Revisiting Wireless Link Layers and In-order Delivery

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

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

Abstract—Wireless link layers, which perform retransmissions to hide transmission errors from upper layers, enforce in-order delivery to avoid triggering TCP’s congestion control mechanisms. However, new reordering robust TCP flavors make it possible to revisit the design of these link layers.

In this paper, we study the effects of the link layer configuration (in-order vs out-of-order) on network layer buffering and transport layer smoothness in a WWAN scenario through simulations. We use a standards-compliant TCP, TCP-Aix, and TCP-NCR. The results show that smoothness is improved and the buffer requirement is reduced when out-of-order delivery is allowed.

I. INTRODUCTION

The poor performance of TCP in the presence of reordering has lead to an informal constraint on network components to deliver packets in-order. However, recently a number of reordering robust TCP flavors have been proposed [1], [2], [3], [4], [5]. Reordering is also likely to become more common as parallelism and mobility in the network increase; creating an incitement to deploy reordering robust TCP flavors. This in turn creates an opportunity to reconsider the design of network elements that naturally cause reordering.

In this study, we focus on Wireless Wide Area Networks (WWANs) that due to capacity and cost considerations have been designed with relatively error prone links. To hide the transmission errors from TCP, which otherwise falsely would assume congestion and reduce its sending rate, the link layer performs retransmissions. It then normally re-establishes the incoming packet sequence before delivering packets to the upper layer.

The main drawbacks of in-order delivery are increased link layer complexity and larger delay variations. Also, the TCP acknowledgment clock may be disrupted causing large bursts of segments to be sent at a time. These bursts have a negative effect on network buffer performance and co-existing delay sensitive traffic.

When the traffic is bursty, buffers must be dimensioned both for the case when the packet arrival rate is relatively smooth and when it exhibits large variations. If the buffers cannot accommodate large bursts of packets TCP performance suffers, because TCP has problems to recover efficiently when multiple packets are dropped from a window of data. On the other hand, large buffers give a late indication to congestion in the network and add delay.

In [6], it was shown that the buffer requirements are smaller when the sending rate of each source is smoothed. They identified bursts in individual flows and used a form of rate pacing to smooth the sending rate.

II. REORDERING ROBUST TCP FLAVORS

We chose TCP-Aix [4] and TCP Non-congestion Robustness (TCP-NCR) [7] to explore possible benefits of allowing out-of-order delivery at the link layer. For reference, we included a standards-compliant TCP with Limited Transmit [8], SACK-based Loss Recovery [9], and Congestion Window Validation [10].

When a segment is reordered, the receiver generates duplicate acknowledgments (dupacks). Upon receiving a certain number, e.g., dupthresh, dupacks, a standards-compliant TCP initiates loss recovery and reduces its congestion window (cwnd).

In TCP-Aix, dupthresh dupacks immediately trigger loss recovery, but the decision whether to reduce cwnd is postponed until a more reliable decision can be made. TCP-Aix waits for the acknowledgment of the segment, which was indicated as lost. We refer to this acknowledgment as the newACK. If the newACK was sent in response to the retransmission, the TCP-Aix sender reduces its cwnd.

TCP-NCR sets dupthresh to correspond to approximately one rtt when the first dupack is received. After receiving dupthresh dupacks, TCP-NCR initiates loss recovery and reduces cwnd. Both, TCP-Aix and TCP-NCR can thus distinguish reordering events when the segment is reordered by less than one rtt.

TCP-Aix can be used with a higher dupthresh setting to allow even longer reordering events to be discerned. The winthresh algorithm [4] computes a suitable dupthresh with regards to the available buffer space at both the sender and the receiver. This dupthresh can be configured with an upper bound. In this study, we applied an upper bound of two send windows.

Both TCP-Aix and TCP-NCR sends new segments in response to each dupack received until congestion control is initiated. This helps to keep the TCP acknowledgment clock alive and to sustain a higher dupthresh.

III. SIMULATION ENVIRONMENT

We used simulations to explore a number of scenarios in which allowing reordering may have a positive effect on TCP smoothness and buffer management. The simulations have been performed in the Network Simulator version 2.27, ns-2.27 [11]. We have implemented the TCP-Aix agent and modified the tcp-sackl and tcp-der agents to follow [9], [12], [8] and [10]. The tcp-dcr agent was retrieved from [13]. Details concerning the changes to the code can be found in [4].
In ns-2.27, Limited transmit [8] is enabled by default. The minimum RTO is 1 s, but we used 200 ms as is the default value in newer ns versions. Furthermore, the variables timerfix_ts, timestamps, ts_reset_RTO, and control_increase were all enabled. Throughout the simulations, the segment size was 1000 bytes, the advertised window unlimited and the delayed acknowledgment algorithm disabled.

The error module from [3] introduced reordering. It models a link layer protocol that repeats transmission until it is successful. Between each attempt a link round trip time must pass. The error module has been extended to have the same error probability for all transmission attempts; first time and retransmissions, over the link. We have also implemented an error module which re-establishes the incoming packet order after the reordering event. It models a link layer protocol that supports in-order delivery. The error rate applies to the segments, thus if the error rate is 10% approximately one out of 10 segments is retransmitted.

Fig. 1 shows our simulation topology. At node N1 the users share the buffer space, whereas data is buffered for each user individually at node N2. When Random Early Detection (RED) [14] gateways are used, the minimum and maximum thresholds are set to one tenth and half the buffer size and the instantaneous buffer size is used.

IV. SIMULATION RESULTS

In this section we present selected results from the simulation study.

A. Smoothness

The sending patterns of standard TCP with in-order delivery and TCP-Aix with out-of-order delivery are shown in Fig. 2. There are almost like two graphs in each figure. The upper graph is the segments, whereas the other graph represent the acknowledgments. For TCP-Aix with out-of-order delivery the graphs are clearer because segments are sent more evenly. As segments and acknowledged arrivals arrive close together periods of inactivity separate groups of arriving packets as in the figure for standard TCP.

The longer the reordering length, the larger the bursts in the in-order delivery scenario. The explanation is that the link layer holds on to successfully transmitted packets until they can be delivered in-order. When the packets are forwarded to upper layers after a reordering event, the TCP receiver generates closely spaced acknowledgments that in turn trigger bursts of segments. In out-of-order delivery, TCP-Aix can keep the acknowledgment clock alive during the reordering event by releasing a new segment on the receipt of each dupack. This prevents large bursts from being released when a reordering event is resolved. In this scenario, the segments were reordered between the sender and the receiver, i.e., in the forward path.

If instead reordering occurs in the reverse path, between the receiver and the sender, an acknowledgment may either acknowledge a number of segments or it may bring no new information. We call an acknowledgment that arrives with no new information a late acknowledgment. From the sender’s point of view, a late acknowledgment is the same as a lost acknowledgment because late acknowledgments are usually disregarded.

TCP-Aix and TCP-NCR would also produce bursts if fewer acknowledgment arrived in-order. A solution is to let late acknowledgments clock out new segments and limit the number of segments that may be released by a single acknowledgment to make the flow smoother as suggested in [15]. Byte counting [16] may be useful to compensate for the decreased number of acknowledgments that arrive in-order. With byte counting the number of bytes acknowledged and not the number of acknowledgments are counted.

A number of metrics have been proposed to quantify smoothness and burst characteristics. We chose the Coefficient of Variation (CoV) that can measure smoothness over several time scales. The CoV of a flow can be computed by dividing the simulation into time intervals of length δ and considering the sending rate in each time interval a sample. CoV in the time scale δ is then the standard deviation of these samples divided by the mean. A low CoV means that the flow is sending data at a steady rate.

With in-order delivery in both directions it does not matter if it is the acknowledgments or the segments that are reordered; the bursts are of the same magnitude as the reordering length. When we have out-of-order delivery on the forward path,
we have shown that bursts can be avoided. If instead some acknowledgments arrive late, there will be small bursts corresponding to the number of consecutive acknowledgments that suffer from delay. That is, the bursts would be much smaller than they are with in-order delivery.

We computed the CoV of each flow during the last 150 seconds of a 200 seconds long simulation. For each type of flow, ten repetitions were conducted. The average CoV that is shown in Fig. 3 with confidence intervals for 95%. When there are no errors, the standards-compliant TCP, TCP-NCR, and TCP-Aix all yield a CoV below 0.1 at all time scales, see Fig. 3(a).

When there are transmission errors, the shape of the curves for in-order delivery is different from the shape of the curves that out-of-order delivery generates. It is primarily at the shortest time scale, 50 ms, that in-order delivery causes larger burstiness than out-of-order. A retransmission takes 40 ms and thus the idle periods are in the range of the shortest time scale.

As expected, standard TCP does not perform well with out-of-order delivery. It can not utilize the capacity, because it can not separate between reordering and congestion events, which also results in a bursty sending pattern. TCP-NCR is able to identify most of the reordering events when the error rate is 3%, but fails to detect all reordering events when it is 10%. At 10%, more than one link layer retransmission is often required, which leads to reordering durations longer than one rtt. Similar to standard TCP, the smoothness of TCP-NCR decreases when reordering events are mistaken for congestion. TCP-NCR still has a higher throughput than standard TCP, which yields even larger fluctuations in the sending rate. TCP-Aix has approximately the same CoV for 0, 3 and 10% error rate with out-of-order delivery, because it is capable of separating between congestion and reordering at all the investigated error rates.

We conclude that out-of-order delivery improves smoothness over short time scales compared to in-order delivery, as long as the TCP flavor in question can cope with the reordering durations. The reordering robust TCP flavors used in this study both send a new segment in response to each dupack they receive, at least until they perform a fast retransmit. Thereby the acknowledgment clock can be kept alive with out-of-order delivery.

B. Buffering requirements

For the following discussion of the required drop-tail buffer size we assume that there is a single transfer, a fixed network capacity, and no reordered acknowledgments.

To fully utilize the bottleneck link, TCP must have a cwnd that is as large as the bdp. TCP continuously probes for bandwidth and reduces its cwnd by half when encountering a loss. For cwnd to stay in parity to bdp, losses should occur when cwnd is about two bdp. When cwnd is larger than bdp, the transport layer rtt increases due to queuing delay. The segments that cannot be transmitted during the link layer rtt builds up the rtt. When cwnd exceeds the bdp by the number of segments that the bottleneck buffer can hold, there will be a loss. For cwnd to be two bdp when this happens, the buffer should be able to hold one bdp of data.

With our simulation topology, where the rtt is 60 ms and the bottleneck capacity 10 Mbps, bdp is 75 segments. Our simulations confirm that increasing the size of the drop-tail buffer beyond 75 segments does not increase the throughput for any of the TCP flavors when there are no transmission errors.

When there are transmission errors, the buffer need is larger; not primarily because of the increased burstiness, but because of the increased bdp that in-order delivery gives rise to. When a frame is lost and has to be retransmitted the link layer buffer on the receiver side fills up with the frames that arrive after the gap. As a consequence of the segments not being forwarded to the TCP receiver until the missing data is recovered, no TCP acknowledgments are sent during the retransmission interval. After the retransmission the link layer releases all the buffered segments, assuming that there are no other losses. The sending rate has to be high enough to compensate for the additional delay that in-order delivery cause by holding on to the segments during the link layer retransmission. It is the experienced delay, which includes the retransmission delay, that should be used when computing bdp.

In our case the end-to-end delay is 60 ms and the link layer retransmission delay is 40 ms, which gives an experienced delay of 100 ms. Thus, a drop-tail buffer size of 75 + 50 segments should be enough when few segments are retransmitted by the link layer more than once. When the error rate is 10%, segments are often retransmitted more than once. To achieve the highest possible throughput the buffer needs to be increased to 75 + 50 + 50 segments.

With out-of-order delivery only the retransmitted segment suffers from a longer delay. The segments that are sent imme-
buffer requirement at the investigated error rates with out-of-order delivery. At 3% error rate, TCP-NCR has a lower buffer requirement than with in-order delivery.

The simulations also show that RED generally has worse performance than drop-tail when the buffer is dedicated to a single flow, Fig. 4(c). The simulation of each point has been repeated ten times with different seeds for RED. The confidence intervals are for 95%. In most cases they are too narrow to be distinguished.

V. CONCLUSIONS

Recently a number of reordering robust TCP flavors have been proposed. This makes it possible to design wireless link layers to allow out-of-order delivery without impairing TCP performance. The major benefit would be a simpler link layer.

We contrasted in-order and out-of-order delivery at the link layer and the interaction with TCP through simulations in ns-2. We found that out-of-order delivery can bring benefits such as improved smoothness especially at short time scales corresponding to the retransmission delay. A smooth sending rate gives a better estimate of the round trip time used in TCP congestion control and reduces network jitter also for other applications. The buffer requirement is also smaller when TCP-Aix or TCP-NCR is combined with out-of-order delivery, compared to in-order delivery with a standard TCP implementation. The benefits should be further evaluated for specific network technologies.

REFERENCES