EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Rate control for semi-TCP in multihop wireless networks

Weiqi Chen[1], Hua Yu[1], Quansheng Guan[1*], Yide Wang[2] and Shengming Jiang[3]

Abstract
This paper studies the rate control algorithm to improve the performance of semi-TCP, which adopts a buffer-based hop-by-hop congestion control mechanism. By analyzing the impact of the buffer-based sending strategy on the medium contention and interference, we find that this sending strategy is too aggressive to increase the throughput, which leads to severe contention and interference in the channel. Extensive simulations are then carried out and reveal the significance of the channel status of the first hop in rate control. Based on the findings in the simulations, we propose a binary search-based rate control to improve semi-TCP using only local information. The proposed rate control relies on the media status in the first hop to adjust the source rate to approach the peak throughput, without the assistance of end-to-end feedback. Simulation results show the effectiveness of the proposed algorithm in improving the end-to-end throughput and delay.

Keywords: Rate control, Semi-TCP, Congestion control, Multi-hop wireless networks

1 Introduction

Transmission control protocol (TCP) has shown severe performance degradation in wireless multi-hop networks [1, 2]. The main reason is that the congestion in the network is frequently misjudged by TCP, since a packet loss is usually considered as a signal of the network congestion. However, in addition to congestion, there are many other reasons that lead to packet losses in wireless networks, to name some, the high bit error rate in the radio channel, the collisions in the open shared medium, the route unavailability due to terminal mobility, and etc. The frequent packet losses suppress the increase of the sending window in TCP, thus degrading the throughput.

Many existing works have been proposed to address the congestion misjudgement issue in wireless TCP. Packet losses due to route breaks in ad hoc wireless networks have been noticed in [3–8]. TCP-F in [3] is proposed to feed back the route break information to the source. On receiving the feedback, the source then freezes the sending window and suspends sending packets. The contention in the shared medium is another reason that causes packet losses. It is shown in [9] that link layer drops due to medium contention are the first sign for network overload. Link-layer random early dropping (LRED) and adaptive pacing are then proposed in [9] to stabilize the TCP window size around the best size, where LRED tunes up the link-layer dropping probability according to channel conditions and adaptive pacing extends the range of link-layer contention coordination. Rather than back-offing an additional packet transmission time in [9], the cross-layer TCP pacing in [10] adjusts the contention window to improve throughput. Backward end-to-end acknowledgement (ACK) transmissions also involves in the contention with forward data transmissions for the network resources. Compressing ACKs is another means to improve TCP throughput. The reference [11] proposed to minimize the number of ACK packets by delaying ACK packets according to the channel condition, thus mitigating channel contention and improve TCP's throughput.

The above TCP schemes employ the ideas of improving the judgement of network congestion and retain the end-to-end semantics of congestion control. The end-to-end congestion control usually needs several round-trip times (RTTs) to detect the congestion in the network, which may respond slowly to the congestion. Particularly for the dynamically changing wireless network, the delayed response may become invalid. Semi-TCP [12, 13] judges the network congestion via the buffer occupancy at each

*Correspondence: eeqshguan@scut.edu.cn
[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, People's Republic of China
Full list of author information is available at the end of the article

node, in the sense that a congestion means the run-out of the buffer resources. A hop-by-hop congestion control is then triggered promptly by the congested node. The source just keeps sending packets as long as there exist available queueing buffers. Semi-TCP can judge accurately the congestion status, response promptly to the congestion and the release of the congestion, as well as provide the mobility support for wireless networks. Although the idea of semi-TCP is interesting, the study in this paper shows that the sending rate in semi-TCP is relatively aggressive, thus makes the network overloaded and leads to frequent congestion. In this sense, semi-TCP demands an efficient control mechanism for the semi-TCP source to improve its performance.

Rate control has been considered as a useful tool for congestion control [14–16]. The reference [17] considers a rate-based end-to-end flow/congestion control with the objective to maximize the aggregate source utility, subject to link capacity in wired networks. The reference [18] proposes a hop-by-hop rate-based congestion control. The optimization flow control is extended to wireless multi-hop networks, and a hop-by-hop scheme is developed by allowing simultaneous transmissions which do not share a common node in [19, 20]. The study in [21] shows that the wireless interference usually covers four hops. Thus, TCP with adaptive pacing (TCP-AP) is proposed to adapt the sender's transmission rate close to the value estimated by the four-hop propagation delay. Active queue management is another technique to control the source rate. Active queue management provides incipient congestion notification to the source by dropping/marking packets, which can decrease the size of the source's sending window [22, 23].

The simulation results in our previous work [12] has shown that the minimum buffer reservation in semi-TCP reaches its maximum throughput in the simulated scenario. The study in this paper also shows that the throughput of semi-TCP can be further improved by controlling the source rate (in segments per second or bytes per second), taking into account not only the available buffer but also the medium contention and interference in the downstream nodes. To this end, this paper studies the source rate control for semi-TCP to further improve its performance.

The contributions of this paper are summarized as follows:

- *Analysis of performance limitation:* Our study shows that the aggressive sending rate of semi-TCP results in severe retransmissions and drops in links, particularly in the first hop. This finding suggests a localized rate control algorithm to improve semi-TCP.
- *Binary search-based rate control:* By exploiting the relationship among the per packet sending time in the MAC layer, average attempts per ready-to-send (RTS) frame and the end-to-end throughput, we propose a binary search-based rate control algorithm. The proposed rate control relies on the media status in the first hop to adjust the source rate to approach its peak throughput without the assistance of end-to-end feedback. The simulations show the improvement in end-to-end throughput and delay comparing to the original semi-TCP.

The rest of this paper is organized as follows. Section 2 presents an overview of semi-TCP. Section 3 analyzes the performance limitation in semi-TCP by simulations and obtains some insights for performance improvement. A binary search-based rate control algorithm is then proposed in Section 4 to improve semi-TCP. Section 5 verifies the improvement via simulations. Finally, Section 6 concludes the paper.

2 Overview of semi-TCP

Semi-TCP removes the congestion control functionality from the transport layer to the MAC layer, to implement a hop-by-hop congestion control.

- *Congestion judgement:* Since the congestion control has been decoupled from the transport layer and moved down to the link layer, semi-TCP allows each node to judge the congestion based on its buffer occupancy.
- *Congestion avoidance:* When a node is congested, which is indicated by the running out of the queue buffer, the node will inform its direct upstream node to stop sending packets. Without outputting packets, the upstream node will run out its queue buffer quickly, becoming congested. In this manner, the congestion status will be propagated back to the source along the path. Then, the source's sending is suspended until its buffer becomes available again. Semi-TCP takes actions at the node where the congestion begins and releases the congestion immediately along the inverse direction of the traffic flow.
- *Buffer-based sending at the source:* Semi-TCP reserves some buffers for the source in the queue. When the reserved buffer has space for more packets, it initiates a request to pull down the packets from the transport layer. Semi-TCP does not rely on ACKs to adjust the sending rate. Thus the congestion window in TCP becomes unnecessary, and is cancelled in semi-TCP. It adjusts its sending rate directly based on the congestion status, instead of the judgement from the arrivals of ACKs.

Comparing to the traditional TCP, semi-TCP has the following advantages.

- The buffer-based congestion judgement is more accurate than the inference from packet losses.
- Congestion avoidance can be carried out exactly at the congested node, and the sending rate can be restored immediately when the congestion is released.
- The hop-by-hop congestion control provides more mobility supports for wireless networks.

It is worth noting that the congestion control and reliability control are decoupled in semi-TCP. Instead of relying on the feedback to implement both congestion control and reliability control in traditional TCPs, semi-TCP judges congestion based on the buffer occupation in each hop, and each hop can response to release congestion once detected. Thus, the acknowledgement has no impact on congestion control in semi-TCP.

3 Analysis of performance limitation in semi-TCP

The buffer-based sending prevents from packet drops thanks to the overflow in the queue. However, the buffer-based sending in semi-TCP is still too aggressive to overload the network quickly, leading to severe contention among adjacent nodes in the network, who compete the access to the shared medium. In addition to the severe medium contention, the conflicts of simultaneous transmissions happen frequently in this case. The results in [12] have shown that semi-TCP performs the best when the reserved buffer for the source is set to only one packet. From the perspective of medium contention, the node will keep competing the access to the medium as long as there is one packet in the buffer. In this sense, controlling the source's rate by the buffer does not help significantly in improving semi-TCP's performance.

We use simulations to verify the above analysis in this section.

3.1 Simulation settings

In the simulations, we adopt IEEE 802.11 and AODV as the MAC and routing protocols respectively. The bandwidth is 2 Mbps. The source sends its FTP traffic to the destination with a packet size of 512 B. The transmission range for each node is 250 m, and the interference range is 500 m.

We set up a chain network topology with 15 nodes, and the distance between adjacent nodes is 200 m, as shown in Fig. 1. FTP applications are attached to semi-TCP. In this scenario, the FTP traffic goes through 14 hops, from node 1 to node 15. The reserved buffer for the semi-TCP source has a length of one packet. We investigate the medium contention at each node by studying the average number of attempts per RTS and data packets, the transmissions of RTS-congestion (RTSC), the RTS dropping ratio after seven attempts[1], RTSC ratio[2], the average sending time in the MAC layer, and the average queue length.

3.2 Performance limitation of semi-TCP

As shown in Fig. 2, the sending attempts per RTS decrease along the downstream of the path. The attempts per data packet also decline slightly along the path, though they are much smaller than the RTS attempts. The RTS attempts in the first hop are extremely higher than those in the downstream hops, and are decreased into almost one in the final hops. This indicates that, the medium contention in the upstream hops, particularly in the first hop, is much severe than that in the downstream hops. The reason is that semi-TCP propagates the congestion status in a backward direction using the hop-by-hop RTSC/CTSC exchanges by each intermediate nodes. Due to the severe medium contention, RTS/clear-to-send (CTS) handshakes fail frequently. It is not surprising to see in Fig. 3 that the average sending time for a data packet is longer in the source than that in the downstream nodes, since the time is mainly consumed by the repeated RTSs. The worse is that, the forwarding path will be regarded as broken when the RTS is dropped (see Fig. 4), which triggers a re-routing process. The frequent re-routings result in a waste of wireless bandwidth and degrade dramatically the throughput of semi-TCP.

It is revealed in [21] that the wireless transmission has an interference range of four-hops. Thus, transmission conflicts in the intermediate nodes happen more frequently than those in the end nodes in the chain topology. The average queue length in Fig. 3 shows that, being in a situation with more severe interference, the second node in the chain topology has less chances to access the channel, and its buffer is quick flushed by incoming

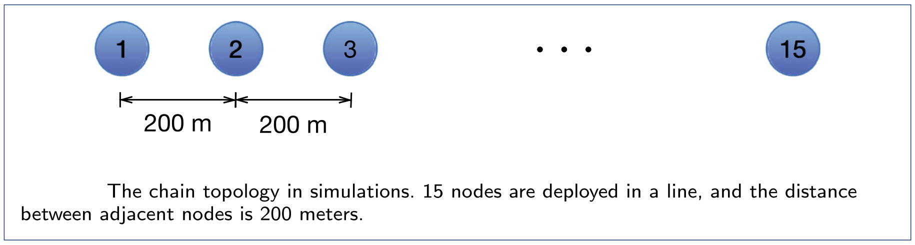
1
2
3
. . .
15
200 m
200 m
The chain topology in simulations. 15 nodes are deployed in a line, and the distance between adjacent nodes is 200 meters.

Fig. 1 The chain topology in simulations. Fifteen nodes are deployed in a line, and the distance between adjacent nodes is 200 m

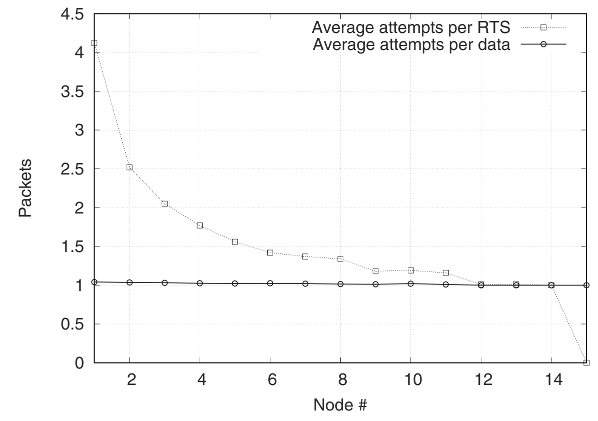

Fig. 2 Average number of attempts per RTS and data packets

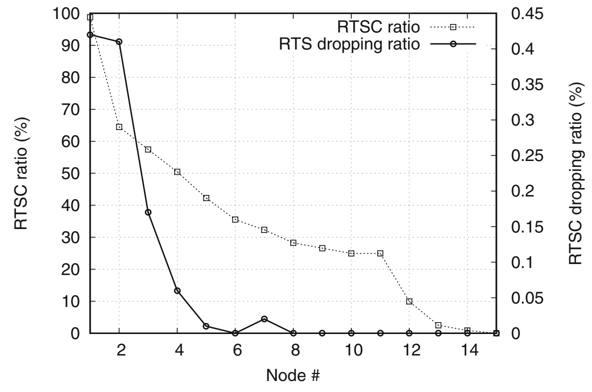

Fig. 4 RTS dropping ratio after seven attempts and RTSC sending ratio

packets from the source node. In this sense, the failures and drops of RTSs in the first hop are mainly due to the aggressively sending in the semi-TCP source. Reserving the buffer of one packet in semi-TCP still results in a over-provisioned sending rate for the network. The aggressive sending makes the upstream nodes congested, leading to low transmission rates in the downstream links. This is confirmed by the curve of RTSC transmissions in Fig. 4, which shows that the source node sends the most RTSCs to broadcast its congestion state in its buffer[3].

The above simulation results reveal some insights into semi-TCP.

- *Inefficiency of the buffer-based sending:* The buffer-based sending in semi-TCP aggravates the medium contention in wireless networks, resulting in a large amount of drops and retransmissions, which wastes the network resources. In addition, the buffer-based rate control also triggers the transmissions of RTSCs frequently to avoid the buffer overflow and release congestion at each node. In this sense, semi-TCP is inefficient in using the network resources. It still has much room for improvement.
- *Significance of the source:* The upstream hops, especially the first hop, have much severer medium contention and congestion than the downstream hops. It is significant for the source to release the contention and congestion by controlling its sending rate. Therefore, rate adjustment based on the local (i.e., the first hop) information is important for the improvement.
- *Demand of new rate control:* Note that we have set the minimum buffer for the semi-TCP source. It is not possible to improve the performance of semi-TCP by adjusting the buffer reservation. Thus, the rate control in semi-TCP has to consider not only the buffer occupancy but also the medium contention.

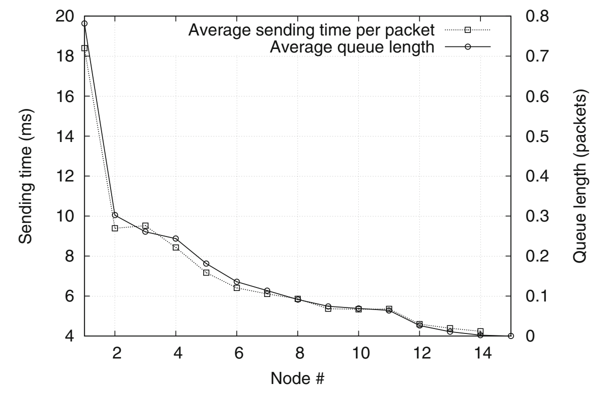

Fig. 3 Average sending time per packet in MAC and the average queue length

The above insights have suggested a localized rate control algorithm, which will be discussed next.

4 Rate control to improve performance of semi-TCP

The studies in Section 3 have shown that the sending rate in semi-TCP is too aggressive to cause severe medium contention. To design an optimal rate control, we have to identify an indicator that is closely related to the end-to-end throughput, and have a new mechanism to adjust the source's sending rate.

4.1 Indicators for throughput performance

We have shown in the previous section that the medium condition is much severer in the first hop, and the medium contention and interference are the main reasons that degrade the end-to-end throughput. The importance of the first hop in end-to-end control facilitates the design of rate control in the semi-TCP source. We next focus on the first hop to design a rate control algorithm.

There are many metrics indicating the medium condition. To avoid buffer overflow, the source's rate should not be larger than the sending rate in the MAC, which is the reciprocal of the sending time in MAC. Generally, a successful data transmission experiences the medium access, the exchanges of RTS, CTS, data, and ACK, as well as their retransmissions. Particularly in severe contention, the average attempts of an RTS dominate the sending time in MAC. Therefore, we consider the average attempts per RTS (APR) and the MAC sending time (MST) in adjusting the source' rate to improve the end-to-end throughput.

We use Figs. 5 and 6 in the chain topology with 14 hops to reveal how the end-to-end throughput is affected by the source rate. We can observe that the end-to-end throughput is almost increased linearly with the source rate when the source rate is less than 140 kbps. The MST is small in this case. Thus, the incoming packets from the semi-TCP source can be sent out by MAC immediately. The throughput deceases dramatically when the source rate continues to increase. We can observe at the same time an increase in the average APR and an increase in the MST, which means that the medium has been in severe contention and interference.

We also observe that the APR and the MST switch between a low and a high levels. The end-to-end throughput is low in the high-level APR and the high-level MST, and vice versa. Thus, we have the following observation from the simulation results in this section.

- The best source rate that maximizes the throughput exists at around the switching points of APR and MST.

The above observation has given rise to an algorithm to find the optimal source rate that will maximize the end-to-end throughput, which will be elaborated in Section 4.3.

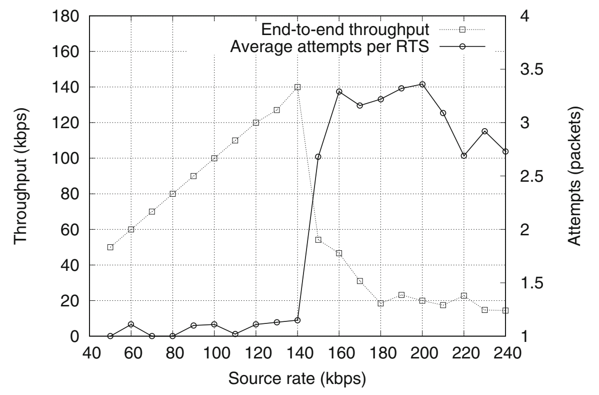

Fig. 5 End-to-end throughput and the average attempts per RTS at the source

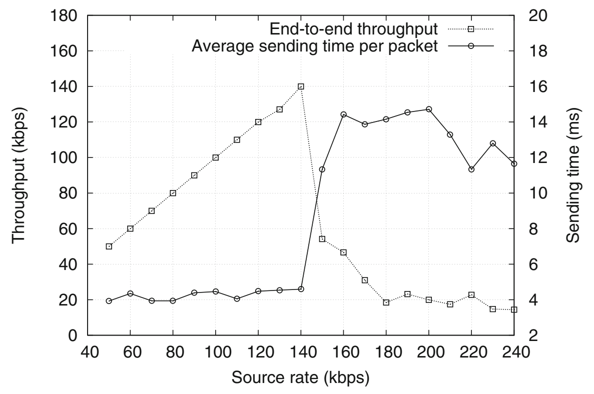

Fig. 6 End-to-end throughput and the sending time per data packet in MAC at the source

4.2 Timer-based adjustment for source rate

The traditional TCPs rely on either the end-to-end feedback from the destination or the notifications from the intermediate nodes to adjust the source rate, which retains the end-to-end control semantics. Instead, semi-TCP uses a buffer-based congestion control in a hop-by-hop manner to achieve accurate congestion detection and prompt response for congestion. However, the results in Section 3 has shown that the semi-TCP using even the minimum buffer reservation also leads to severe medium contention and interference. That means the buffer-based sending rate in semi-TCP is too aggressive to cause congestion in the network. Semi-TCP needs a new mechanism to adjust the source rate, rather than the available buffer.

We use a sending timer (Stimer) to control the source's sending rate. The duration of the timer is the interval for consecutive packets. The source rate is determined by reciprocal of the timer duration and the packet size as follow:

$$\text{Source rate} = \frac{\text{Packet size}}{\text{Duration of Stimer}}. \quad (1)$$

From the above equation, it is known that a timer with a long duration reaches a low source rate and vice versa. We can half the timer duration to double the source rate, which will be adopted in the slow start in Algorithm 1 in the next section.

4.3 Binary search-based rate control algorithm

Our objective is to find out the source rate that maximizes the end-to-end throughput. The intuitive method to achieve this objective is to establish a relationship between the source rate and the end-to-end throughput, and then adjust the source rate according to the changes of throughput. As shown in Fig. 6, there is a turning point at around the source rate of 140 kbps for the slope of the end-to-end throughput. The source rate should be set to the turning point to maximize the throughput. However,

Algorithm 1 Binary search-based rate control algorithm

```
Step 1: Aggressive sending to detect the low APR (LAPR) and the high APR (HAPR)
1: Send packets when the buffer is available
2: Set LAPR = 1, HAPR=$\hat{A}(n)$
3: Set the high MST (HMST)=$\hat{T}_s(n)$
Step 2: Slow start to determine the search range of the sending rates
1: Set Stimer=HMST, the high Stimer (HStimer)=HMST (corresponding to the low source rate)
2: while $\hat{A}(n) < \frac{\text{HAPR+ LAPR}}{2}$ do
3:     Stimer=$\frac{1}{2}\times$Stimer (i.e., double the source rate)
4: end while
5: Set the low Stimer (LStime)=Stimer (corresponding to the high source rate)
Step 3: Binary search for the rate to improve the throughput
1: while the difference of the high-level and the low-level source rates is larger than 1 kbps do
2:     if $\hat{A}(n) > 0.5\times$(HAPR+LAPR) then
3:         LStimer=Stimer
4:         Stimer=$\frac{Stimer+HStimer}{2}$
5:     end if
6:     if $\hat{A}(n) < 0.3\times$(HAPR+LAPR) then
7:         HStimer=Stimer
8:         Stimer=$\frac{Stimer+LStimer}{2}$
9:     end if
10:    if $\hat{A}(n) > 1.2 \times \hat{A}(n-1)$ OR $\hat{A}(n) < 0.8 \times \hat{A}(n-1)$ then
11:        GOTO Step 1
12:    end if
13: end while
Step 4: Dynamic adjustment
1: if $\hat{A}(n) > 1.2 \times \hat{A}(n-1)$ OR $\hat{A}(n) < 0.8 \times \hat{A}(n-1)$ then
2:     GOTO Step 1
3: end if
```

measuring the throughput in the source has to rely on the end-to-end feedback from the destination. It takes at least several RTTs to measure the throughput, which is inaccurate and obsolete in dynamically changing wireless networks.

We focus on the local information that is available at the source to adjust the source's sending rate, based on the observation in Section 4.1. It has been shown in Figs. 5 and 6 that the metrics of both APR and MST switch between two values. When the throughput is lower than its maximum level, both APR and MST keep in a low level. Once the throughput reaches its peak and continues to grow, both APR and MST suddenly leap to a high level. The switching of APR or MST indicates the saturation of the throughput. Fortunately, it is easy to obtain APR and MST at the source. We instead try to find out the source rate that makes the switching of APR or MST.

To find the switching of APR and MST, the first step is to identify the low and high levels of APR or MST. It is difficult to identify the low-level MST. However, it is easy to know that the low-level APR is around one, in the case that no retransmissions for RTS is needed. We only need to obtain the high-level APR. Therefore, we focus on studying the metric of APR. The difficulty still exists in obtaining the high-level APR, in which case the medium is in severe contention and interference. Our study in Section 3 has shown that the aggressive source rate controlled by the available buffer leads to a high-level APR and a high-level MST. Thus, we use the aggressive source rate in the original semi-TCP to obtain the high-level APR and the high-level MST.

The source rate can change theoretically from 0 to infinity. To accelerate the search for the optimal source rate, we next need to narrow the range of the source rates that includes the rate reaching the maximum throughput, and determine the bounds for the range. From Figs. 5 and 6, it is known that the optimal source rate exists in a narrow range that makes the sudden leap of APR and MST. The sending rate in the MAC layer can be calculated by the reciprocal of the MST and the packet size, i.e.,

$$\text{Sending rate} = \frac{\text{Packet size}}{\text{MST}}. \tag{2}$$

When the MST reaches to a high level, the throughput will decline to a low level. If the source rate is much higher than the throughput in this case, the packets will occupy the queue buffer, and it will result in overflow. The lower bound for the range that includes the optimal source rate should be the low throughput in the case of a high-level MST. The lower bound can be calculated by the reciprocal of the high-level MST, since the end-to-end throughput is not available at the source. In the meanwhile, the source rate that results in a high-level MST should be larger than the optimal source rate, and the source rate in this case can be the upper bound for the rate range. Take Fig. 6 for an example. When we have a source rate at 180 kbps, which is higher than the optimal source rate at around 140 kbps, the MST reaches to a high level at around 14 ms and the throughput reaches to a low level at 20 kbps. Thus, the optimal source rate should be in the range of 20–180 kbps. In practice, we adopt the slow start mechanism in TCP to find out the lower and upper bounds. The slow start increases the source rate exponentially, until the high-level APR or the high-level MST is achieved, to determine the upper-bound source rate.

We now try to find out the optimal source rate between the lower bound and the upper bound. We have known that the optimal source rate makes the switching between the low-level APR and the high-level APR. Considering the sudden decline of throughput with the increase of the source rate (see Fig. 5), we infer that the optimal source rate should lead to an APR in the range of $[0.3, 0.5] \times (\mathrm{HAPR} + \mathrm{LAPR})$.

Note that the APR increases monotonically with the source rate. Thus, we can easily apply the binary search to find the optimal source rate. The initial range for the binary search is set to the range between the upper bound and the lower bound for the source rate as stated above. The initial source rate for the binary search is set to the upper-bound rate. The binary search will narrow the range including the optimal source rate iteratively. When the current source rate obtains an APR that is out of the range of the $[0.3, 0.5] \times (\mathrm{HAPR} + \mathrm{LAPR})$, the search range is halved and the source rate for the next iteration is set to the boundary of the new search range. Otherwise, the search will stop. Finally, the algorithm will achieve an APR in the range of the $[0.3, 0.5] \times (\mathrm{HAPR} + \mathrm{LAPR})$. Instead of finding the exactly optimal source rate, we find the near-optimal source rate to stabilize the source rate around the best operating point, which will accelerate the convergence of the binary search and make the algorithm feasible for the dynamic environment.

The high-level APR may change with the medium conditions after the binary search terminates, due to the node mobility and the flow activities. For example, the incoming of a new flow will increase the high-level APR, while the termination of an existing flow will decrease the high-level APR. In this case, the binary search has to be carried out again to find out the new optimal source rate. When the change of APR is detected, the algorithm has to detect the new high-level APR and the upper-bound source rate again. To avoid the frequent detections, the fluctuation of the measured APR should be large enough to trigger the detection. We re-launch the detection only when the measured APR is higher than 1.2 times the previous APR or lower than 0.8 times the previous APR.

The estimations for MST and APR is important for the binary search. The aggregate past values at each node are used to estimate the current MST and APR of a packet. We use a simple moving average in estimating MST and APR. Let $T_s(n)$ denote the measurement of the MST at time n. Then, the estimated MST can be

$$\hat{T}_s(n) = \sum_{n'=n-n_0+1}^{n} a_s\left(n', n\right) T_s\left(n'\right), \tag{3}$$

where

$$\sum_{n'=n-n_0+1}^{n} a_s\left(n', n\right) = 1, \forall n. \tag{4}$$

The estimated MST $\hat{T}_s(n)$ is calculated by taking average over only the latest n_0 sending time measurements, i.e., $a_s\left(n', n\right) = 1/n_0$ for $n' \in \{n - n_0 + 1, \cdots, n\}$. Then, it can be rewritten as

$$\begin{aligned} \hat{T}_s(n) &= \frac{1}{n_0} \sum_{n'=n-n_0+1}^{n} T_s\left(n'\right) \\ &= \hat{T}_s(n-1) + \frac{T_s(n) - T_s(n-n_0)}{n_0}. \end{aligned} \tag{5}$$

Although the latest n_0 MST samples are used in estimating the current MST, it is shown in Eq. (5) that only the current sample and the latest $n - n_0$ sample are required

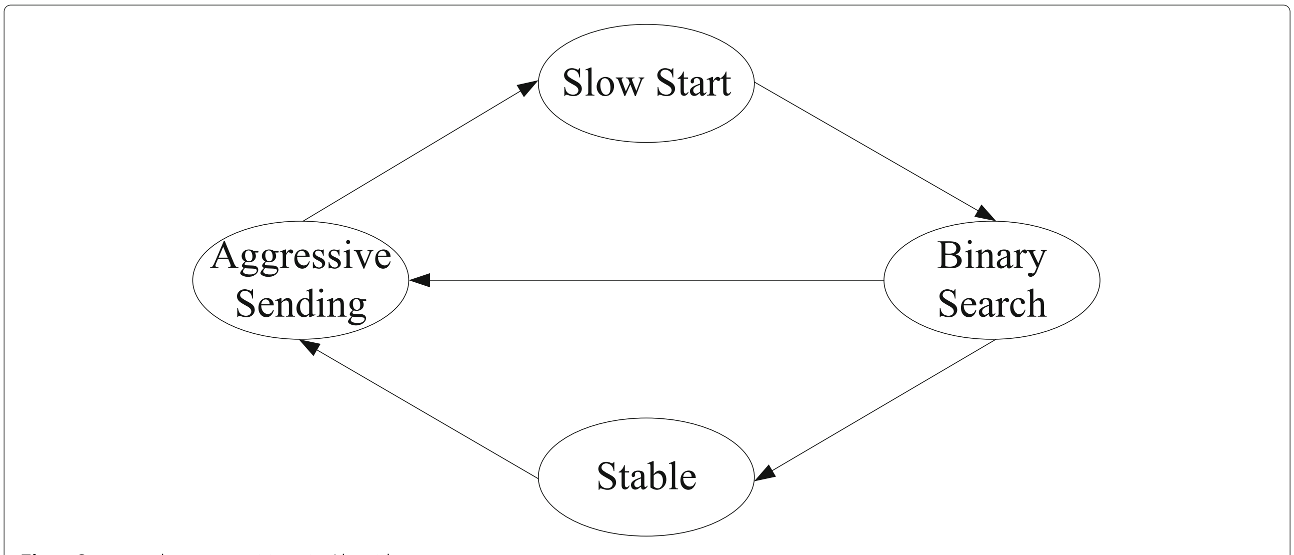

Fig. 7 States and state transitions in Algorithm 1

Table 1 Iterations of Algorithm 1: an example

Iteration	Source rate (kbps)	APR
0	240	2.7 (> 2.15)
1	120	1.2 (< 1.29)
2	180	3.3 (> 2.15)
3	150	3.3 (> 2.15)
4	135	1.2 (< 1.29)
5	143	1.7 (in [1.29, 2.15])

to be kept in the buffer for calculation. The calculation in Eq. (5) saves buffer space and accelerate the MST estimation.

Similarly, the estimate of the current APR, denoted by $\hat{A}(n)$ is obtained by

$$\hat{A}(n) = \hat{A}(n-1) + \frac{A(n) - A(n-n_0)}{n_0}. \quad (6)$$

The above binary search based rate control algorithm is summarized in Algorithm 1. It consists of four operations as follows:

- *Aggressive sending*: The buffer-based source rate in the original semi-TCP has been shown in Section 3 to be to aggressive to result in severe contention and interference in the medium. We exploit the original semi-TCP to artificially construct the severe medium contention to obtain the high-level APR and the high-level MST.
- *Slow start*: When the high-level APR is achieved, the medium is evidently in severe contention. The source rate in this case should be the upper-bound source rate for the binary search. After identifying the high-level APR by aggressive sending, we increase the source rate by slow start, where the source rate is adjusted by the sending timer, to obtain the upper-bound source rate with the high-level APR. Note that the lower-bound source rate can be calculated by directly by the high-level MST using Eq. (2).
- *Binary search*: The binary search is used to find out the optimal source rate which makes the switching of APR.
- *Dynamic adjustment*: The medium condition may change dynamically with the number of users in the network, user mobility, and etc. The source rate is re-adjusted when the change of medium condition is detected.

The above operations makes the algorithm transit in four states as shown in Fig. 7. The aggressive sending first detects the high-level APR and the high-level MST. The slow start identifies the bounds of the source rate. Then, the binary search is used to find the optimal rate. The source will keep stably sending at this rate, unless the change of the medium condition is detected.

4.4 Discussion on convergence

Due to the sharp decline of the end-to-end throughput with the source rate, the binary search will converge in only serval iterations. We use Fig. 5 to exemplify the fast convergence of Algorithm 1. Suppose the aggressive sending in Algorithm 1 has obtained that HAPR = 3.3 and LAPR = 1, and the slow start has showed that the optimal source rate is in the range of 0 and 240 kbps[4]. The output of the slow start in step 2 is 240 kbps, which is also the initial source rate for the binary search in step 3. Thus,

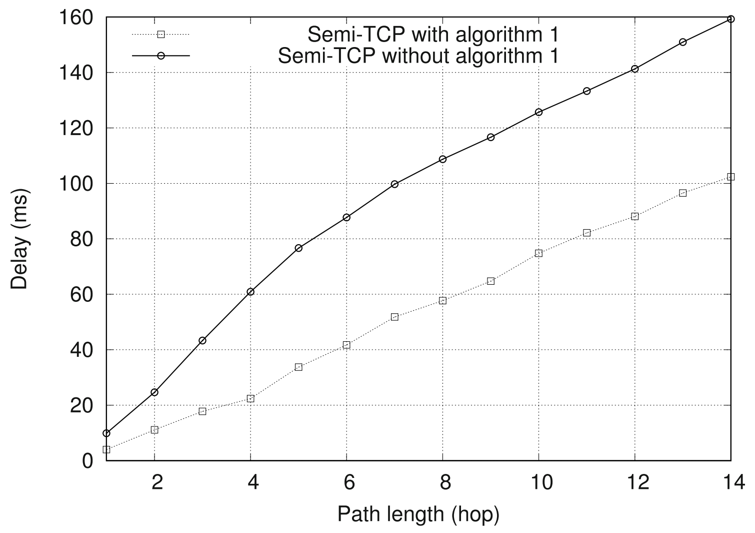

Fig. 8 End-to-end throughput in the chain topology

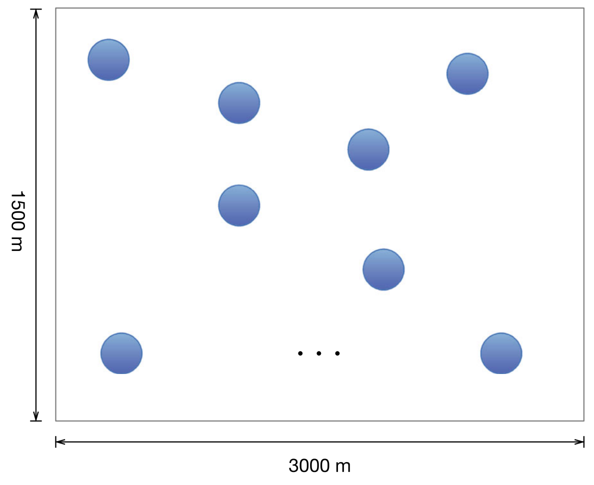

Fig. 9 End-to-end delay in the chain topology

the binary search will stop when APR is in the range of 1.29–2.15 in the fifth iteration. The iterations are shown in Table 1.

Since the proposed Algorithm 1 exploits only the local information available at the source, the time for convergence experiences only several frame transmissions, which consumes around few milliseconds.

5 Simulations and discussions

We study the performance of our proposed rate control algorithm using NS2.

5.1 Simulation settings

We use the same setting in Section 3.1 again. We evaluate the end-to-end throughput and delay of both our proposed algorithm and the original semi-TCP in the chain topology and the mobile random topology. The chain topology is the same as Fig. 1. In the mobile random topology, 48 mobile nodes, following the random waypoint mobility model are deployed in an area of 1500 × 3000 m^2 in the mobile random topology.

5.2 Results and discussions

We first show the results in the chain topology in Figs. 8, 9, and 10. An FTP flow is attached to the source node at one end of the chain topology, and the destination node locates at the other end of the chain topology. The semi-TCP with our proposed algorithm increases the end-to-end throughput and decreases the end-to-end delay, comparing to the original semi-TCP. The performance improvement becomes more significant when the path length gets longer. The reason is that the aggressive sending rate in the original semi-TCP makes the nodes be in the status of congestion avoidance which throttles the increase of throughput. Figure 8 also shows that Algorithm 1 has achieved the near-optimal throughput. Algorithm 1 achieves a throughput at around 140 kbps in

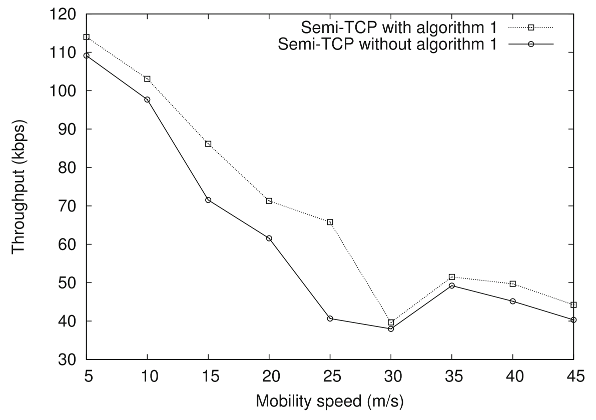

Fig. 11 End-to-end throughput in the random topology

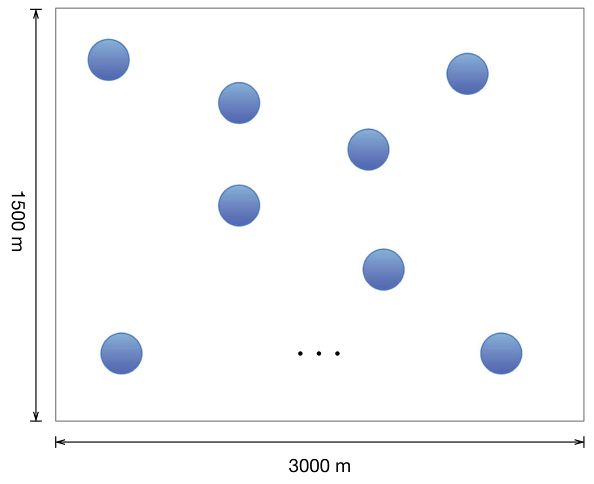

Fig. 10 Mobile random topology in the simulation. There are 48 nodes randomly deployed in an area of 3000 × 1500m^2

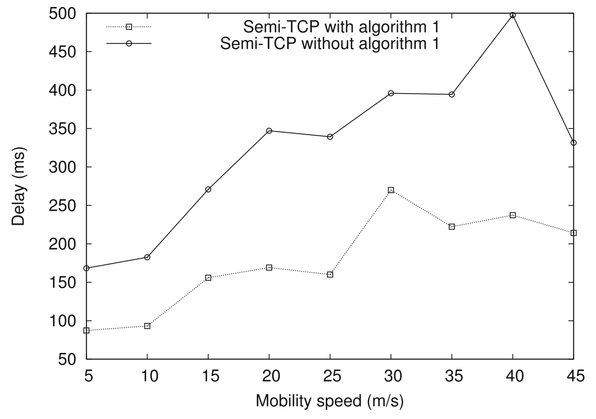

Fig. 12 End-to-end delay in the random topology

the case of 14 hops, which is consistent with the result in Fig. 5. The proposed Algorithm 1 can detect the congestion and interference in the medium locally and then control the source rate to maximize the end-to-end throughput. In the meanwhile, the end-to-end delay is significantly decreased by our algorithm. Particularly, the delay is decreased by almost 60% in path length of 4, from 60 to 20 ms. The medium contention and interference have substantially declined using our algorithm, thus wasting less network resources in retransmissions for both RTS/CTS and data packets. The simulation results indicate that our proposed algorithm can achieve higher throughput by consuming less network resources than the original semi-TCP.

The same improvement is also observed in the mobile random topology in Figs. 11 and 12. FTP flows are attached to 10 randomly selected sources and destinations. Due to the node mobility, the route for data forwarding may change frequently, which is the main reason that causes the congestion in nodes in the mobile random topology. The hop-by-hop strategy in semi-TCP can respond to this kind of congestion promptly. The proposed Algorithm 1 can avoid flushing packets to the medium when the forwarding route is not available, thus improve the end-to-end delay and the throughput, regardless of the mobility speed of nodes.

6 Conclusion

We have studied the rate control algorithm to improve semi-TCP in this paper. Our analysis has shown the aggressiveness of the buffer-based sending behavior in semi-TCP. We have then proposed a binary search-based rate control algorithm for semi-TCP. Simulation results show its improvement in end-to-end throughput and delay comparing to the original semi-TCP.

Endnotes

[1] An RTS will be discarded after the seventh failed attempts.

[2] RTSC ratio is the ratio of the number of RTSCs to the total RTSs.

[3] Semi-TCP makes a node to send RTSC and CTS-congestion (CTSC) when it is congested.

[4] The source rate is adjusted by a sending timer in Algorithm 1. To facilitate the explanation, we use the source rate in kbps directly.

Abbreviations

APR: Average attempts per RTS; CTS: Clear to send; CTSC: CTS-congestion; HAPR: High APR; LAPR: Low APR; MAC: Medium access control; MST: MAC sending time; RTS: Request to send; RTSC: RTS-congestion

Funding

This work was supported in part by the National Natural Science Foundation of China under grants 61431005, 61671208, 61671211, and U1701265; the Guangdong provincial research project under grants 2016A030308006, 2015A030310287 and 2016B090918049; and the Pearl River S&T Nova Program of Guangzhou.

Authors' contributions

WC and QG conceived and designed the study. WC and HY performed the simulations. YW and SJ analyzed the simulation results. WC wrote the paper. QG and YW reviewed and edited the manuscript. All authors read and approved the manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, People's Republic of China. [2]Ecole polytechnique, University of Nantes, Nantes, France. [3]College of Information Engineering, Shanghai Maritime University, Shanghai, People's Republic of China.

Received: 10 October 2017 Accepted: 12 November 2018
Published online: 04 December 2018

References

1. A. Al Hanbali, E. Altman, P. Nain, A survey of tcp over ad hoc networks. IEEE Commun. Surv. Tutor. **7**(3), 22–36 (2005)
2. K.-C. Leung, V. O. K. Li, Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges. IEEE Commun. Surv. Tutor. **8**(4), 64–79 (2007)
3. K. Chandran, S. Raghunathan, S. Venkatesan, R. Prakash, A feedback-based scheme for improving TCP performance in ad hoc wireless networks. IEEE Pers. Commun. **8**(1), 34–39 (2001)
4. G. Holland, N. Vaidya, Analysis of TCP performance over mobile ad hoc networks. Wirel. Netw. **8**(2/3), 275–288 (2002)
5. T. D. Dyer, R. V. Boppana, in *Proc. ACM Mobicom*. A comparison of TCP performance over three routing protocols for mobile ad hoc networks, (Rome, 2001)
6. J. Liu, S. Singh, ATCP: TCP for mobile ad hoc networks. IEEE J. Sel. Areas Commun. **19**(7), 1300–1315 (2001)
7. Z. Fu, B. Greenstein, X. Meng, S. Lu, in *Proc. IEEE ICNP*. Design and implementation of a TCP-friendly transport protocol for ad hoc wireless networks, (Paris, 2002)
8. S. R. Pokhrel, M. Panda, H. L. Vu, M. Mandjes, TCP Performance over Wi-Fi: Joint Impact of Buffer and Channel Losses. IEEE Trans. Mob. Comput. **15**(5), 1279–1291 (2016)
9. Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, M. Gerla, The impact of multihop wireless channel on TCP performance. IEEE Trans. Mob. Comput. **4**(2), 209–221 (2005)
10. H. Xie, A. Boukerche, TCP-CC: cross-layer TCP pacing protocol by contention control on wireless networks. Wirel. Netw. **21**(4), 1061–1078 (2015)
11. R. De Oliveira, T. Braun, in *Proc. IEEE INFOCOM*. A dynamic adaptive acknowledgment strategy for TCP over multihop wireless networks, (Miami, 2005)
12. Y. Cai, S. Jiang, Q. Guan, F. R. Yu, Decoupling congestion control from TCP (semi-TCP) for multi-hop wireless networks. EURASIP J. Wirel. Commun. Netw. **2013**(113), 1 (2013)
13. W. Chen, Q. Guan, S. Jiang, Q. Guan, T. Huang, Joint QoS provisioning and congestion control for multi-hop wireless networks. EURASIP J. Wirel. Commun. Netw. **2016**(113), 1 (2016)
14. F. P. Kelly, A. K. Maulloo, D. K. Tan, Rate control for communication networks: shadow prices, proportional fairness and stability. J. Oper. Res. Soc. **49**(3), 237–252 (1998)
15. D. Kliazovich, F. Granelli, Cross-layer congestion control in ad hoc wireless networks. Ad Hoc Netw. **4**(6), 687–708 (2006)

16. J. Lee, H. Lee, J. Lee, H. Lim, J. Park, J. Lee, in *Proc. IEEE ICC*. A performance study of proxy-based TCP rate control design for mobile video streaming services, (Kuala Lumpur, 2016), pp. 1–8
17. S. H. Low, D. E. Lapsley, Optimization flow controll: basic algorithm and convergence. IEEE/ACM Trans. Networking (TON). **7**(6), 861–874 (1999)
18. P. P. Mishra, H. Kanakia, A hop by hop rate-based congestion control scheme. ACM SIGCOMM Comput. Commun. Rev. **22**(4), 112–123 (1992)
19. Y. Yi, S. Shakkottai, Hop-by-hop congestion control over a wireless multi-hop network. IEEE/ACM Trans. Networking. **15**(1), 133–144 (2007)
20. B. Scheuermann, C. Lochert, M. Mauve, Implicit hop-by-hop congestion control in wireless multihop networks. Ad Hoc Networks. **6**(2), 260–286 (2008)
21. S. M. ElRakabawy, A. Klemm, C. Lindemann, in *Proc. ACM Mobicom*. Tcp with adaptive pacing for multihop wireless networks, (Cologne, 2005)
22. S. Ryu, C. Rump, C. Qiao, Advances in active queue management (aqm) based tcp congestion control. Telecommun. Syst. **25**(3), 317–351 (2004)
23. S. K. Bisoy, B. Pati, C. R. Panigrahi, P. K. Pattnaik, in *Computational Intelligence in Data Mining*. Analysis of tcp variant protocol using active queue management techniques in wired-cum-wireless networks (Springer, Singapore, 2017), pp. 439–448