Reducing the TCP Acknowledgment Frequency

Sara Landström
Luleå University of Technology, Sweden
Sara.Landstrom@ltu.se

Lars-Åke Larzon
Uppsala University, Sweden
lln@it.uu.se

ABSTRACT

Delayed acknowledgments were introduced to conserve network and host resources. Further reduction of the acknowledgment frequency can be motivated in the same way. However, reducing the dependency on frequent acknowledgments in TCP is difficult because acknowledgments support reliable delivery, loss recovery, clock out new segments, and serve as input when determining an appropriate sending rate.

Our results show that in scenarios where there are no obvious advantages of reducing the acknowledgment frequency, performance can be maintained although fewer acknowledgments are sent. Hence, there is a potential for reducing the acknowledgment frequency more than is done through delayed acknowledgments today. Advancements in TCP loss recovery is one of the key reasons that the dependence on frequent acknowledgments has decreased.

We propose and evaluate an end-to-end solution, where four acknowledgments per send window are sent. The sender compensates for the reduced acknowledgment frequency using a form of Appropriate Byte Counting. The proposal also includes a modification of fast loss recovery to avoid frequent timeouts.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Performance

Keywords

Acknowledgments, TCP

1. INTRODUCTION

Acknowledgments are usually much smaller than segments. Certain network devices, however, are limited by the number of packets they can process rather than the size of the packets. When the resources can not be assigned in arbitrarily small pieces or it is costly to assign the resources to a user, acknowledgments may have a high overhead in percentages, making a reduction of the acknowledgment frequency interesting.

There are proposals to reduce the number of acknowledgments sent by TCP in asymmetric and wireless networks [11, 22, 27, 29, 10, 25, 9, 16]. In asymmetric networks, the bottleneck can be in the acknowledgment path. In wireless networks, requiring fewer acknowledgments may decrease the interference and save power in addition to freeing up bandwidth resources. We call scenarios that exhibit such characteristics "gain scenarios". The gain can easily overshadow any negative effects of lowering the acknowledgment frequency in these scenarios. Therefore, the question of whether a lower acknowledgment frequency can be used in a wider range of scenarios without sacrificing performance remains to be answered.

Allman [2], compared the performance of delayed acknowledgments (DA) with byte counting vs. acknowledging every segment (AE) in scenarios not including asymmetry nor wireless links. We use a similar approach, i.e., we study situations where reducing the acknowledgment frequency is the most likely to degrade throughput performance. Since Allman's study TCP loss recovery has been refined, we consider this development and evaluate lower acknowledgment frequencies than represented by DA. With DA every second segment is acknowledged.

To compensate for lowering the acknowledgment frequency we modified Appropriate Byte Counting (ABC) [3] to allow more than a maximum of two segments for each acknowledgment to be considered. Also the behavior following a timeout was modified to allow an exponential increase of the sending rate with a low acknowledgment frequency.

We find that acknowledging data four times per send window yields good results. In theory, sending two acknowledgments per send window allows full utilization but the router buffer requirement for this strategy when there is a single flow is much higher than with AE. In practice, acknowledging data four times per send window is therefore a stronger option. We also identify and solve a problem with loss recovery that arises when the acknowledgment frequency is low. The analysis of this loss recovery problem contributes to understanding how changing the acknowledgment frequency over time affects the data sending pattern and thus performance.

The remainder of this paper is structured as follows; in Section 2, we discuss the many roles of acknowledgments in TCP and techniques that can be applied at the end-points to compensate for a lower acknowledgment frequency. We then introduce the acknowledgment strategies evaluated in this study in Section 3. Thereafter, we illustrate some of the performance problems when reducing the acknowledgment frequency and evaluate potential solutions in Section 4. The related work focused on gain scenarios and congestion con-

trol of acknowledgments is discussed in Section 5, and then we conclude in Section 6.

2. THE ROLES OF ACKNOWLEDGMENTS

Acknowledgments serve several purposes in TCP. Apart from acknowledging that segments have been received, acknowledgments clock out new segments, and the sending rate increase is based on the number of acknowledgments that arrive. Therefore, a reduction of the acknowledgment frequency affects the micro burstiness, sending rate and acknowledgment pattern.

2.1 Micro burstiness

TCP is called an ack-clocked protocol because each acknowledgment releases new segments. The number of new segments released together corresponds to the number of segments acknowledged simultaneously. The fewer acknowledgments sent, the larger the bursts of segments released. A burst sent in response to a single acknowledgment is called a micro burst. TCP usually sends micro bursts of one to three segments when each or every second segment is acknowledged.

There are concerns about increasing the burstiness of TCP, as lowering the acknowledgment frequency would. Through queuing theory it is known that bursty traffic has larger buffer requirements, higher drop rates and longer queuing delays [23]. Most existing techniques [5] for dealing with large micro bursts are designed for when they rarely occur. A low acknowledgment frequency continuously cause micro bursts that are larger than three segments. Instead of relying on acknowledgments to clock out segments, the sender can pace out segments using a timer. This technique is called pacing and can be used throughout a connection [1, 30] to reduce the micro burstiness of a flow.

More studies on the importance of micro burstiness are needed. A recent measurement study shows that only micro bursts of considerable size (hundreds of segments) cause aggravated loss [12], but it is not possible to generalize from one measurement study.

2.2 Byte counting

The *number* of acknowledgments determines the sending rate. The fewer acknowledgments, the slower is the sending rate increase. Byte counting lessen this dependence by considering the *amount of data* acknowledged by each acknowledgment instead of the actual number of acknowledgments.

Allman, [2], evaluated two byte counting algorithms for use with DA. The Unlimited Byte Counting algorithm (UBC) counts all the acknowledged data, whereas Limited Byte Counting (LBC) counts at most two maximum sized segments per acknowledgment. LBC gave better results than UBC for TCP Reno, but there was no difference in performance between LBC and UBC for TCP SACK [26], [17]. These results suggest that UBC may be suitable for use with improved loss recovery algorithms.

The standardized version of byte counting, Appropriate Byte Counting (ABC) [3], limits the amount of data counted for one acknowledgment to at most $L * MSS$ bytes. MSS is the Maximum Segment Size of TCP in bytes and L a configurable parameter called the limit. It is encouraged to experiment with L set to two to compensate for DA, but higher values are discouraged because byte counting increases the micro burstiness. During slow start periods that follow after a timeout, L should be set to one, as first suggested in [7]. Otherwise, segments that have left the network prior to the timeout, can be taken as an indication of the current network status resulting in a too aggressive sending rate increase.

2.3 Acknowledgment patterns

During periods of loss, the receiver sends an acknowledgment in response to each segment that arrives. The acknowledgments inform the sender of which segments that have arrived through the SACK option and support retransmission decisions. If the acknowledgment frequency is low when a loss occurs, the increase in acknowledgment traffic, when the receiver starts to acknowledge every segment, can be large. Also, the data sending pattern changes as a result of the changed acknowledgment frequency, since the acknowledgments clock out new segments.

3. ACKNOWLEDGMENT FREQUENCY

The acknowledgment bit rate could be reduced either by the TCP receiver sending fewer acknowledgments or by an intermediate node dropping selected acknowledgments. To be able to select acknowledgments for dropping, an intermediate node would have to inspect the TCP header and keep state for each connection or inspect its buffer for each arriving acknowledgment. Even then, the TCP end-points have superior information about the importance of individual acknowledgments, which motivates an end-to-end solution.

In this section, we present the acknowledgment strategies that we evaluate. We also establish a lower bound on the acknowledgment frequency. Throughout the study we use this lower bound to better understand the drawbacks of reducing the acknowledgment frequency.

3.1 A Lower Bound

We first establish a lower bound for the acknowledgment frequency from a TCP perspective. To facilitate the analysis, we assume that the last segment in a micro burst is acknowledged, negligible buffering delay, and a very large delayed acknowledgment interval.

TCP is a sliding window protocol. Thus, at least an acknowledgment per send window of data must be sent to avoid timeouts. In practice, one acknowledgment per send window transforms TCP into a stop-and-wait protocol. The micro bursts are separated by one RTT, i.e., the time it takes for an acknowledgment to reach the sender and the segments it releases to propagate to the receiver.

To allow full utilization of the available bandwidth, two acknowledgments per send window of data are required and micro burst i should start to arrive at the receiver as the receiver sends the acknowledgment for micro burst $i-1$. The size of a micro burst corresponds to the time from when an acknowledgment is sent until the data it clocks out reaches the receiver. That is, a micro burst must correspond to one bandwidth×delay product (BDP) of data. The send window thus has to be two BDPs before the full capacity can be used. Furthermore, one cycle of the protocol takes two RTTs to complete when approaching full utilization. A cycle is the time it takes for a send window of segments to be acknowledged.

We refer to the strategies based on send window size as xW, where x is the number of acknowledgments sent per send window.

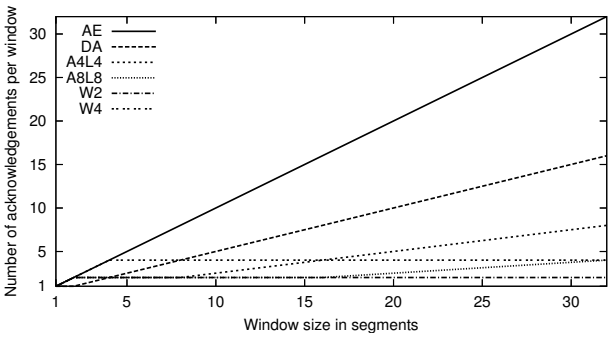


Figure 1: The potential of different acknowledgment strategies to reduce the acknowledgment bit rate requirement. The window is the TCP send window.

3.2 Acknowledgment Strategies

We consider the following acknowledgment strategies:
AE – *acknowledge every segment (AE),*
DAL2 – *delayed acknowledgments with* $L = 2$*,*
A4L4 – *acknowledging every fourth segment with* $L = 4$*,*
A8L8 – *acknowledging every eighth segment with* $L = 8$*,*
2W – *acknowledging two segments per send window, and*
4W – *acknowledging four segments per send window.*
L is the byte counting parameter that limits the amount of data to be counted for a single acknowledgment.

AE and DAL2 are being used today and are supported by IETF standards [15, 14, 8]. We include A4L4 and A8L8 for their simplicity, similarity to DAL2 and because they reduce the acknowledgment traffic more than DAL2. 2W has been chosen because it represents the lower bound and the effects from reducing the acknowledgment frequency should therefore be emphasized. 4W has a cycle of about 1.33 RTTs, and provides redundancy compared to 2W.

Depending on the path and the instantaneous conditions on that path, the TCP send window may vary between one and several thousands of segments. The performance of an acknowledgment strategy that acknowledge every ith segment, suffers if fewer than two segments are acknowledged per send window. We therefore always apply the computed lower bound when acknowledgments are sent less often than for every second segment. A fixed acknowledgment interval also leads to more acknowledgments being sent per send window when the send window is larger, whereas the xW strategies always generates a maximum of x acknowledgments per send window. Therefore it is interesting to study different send window sizes. The ratio of the number of segments covered by an acknowledgment to the send window size will most likely influence the results.

The strategies have different potential for reducing the acknowledgment frequency. Figure 1 shows the behavior for different window sizes. The larger the send window, the better 2W and 4W are in comparison to the other strategies. For small send windows, 2W gives the largest bit rate reduction. The acknowledgment frequencies of all acknowledgment strategies is similar when there are many segments lost, since the receiver acknowledges all segments if a segment is missing.

The receiver has to be part of any scheme to reduce the acknowledgment frequency. The TCP sender needs to compensate for a large reduction in the acknowledgment frequency to avoid a throughput decrease. Therefore, taking advantage of the sender's knowledge of the RTT, send window and congestion control phase to choose the acknowledgment frequency only increases the sender's involvement. When the connection is set-up a handshake should be performed to determine if both the receiver and the sender support the desired acknowledgment strategy.

4. DESIGN AND EVALUATION

We want to reduce the acknowledgment frequency without decreasing throughput performance. To achieve this goal, we need to study a number of issues. First, to compensate for sending fewer acknowledgments we want to use byte counting, but currently ABC supports only low L values. Second, the increased micro burstiness, when an acknowledgment clocks out several segments, can influence drop rates and loss patterns. Third, especially in the low multiplexing case, the increased micro burstiness affects the router buffer requirement. Last, the acknowledgment pattern influences the sending pattern and can lead to different estimates of the congestion level resulting in fairness problems. In this section we investigate these issues and evaluate potential solutions.

4.1 Simulation Platform

We used the Network Simulator version 2.29, ns-2, for our simulations. The tcp-sack1 agent was modified to perform SACK-based loss recovery according to [13]. We implemented A4L4, A8L8, 2W, and 4W through a simple poll mechanism. The sender sets a bit in the TCP header to tell the receiver to acknowledge the segment. We also use the poll mechanism to ensure an immediate reply to the SYN/ACK during the initial handshake. The receiver only needs to be modified to acknowledge marked segments, which is a minor modification.

We enabled timestamps to get a better estimate of the effective RTT and a large initial send window [6]. The segment size was 1000 bytes. We also varied the delayed acknowledgment interval. The default value was 100 ms. We experimented with values as large as 1000 ms to avoid interactions between the acknowledgment frequency, the RTT and the delayed acknowledgment interval.

4.2 ABC with a High Limit

ABC is designed for use with L less than or equal to two. Setting L to one during slow start following a timeout therefore has limited effect on the sending rate increase. For higher values of L, however, we need to reconsider the design. We want to avoid counting segments that left the network prior to the timeout, but still allow an exponential sending rate increase during slow start.

The receiver will revert to a lower acknowledgment frequency after a timeout as soon as the send window is large enough and the sequence of received segments is without holes. This prevents the sender from doubling the sending rate for the remainder of the slow start period. To allow an exponential increase, we take the minimum of L, and the current send window to determine how many segments that may be counted per acknowledgment. If many acknowledgments acknowledge both newly retransmitted data and old data, the sender can more than double its sending rate based on out-dated information, The results reported herein all include ABC with this modified behavior following a timeout.

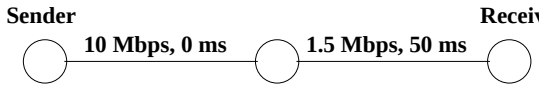


Figure 2: The simulation topology used to generate Figure 3.

There are no indications that the change causes any problems.

SACK information can be used to reduce the risk of a too aggressive sending rate increase further. Only segments that have not previously been acknowledged through the SACK option should be counted when they are cumulatively acknowledged. We have not implemented this addition.

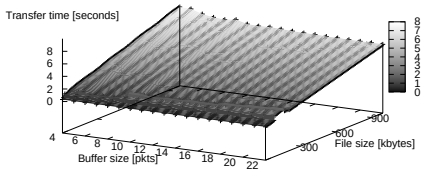
We repeated the simulations in [2] to determine whether ABC and advances in TCP loss recovery, such as Limited Transmit [4] and SACK-based loss recovery [13], have an effect on the results. We also extended the study to consider lower acknowledgment frequencies than represented by DA.

The simulation topology is described in Figure 2. It has a BDP of $1.5Mbps \times 100ms$, or about 19 segments. One flow is transferred at a time. Both the bottleneck buffer size and the transfer size have been varied. The delayed acknowledgment interval is 100 ms.

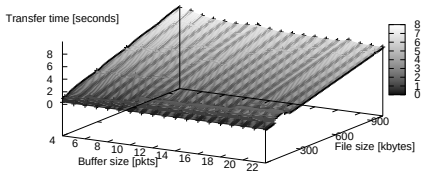
In Figure 3, we show selected results. Ideally the plane should be smooth and dark because it indicates short transfer times and a predictable behavior. DAL2, Figure 3(b), performs as well as AE, Figure 3(a). The performance of both configurations is stable for all buffer sizes. This is an improvement compared to the results presented in [2] that exhibited more variability. A8L8 and 2W perform poorly when the buffer size is small, Figures 3(d) and 3(e). Also for the largest investigated buffer sizes, the performance is less predictable than for AE and DAL2. It is the initial slow start overshoot that often results in a timeout when the acknowledgment frequency is low. As shown in Figure 3(f), 4W performs better than A4L4, A8L8, and 2W. When the buffer is small, it keeps the micro bursts small and the acknowledgment pattern does not vary much.

A side-by-side comparison of DAL2 and AE, in Figure 4, reveals that by delaying the acknowledgments a penalty corresponding to the transmission delay of the link is introduced each RTT until the BDP is reached. The transmission delay separates two segments that are sent back-to-back from each other. If every fourth segment is acknowledged the penalty will be three times the transmission delay each RTT. Other factors that determine the penalty are the BDP, the delayed acknowledgment interval, and the initial send window. This means that reducing the number of acknowledgments can not be entirely compensated for by byte counting. We call this penalty, "the byte counting delay". It is also present in congestion avoidance, reducing the aggressiveness. The effect of the byte counting delay can be reduced by acknowledging every segment in slow start, as in [2]. If the sender is polling the receiver for acknowledgments, this is straightforward to implement. We do not acknowledge every segment during slow start in this study though.

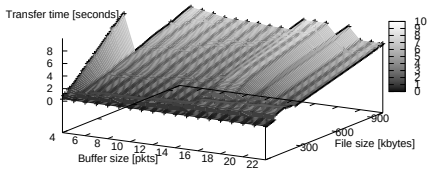
We have shown that 2W and A8L8 cause increased transfer times both for small and large buffers, when there are only one active connection. 4W and A4L4 yield a more stable performance, even if the variability is larger for larger



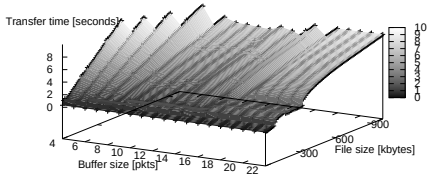
(a) AE.



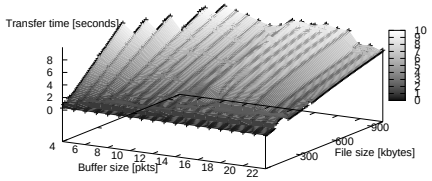
(b) DAL2.



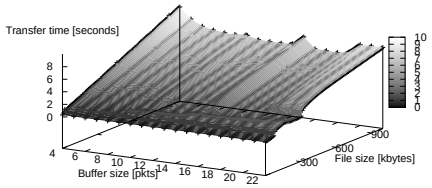
(c) A4L4.



(d) A8L8.



(e) 2W.



(f) 4W.

Figure 3: Throughput for different buffer sizes and file sizes.

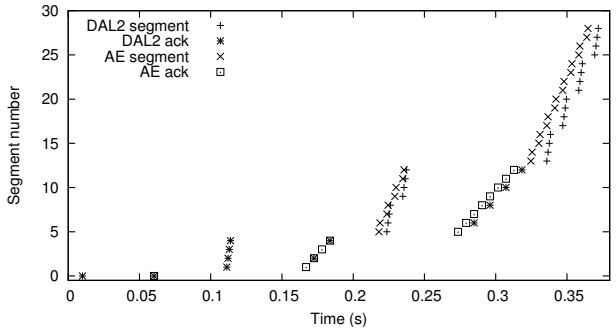


Figure 4: A study of DAL2 and AE.

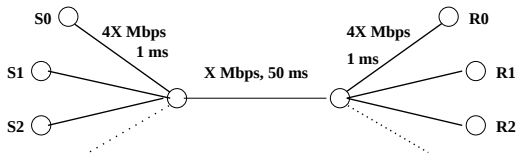


Figure 5: The simulation topology used to study flow multiplexing and varying bandwidths.

buffer sizes. These results indicate that there is a potential for lowering the acknowledgment frequency.

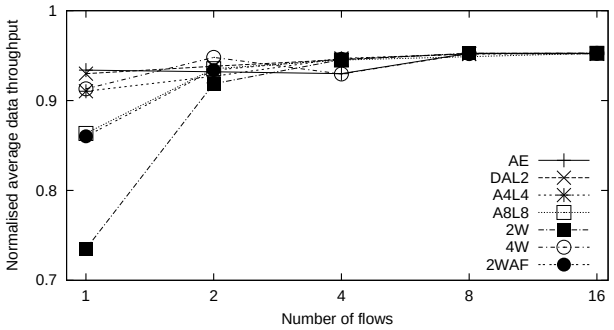
4.3 Drop Rates and Loss Recovery

We continue to study one-way traffic, but introduce flow multiplexing and vary the bandwidth. The flows have homogeneous RTTs and the topology is shown in Figure 5. We are interested in the worst case performance and therefore we use drop-tail buffering, which is known to be biased against bursty traffic. The bottleneck buffer size is set to one BDP and the delayed acknowledgment interval to 1000 ms. We simulate 50 s when the bottleneck is 1 Mbps, and 2000 s when the bottleneck is 100 Mbps. The initial 40% of each simulation is disregarded.

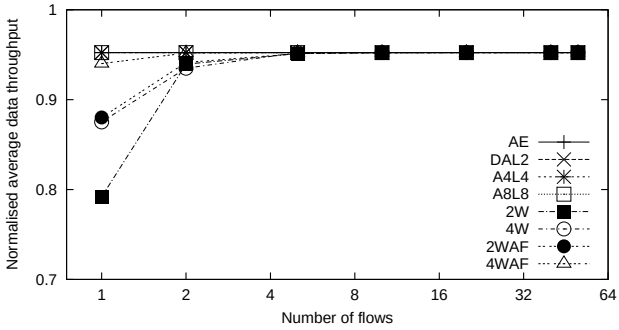
Figure 6 shows that bandwidth utilization is the lowest when there is only one or a few flows active. 2W is not able to utilize the capacity when multiplexing is low, see Figures 6(a) and 6(b). 4W has a lower utilization at 100 Mbps than at 1 Mbps. A8L8 negatively influences the utilization when the bottleneck is 1 Mbps, but not when it is 100 Mbps. This supports the intuition that the ratio of the segments acknowledged together with the send window size is important.

The drop rates, Figure 7, are similar for all strategies with few flows although 2W has a lower throughput in the one flow case. Normally there should be a strong correspondence between the drop rates and the throughput, therefore we investigate which segments 2W loses. We find that the dropped segments are the segments sent in response to the cumulative acknowledgment, that is triggered when segments start to arrive out-of-order at the receiver, and the fast retransmit. It is unfortunate to lose the fast retransmit, because it leads to a timeout. The congestion window trace also reveals that timeouts are common with 2W.

With AE, the sender decreases its sending rate slightly before sending the fast retransmit. Normally each acknowledgment clocks out a segment, but there is no acknowledgment for the lost segment. With Limited Transmit enabled,

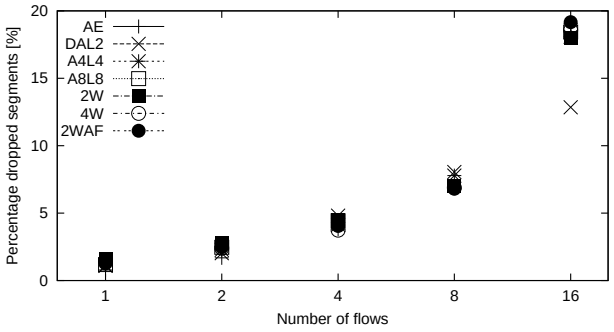


(a) Bottleneck of 1 Mbps.

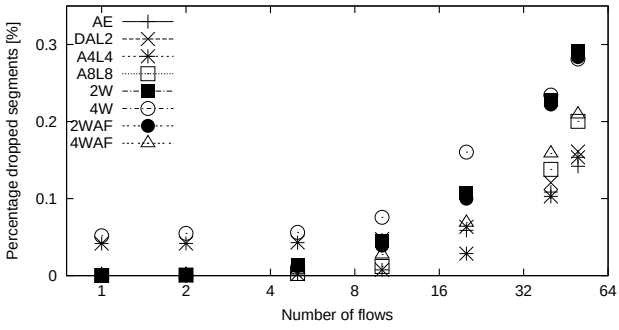


(b) Bottleneck of 100 Mbps.

Figure 6: Average throughput normalized by the expected throughput of a flow sharing the capacity fairly with the other flows.



(a) Bottleneck of 1 Mbps.



(b) Bottleneck of 100 Mbps.

Figure 7: The percentage of segments dropped.

new segments are sent in response to the duplicate acknowledgments until the duplicate acknowledgment threshold is reached and the fast retransmit performed.

With 2W, the sender often increases its sending rate before sending the fast retransmit as follows:

- When there are no losses, segments are sent back-to-back upon the receipt of an acknowledgment. In congestion avoidance, the time interval between the acknowledgments increases slowly as buffers build up.
- When the receiver detects a gap in the receive sequence, it sends a cumulative acknowledgment for the segment prior to the gap. This segment would normally not be acknowledged until later, after the arrival of the whole micro burst.
- In response to the early cumulative acknowledgment, the sender releases a number of segments. These segments would have been sent later if there had been no loss.
- The fast retransmit is then sent in response to the duplicate acknowledgments that follow, right after the burst sent out in response to the cumulative acknowledgment.

The sending rate increase is the reason that a large number of segments are lost, instead of only one segment as is normal in congestion avoidance.

We solve this problem by checking whether a cumulative acknowledgment has a SACK option attached when we are not in loss recovery. If so, we do not send any new segments. We also postpone the fast retransmit until the time when the next acknowledgment is expected. To estimate this time, we measure the interval between acknowledgments. Limited transmit is enabled as usual though. The changes are called the acknowledgment fixes (AF) and hence 2W with AF is denoted 2WAF.

The drop rates stay the same with AF, but the utilization is increased from 73% to 86% when 2W is used for one flow in the 1 Mbps case. When the bottleneck is 100 Mbps, utilization goes from 79% to 89%. Enabling AF for 4W when the bottleneck is 100 Mbps, improves utilization from 88% to 94% when there is one flow. We also enabled AF for A8L8 and the smaller bottleneck, which only led to a small improvement. There is another limitation present.

4.4 Buffer Requirement

The increased burstiness when sending acknowledgments only rarely puts larger demands on buffering in the routers. Figure 8 shows the throughput as a function of the buffer size for the 1 Mbps bottleneck. AE has the smallest buffer size requirement and 2W the largest. Although not shown, A4L4 has similar requirements to 4W and the buffer requirements of A8L8 is best approximated by 2WAF.

AE increases its sending rate by one segment per RTT. When a loss is detected through three duplicate acknowledgments, the sending rate is halved. To be able to utilize the capacity, the minimum send window needs to be one BDP. Consequently, the send window should be two BDPs before a loss event and the buffer requirement is one BDP.

2W requires a twice as large window as AE to fully utilize the capacity. Acknowledging more frequently rapidly decreases the buffer requirement, but increases the network

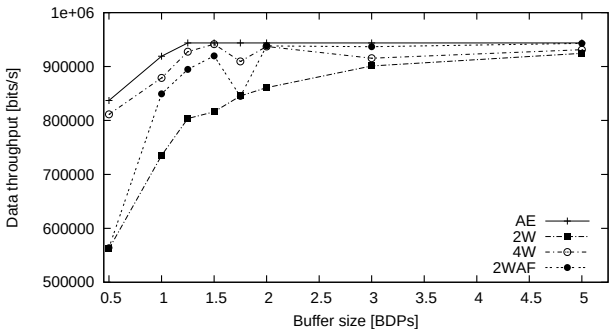


Figure 8: The throughput as a function of the buffer size for different acknowledgment strategies and a bottleneck of 1 Mbps. The throughput is computed over a simulation of 50 s.

traffic. The relation between the number of acknowledgments, A_{num}, sent per send window and the minimum send window, W_{min}, is roughly given by Equation 1 for one flow.

$$W_{min} = A_{num} \frac{1}{A_{num} - 1} BDP \quad (1)$$

The buffer requirement depends on the minimum send window, but also the bandwidths of the preceding links relative to the the bandwidth of the bottleneck. The link speeds determine how closely the segments that are sent back-to-back will arrive to the bottleneck buffer.

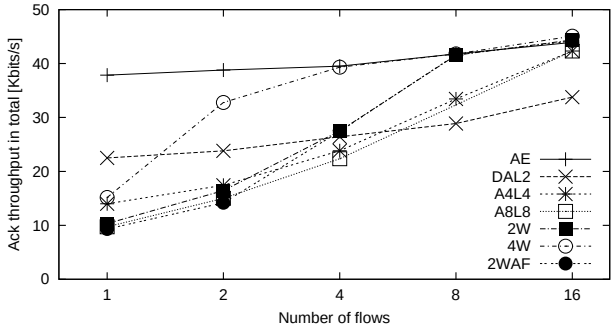
If segments are paced out instead of sent in a burst, the buffer requirement is reduced. But pacing works by delaying segments to smooth the sending rate, which would add another delay component in addition to the byte counting delay. We would therefore like another solution.

From the analysis of 2W, we know that a cycle of the protocol takes two RTTs. This implies that 2W has lower aggressiveness than AE. Therefore, it is possible to reduce the responsiveness while fulfilling the condition for TCP-friendliness for AIMD presented in [19]. We repeat this condition here, where α is the increase and β the decrease parameter.

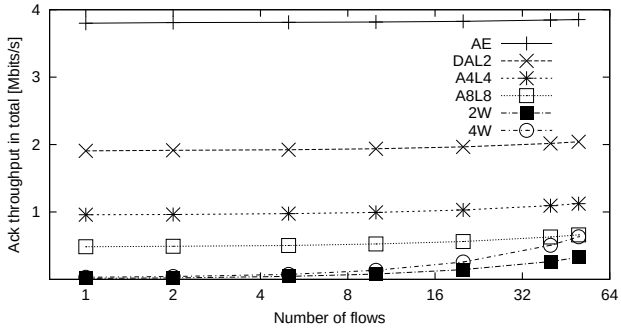
$$\alpha = \frac{3\beta}{(2 - \beta)} \quad (2)$$

Solving for $\alpha = 1/2$ gives $\beta = 2/7$. We use $\beta = 2/7$ after the initial slow start period with 2W, which improves utilization in the one flow case from from 86% to 92% in the 1 Mbps scenario and from 89% to 94% in the 100 Mbps scenario. For two flows, the utilization decreases. The relatively small send window reduction makes the second flow suffer a loss close to the first flow, reducing performance. For higher degrees of multiplexing, the performance is not affected. This change is specific to 2W and difficult to apply to varying acknowledgment frequencies or when there is no fixed relation between the acknowledgment frequency and the send window size.

A4L4 and A8L8 utilize the capacity as well as AE when the bottleneck is 100 Mbps, but not when it is 1 Mbps. The larger the send window, the more acknowledgments are sent per send window with A4L4 and A8L8. The minimum send window requirement for full utilization is thereby reduced and so is the size of the micro bursts relative the buffer size.

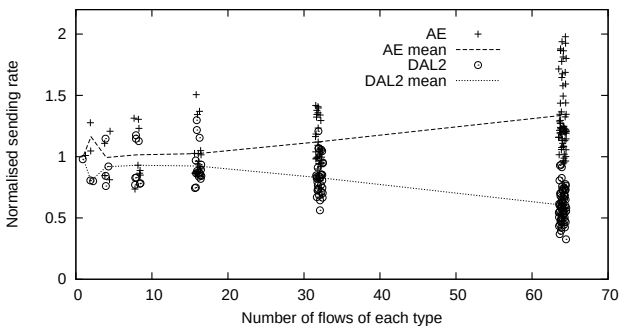


(a) Bottleneck of 1 Mbps.

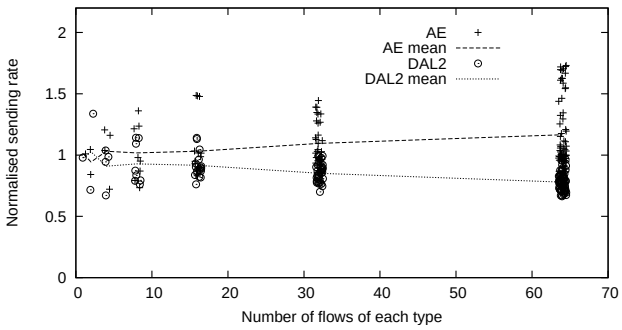


(b) Bottleneck of 100 Mbps.

Figure 9: The total bit rate generated by the acknowledgment traffic for all flows during a simulation.



(a) DAL2, delayed acknowledgment interval of 1000 ms.



(b) DAL2, delayed acknowledgment interval of 100 ms.

Figure 10: Fairness between AE and DAL2 with drop-tail buffer management and a 15 Mbps bottleneck.

On the other hand, 2W generates the same number of acknowledgments per send window and therefore has similar performance regardless of the bottleneck bandwidth. We adapt β to reduce the buffer requirement of 2W for a single flow, this approach is not easily applied to other acknowledgment strategies. A simpler approach is to acknowledge data twice as often (4W), which is sufficient to drastically reduce the buffer requirement.

4.5 Acknowledgment Throughput

The acknowledgment bit rates during the simulations are shown in Figure 9. The send window for each flow decreases when the number of flows increases. When the lower bound for the acknowledgment frequency is encountered, the total acknowledgment bit rate starts to increase. 4W has the highest lower bound and therefore the increase comes when there are fewer active flows for this strategy than for 2W. The simulated results are as expected.

4.6 Fairness

A TCP connection applying a low acknowledgment frequency also has to be able to compete for bandwidth with other TCP connections sending frequent acknowledgments.

In this scenario, half the flows uses AE and the other half uses DAL2, A4L4, A8L8, 2W, or 4W. We used a dumbbell topology with an RTT of approximately 50 ms. There were four reverse flows with the receiver buffer space limited to twenty segments. We studied the performance both with RED and drop-tail buffer management for bottleneck bandwidths of 8 and 15 Mbps.

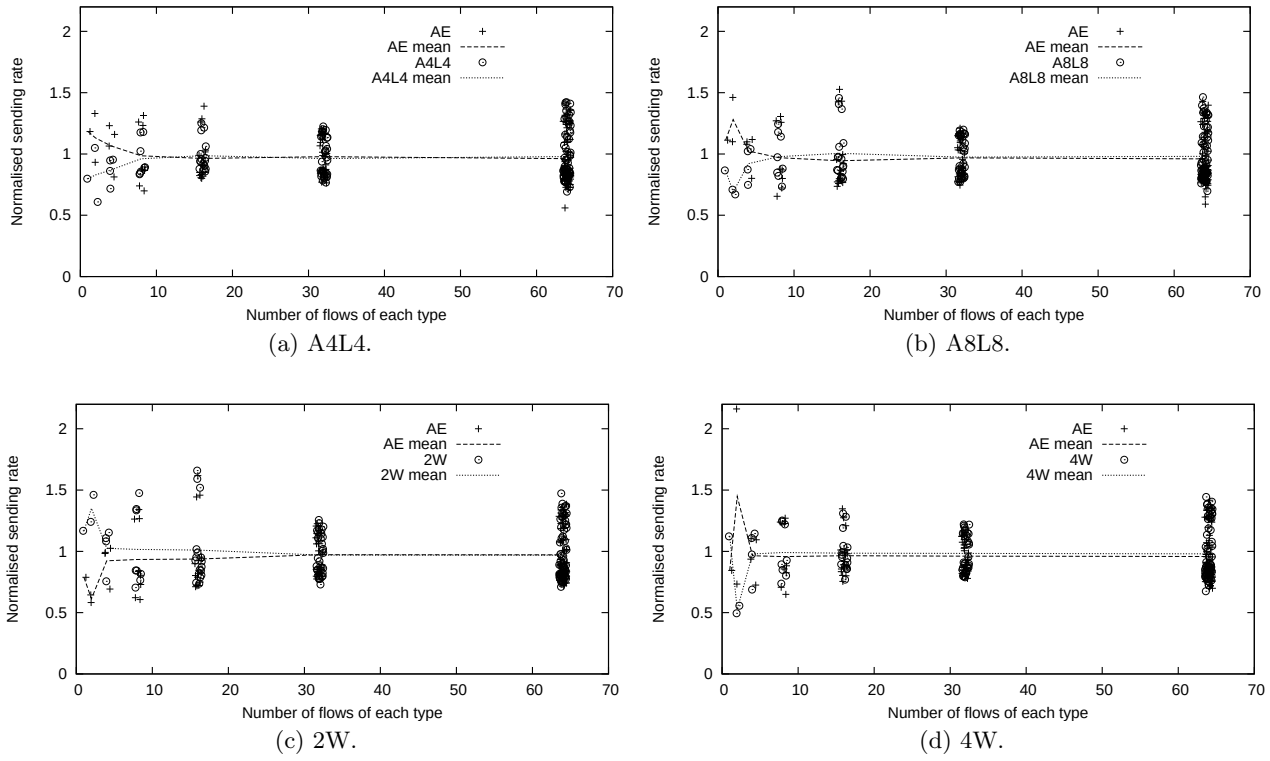
RED was configured to have an upper threshold at 1.25BDP and a lower threshold at 0.25BDP. The maximum drop probability between the thresholds is 10% and gentle mode is enabled. The drop-tail buffer size was set to one BDP.

Figures 10 and 11 show the results for drop-tail buffer management and a bottleneck of 15 Mbps. The results are similar for 8 Mbps. When multiplexing is low, the results exhibit more variations because there are fewer flows on which to base the computations. Each marker represents a flow and the graphs are the average throughput of the flows of the same type. All acknowledgment schemes, but DAL2, are able to take their fair share of the bandwidth.

DAL2 gets less than its fair share of the bandwidth, especially when the loss rates are high, see Figure 10(a). At high loss rates, timeouts are more frequent. After a timeout the send window is reduced to one segment. With DAL2, this segment will not be immediately acknowledged. If the delayed acknowledgment interval is too large, consecutive timeouts are likely. We therefore repeated the simulation for DAL2 with the delayed acknowledgment interval set to 100 ms. The difference in performance at higher loss rates is reduced as shown in Figure 10(b), but DAL2 still has a lower average throughput than AE.

When RED is used the spread is smaller, especially at low levels of multiplexing. As with drop-tail, fairness is achieved with all acknowledgment strategies but DAL2.

The number of segments lost as a function of the number of flows is shown in Figure 12. At higher loss rates, DAL2 yields a lower loss rate than the other acknowledgment strategies because it is unable to compete for the band-



(a) A4L4. (b) A8L8.

(c) 2W. (d) 4W.

Figure 11: Fairness between AE and other acknowledgment strategies with drop-tail buffer management and a 15 Mbps bottleneck.

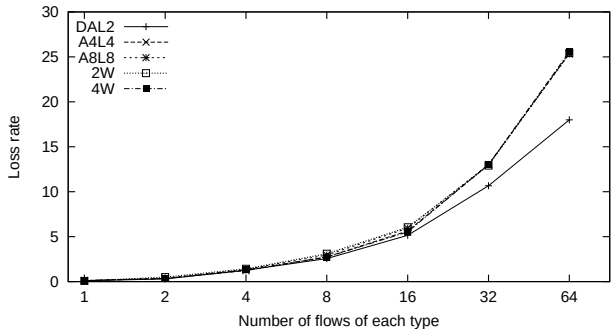


Figure 12: The loss rates for a bottleneck of 15 Mbps bottleneck with drop-tail buffer management.

width. Thus, there are no indications that the increased micro burstiness when the acknowledgment frequency is low leads to more segments being lost.

4.7 Pacing

To reduce the micro burstiness of a TCP flow with a low acknowledgment frequency, pacing is attractive. Pacing can be completely rate-based or window controlled. When window controlled, the sender is not allowed to have more segments in flight than currently allowed by the send window. An advantage of dropping this requirement is that the significance of a single acknowledgment would be smaller. On the other hand, a mechanism for reducing the sending rate in the absence of acknowledgments becomes necessary. The acknowledgments may be lost due to congestion or no segments may arrive to trigger acknowledgments.

Pacing breaks up the relationship between the acknowledgment frequency and the micro burst size, thereby reducing the buffer requirement. It is difficult to predict how pacing will interact with SACK-based loss recovery though.

The main drawbacks of pacing are that it introduces a delay component because segments are delayed to smooth the sending rate. Increasing the sending rate slightly more aggressively than in ack-clocked TCP could improve performance, but this approach requires further studies. Pacing has also been shown to get less than its fair share of the bandwidth when competing against a reference TCP version [1].

We believe that the byte counting delay, in addition to the pacing delay, will be significant and reduce the possibility for a TCP with a low acknowledgment frequency to compete with AE traffic. Reducing the acknowledgment frequency results in larger bursts, whereas pacing smooths the sending rate compared to reference TCP. The data sending pattern will therefore be different than for AE in either case, which may lead to fairness problems.

Our results indicate that pacing may not be necessary. By acknowledging data four times per send window, micro burstiness is kept at a level that only slightly increases the router buffer requirement. It remains to be investigated how delay sensitive flows can co-exist with a burstier TCP. So far, we have not included pacing in our proposal.

5. RELATED WORK

In this section, we will discuss the related work focused on asymmetric and wireless networks, as well as congestion control of acknowledgments.

In asymmetric networks with one-way data traffic, acknowledgments are dropped if the bit rate requirement exceeds the capacity of the reverse path. If TCP keeps the acknowledgment traffic to a minimum, this situation is less likely to arise. In [11], acknowledgments are filtered out at the buffer prior to the constrained link. The disadvantages of this approach are that each packet needs to be inspected and the filtering decision depends on the current content of the buffer. This makes packet management in the network more complex and requires more processing time.

Two-way data traffic can aggravate the asymmetry and lead to dropped acknowledgments also in symmetric networks. Keeping the acknowledgment traffic to a minimum can reduce the load, but greedy applications grab as much bandwidth as possible and therefore acknowledgments may still be lost. In addition to dropped acknowledgments, interactions between segments and acknowledgments lead to ack compression [30] and fairness problems [11, 22, 24]. To improve utilization of the faster forward channel, [11] suggests giving acknowledgments priority over segments in the reverse channel. This improves utilization of the fast forward channel, but may starve the slow connection. The data traffic over the slow link can be guaranteed a minimum throughput using a connection-level bandwidth allocation mechanism [22]. In [24], each user is served in a weighted round robin fashion to improve fairness between several connections. Thus, reducing the acknowledgment traffic can not solve all the problems associated with two-way data traffic and a solution involving both a reduction of the acknowledgment traffic and scheduling is most likely required.

If we want to reduce the acknowledgment traffic only when the reverse path is congested, we need to be able to detect when congestion occurs. In [11], RED [20] coupled with ECN [28] is used to mark acknowledgments during periods of congestion. Congestion control is then performed on the acknowledgments. Whenever a marked acknowledgment is detected the interval between acknowledgments is doubled. The interval is reduced by one segment at a time similar to the congestion control algorithm for TCP data traffic. The minimum acknowledgment frequency is set to one acknowledgment per send window and the receiver is explicitly informed of the current send window through an option in the TCP header. However, it can not be assumed that all nodes along the path support RED and ECN, and therefore lost acknowledgments will also have to be considered signs of congestion, which increases the error sensitivity of TCP.

In the TCP-like profile for the Datagram Congestion Control Protocol (DCCP) [21], congestion control of the acknowledgment traffic is mandatory and work in the IETF on TCP acknowledgment congestion control has been proposed [18]. DCCP uses a minimum acknowledgment frequency of two acknowledgments per send window when the window is larger than one, which is similar to the minimum acknowledgment frequency considered here.

Byte counting is used to compensate for a lower acknowledgment ratio in DCCP and we expect a similar approach for TCP. The changes we have made to ABC are therefore highly relevant to the work on congestion control for acknowledgments. However, if we use a low acknowledgment frequency, like 4W, acknowledgment congestion control is unlikely to be of interest.

In DCCP, the sender informs the receiver of which acknowledgment frequency to use. If a segment is lost, the receiver will acknowledge the next segment that arrives and therefore this acknowledgment strategy is relatively robust. At present, TCP does not have any options that allow the sender to control the acknowledgment frequency. Different implementations are possible, such as the sender polling the receiver for acknowledgments. To improve robustness, the sender can poll the receiver for acknowledgments using a counter that decreases for each segment. The receiver would then be able to detect when an acknowledgment should be sent even if the segment requesting an acknowledgment is lost.

The minimum acknowledgment frequency also depends on the nature of the links along the path. We have assumed that the links are full-duplex and that there is essentially no interference between transmissions over different links. In [29] it is shown that acknowledging data only once per send window is desirable for certain topologies in ad hoc networks. The explanation is that data and acknowledgments in different directions contend for the same medium.

6. CONCLUSIONS

The TCP end-points know little about the constraints on the path the data travel, which makes it difficult to determine when the gain of sending fewer acknowledgments would be significant. Therefore, a generally applicable solution, that does not degrade performance when the acknowledgment traffic is not necessarily a constraint, is attractive.

The sender's knowledge of the current send window and congestion control phase makes it better equipped than the receiver to determine when segments should be acknowledged. Also, the sender needs to be modified to compensate for a lower acknowledgment frequency, which makes it necessary to introduce changes at both end-points even if the responsibility for determining the acknowledgment frequency is placed on the receiver.

Byte counting is used to compensate for a lower acknowledgment frequency. We show that the limitation on the amount of data that may be counted for each acknowledgment can be set to correspond to the acknowledgment frequency and thus can be higher than two maximum sized segments. Currently the two maximum sized segments is the recommended upper bound [3]. A low acknowledgment frequency, and consequently a high byte counting limit, requires modifying the byte counting behavior following a timeout.

We show that acknowledging a fixed number of segments each time suffers from several drawbacks. When the send window is small, a lower bound is needed to avoid timeouts. The throughput performance also varies with the available capacity. By acknowledging data a fixed number of times per send window, we can ensure that timeouts are not caused by a small send window and the performance is more predictable.

Acknowledging data once per send window is enough to avoid timeouts, but does not ensure good utilization. In theory, acknowledging data twice per send window should be sufficient to also ensure utilization, but the router buffer demand in low multiplexing environments makes it necessary

to acknowledge data more often. Doubling the acknowledgment frequency to four times per send window reduces the router buffer requirement substantially.

Fast recovery is often unsuccessful when the acknowledgment frequency is low. The reason for this is a temporary increase in the sending rate when the acknowledgment pattern is changed because of a segment loss. We present a solution to this problem, which reduces the number of timeouts.

Our results support the view that a lower acknowledgment frequency than provided by delayed acknowledgments is possible today with the use of byte counting. In particular, acknowledging data four times per send window is sufficient to preserve throughput performance, but changes both to byte counting and loss recovery are necessary.

We plan to continue investigating the robustness of different acknowledgment strategies in the presence of transmission errors and heavier two-way traffic. We are also interested in investigating the fairness of flows with different RTTs.

7. REFERENCES

[1] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In *Proc. IEEE INFOCOM*, volume 2, pages 1157–1165, Mar. 2000.

[2] M. Allman. On the Generation and Use of TCP Acknowledgments. *ACM SIGCOMM Computer Communication Review*, 28(3):4–21, Oct. 1998.

[3] M. Allman. TCP Congestion Control with Appropriate Byte Counting. Experimental RFC 3465, IETF, Feb. 2003.

[4] M. Allman, H. Balakrishnan, and S. Floyd. Enhancing TCP's Loss Recovery Using Limited Transmit. Standars Track RFC 3042, IETF, Jan. 2001.

[5] M. Allman and E. Blanton. Notes on Burst Mitigation for Transport Protocols. *ACM Computer Communication Review*, 35(2):53–60, Apr. 2005.

[6] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window. Standards Track RFC 3390, IETF, Oct. 2002.

[7] M. Allman and V. Paxson. On Estimating End-to-End Network Path Properties. In *Proc. ACM SIGCOMM*, pages 263–274, Sep. 1999.

[8] M. Allman and V. Paxson. TCP Congestion Control. Standards Track RFC 2581, IETF, Apr. 1999.

[9] E. Altman and T. Jimnez. Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks. In *Proc. PWC*, Sep. 2003.

[10] A. C. Aug, J. L. Magnet, and J. P. Aspas. Window Prediction Mechanism for Improving TCP in Wireless Asymmetric Links. In *IEEE GLOBECOM*, volume 1, pages 533–538, Nov. 1998.

[11] H. Balakrishnan, S. Seshan, and R. Katz. Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks. *ACM Wireless Networks*, 1(4):469–481, Dec. 1995.

[12] E. Blanton and M. Allman. On the Impact of Bursting on TCP Performance. In *Proc. PAM*, Mar. 2005.

[13] E. Blanton, M. Allman, K. Fall, and L. Wang. A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP. Standards Track RFC 3517, IETF, Apr. 2003.

[14] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC Standard 1122, IETF, Oct. 1989.

[15] D. D. Clark. Window and Acknowledgement strategy in TCP. RFC 813, IETF, Jul. 1982.

[16] R. de Oliveira and T. Braun. A Smart TCP Acknoledgement Approach for Multihop Wireless Networks. *IEEE Transactions on Mobile Computing*, 6(2), 2007.

[17] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *ACM SIGCOMM Computer Communication Review*, 26(3):5–21, Jul. 1996.

[18] S. Floyd. http://www.icir.org/floyd/papers/draft-floyd-tcpm-ackcc-00a.txt, Nov. 2006.

[19] S. Floyd, M. Handley, and J. Padhye. A Comparison of Equation-based and AIMD Congestion Control. Technical report, ICIR, May 2000.

[20] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, Aug. 1993.

[21] S. Floyd and E. Kohler. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. Standards Track RFC 4341, IETF, Mar. 2006.

[22] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan. Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions. In *Proc. ACM SIGMETRICS*, pages 78–89, Jun. 1998.

[23] L. Kleinrock. *Queueing Theory*. Wiley, 1975.

[24] T. Lakshman, U. Madhow, and B. Suter. Window-based Error Recovery and Flow Control with a Slow Acknowledgement Channel: a Study of TCP/IP Performance. In *Proc. IEEE INFOCOM*, volume 3, pages 1199–1209, Apr. 1997.

[25] W. Lilakiatsakun and A. Seneviratne. TCP Performance over Wireless Link Deploying Delayed ACK. In *Proc. VTC*, pages 1715–1719, Apr. 2003.

[26] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options. Standards Track RFC 2018, IETF, Oct. 1996.

[27] I. T. Ming-Chit, D. Jinsong, and W. Wang. Improving TCP Performance Over Asymmetric Networks. *ACM SIGCOMM Computer Communication Review*, 30(3):45–54, Jul. 2000.

[28] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. Standards Track RFC 3168, IETF, Sep. 2001.

[29] A. K. Singh and K. Kankipati. TCP-ADA: TCP with Adaptive Delayed Acknowledgement for Mobile Ad Hoc Networks. In *Proc. WCNC*, volume 3, pages 1685–1690, Mar. 2004.

[30] L. Zhang, S. Shenker, and D. D. Clark. Observations on the Dynamics of a Congestion Control Algorithm: the Effects of Two-way Traffic. *ACM SIGCOMM Computer Communication Review*, 21(4):133–147, Sep. 1991.