSC is congested; clear-to-send-congested (CTSC) indicates that the sender of this CTSC is congested. We can implement these two subtypes by setting the idle bits in the original RTS and CTS frames.

In the following, we will introduce the procedure of the hop-by-hop congestion control algorithm based on the widely used RTS/CTS protocol. For the ease of presentation, we denote the two nodes involved as node A and node B. Suppose node A first senses the idle channel, and it has a packet for node B.

According to the situations of node A and node B which are in, all possible combinations of RTS(C) and CTS(C) used by this semi-TCP implementation are depicted in Figure 3 and are discussed below. When node A has a frame to send to node B, node A needs to first decide whether it should contend for the channel according to

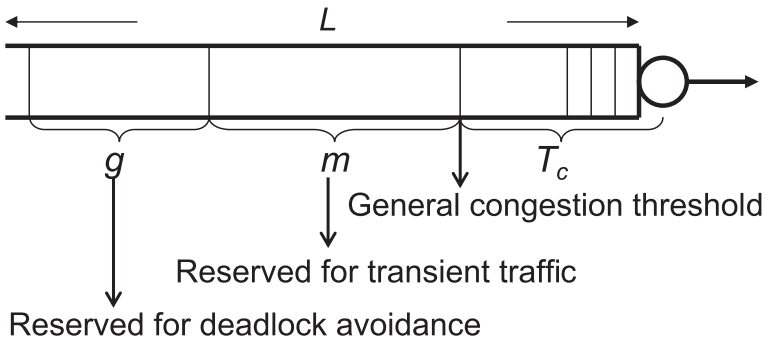

Figure 2 Threshold settings of logical congestion status at the MAC buffer.

Algorithm 1 Node *A* has a packet for node *B*

```
if Node B is congested then
    Defer transmission
else
    if Node A is congested then
        Send RTSC
    else
        if There is no neighboring node congested then
            Send RTS
        else
            Defer transmission
```

the congestion status of its neighboring nodes. Node *A* detects the congestion situation of the neighboring nodes (e. g., node *B*) by overhearing the channel. Whenever an RTSC or a CTSC frame transmitted by node *B* is monitored, node *B* is regarded as congestion by node *A*, and a timer is started for node *B*. When node *B* has transmitted *k* packets or the timer times out, node *A* will justify that the congestion in node *B* has been released. For the nodes overhearing either an RTS or a CTS, they just follow the network allocation vector carried by the RTS and CTS to decide their behaviors, which are the same as defined in the IEEE 802.11 DCF.

Algorithm 1 is the procedure when node *A* has a packet to send to node *B*. If node *A* is not congested and some node(s) in its vicinity are congested, node *A* needs to postpone the channel contention until the congestion is released in order to accelerate congestion release.During node *A*'s waiting for an RTS or RTSC resubmission, if it receives an RTS(C) from another node, say node *C*, node *A* should not grant this request in order to release its own congestion. Then, node *C* will receive nothing and needs to resubmit this RTS(C) according to the IEEE 802.11 DCF, which defines a station short retry limit (SSRL) for the resubmission. However, a large SSRL may affect congestion release at node *B* since its transmission may be affected by such RTS resubmission. Therefore, the number of such retransmissions should be limited.

Once node *B* receives an RTS(C) from node *A*, if there exists the hidden terminal problem, it just does nothing as defined in the original DCF protocol; otherwise,

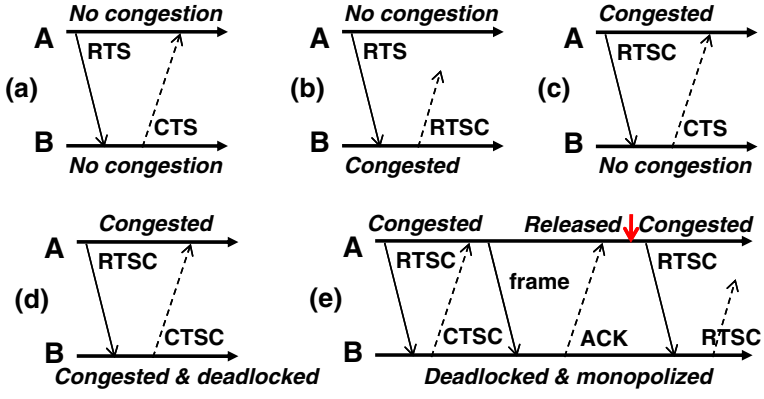

Figure 3 Combinations of RTS(C)/CTS(C) handshaking without hidden terminals: shorter dashed line means that node A may not be the receiver. (a) No congestion. **(b)** Node B is congested. **(c)** Node A is congested. **(d)** Both are congested and the deadlock situation. **(e)** Deadlock and monopolized.

Algorithm 2 Node *B* receives RTS or RTSC from node *A*

```
if Node B is congested then
    if Node A had sent node B a CTSC then
        Send CTS
    else
        if The type of request from node A is RTSC then
            if The HOL packet in node B is to node A, i.e., deadlock then
                Send CTSC
            else
                Do not reply, and set the minimum contention window as MIN_CONGESTION
        else
            Send RTSC for the HOL packet immediately to reuse the channel
else
    Send CTS
```

node *B* simply returns a CTS to node *A* if node *B* is not congested, as illustrated in Figure 3a,c. However, if node *B* is congested, it needs to further take into account the following factors in making its decision. (1) If node *B* receives an RTS from node *A*, it knows that node *A* is not congested and tries to release its own congestion first. To this end, node *B* submits an RTSC to the destination of the first frame in the queue, as illustrated in Figure 3b. Once node *A* receives or overhears this RTSC, it knows that its RTS is rejected. (2) If node *B* receives an RTSC from node *A*, node *B* has to return a CTSC to node *A* if there is a deadlock situation, as illustrated in Figure 3d; otherwise in the case of monopolization, node *B* sends an RTSC to release its own congestion, as illustrated in Figure 3e. Algorithm 2 depicts the above decision process when node *B* receives a request from node *A*. More discussion on the deadlock situation is given in Section 3.3.

3.3 Deadlock and monopolization situations

Since in the proposed hop-by-hop congestion control, an RTS may be rejected due to the congestion status at the receiver. This rejection may lead to a deadlock situation, as illustrated in Figure 4a, where both node *A* and node *B* just reject each other since they both are congested as discussed below. To release the congestion in node *A*, node *A* has to send out frames in its buffer. If the destination of the first frame queued in node *A* is just node *B*, node *A*'s request will be rejected by node *B* if it is also congested. A deadlock takes place if the destination of the first frame queued in node *B* is just node *A* since node *A* will also reject node *B*'s request. In this case, the congestion in both node *A* and node *B* cannot be released unless dropping is carried out.

A solution of this deadlock situation is to reserve some buffer spaces (g) to avoid the above mutual rejections, as illustrated in Figure 4b. That is, if node *B* receives an RTSC from node *A* while node *B* is also congested with its first frame to be sent to node *A*, it then infers that a deadlock situation takes place. In this case, it has to grant node *A*'s request by returning node *A* a CTSC rather than a CTS to indicate its congestion status. Since some buffer spaces have been reserved for transient traffic discussed above and the deadlock is relevant to transient traffic, the buffer reservation for deadlock should be privileged over

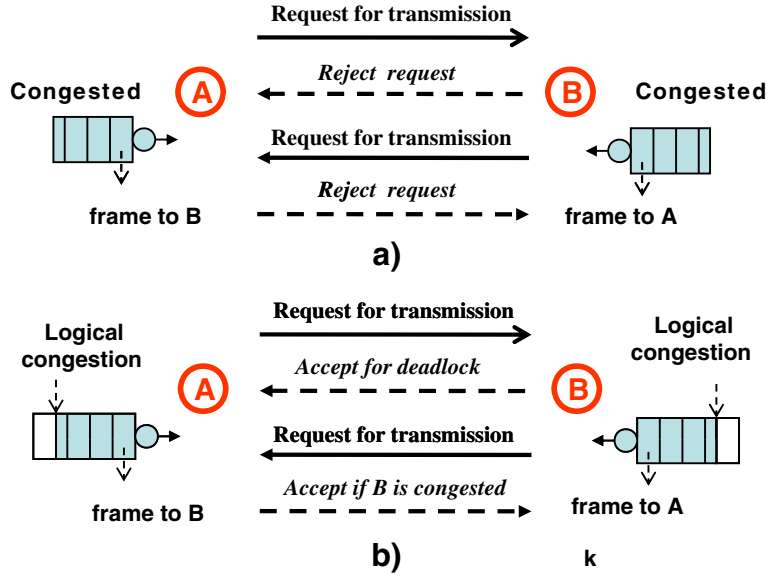

Figure 4 Deadlock situation and an avoidance solution; (**a**) deadlock situation, (**b**) solution to avoid deadlock.

transient traffic. Therefore, the buffer availability to avoid the deadlock situation is justified by

$$\aleph < T_c + m + g. \tag{5}$$

However, the above solution may lead to node A's monopolization as illustrated in Figure 5, where both node A and node B are congested and deadlocked each other. That is, after node B returns a CTSC to node A, node A has successfully transmitted a frame and released its congestion. Then, node A can accept a request from other nodes such as node B or node C. If node B fails in channel contention while node C wins, node A becomes congested again after it receives a frame from node C. In this case, if the first frame queued in node A is still toward node B, a deadlock between node A and node B occurs again, which forces node B to accept another frame from node A, causing node B more congested. To solve this problem, node B should only grant the first RTSC from node A while rejecting its subsequent RTSCs if any.

3.4 Settings of T_c, m, and k

The condition for the transport layer to send a segment to the MAC layer is dominant by Eq. (3), where T_c needs to be determined. Obviously, the larger T_c, the higher link utilization is since in this case, the link need not wait for traffic from the upper layer, instead it will start to content the channel once the wireless channel is available. However, in a shared-media network where MAC has to be used to arbitrate nodes for using media, the heavier traffic load, the higher MAC contention will be, which causes the hidden terminal problem more severe. Thus, T_c should not be set too large. Intuitively, if the queue is not empty, the link will not be wasted; hence, T_c can be set to 1 by default.

To balance media access opportunity between the source traffic and transient traffic at a node, Eq. (4) is used to give a priority to transient traffic with a reservation of m buffer spaces, as illustrated in Figure 2. As mentioned above, high buffer occupancy is undesirable in a shared-media network so that each neighboring node should have only a buffer space reservation; hence, a maximum m equal to the number of its neighboring nodes while a minimum m equal to 1 in the case of $T_c = 1$. Similarly for g used in Eq. (5) to avoid the deadlock situation, we can roughly reserve one buffer space for each node while g can also be simply set to 1 in the s case of $T_c = 1$.

Regarding k, its minimum value is 1, which means that a congestion state can be released after a frame has been sent out by other nodes. Consider the worst case in which a congested node has to transmit $g + m$ frames to release a congestion, the maximum k should beset to $g + m$ as well.

4 Simulation investigation

Here, we study the performance of semi-TCP through simulation in NS-2 by focusing on some fundamental performance characteristics of semi-TCP[a].

4.1 Simulation model

The simulation adopts AODV [29] for routing and the IEEE 802.11 MAC with the modified RTS/CTS protocol discussed in Section 3.2. Similar to [30,31], chain, parallel, spindle, cross, and random mobile topologies illustrated in Figure 6 are simulated. The successful radio transmission range is set to 250 m while the interference range to 500 m. The default two-ray ground reflection channel model is used. A channel data rate of 2 Mbps is used for data transmissions and a basic rate of 1 Mbps for control frames, i.e., RTS(C), CTS(C), and ACK. FTP is simulated with a segment size of 512 bytes.All simulations last more than 500 s to ensure enough packets to go through the network. A sub-queue is provided to store control frames, which are privileged over data frames.The simulated environment for the random and mobile scenario as illustrated in Figure 6e is an $a \times b$ rectangle. For node motion, the random way point mobility model is adopted, which is configured with a maximum

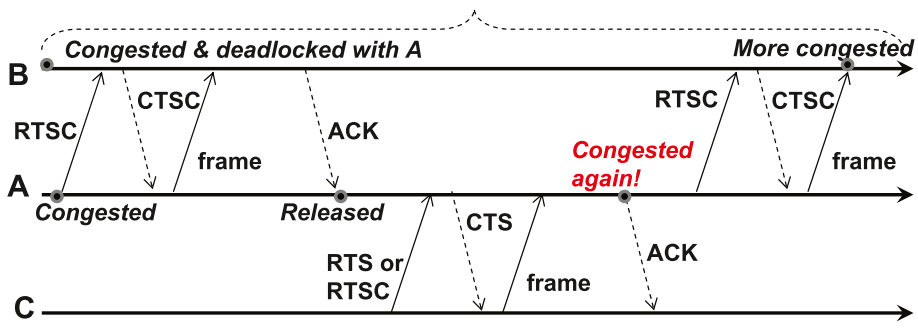

Figure 5 Monopolization situation.

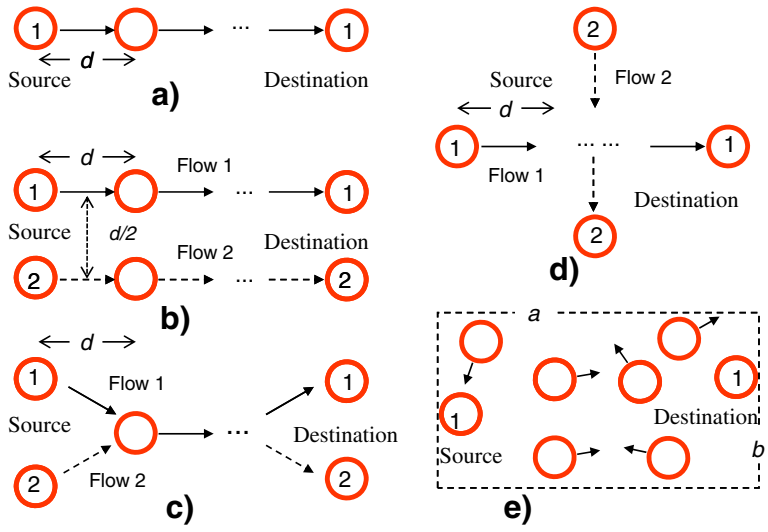

Figure 6 Simulated network topologies: the distance between two adjacent nodes $d = 200$ m. (a) Chain, **(b)** parallel, **(c)** spindle, **(d)** cross, and **(e)** random mobile.

speed ($S_{\max}$), a minimum speed ($S_{\min}$), and a pause time (PT).

This investigation compares semi-TCP with TCP-AP [32] as explained below. TCP-AP has been demonstrated much better than TCP-NewReno [33] in IEEE 802.11 multi-hop *ad hoc* networks. The basic idea of TCP-AP is that each TCP source node needs to control the interval of sending packets down to be close to the current four-hop propagation delay by jointly using TCP congestion window control. This is because this delay actually corresponds to the distance between two mutually hidden terminals in the multi-hop *ad hoc* network using the IEEE 802.11 MAC protocol. Therefore, the number of collisions caused by hidden terminals can be largely reduced through AP. Obviously, such efficiency largely depends on the estimation accuracy for the four-hop propagation delay, which however is dynamic by nature especially in mobile environments or with time-varying traffic loads.

The major performance indicators for comparison include throughput, delay, dropping ratio, and path length, which are defined below.

- Throughput: average number of data bytes successfully received by the destination node per time unit.
- Delay: average time interval between a frame's arrival at the MAC layer of the source node and its arrival at the MAC layer of the destination node.
- Dropping ratio: ratio of the total number of data segments dropped to the total number of data segments transmitted in the network during a simulation.
- Path length: average number of hops that a frame has traveled from the source node to its destination node.

4.2 Effect of station short retry limit

As mentioned earlier, SSRL defines the maximum number of RTS retrials per data frame. An RTS failure after SSRL retransmissions will cause a frame dropping. Therefore, SSRL is an important parameter to the network performance. Figures 7, 8, and 9 plot the throughput, dropping ratio, and delay given by TCP-AP (AP) and semi-TCP (Semi) against SSRL for three topologies. Here, both the source and destination are set to be fixed in order to focus on SSRL's effect.

As shown in Figure 7, for both TCP-AP and semi-TCP, their throughput increases with SSRL until SSRL is roughly equal to 9 for the chain and parallel topologies. For the random mobile topology, two scenarios are simulated. In scenario 1 (S1), the space size is set to 1,500 m $\times$ 300 m, and the speed is fixed at 20 m/s for each node except for the source and destination which are stationary, while in scenario 2 (S2), the size is 1,500 m $\times$ 4,800 m, and the speed ranges from 5 to 45 m/s for each node. The above phenomenon can also be found in S1 but not in S2. In general, a large SSRL can provide more chance for an RTS to pass through, resulting in low dropping ratio, as illustrated in Figure 8. However, as SSRL continues to increase, for both the chain and parallel topologies as illustrated in Figure 8a,b, dropping is almost zero but their throughput decreases or converges, as illustrated in Figure 7a,b. This is due to that in this case, frequent RTS retrials intensify the MAC contention, leading to a decrease in MAC effective throughput. Once MAC contention degree reaches its maximum, the throughput

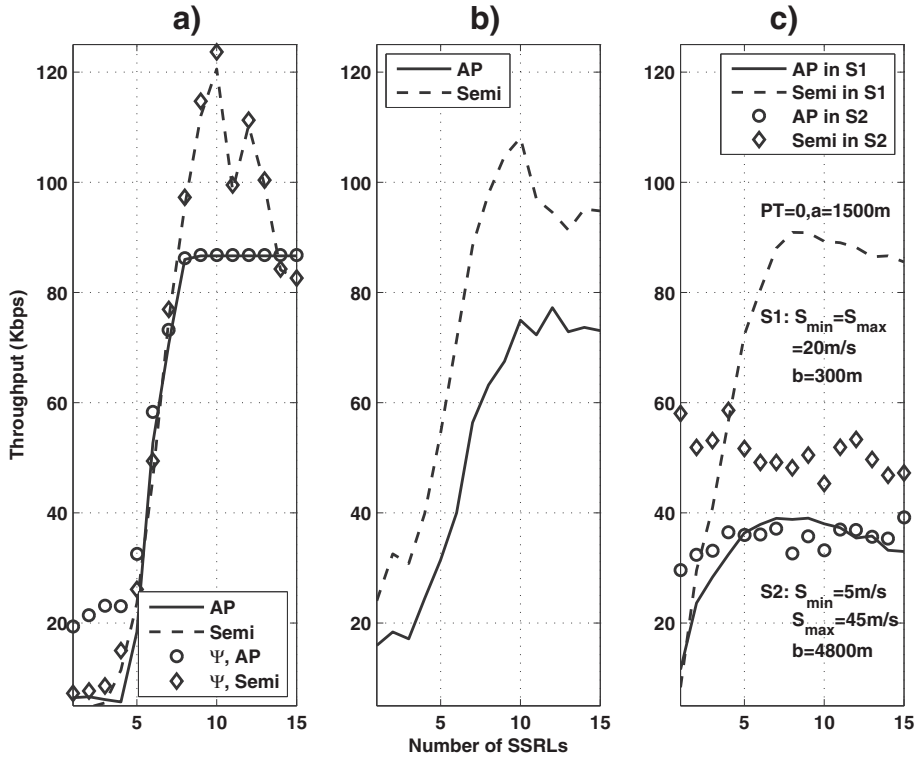

Figure 7 Effect of SSRL on throughput: $T_c = k = 1$. (a) Chain (one flow), **(b)** parallel (one flow), and **(c)** random mobile (two flows).

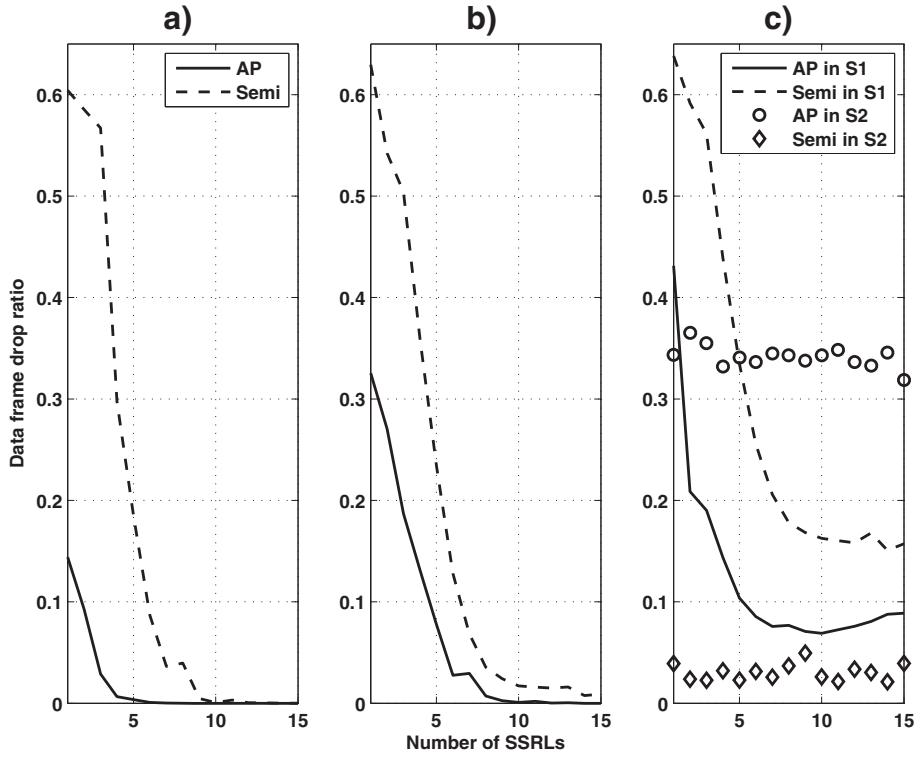

Figure 8 Effect of SSRL on dropping ratio with Figure 7's setting. (a) Chain, **(b)** parallel, and **(c)** random mobile.

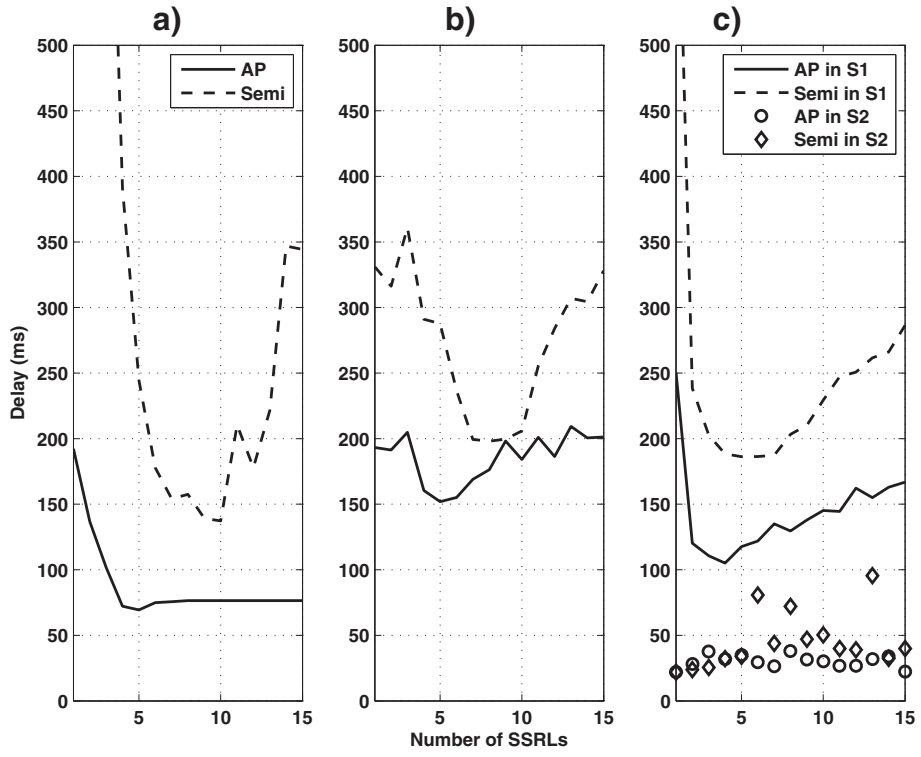

Figure 9 Effect of SSRL on delay with Figure 7's setting. (a) Chain, **(b)** parallel, and **(c)** random mobile.

converges. The same explanation holds for the delay depicted in Figure 9. For both TCP-AP and semi-TCP, the source node will carry out retransmission for every unacknowledged segment. The less MAC frame dropping with larger SSRL, the less retransmissions to be carried out by the source node, resulting in short delay. However, high MAC contention will contribute significantly to long delay. To further verify the above analysis, the effective MAC throughput (Ψ, which is defined as the average number of data bytes successfully transmitted by the MAC layer) of the bottleneck node in a connection is also plotted for the chain topology, as illustrated in Figure 7a. For TCP-AP, when SSRL $<$ 5, the throughput is much lower than Ψ. This is mainly due to TCP's misjudgment of non-congestive frame dropping caused by RTS failure, which leads to unnecessarily shrinking of its congestion window. Since the MAC layer may retransmit several times a MAC frame for one TCP segment, Ψ is higher than the throughput.

Now, we discuss performance differences between stationary and random mobile topologies. The convergence depicted in Figure 7a,b for the stationary topologies becomes slower in the random mobile topology, as illustrated in Figure 7c. This is because in the former two topologies, the available links will not be broken by node mobility, which however takes place in the random mobile topology. If a link is broken, any RTS retrial cannot increase successful transmission (Figure 8) but only MAC contention degree, leading to more MAC contention overhead. Also, due to node mobility, a transmitted frame may not reach its destination or cannot be acknowledged due to sudden link break. Both reduce the throughput in the random mobile topology. Compared to S1, S2 is set with a looser node density and higher mobility so that link break takes place more frequently in S2. As illustrated in Figure 7c, the setting of SSRL $=$ 4 yields the maximum throughput for semi-TCP in S2 rather than SSRL $=$ 9 in S1 because large SSRL does not make sense in increasing throughput in this case. As illustrated in Figure 8c, frequent link break in S2 causes much higher dropping ratio than in S1 for TCP-AP, but this does not happen to semi-TCP. This is because frequent link break causes more droppings for both, but TCP-AP takes a long time for segment retransmission due to its slow reaction as discussed in Section 2.4. However, this is not the case for semi-TCP. Furthermore, due to lower traffic load caused by the misjudgment on congestion status in TCP-AP, the delay here is much shorter than that in S1, as illustrated in Figure 9c.

As discussed in Section 2, semi-TCP outperforms TCP in terms of throughput, which is illustrated in Figure 7. For the parallel topology depicted in Figure 7a, semi-TCP

Table 1 Effect of T_c and k on throughput: two flows, SSRL $= 9, S_{min} = S_{max} = 20$ m/s, PT $= 0$

Topology	Parallel				Random mobile			
T_c	1	2	3	4	1	2	3	4
$k = 1$	89.69	79.69	99.81	88.77	95.93	93.52	88.95	85.25
$k = 2$	117.1	100.0	99.35	87.52	95.54	92.42	87.90	87.11
$k = 3$	90.01	94.23	89.88	85.24	91.67	90.52	88.51	84.27

outperforms TCP-AP for SSRL roughly between 4 and 14 with an improvement up to about 40%, but it then performs poorer because the high MAC contention with semi-TCP makes the hidden terminal problem more severe. This is because that traffic load at the MAC layer with semi-TCP is heavier than with TCP-AP, which also causes longer delays, as illustrated in Figure 9. However, for the parallel topology depicted in Figure 7b, semi-TCP always outperforms TCP-AP with a maximum improvement of about 40%, and the same for the random mobile topology but with a much larger improvement up to 120%, as illustrated in Figure 7b. The reason is that TCP misjudges the congestion status much more frequently in the mobile case, which makes TCP-AP less efficient.

4.3 Settings of parameters T_c and k

As discussed in Section 3.4, some parameters need to be decided for semi-TCP. From the above discussion, SSRL = 9 is good for both TCP-AP and semi-TCP to yield the highest throughput in the most of the scenarios discussed above. Therefore, this setting is adopted here to investigate the setting of congestion threshold (T_c) and the number of frames necessarily to be transmitted to release a congestion (k). The buffer reservation (g and m) here is set to 1 to simplify the discussion since these parameters are less important than T_c and k for congestion control.

As illustrated in Table 1, for the random mobile topology, the setting of $[T_c = 1, k = 1]$ is the best for the throughput and $[T_c = 1, k = 2]$ for the parallel topology. As discussed in Section 3.4, with semi-TCP, traffic is distributed in each node with $T_c > 0$, leading to higher MAC utilization than with TCP-AP. However, with a large T_c, MAC contention degree will also become higher so that nodes need to take more time to contend for channel access, resulting in throughput decrease. Meanwhile,

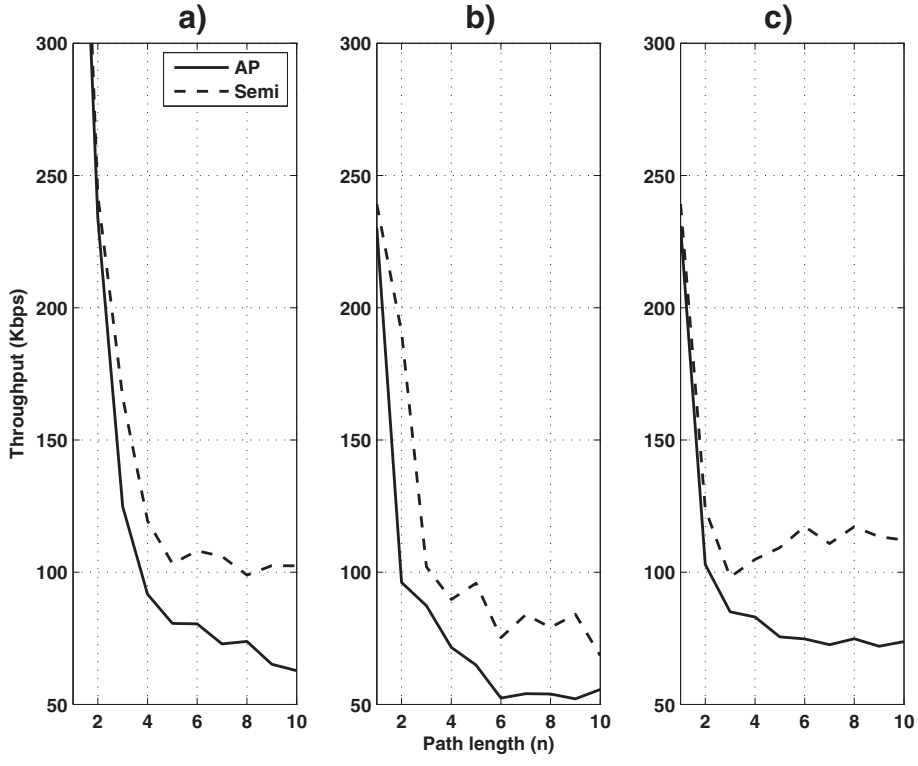

Figure 10 Throughput against path lengths (n): two flows; (a) parallel, (b) spindle, and (c) cross.

higher MAC contention may also make the hidden terminal problem more severe, leading to more collision, which also degrades the performance. Regarding k, the larger k, the longer a node may need to wait before contending for channel access, which causes a channel waste especially if the link with the congested node is broken. Therefore, k should not be set larger, which is demonstrated by the most of the results listed in Table 1.

From the above discussion, the setting of SSRL = 9, $T_c = 1$, and $k = 1$ can yield the best performance in the most of the scenarios. Therefore, this setting along with $g = m = 1$ is used in the following simulation study if not specified otherwise.

4.4 Effect of path lengths

Only with stationary topologies, path length can be preset. Therefore here, the parallel, spindle, and cross topologies are investigated with two flows and path lengths (n) ranging from one to ten hops.

Figure 10 compares the throughput given by TCP-AP and semi-TCP. We can find that semi-TCP outperforms TCP-AP in most cases as n increases. In some cases, the improvement can reach 65% in the cross topology, as illustrated in Figure 10c. For both semi-TCP and TCP-AP, their throughput declines dramatically with n. This is mainly due to as n increases, the hidden terminal problem becomes more severe as well as the channel sharing limit imposed by the MAC-layer contention. As discussed in [34], for the chain topology, this factor will bound per-node throughput up to one-third of the channel capacity. Therefore, the throughput of semi-TCP trends to be stable when $n > 3$. However, the throughput of TCP-AP becomes lower than that of semi-TCP as n increases. A similar phenomenon can also be found for the spindle and cross topologies but with a lower throughput, as illustrated in Figure 10b,c. This is because the MAC contention in these topologies is higher than in the parallel. As illustrated in Figure 6c,d, one segment between the source and destination is shared by two flows in the spindle while one node in the cross topology, and this node becomes the bottleneck of both flows. In the parallel topology, two flows travel along two separate paths.

Regarding the delay plotted in Figure 11, it is not surprising that the delay increases with n since the longer the path, the more travel time a packet has to experience. The delay given by semi-TCP is almost always longer than that by TCP-AP since semi-TCP needs to handle more traffic than TCP-AP.

4.5 Mutual effect of TCP and UDP

Figure 12 shows the ratio of performance indicators against the number of TCP and user datagram protocol (UDP) flows in the random mobile topology. As illustrated in Figure 12a where no UDP flows compete with

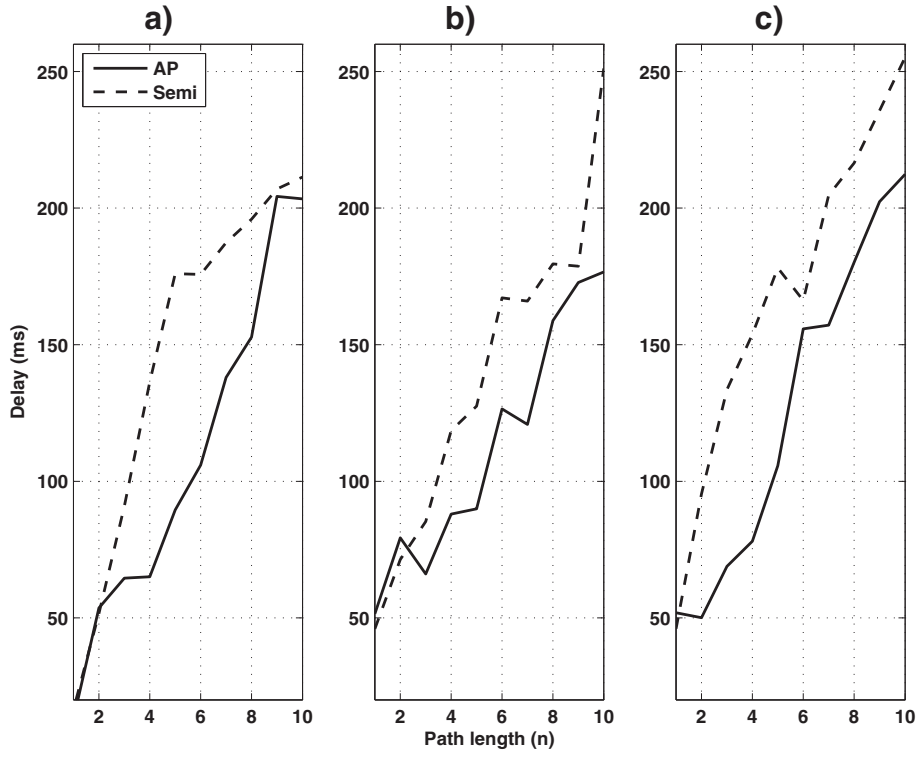

Figure 11 Delay against n: two flows; (a) parallel, (b) spindle, and (c) cross.

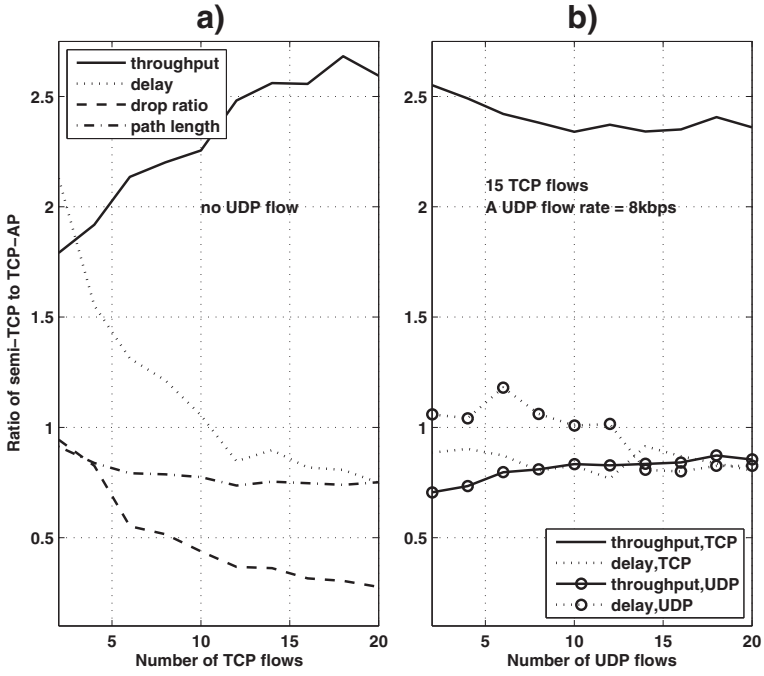

Figure 12 Effect of TCP and UDP: $S_{\min} = 5$ m/s, $S_{\max} = 45$ m/s, PT = 60 s, $a \times b = (1,500\text{ m})^2$. (a) Effect of TCP and **(b)** effect of UDP.

TCP flows, as the number of TCP flows increases, the superiority of semi-TCP over TCP-AP increases too, with a maximum ratio up to about 2.6 for throughput, and similar for delay, path lengths, and dropping ratio. This is because the more TCP flows, the more misjudgment on network congestion status may happen with TCP-AP, which degrades its performance while such misjudgment rarely happens in semi-TCP. When UDP flows join the competition as illustrated in Figure 12b, where the number of TCP flows is fixed at 15 and that of UDP flows is increased, the superiority of semi-TCP over TCP-AP decreases in terms of TCP throughput. The reason is that in this case, more bandwidth which should have been allocated to TCP is taken away by UDP. In semi-TCP, a UDP segment is treated in the same way as that for a TCP segment with the hop-by-hop congestion control.

For UDP, TCP-AP can provide better support than semi-TCP as illustrated in Figure 12b. This is because semi-TCP can allocate more bandwidth to TCP than TCP-AP so that the remaining bandwidth that UDP can use with semi-TCP is less than that with TCP-AP. However, this superiority slightly decreases with the number of UDP flows. Recall that TCP-AP adopts AP to alleviate the impact of the hidden terminal problem by spacing the departure time interval between two consecutive segments. However, UDP segments are not regulated by either congestion window or AP, which weakens the efficiency of AP for TCP segments since UDP segments also cause the hidden terminal problem, leading to more TCP segments to be retransmitted. This reduces the remaining bandwidth that UDP flows can use, and such impact increases with the number of UDP flows.

5 Conclusions

Since the hop-by-hop congestion control is more efficient than the end-to-end control used by TCP, and TCP's congestion window limits further performance improvement offered by the hop-by-hop congestion control, decoupling congestion control from TCP, called semi-TCP, becomes an alternative approach to solve TCP's problems in multi-hop *ad hoc* networks, especially multi-hop MANETs. This paper investigates such a semi-TCP implementation in NS-2 using a hop-by-hop congestion control, which slightly modifies the IEEE 802.11 RTS/CTS protocol. The simulation studies show that semi-TCP can much outperform TCP-AP. Compared with many other proposals only modifying TCP, the semi-TCP may not be the best solution in terms of end-to-end inter-operability with TCP. However, since TCP has so many problems in mobile wireless networks while some characteristics of wireless networks favorable for the hop-by-hop congestion control, it may be worth of exploiting such decoupling approach in order to achieve higher performance for multi-hop *ad hoc* networks.

Note that the RTS/CTS handshake protocol may not always be used in the implementation. In this case, the

information on the congestion status of a node can be piggybacked by other frames such as data frames or MAC ACK frames, which is under investigation by one of our ongoing research programs. Another research activity is to investigate semi-TCP against the effect of other networking functions such as routing protocols and acknowledgment schemes.

Endnote

[a]The source code is available at http://sce.carleton.ca/~ycai/semitcp.tar.gz

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
We thank the reviewers for their detailed reviews and constructive comments, which have helped to improve the quality of this article.

Author details
[1]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada. [2]Marine Internet Laboratory (MILAB), College of Information Engineering, Shanghai Maritime University, Shanghai, China . [3]School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, People's Republic of China.

Received: 12 November 2012 Accepted: 10 May 2013
Published: 3 June 2013

References

1. AA Hanbali, E Altman, P Nain, A survey of TCP over Ad hoc networks. IEEE Commun. Surv. Tutorials. **7**(3), 22–36 (2005)
2. KC Leung, VOK Li, Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges. IEEE Commun. Surv. Tutorials. **8**(4), 64–79 (2006)
3. S Chokhandre, U Shrawankar, in *Proc. Int. Conf. Computer Commun. & Management (CSIT)*. TCP over multi-hop wireless mesh network, vol. 5 (IACSIT Press Singapore, 2011)
4. H Balakrishnan, S Sehan, R Katz, Improving reliable transport and handoff performance in cellular wireless networks. ACM Wireless Netw. (WINET). **1**(4), 469–481 (1995)
5. S Vangala, M Mehta, in *Proc. IEEE Veh. Tech. Conf. (VTC) - Fall*. The TCP SACK-aware snoop protocol for TCP over wireless networks, vol. 4 (Orlando, FL, 6–9 October 2003), pp. 2624–2628
6. FL Sun, VOK Li, SC Liew, in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*. Design of SNACK mechanism for wireless TCP with new snoop, vol. 5 (Atlanta, GA, 21–25 March 2004), pp. 1046–1051
7. CD Lai, KC Leung, VOK Li, in *Proc. IEEE INFOCOM*. Enhancing wireless TCP: a serialized-timer approach (San Diego, CA, 14), pp. 1–5
8. SG Holland, N Vaidya, Analysis of TCP performance on mobile Ad Hoc network on wireless. ACM Wireless Netw. (WINET). **8**(2-3), 275–288 (2002)
9. V Tsaoussidis, H Badr, in *Proc. IEEE Int. Conf. Net. Protocols (ICNP)*. TCP-probing: towards an error control scheme with energy and throughput performance gains (Osaka, 2000), pp. 12–21
10. M Gerla, MY Sanadidi, R Wang, A Zanella, C Casetti, S Mascolo, in *Proc. IEEE Global Tele. Conf. (GLOBOCOM)*. TCP Westwood: congestion window control using bandwidth estimation, vol. 3 (San Antonio TX, 2001), pp. 1698–1702
11. B Sardar, D Saha, Survey of TCP enhancements for last-hop wireless networks. IEEE Commun. Surv. Tutorials. **8**(3), 20–34 (2006)
12. AM Al-Jubari, M Othman, BM Ali, NAWA Hamid, TCP performance in multi-hop wireless ad hoc networks: challenges and solution. EURASIP J. Wireless Commun. Netw. **2011**, 198 (2011)
13. C Luo, FR Yu, H Ji, VCM Leung, Cross-layer design for TCP performance improvement in cognitive radio networks. IEEE. **59**(5), 2485–2495 (2010)
14. B Sadeghi, A Yamdad, A Fujiwara, L Yang, in *Proc. Annual Int. Wireless Internet Conf. (WICON)*. A simple and efficient hop-by-hop congestion control protocol for wireless mesh networks (Boston, TX, 2006)
15. Y Yi, S Shakkottai, Hop-by-hop congestion control over a wireless multi-hop network. ACM/IEEE Trans. Netw. **15**, 133–144 (2007)
16. B Scheuermann, C Locherta, M Mauve, Implicit hop-by-hop congestion control in wireless multihop networks. Ad Hoc Netw. **6**, 260–288 (2008)
17. XY Wang, D Perkins, in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*. Cross-layer hop-by-hop congestion control in mobile *ad hoc* networks (Las Vegas, NV, March 31–April 3 2008), pp. 2456–2461
18. S Jiang, Q Zuo, G Wei, in *Proc. of the 4th ACM workshop on Challenged networks*. Decoupling congestion control from TCP for multi-hop wireless networks: semi-TCP (CHANTS '09, ACM New York, 2009), pp. 27–34
19. K Sundaresan, V Anantharaman, HY Hsieh, R Sivakumar, ATP: A reliable transport protocol for Ad Hoc networks. IEEE Trans. Mobile Comput. **4**(6), 588–603 (2005)
20. SM ElRakabawy, C Lindemann, A practical adaptive pacing scheme for TCP in multihop wireless networks. IEEE/ACM Trans. Netw. **19**(4), 975–988 (2011)
21. S Floyd, K Fall, Promoting the use of end-to-end congestion control. ACM/IEEE Trans. Netw. **7**(4), 458–472 (1999)
22. S Xu, T Saadawi, Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? IEEE Commun. Mag. **39**(4), 130–137 (2001)
23. E Altman, T Jimenez, in *Proc. IEEE Int. Conf. on Personal Wireless Comm.* Novel delayed ACK techniques for improving TCP performance in multihop wireless networks (Venice, 23–25 September 2003), pp. 237–242
24. AK Singh, K Kankipati, in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*. DTCP-ADA: TCP with adaptive delayed acknowledgement for mobile *ad hoc* networks, vol. 3 (Atlanta, GA, 2004), pp. 1685–1690
25. IEEE Std 802.11, Medium Access Control (MAC) sub layer and 3 Physical Layer Specifications, IEEE, (1997)
26. HQ Zhai, JF Wang, YG Fang, in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*. Distributed packet scheduling for multihop flows in *ad hoc* networks, vol. 2 (Atlanta, GA, 21–25 March 2004), pp. 1081–1086
27. K Chen, K Nahrstedt, N Vaidya, in *Proc. IEEE Wireless Commun. & Networking Conf. (WCNC)*. The utility of explicit rate-based flow control in mobile ad hoc networks, vol. 3 (Atlanta, GA, 21), pp. 1921–1926
28. W Kiess, M Mauve, A survey on real-world implementations of mobile ad-hoc networks. Ad Hoc Netw. **5**(3), 324–339 (2007)
29. CE Perkins, EM Belding-Royer, SR Das, *Ad hoc* on-demand distance vector (AODV) routing. IETF RFC3561 (2003)
30. M Gerla, K Tang, R Bagrodia, in *Proc. IEEE WS. Mobile Computing Systems & App. (WMCSA)*. TCP performance in wireless multi-hop networks (New Orleans, LA, 25–26 Feb 1999), pp. 41–50
31. Z Fu, P Zerfos, H Luo, SW Lu, LX Zhang, M Gerla, The impact of multihop wireless channel on TCP throughput and loss. IEEE Trans. Mobile Comput. **4**(2), 209–221 (2005)
32. SM ElRakabawy, K Alexander, L Christoph, in *Proc. ACM Int. Sym. Mobile Ad Hoc Networking and Computing (MobiHoc)*. TCP with adaptive pacing for multihop wireless networks (New York, NY, 2005), pp. 288–299
33. S Floyd, T Henderson, The New-Reno modification to TCP's fast recovery algorithm. IETF RFC 2582 (1999)
34. D Berger, ZQ Ye, P Sinha, S Krishnamurthy, M Faloutsos, SK Tripathi, in *Proc. IEEE Int. Conf. Mobile Ad-hoc & Sensor Systems*. TCP-friendly medium access control for *ad-hoc* wireless networks: alleviating self-contention (Port Lauderdale, FL, 2004), pp. 214–223

doi:10.1186/1687-1499-2013-149
Cite this article as: Cai *et al.*: **Decoupling congestion control from TCP (semi-TCP) for multi-hop wireless networks.** *EURASIP Journal on Wireless Communications and Networking* 2013 **2013**:149.