LETTER

On Window Control Algorithm over Wireless Cellular Networks with Large Delay Variation*

Ho-Jin LEE†, Hee-Jung BYUN†, Nonmembers, and Jong-Tae LIM†, Member

SUMMARY In addition to high bit error rates, large and sudden variations in delay often occur in wireless cellular networks. The delay can be several times the typical round-trip time, which can cause the spurious timeout. In this letter, we propose a new window control algorithm to improve TCP performance in wireless cellular networks with large delay variation and high bit error rates. Simulation results illustrate that our proposal improves the performance of TCP in terms of fairness and link utilization.

key words: congestion avoidance, wireless cellular networks, delay variation

1. Introduction

The wireless networks have several features such as high bit error rates and large delay variation. In particular, the wireless cellular networks accommodating a variety of mobile devices has a sudden change of round-trip time (RTT) [1]. The delay variation can be caused by several factors such as the time-varying quality of the wireless link, handoff delay, and transmission interruptions due to the priority scheduling and the preemptive service [2].

The sudden and large increase in RTT can lead to the spurious timeout and trigger two undesirable behaviors [4]. First, TCP interprets the spurious timeout as being caused by packet losses and unnecessarily retransmits packets which are presumed lost. Moreover, the congestion avoidance mechanism falsely reduces the window size leading to low throughput. Although several solutions for wireless TCP such as TCP-Jersey [3] can improve TCP performance under wireless losses, they do not consider the spurious timeout. To alleviate the effect of delay variation, Eifel [4] and DualRTT [5] have been proposed. They can effectively detect the spurious timeout and improve TCP performance. However, after detecting the spurious timeout, they set the window size in appropriately resulting in poor fairness and low throughput. In this letter, we propose a new window control algorithm to improve TCP performance in the wireless cellular networks with large delay variation.

2. Window Control Algorithm

In this letter, we consider TCP flows from the fixed hosts to the mobile hosts and a single bottleneck link. Due to the difference of network bandwidth between wired and wireless links, a base station is liable to be a bottleneck. We assume that the outgoing link of the base station is the bottleneck link. Let \( M \) be the number of active flows routed through the base station. Each flow has a different RTT and experiences a different sudden change of RTT. Let \( \Delta_i(n) = \lfloor \frac{RTT_i(n)}{\tau_s} \rfloor \) where \( \lfloor x \rfloor \) is the smallest integer larger than \( x \), \( \tau_s \) is a pre-defined common value, and \( RTT_i(n) \) is the estimated RTT of flow \( i \) at time \( n \), respectively. Moreover, we assume that a sender updates its window size every its RTT. Then, the system can be described by a discrete-time model where the duration of a time slot is determined by \( \tau_s \).

First of all, we consider the packet loss due to high bit error rates. The existing TCP algorithm, such as Reno, treats the occurrence of packet loss as a manifestation of network congestion. This assumption may not be applied to the networks with wireless links, in which packet loss is often induced by link error, or causes other than network congestion. Misinterpretation of random packet loss as an indication of network congestion causes TCP senders to back down the window size unnecessarily, resulting in significant TCP performance degradation. If a packet loss due to the network congestion is eliminated, senders can see only the wireless losses. In this case, when a packet loss occurs, we can improve TCP performance by not halving the window size and just retransmitting the lost packet. For the sake of this purpose, we introduce the feedback value, \( FV \), computed by the router, as follows:

\[
FV = \frac{1}{M} (q_{ih} - q(n))
\]

where \( q_{ih} \) is the queue threshold and \( q(n) \) is a queue length at time \( n \). The queue length can be described as

\[
q(n) = \left[ q(n-1) + \sum_{i=1}^{M} A_i(n-1) - C \tau_s \right]^+
\]

where \( C \) is the bottleneck link bandwidth and \( [ \cdot ]^+ = \max(0, \cdot) \). Moreover, \( A_i(n-1) \) is the number of arriving packets at the base station from flow \( i \) during the time slot \( [n-1, n) \). The feedback value is computed every \( \tau_s \) and carried by returning TCP acknowledgements (ACKs) in the
advertised window (AWND) field. If the current value in the AWND field exceeds the feedback value computed by the router, it is overwritten newly by the computed feedback value. Sends receive the smallest feedback value. This value explicitly does not only notify senders of the bottleneck queue length, but also helps senders to avoid the overflow.

Secondly, we consider another feature, the sudden and large increase or decrease of RTT. We denote a set \( K_i = \{ k_{ij} | j = 0, \cdots, N_i \} \) where \( k_{ij} \) represents time \( n \) when flow \( i \) experiences the sudden change of RTT, and \( N_i \) is the frequency of sudden RTT change of flow \( i \) throughout the live time of the flow. Then, \( \Delta_i(n) = \Delta_i(k_{ij}) \) for \( k_{ij} \leq n < k_{i,j+1} \). If there are 6 sudden changes of RTT, then \( N_i = 6 \). For the sake of convenience, we denote \( \Delta_i(k_{ij}) = l_{ij} \).

Eifel [4] and DualRTT [5] propose algorithms to detect the spurious timeout. When timeout occurs, they store the current values of the slow start threshold and the window size. If the spurious timeout is detected, the sender simply restores the slow start threshold and the congestion window to the stored values. However, since a flow with a large increase of RTT can not send new packets nor increase its window size during the spurious timeout, its throughput is lower than those of other flows without large RTT increase. In addition, since the throughput is inversely proportional to the RTT, fairness also deteriorates. Thus, instead of setting the congestion window to the previous size, a sender should increase the window size fast to improve throughput and fairness after detecting the spurious timeout. Moreover, in case of decrease of RTT, since the increased window size may impose traffic burden on the network, the window size should decrease fast.

Therefore, from the assumption that the sender updates its window size every its RTT, considering two features such as high bit error rates and RTT variation in wireless cellular networks, we introduce a new window control algorithm as follows:

\[
w_i(n) = \begin{cases} 
\frac{\alpha \cdot \Delta_i(n)}{\Delta_i(n-\tau_i)} w_i(n-\tau_i) + \frac{\beta}{\tau_i} (q_{ib} - q(n-d_{ib)}) \cdot \Delta_i(n) & \text{if } \text{mod}(n, \Delta_i(n)) = \text{mod}(k_{ij}, l_{ij}), \\
\frac{\beta}{\tau_i} (q_{ib} - q(n-d_{ib})) & \text{else}, \\
w_i(n-1) & \text{if } n \leq \tau_i.
\end{cases}
\]

where \( \alpha \) and \( \beta \) are positive control parameters, \( d_{ib} = \left\lceil \frac{n}{\tau_i} \right\rceil \), and \( \tau_i \) is the propagation delay of flow \( i \) from the base station to a sender. Multiplying \( FV \) by their RTTs, we can guarantee fairness among flows with different RTTs. After detecting the spurious timeout, to compensate a low throughput due to the large increase of RTT, we increase the congestion window fast by multiplying the ratio \( \frac{\alpha \cdot \Delta_i(n)}{\Delta_i(n-\tau_i)} > 1 \), by the previous window size. Hence, fairness can improve and the link utilization can also increase fast. In the case of the large decrease of RTT, to avoid congestion, we decrease the large congestion window by multiplying the ratio \( \frac{\beta}{\tau_i} \) fast. Furthermore, through \( FV \), the queue length will be adjusted after one RTT. If the router has enough buffer space to hold the sudden burst of packets, there will be no packet loss due to the overload. Thus, when a packet loss occurs, we treat it as an indication of link error and just retransmit it without decreasing the window size. In our proposal, the senders do not need any additional information except the feedback value from the bottleneck link. The receivers also do not need additional functions. Thus, since the proposal uses the existing AWND field and the estimated RTT without additional fields and information, it can be easily deployed to the wireless cellular networks. Consequently, we can improve TCP performance in wireless cellular networks with high bit error rates and large RTT variation.

### 3. Equilibrium Point and Asymptotic Stability

In this section, we investigate the equilibrium point and the condition for asymptotic stability. To analyze the asymptotic stability, we approximate the system based on the sense of average. Let \( \hat{w}_i(n) \) denote the average window size of TCP flow \( i \) at time \( n \). Since each TCP flow \( i \) updates its window size every its RTT, we assume that flow \( i \) sends \( \frac{1}{\tau_i} \) every time slot on the average sense. Then, from (3), the average window size denoted by \( \hat{w}_i(n) = \frac{w_i(n)}{\Delta_i(n)} \) is described as

\[
\hat{w}_i(n) = \alpha \cdot \hat{w}_i(n-\tau_i) + \frac{\beta}{\tau_i} (\hat{q}(n) - \hat{q}(n-d_{ib}))
\]

where \( \hat{q}(n) \) is the approximate queue length at time \( n \). Let \( \tau_{sf} \) be the propagation delay of flow \( i \) from a sender to the base station and \( d_{sf} = \left\lceil \frac{n}{\tau_{sf}} \right\rceil \). Since flow \( i \) sends \( \hat{w}_i(n) \) packets every time slot, we approximate \( A_i(n-1) \equiv \hat{w}_i(n-d_{sf}) \). Then, as it follows from (2), \( \hat{q}(n) \) can be written as

\[
\hat{q}(n) = \hat{q}(n-1) + \sum_{i=1}^{M} \hat{w}_i(n-d_{sf}) - CT_s
\]

Let \( \hat{w}_{is} \) and \( \hat{q}_s \) be the equilibrium point of \( \hat{w}_i(n) \) and \( \hat{q}(n) \). Then, from (4) and (5), we have \( \hat{w}_{is} = \frac{\beta}{\tau_{sf}}, \hat{q}_s = \frac{\beta}{\tau_{sf}} \cdot \frac{\Delta_i(n)}{\alpha \cdot \Delta_i(n)} \cdot \frac{1}{\tau_{sf}}, \) where \( 0 < \alpha < 1 \) and \( \hat{q}_{ib} > \frac{1}{\tau_{sf}} \cdot \frac{\Delta_i(n)}{\alpha \cdot \Delta_i(n)} \cdot \frac{1}{\tau_{sf}} \) to guarantee \( \hat{w}_{is} > 0 \) and \( \hat{q}_s > 0 \). Then, \( \hat{q}_s \) is stabilized at a certain value smaller than \( \hat{q}_{ib} \) and greater than zero. Moreover, the throughput of each flow, which is represented by \( \hat{w}_{is} / \tau_{sf} \) for all \( i \), is same as \( \hat{q}_s \). Therefore, each flow achieves the high throughput and shares the bandwidth fairly.

Let \( x_i(n) = \hat{w}_i(n) - \hat{w}_{is} \) and \( y(n) = \hat{q}(n) - \hat{q}_s \). Then, we rewrite (4) and (5) as follows:

\[
x_i(n) = x_i(n-\tau_i) - \frac{\beta}{\tau_i} \cdot y(n-d_{ib})
\]

\[
y(n) = y(n-1) + \sum_{i=1}^{M} x_i(n-d_{sf})
\]

When packets are considered as losses due to the spurious timeout, a sender is in the slow start phase, the window size is set to one, and the retransmission of them is triggered. However, in case that the retransmission packets arrive at receivers earlier than the delayed packets do, we
cannot detect the spurious timeout with Eifel [4] and DualRTT [5] which use ACKs. This is because a receiver does not generate ACKs by the duplicate packets. Thus, since $l_i$ is bounded, we denote $\phi = \max(l_i)$.

For the purpose of simplicity, we assume that the delay from each sender to the bottleneck link is identical with $d_{lf} = F$. Moreover, let $\bar{x}(n) = \sum_{i=1}^{M} x_i(n)$. Then, (6) and (7) can be rewritten as follows:

$$\bar{x}(n) = \alpha \cdot \bar{x}(n-\phi) - \frac{\beta}{M} \sum_{d=1}^{D} \gamma_d \cdot y(n-d)$$

(8)

$$y(n) = y(n-1) + \bar{x}(n-F)$$

(9)

where $\gamma_d$ is the number of flows with delay $d$, $\sum_{d=1}^{D} \gamma_d = M$, $D = \text{max} \{d_{lf}\}$, and $\phi > D + F$. Let $X(n)$ be the state vector by $X(n) = [\bar{x}(n) \; \bar{x}(n-1) \; \cdots \; \bar{x}(n+1-\phi) \; y(n) \; \cdots \; y(n+1-D)]^T$. Then, we obtain

$$X(n+1) = AX(n)$$

(10)

where

$$A = \begin{pmatrix}
0 & 0 & \cdots & 0 & -\frac{\beta}{M} \gamma_1 & \cdots & -\frac{\beta}{M} \gamma_D \\
1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0
\end{pmatrix}$$

The characteristic polynomial of $A$ is described by $\Phi(z) = z^{D+1} - \frac{\beta}{M} \gamma_1 z^D - \cdots - \frac{\beta}{M} \gamma_D$. The roots of $\Phi(z)$ are $R(h)$, where $h = \alpha, \beta, M, \gamma_1, \cdots, \gamma_D$. Thus, if all $R(h)$ are inside the unit circle, the equilibrium point is asymptotically stable.

**Remark.** In contrast to the wired networks, the wireless networks have the variable capacity, depending on many factors such as distance and multipath interference [6]. Then, the capacity, $C$, can be rewritten as $C = (1-e(n))$, where $e(n)$ denotes the stochastic process of packet loss ratio at time $n$. However, since $C$ can generally take arbitrarily large value, but $0 \leq e(n) \leq 1$, on the average sense, $C = (1-e(n))$ can be represented as the constant value, $C = (1-P_e)$, where $P_e = E[e(n)]$. Let $C' = C = (1-P_e)$. Then, our results including the stability condition in this letter do not change except $C$ is replaced by $C'$. Particularly, the throughput of each flow can be expressed as $\frac{C'}{M} = \frac{C(1-P_e)}{M}$. We verify this result through simulations.

**4. Simulation**

In this section, we compare our proposal with TCP-Jersey [3] and DualRTT [5] and illustrate improvement in terms of fairness and link utilization. To compare fairness, we use the Jain’s fairness index, $\Lambda$, which is defined as follows [7]: If there are $M$ flows and each flow is characterised by its average throughput $s_a$, $a \in [1,M]$, then $\Lambda = \frac{(\sum_{i=1}^{M} s_i)^2}{M \sum_{i=1}^{M} s_i^2}$, where $1/M \leq \Lambda \leq 1$. A perfectly fair bandwidth allocation would result in $\Lambda = 1$. Moreover, in order to detect the spurious timeout, we use Eifel algorithm [4]. For DualRTT [5], we set a parameter $\tau = 0.5$. The network topology with a single bottleneck link is a dumbbell. The bottleneck link bandwidth, $C$, is 2500 packets/s. We set $\tau_s = 0.1$ sec, $q_{th} = 400$ packets. The value of $\alpha = 0.6$, and $\beta = 0.3$ are used for stability unless otherwise indicated. The simulation time is 100 sec. The throughput is in the unit of packets/s.

In the first simulation, to investigate the TCP performance only in the presence of the delay variation, we assume $P_e = 0\%$. There are two flows with the same minimum RTT, 0.1 sec. TCP flow 1 experiences the sudden and large change of RTT and TCP flow 2 has no sudden variation of RTT. Table 1 compares the case of more frequent variation of RTT, RTT A ($N_1 = 30$), with the case of less frequent variation of RTT, RTT B ($N_1 = 6$). Despite the frequent variation of RTT, Table 1 shows that the throughput of each flow in our proposal is similar. Moreover, our proposal has higher throughput than those of TCP-Jersey and DualRTT. Since TCP-Jersey cannot detect the spurious timeout, it has the lowest throughput. Therefore, our proposal guarantees fairness among flows and achieves the high throughput.

In the second simulation, we assume that there are ten flows with different minimum RTTs. The minimum RTT of TCP flow 1-3 and TCP flow 10 is 0.1 sec, the minimum RTT of TCP flow 4-6 is 0.2 sec, the minimum RTT of TCP flow 7 is 0.3 sec, the minimum RTT of TCP flow 8 is 0.4 sec, and the minimum RTT of TCP flow 9 is 0.5 sec. Each flow experiences a different delay variation. The delay variations caused by several reasons such as handoff delay are likely to exceed the typical round-trip time several times. To simulate such scenarios, the length of large change of RTT is distributed between (0.2, 1) sec, with interval between delay variations being distributed between (5, 20) sec. TCP flow 10 does not experience the sudden change of RTT. Irrespective of different RTTs and large variation of RTTs, Table 2 shows that our proposal provides flows with bandwidth fairly. As $P_e$ increases, it is also observed that our proposal achieves significantly higher throughput than that of DualRTT. This is because DualRTT unnecessarily decreases the window size under the wireless packet loss. Although the throughput does not severely degrade by wireless packet losses, TCP-Jersey leads to poor fairness due to large variation of RTT. Considering the varying capacity, we can observe that our throughput from the simulations is similar to the estimated results, $\frac{C(1-P_e)}{M}$.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Variation of RTT</th>
<th>TCP1</th>
<th>TCP2</th>
<th>Fairness</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP-Jersey</td>
<td>RTT A</td>
<td>253</td>
<td>2148</td>
<td>0.616</td>
</tr>
<tr>
<td></td>
<td>RTT B</td>
<td>797</td>
<td>1643</td>
<td>0.893</td>
</tr>
<tr>
<td>DualRTT</td>
<td>RTT A</td>
<td>561</td>
<td>1802</td>
<td>0.783</td>
</tr>
<tr>
<td></td>
<td>RTT B</td>
<td>891</td>
<td>1572</td>
<td>0.929</td>
</tr>
<tr>
<td>Proposal</td>
<td>RTT A</td>
<td>1036</td>
<td>1377</td>
<td>0.980</td>
</tr>
<tr>
<td></td>
<td>RTT B</td>
<td>1205</td>
<td>1283</td>
<td>0.999</td>
</tr>
</tbody>
</table>
Table 2  Average throughput and fairness. (10 flows)

<table>
<thead>
<tr>
<th>$P_e$ (%)</th>
<th>Algorithms</th>
<th>Fairness</th>
<th>Throughput Simulation</th>
<th>$C(1-P_e)$ M</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TCP-Jersey</td>
<td>0.690</td>
<td>235.2</td>
<td>·</td>
</tr>
<tr>
<td></td>
<td>DualRTT</td>
<td>0.576</td>
<td>230.0</td>
<td>·</td>
</tr>
<tr>
<td></td>
<td>Proposal</td>
<td>0.998</td>
<td>247.9</td>
<td>250.0</td>
</tr>
<tr>
<td>0.1</td>
<td>TCP-Jersey</td>
<td>0.688</td>
<td>232.8</td>
<td>·</td>
</tr>
<tr>
<td></td>
<td>DualRTT</td>
<td>0.752</td>
<td>197.2</td>
<td>·</td>
</tr>
<tr>
<td></td>
<td>Proposal</td>
<td>0.997</td>
<td>246.1</td>
<td>249.7</td>
</tr>
<tr>
<td>1</td>
<td>TCP-Jersey</td>
<td>0.690</td>
<td>227.9</td>
<td>·</td>
</tr>
<tr>
<td></td>
<td>DualRTT</td>
<td>0.806</td>
<td>71.2</td>
<td>·</td>
</tr>
<tr>
<td></td>
<td>Proposal</td>
<td>0.997</td>
<td>243.3</td>
<td>247.5</td>
</tr>
</tbody>
</table>

As shown in Table 3, TCP Reno flows have the lower average throughput than proposal flows do. This degradation of TCP Reno flows is not due to our proposal, but due to the problem of TCP Reno flows, i.e., TCP Reno flows unnecessarily reduce their transmission rates at a wireless packet loss and the spurious timeout. Moreover, irrespective of the number of proposal flows, TCP Reno flows have the similar average throughput. It means that our proposal does not penalize the existing TCP Reno flows.

5. Conclusion

In this letter, we propose a new window control algorithm to regulate window size according to the change of RTT. It uses the queue length and the ratio of the current RTT to the previous RTT. The simulation results show that the proposed algorithm helps us to utilize the link bandwidth efficiently while providing fairness among flows in the wireless cellular networks with high bit error rates and large variation of RTT.

References