Improving Retransmission Performance of IP-Based Transport Protocols

Stan McClellan
Ingram School of Engineering
Texas State University
San Marcos, TX USA
stan.mcclellan@txstate.edu

Wuxu Peng
Department of Computer Science
Texas State University
San Marcos, TX USA
wuxu@txstate.edu

***Abstract*—This paper analyzes the algorithm used for estimating retransmission timeouts in connection-oriented IP-based transport protocols, such as the Transmission Control Protocol (TCP) and the Stream Control Transmission Protocol (SCTP). The estimation algorithm uses historical values of the round-trip time to estimate future round-trip delays, and so creates a maximum waiting time before triggering retransmission attempts. The purpose of the analysis is to question / validate some of the fundamental assumptions used in the estimation algorithm. The conclusion of the analysis is that the algorithm is somewhat mismatched to the dynamics of the current Internet. Alternative algorithms are discussed, and potential modifications are presented. Impact of the suggested alternative algorithm on the well-known selective acknowledgement and fast retransmit mechanisms is discussed.**

***Keywords*-SCTP; retransmission timeout; round-trip time; RTT; RTO; selective acknowledgment; fast retransmits; Jacobson algorithm; Chebyshev approximation; parameter estimation; upper bound.**

I. Introduction

This paper looks at the timing aspect of IP-based transport protocols (TCP and SCTP) of the Internet. Although this topic has been studied extensively in the literature in the past, we believe that our paper has made valuable new contribution on it. The theme of this paper is that the timeout aspects of current TCP and SCTP protocols can no longer reflect the current infrastructure and traffic dynamics of today's Internet. Current and future research should work on new and practical timeout mechanisms that meet the need of changed/changing landscapes of Internet. The paper is an extended and enhanced version of our conference paper [1].

IP-based transport protocols such as the Transmission Control Protocol (TCP) [2] and the Stream Control Transmission Protocol (SCTP) [3], [4] estimate maximum round-trip times using data from prior successful transmissions. The purpose of this estimation process is to create a triggering mechanism for retransmission procedures when transmissions are lost or seriously delayed. Estimation of the maximum round-trip time is performed via the Jacobson algorithm [5], which is codified in several IETF RFC's, including RFC 6298 [6]. The Jacobson algorithm has an interesting basis in fundamental theory, but suffers from some performance issues due to a mismatch between the theory and the application area. Performance issues related to the Jacobson algorithm and other retransmission procedures have been noted and addressed in several alternative approaches, including [7]–[12]. The paper discusses the Jacobson algorithm, the theory which motivates it, and several alternative algorithms including a new approach which is a modified form of Jacobson. Section II describes the estimation process and its use in establishing timeouts for retransmission procedures. Related work in retransmission optimization and timeout boundaries is also summarized. In Section III, the parameters of the existing algorithm are analyzed, and in Section IV an alternative approach is presented based on similar theoretical concepts, and achieving improved results. Performance results based on implementation and simulation are summarized in Section V. In Section VI, the impact on SCTP selective acknowledgments (SACK) and fast retransmits (FRT) are discussed. Our simulation results showed that proposed approach has either minimal or manageable impact on both SACK and FRT. Section VII concludes the paper. Note that much of this work is presented in the context of SCTP, but is also applicable to TCP since the timeout estimation processes are identical.

II. Retransmission mechanisms

When an SCTP sender transmits a unit of data, called a *chunk*, it also initializes a *retransmission timer* with an estimated value of the round-trip time (RTT). The value of this timer is the *retransmission time-out* (RTO). When an acknowledgment arrives, the timer is cancelled. If the timer expires before an acknowledgment arrives, the chunk may be retransmitted. The value of RTO is calculated from observed / actual values of RTT using the *Jacobson Algorithm*, which is detailed in Section III. A too optimistic retransmission timer may expire prematurely, producing *spurious timeouts* and *spurious retransmissions*, reducing a connection's effective throughput. On the contrary, a retransmission timer that is too conservative may cause long idle times before lost packets are detected and retransmitted. This can also degrade performance [7]. So, the difficulty lies in finding an algorithm which has a solid theoretical basis, is not computationally expensive, and can predict RTT with enough

bias to minimize retransmission events and waiting time *simultaneously*.

Performance issues related to retransmission procedures, including alternatives to the Jacobson Algorithm, have been noted and addressed several times in the literature. Much work has been focused on late retransmission and other optimizations of the overall retransmission scenarios [13], [14]. Many authors approach this problem with a "holistic" or overall perspective on the retransmission procedures where RTT estimation contributes to triggering these procedures. Other authors specifically address the estimation of RTT, and propose completely new algorithms. However, the Jacobson Algorithm is deeply rooted in the fabric of connection-oriented IP transport protocols, and its basis in fundamental theory is well-established. In following subsections, we briefly summarize several important approaches to retransmission and RTO estimation.

A. Holistic approaches

Holistic or overall approaches to improving retransmission performance typically address the state machines surrounding the retransmission process, including the estimation algorithm which may be used to trigger these procedures. Examples of holistic approaches include *Early Fast Retransmit* (EFR) [11], *Early Retransmit* (ER) [12], and *Window-Based Retransmission* (WB-RTO) [15] as well as protocol-specific techniques such as *Thin Streams* [10], [11], [16].

Early Fast Retransmit (EFR) is an optional mechanism in FreeBSD which is active whenever the congestion window is larger than the number of unacknowledged packets, and packets remain to be sent. When the RTO timer expires and the entire congestion window is not used, EFR retransmits all packets that could have been acknowledged [11]. The *Early Retransmit* (ER) algorithm [12] suggests that a mechanism should be in place to recover lost segments when there are too few unacknowledged packets to trigger Fast Retransmit. The Early Retransmit algorithm reduces waiting time in four specific situations [11]. The *Window-Based Retransmission Timeout* (WB-RTO) [15] asserts that timeout mechanisms based solely on RTT estimates lead to unnecessary retransmissions and unfair resource allocation, and proposes to schedule flows on the basis of their contribution to congestion. *Thin Streams* [10], [11], [16] optimizes throughput for "thin streams" which are often used in control applications, and often depend on SCTP for transport. When stream characterization is accurate, throughput can be improved by adapting specific sections of the retransmission procedures to match flow characteristics.

B. Alternative estimation algorithms

Alternative estimation algorithms address specific performance issues which have been noted in the Jacobson Algorithm. These issues may be related to the over-estimated value, spurious behaviors for certain traffic characteristics, or inefficient bounding computations. In some cases, heuristic state-machine approaches are also included because of complexities associated with the retransmission process. Examples of alternative or modified RTT estimation algorithms include *Peak-Hopper* [8], *Eifel* [7], and *Weighted Recursive Median* (WRM) [9].

The *Eifel* approach [7] notes a particular style of erroneous performance in the Jacobson algorithm, and adapts the algorithm in several ways to compensate for this performance oddity. As a result, Eifel eliminates unnecessary retransmissions which can result from spurious RTO violations. Similar to Eifel, the *Peak-Hopper* algorithm [8] also observes that the Jacobson algorithm responds inappropriately to certain fluctuations in RTT. This behavior produces "spikes" in RTO values because the algorithm does not distinguish between positive and negative variations. The modification proposed in [8] reduces this effect for a wide range of cases, and the findings in [10] concur. However, this solution results in higher average RTO values than the RFC approach [6], which can be a problem [11]. The *Weighted Recursive Median* (WRM) algorithm [9] redefines RTT estimation from a signal processing standpoint. WRM is effective, but tends to be computationally expensive even in a recursive form, which is a problem for per-packet operations.

C. Other considerations

The remainder of this paper addresses the estimation process for the maximum RTT, or the value which establishes the RTO timer. When the RTO timer has expired, retransmission procedures commence, and may include various conditionally executed processes. We do not address those processes, or the overall retransmission procedure. In most cases, we use the Jacobson and Eifel estimation algorithms for comparison because they are widely accepted or implemented.

For reference, the performance of the Jacobson and Eifel algorithms are shown in Figure 1 along with the modified version of the Jacobson Algorithm which is discussed in detail in Section IV. The traces in the figure are all driven by a common RTT sequence created using our testbed of systems with modified networking stacks, as described in more detail in Section V. In the figure, the quiescent sections of the RTT sequence ($t < 170$ms, $260\text{ms} < t < 430$ms) have a fairly low mean, with similarly low standard deviation. This is typical of modern, high-speed networks. Also note that the RTT sequence has abrupt increases ($t \approx 170$ms, $t \approx 430$ms) followed by a stable period ($170\text{ms} < t < 260$ms), and then an abrupt decrease ($t \approx 260$ms). Note that Jacobson and Eifel both "overshoot" after the abrupt positive discontinuity in RTT. However, at the second discontinuity which is abrupt but negative, Eifel corrects downward, whereas Jacobson again corrects upward. This is a primary beneficial feature of the Eifel approach. Unfortunately, the tuning of the

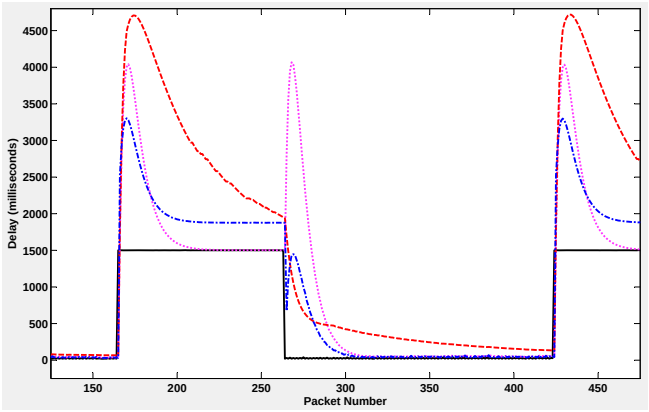

Figure 1. Relative performance of RTO estimators to large discontinuities in RTT. Estimators shown are the Jacobson Algorithm (dotted line), the Eifel Algorithm (dashed line), and the Modified Jacobson Algorithm (dash-dot line). The common RTT sequence driving all algorithms is shown as a solid line.

algorithm creates a larger deviation or "upper bound" for the RTT sequence than is available through the Jacobson Algorithm.

Abrupt changes in RTT (such as those shown in Figure 1) cause the RTO timer to expire, resulting in binary backoff of the timer (BEBO) and retransmission procedures. The RTO values associated with BEBO and retransmissions are not shown, because they do not affect future estimated values of RTO. Post-processing of the RTT estimate to enforce minimum values for RTO is also not considered here. In the RFCs for TCP and SCTP, a hard-coded minimum value for RTO is specified as 1000 milliseconds [2]–[4], [6], whereas the minimum value of the Eifel algorithm is defined as "RTT + 2 $\times$ ticks" (or 200 msec) [7]. The definition of the minimum RTO in TCP-Lite is typically 2 times the clock granularity, which is often taken as 500 msec [4], [7]. This minimum procedure is secondary to the estimation algorithm, and often completely replaces the RTT estimate. For example, in a broadband network with large minimum RTO, the RTT estimate can be orders of magnitude below the minimum, resulting in long wait-times for triggering retransmission. So, in all comparisons here we disable the enforcement of a minimum RTO and focus on the performance of the estimation algorithms.

In [17], a new algorithm to improve the SCTP's retransmission mechanism is proposed. The algorithm uses several fixed values of minimum RTO values, instead of a single minimum RTO value of 1000 msec. It is shown that the algorithm can improve SCTP's performance by as much as five percent.

III. The Jacobson Algorithm

The Jacobson Algorithm, originally proposed in [5], uses the Chebyshev Bound [18] to produce a reasonable value of RTO, or the maximum time the sender will wait for an acknowledgment. After exceeding RTO, a transmission is declared lost and retransmission procedures commence. Interestingly, Jacobson noted the poor performance of the algorithm, since loads higher than 30% resulted in retransmission of packets that were only delayed in transit (i.e., not lost) [5]. This behavior is also noted in later literature, including [7], [8], [16].

The specific computations for the bounding procedure are driven by estimates of the mean (μ) and the standard deviation (σ) of an assumed RTT distribution. The estimates of μ and σ are not conventional parameter estimates, as independent and identically distributed samples from a population. Rather, $\hat{\mu}$ and $\hat{\sigma}$ are *predicted* using prior RTT values. Jacobson estimates $\hat{\mu}$ and $\hat{\sigma}$ from observed values of RTT, and then computes an "overbound" for RTO using Chebyshev. This is the same as saying "we waited a reasonable time ($\hat{\mu}$) and *then* some ($K\hat{\sigma}$), but the ACK didn't come back, so the packet must have gotten lost". This RTO calculation is invoked by the sender for each unique transmission. As such, it is optimized for integer arithmetic and all coefficients are diadic.

Regardless of the specific assumptions or optimizations, Jacobson computes the RTO threshold as

$$x_{thr} = \hat{\mu}_{n+1|n} + K \cdot \hat{\sigma}_{n+1|n} \tag{1}$$

where x_{thr} is the RTO threshold or "overbound" for RTT, K is the number of "standard deviations past the mean," and $\hat{\mu}_{n+1|n}$ and $\hat{\sigma}_{n+1|n}$ are the estimates (predictions) of the mean and standard deviation of the RTT distribution for the next iteration (subscript $n+1$) given some data up to the current time (subscript n).

A. Jacobson and Chebyshev

The Chebyshev bound (2) is a universal bound applicable even for unknown distributions. Chebyshev shows that the probability of the random variable occurring outside a range around the mean (μ) depends on the standard deviation (σ).

$$\Pr\left[|X - \mu| \geq \epsilon\right] \leq \frac{\sigma^2}{\epsilon^2} \tag{2}$$

For RTT estimation, the Jacobson Algorithm uses a fixed offset from the mean ($K\sigma$) [3], [4]. This simplifies the bound and allows a direct computation of the "violation probability". With $\epsilon = K\sigma$ and $K = 4$ in (2),

$$\Pr\left[|X - \mu| \geq K\sigma\right] \leq \frac{1}{K^2} \tag{3}$$

and the fixed, double-sided "violation probability" is $\frac{1}{16} = 0.0625$. However, the RTO timeout is single-sided because the timeout algorithm is only concerned with the case where $X > \mu$. So for a symmetric distribution and $K = 4$, the RTO timeout will be exceeded roughly 3% of the time, with the assumption that the RTT values are reasonably independent, identically distributed.

B. Predicting μ and σ

Rather than using conventional parameter estimation algorithms requiring storage of historical values and more complex calculation, Jacobson estimates μ and σ using simple *prediction algorithms* as in (4). These algorithms rely on current values of the quantities (subscript n) as well as the measured RTT value (x_n). Interestingly, the use of *filtered* (predicted) values for μ and σ implicitly contradicts the assumption of a valid distribution in (2). However, this approach gains computational efficiency and reflects the nonstationary nature of RTT values.

$$\begin{aligned} \hat{\mu}_{n+1|n} &= \hat{\mu}_n + \alpha(x_n - \hat{\mu}_n) \text{ and} \\ \hat{\sigma}_{n+1|n} &= \hat{\sigma}_n + \beta\left(|\hat{\mu}_n - x_n| - \hat{\sigma}_n\right). \end{aligned} \tag{4}$$

When viewed as a time-series prediction or filtering algorithm, it is clear that the relations in (4) are single-pole, lowpass IIR (infinite impulse response) filters (predictors).

C. Selective Acknowledgment

Selective acknowledgment (SACK) was initially introduced in RFC 1072 [19], [20] to handle multiple dropped packets within a window [21]. This feature was omitted in RFC 1323 [22] as the authors felt the details of SACK was still to be worked out. SACK was reintroduced in RFC 2018 [23] and is now an important performance enhancement mechanism in TCP initially and now SCTP as well.

The usefulness and effectiveness of SACK mechanism have been extensively studied. In [24], it was found that "SACK improves TCP throughput significantly in moderate congestion (with a packet loss rate between 2 and 4%), and that the negative impact of SACK on competing non-SACK TCP connections is small".

SACK was proposed to enhance TCP performance over *long delay paths* [19], [23]. We observe that the RTT over the Internet has been steadly decreasing over the last thirty years. This decrease is primarily due to the improvement of Internet infrastructure/software that shortens the store/forward delays and transmission delays.

D. Fast Retransmit

TCP fast retransmit (FRT) and fast recovery algorithm, originally formally defined in [25], and subsequently refined in [26]–[28], is another timing related performance enhancement mechanism. The algorithm is based on a simple heuristic.

In Berkeley-derived kernel implementation FRT works by retransmitting a specific un-acknowledged TCP segment B that was sent after segment A when the third duplicate acknowledgment for segment A is received. The logic reasoning is that by the time the third duplicate acknowledgment for segment A is received, segment B must have lost and hence should be retransmitted. If segment B was not lost, then it should have been acknowledged because four acknowledgments for segment A (one original plus three duplicate acknowledgments) have been received.

It is observed here that the FRT action for segment B occurs before the timer associated with B expires. Otherwise, the retransmission action would be a normal retransmission, not a fast retransmission.

IV. The Modified Jacobson Algorithm

The use of Chebyshev to bound the retransmission timeout is reasonable, since it provides a "target probability" for the timeout calculation. However, the Markov bound [18] is also applicable for RTT estimation since it explicitly uses knowledge of the positivity of the random variable, as in (5) which is valid when $f_X(x) = 0$ for $x < 0$ and $\alpha > 0$.

$$\Pr[X \geq \alpha] \leq \frac{\mu}{\alpha}. \tag{5}$$

Using $\alpha = \mu + K\sigma$ in (5) produces an expression similar to (2). However, the Markov formulation results in a *variable* probability for RTO violation. This concept is particularly unappealing for small, relatively stable values of RTT, since the overbound RTO might have a high probability of violation, which would create spurious retransmissions and a large amount of unwanted network traffic. Recall that in the Chebyshev case, the choice of $\epsilon = K\sigma$ produced a fixed violation probability around 3%. The violation probability of the Markov bound can also be fixed as in (3) if $\alpha = 32\mu$:

$$\Pr[X \geq 32\mu] \leq \frac{\mu}{32\mu} = \frac{1}{32} = \frac{1}{2K^2}. \tag{6}$$

Unfortunately, a bias of 32 times the mean would not produce a viable estimate of RTT, and the overbound RTO would be extremely loose. Therefore, the Markov bound *alone* is not be a reasonable choice to estimate the RTO timeout. However, a *combination* of Markov and Chebyshev approaches seems to produce an effective estimator.

Combining an estimate of σ as in the Jacobson Algorithm with a *biased* estimate of μ as in the Markov bound results in a formulation that retains the Chebyshev structure but improves certain performance aspects. So, we use a slightly revised version of (1) and call this approach the *Modified Jacobson Algorithm*,

$$x_{thr} = \mathcal{A} \cdot \hat{\mu}_{n+1|n} + \mathcal{B} \cdot \hat{\sigma}_{n+1|n}. \tag{7}$$

In (7), the standard deviation estimator $\hat{\sigma}_{n+1|n}$ is identical to the estimator (4) used in the Jacobson Algorithm. However, the multiplier for $\hat{\sigma}_{n+1|n}$ is *reduced* (i.e., $\mathcal{B} = 2$ whereas $K = 4$). Further, the mean estimator $\hat{\mu}_{n+1|n}$ is replaced with the current value of RTT (x_n), and is *biased* in the spirit of a Markov estimator. In this case, we choose $\mathcal{A} = 1.25$ as a reasonable bias term, whereas Jacobson uses $\mathcal{A} = 1.0$.

The values of $\mathcal{A} = 1.25$ and $\mathcal{B} = 2.0$ which are used in (7) were determined experimentally, using the Jacobson values ($\mathcal{A} = 1.0$ and $\mathcal{B} = 4.0$) as a starting point.

Jacobson's mean estimator μ is a filtered version of the time series x_n. The filter structure (in (4)) is a single-pole predictor, or infinite impulse response (IIR) filter. Since the structure is IIR, it embodies some instabilities. As a result, under certain circumstances in the input sequence (such as abrupt changes in x_n) the output value of the IIR filter ca become very large. This is problematic for the combined estimation problem which involves estimators of both μ and σ, particularly when σ is also estimated using an IIR structure. Furthermore, the filtered estimate of μ contains at least a one-sample (one-packet) delay. This can be seen from (4), where μ for time $(n+1)$ is composed using values of μ and x from time (n). This delay in estimation also produces a delay in reaction to instantaneous changes in x, and it is precisely these changes in x that we are attempting to model more accurately. Thus, we replace the IIR-filtered estimate of μ with the instantaneous value x_n which is a viable estimator of the instantaneous mean for this sequence. After replacing μ with x_n, we adjust the bias value of the coefficient according to Markov's relation so that in cases where σ goes to zero, the combined estimation of RTT will not "collapse" exactly onto x_n, which would trigger many more timeouts. In other words, in (7), as σ goes to zero, the value of x_{thr} remains biased above x due to the use of $\mathcal{A} = 1.25$ rather than $\mathcal{A} = 1.0$, as in Jacobson, and this reduces timeouts in quiescent channels. The use of $\mathcal{A} = 1.0$ in Jacobson can be problematic in this regard, but introducing a constant bias (e.g., $\mathcal{A} = 1.25$) in Jacobson can be counterproductive because of the IIR nature of the prediction of μ and the IIR nature of the prediction of σ.

The use of $\mathcal{B} = 2.0$ in the Modified Algorithm follows similar logic. Because the predictor of σ in (4) is an IIR structure, the possibility of overshoot can be dramatic in cases where the RTT sequence x_n has sudden, sharp variations. A multiplier of $\mathcal{B} = 4.0$ or $K = 4.0$ (as in Jacobson) accentuates this overshoot, and is largely responsible for the massive over-estimation of RTT in cases where the underlying network exhibits certain types of instabilities. The specific choice of $\mathcal{B} = 2.0$ was driven by experimentation and is heuristically motivated. The conceptual explanation in terms of the Chebyshev relation is that we "tighten" the timeout boundary by using a smaller multiplier for σ.

Jacobson essentially computes the RTO boundary as "a few sigmas past the mean" where both μ and σ are estimated using IIR filters. We essentially compute a similar RTO boundary as "a few sigmas past the mean" where the estimator of the mean is the actual sequence value plus a small bias (this is a Markov-like formulation), and the estimator of σ is the same as in Jacobson. However, because of the relatively "more accurate" estimate of the mean in the Modified Algorithm (it has no delay), and because the mean estimator in the Modified Case is already biased ($\mathcal{A} = 1.25$ rather than 1.0), we essentially "back off" the multiplier of σ to achieve a similar overall formulation.

Note that the bias term $\mathcal{A}$ for $\hat{\mu}_{n+1|n}$ in (7) is *not equivalent* to the use of gain in the prediction of $\hat{\mu}_{n+1|n}$ in (4). The structure of the prediction filter for $\hat{\mu}_{n+1|n}$ causes delay in the formulation of the overbound, which is problematic. There are no coefficients for the prediction filter which will simultaneously improve delay and maintain stability in the estimation of μ, and incorporating gain in the prediction does not improve the estimate. These undesirable effects are completely eliminated in the Modified approach with the use of x_n as the estimator of μ. This adjustment allows for the use of a Markov-like bias term $\mathcal{A}$ and significantly enhances the performance of the Modified Algorithm.

Several factors must be specifically noted for the Modified Algorithm. First, dependence on the variance of the RTT sequence is preserved via $\hat{\sigma}_{n+1|n}$ and the Chebyshev-like formulation. Some dependence on σ must be maintained in the estimation procedure for cases where the RTT values exhibit significant variability. However, the multiplier $\mathcal{B}$ can be different (smaller) than in Jacobson. This reduced dependence on σ mediates undesirable "overshoot" which is problematic in Jacobson, and has been addressed heuristically in Eifel. Refer to abrupt changes in RTT as shown in Figure 1 for examples.

Secondly, dependence on the mean of the RTT sequence is preserved via the use of x_n for $\hat{\mu}_{n+1|n}$, and a bias is incorporated via the Markov-like formulation for cases where $\sigma \rightarrow 0$. Some dependence on μ is important, since this allows isolation of the variability. However, in cases where $\sigma \rightarrow 0$, Jacobson tends to "settle" directly onto RTT, leading to heuristic modifications including static minimum values which override the Jacobson estimates. This undesirable behavior of Jacobson is clearly evident in Figure 1.

Thirdly, explicit dependence on both μ and σ is retained via the hybrid Markov/Chebyshev formulation which *biases* the estimate higher and reduces the need for secondary minimum computations. Also, the prediction structure for $\hat{\mu}$ and $\hat{\sigma}$ is preserved, which is an important consideration.

Finally, the computational complexity of the Modified Algorithm is essentially the same as the original Jacobson Algorithm. Elimination of the prediction structure for $\hat{\mu}$ and the use of a bias term along with a simplified multiplier for $\hat{\sigma}$ results in an algorithm with the same operational complexity and no heuristic conditional logic steps.

Besides, the Modified Algorithm is designed so that two important performance enhancement mechanisms of current TCP/SCTP implementation will not be adversely affected or impacted.

V. Performance results

To validate algorithm performance, we constructed a "real-world" test environment which pairs client and server computers with modified network stacks via a controllable network infrastructure. In the network testbed, a client

system transmits data to a server system using a specially-constructed user-space application which can vary overall payload length and SCTP chunk size, as well as other parameters such as the number of test iterations. The network stack of the client systems also implement user-selectable timeout estimation algorithms and record important parameters for each transmission. Parameters recorded by the client's network stack include the timestamped, per-chunk values of the actual (measured) RTT, estimated RTO, $\hat{\mu}$, $\hat{\sigma}$, and so on. As described in Section II-C, post-processing of the RTO estimate to enforce minimum values for RTO is disabled since we are investigating the effect of estimation algorithms.

Additionally, the network devices and SCTP server are modified to introduce algorithmically controllable delays in acknowledgments of chunks and delays in delivery of various classes of network traffic. This feature results in an ability to introduce specific "delay profiles" which duplicate other known results (as in Figure 2 [7]) or randomize the round-trip time of the network. Using trace data collected directly from the network stacks of the client & server computers, we were also able to create simulations of system performance which have been cross-checked for accuracy against the delay and estimation performance of the actual systems. All performance data described in this paper was produced using our "real-world" testbed, and has been incorporated into accurate simulations of the estimation algorithms.

The performance of the Modified Jacobson Algorithm is shown in the context of various RTT sequence characteristics or "delay profiles" in Figure 2, Figure 3, and Figure 4. Figure 2 reproduces an important delay profile from literature describing the Eifel algorithm (Fig. 6 of [7]). Figure 3 contains the delay profile for a real, quiescent network with 100 msec average delay and short, artificially induced delay spikes. Figure 4 contains the delay profile for a long-term delay burst on an otherwise quiescent network with 80 msec average delay.

Quantitative assessment of algorithm performance gathered from a large number of packet transmissions is shown in Table I and Table II for each of the delay profiles described by Figure 2, Figure 3, and Figure 4. The data in Table I is presented in terms of Mean Absolute Error (MAE) in milliseconds between the RTO estimate and the actual RTT value for the same packet transmission, according to (8). Table II shows the number of induced timeout events (TO) for each estimation algorithm, and is measured in number of events per 10,000 packets transmitted.

$$\text{MAE} = \frac{1}{N} \sum_{N} |x_{thr} - x_n| \qquad (8)$$

Using our network testbed, we duplicated the RTT sequence in Figure 2 from [7]. This "delay profile" clearly shows the improvement of Eifel over Jacobson, particularly at the termination of the "ramp" sequences. Note

Table I
ESTIMATION PERFORMANCE GATHERED USING NETWORK TESTBED AND SIMULATION. PERFORMANCE IS MEASURED AS MEAN ABSOLUTE ERROR (MAE) IN MILLISECONDS.

Delay profile	Figure	Jacobson	Eifel	Modified
Eifel ramp	Figure 2	1731	2091	1577
Quiet/Spikes	Figure 3	39.81	140.7	43.29
Delay burst	Figure 4	11.31	41.79	2.00

Table II
ESTIMATION PERFORMANCE GATHERED USING NETWORK TESTBED AND SIMULATION. PERFORMANCE IS MEASURED IN TERMS OF TIMEOUT EVENTS (TO) PER 10,000 PACKETS TRANSMITTED.

Delay profile	Figure	Jacobson	Eifel	Modified
Eifel ramp	Figure 2	1	1	1
Quiet/Spikes	Figure 3	99	51	79
Delay burst	Figure 4	378	82	12

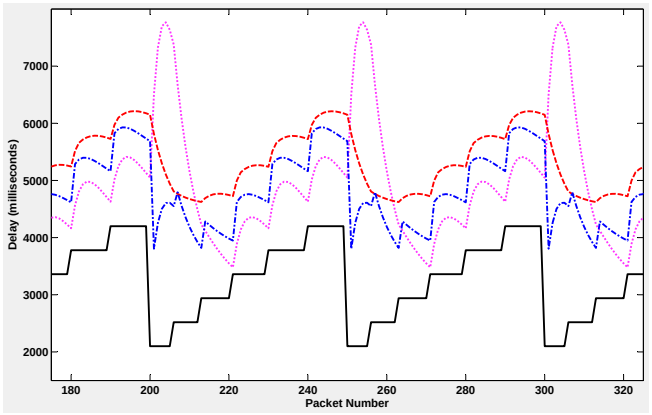

Figure 2. Performance of RTO estimators for the "ramp" RTT sequence described in [7]. Jacobson (dotted), Eifel (dashed), and Modified (dash-dot). The common RTT sequence driving all algorithms (solid).

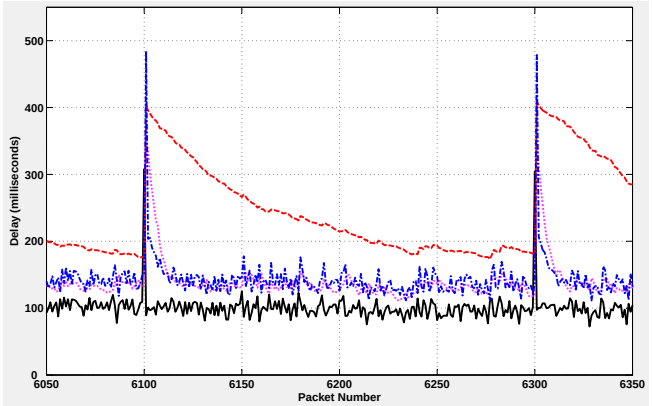

Figure 3. Performance of RTO estimators for a quiescent network with short, artificially induced delay spikes. Jacobson (dotted), Eifel (dashed), and Modified (dash-dot). The common RTT sequence driving all estimation algorithms (solid).

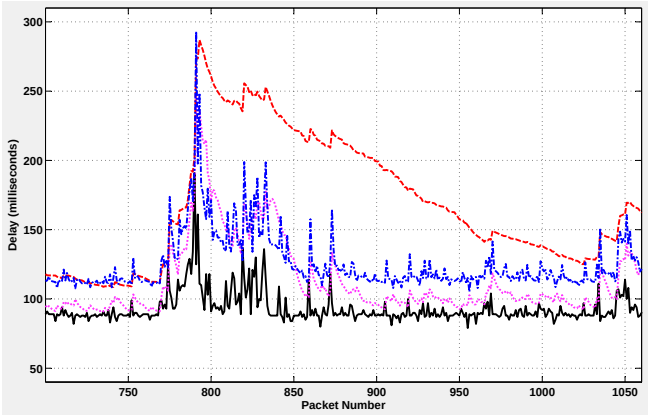

Figure 4. Performance of RTO estimators for a quiescent network with a large, naturally-occurring multi-packet delay burst. Jacobson (dotted), Eifel (dashed), and Modified (dash-dot). The common RTT sequence driving all algorithms (solid).

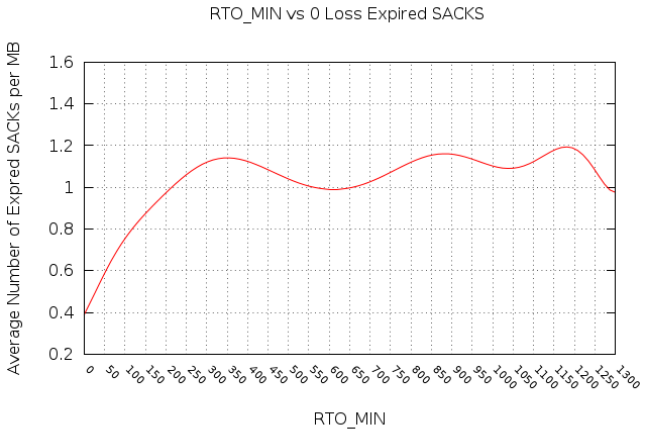

Figure 5. Relationship between RTO-min and expired SACKs

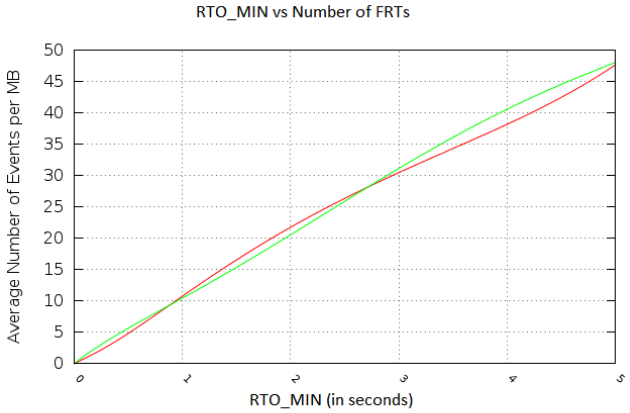

Figure 6. Relationship between RTO-min and FRTs

that Jacobson overshoots *upward* even though the RTT sequence has rapidly declining values. Eifel compensates for overshoot, but at the expense of slightly higher bias from the RTT sequence. Note that the Modified approach mimics Jacobson during ramp ascension, and also compensates for the overshoot at ramp termination. Modified has a smaller bias than Eifel. Interestingly, each algorithm induces a single timeout event, but the Modified Jacobson algorithm does so with an MAE 154 msec smaller than Jacobson, and 514 msec smaller than Eifel. Such a significant difference in MAE can translate to a smaller task-completion time in cases where a small differential in timeouts is encountered.

A delay profile from a relatively quiescent network with artificially induced delay spikes is shown in Figure 3. The network exhibits an average delay of around 100msec, and has artificially induced delay spikes which are typical of an unstable link. Note that with this network profile, delay spikes occur every 200 packets, forcing a timeout. After each delay spike, the estimation algorithms recover in very different manners: Modified and Jacobson fall quickly toward the quiescent RTT sequence, while Eifel decays very slowly, creating a relatively large wait-time for over 100 subsequent packets. As a result, the MAE for the Eifel algorithm with this delay profile is more than 3 times larger than the other estimation algorithms while still creating only 35% fewer timeout events. Regardless, each estimation algorithm creates a very small proportion of timeout events relative to the number of packets transmitted.

A delay profile from another network test is shown in Figure 4. In this figure, the network exhibits an average delay during quiescent periods of around 80 msec. However, between packets 750 and 850 a large, naturally occurring, correlated delay burst is observed, which disrupts the estimation algorithms. Note that Modified and Jacobson both fall quickly after individual, large delay spikes. However, Modified maintains a larger offset during the quiescent period between packets 900 and 1000 due to the specific bias for $\hat{\mu}$. The failure of Jacobson to maintain a bias during periods of low RTT variance ($\sigma \rightarrow 0$) is responsible for many timeout events, with Jacobson inducing *30 times more timeouts* than Modified, and *almost 5 times more* than Eifel. Eifel again exhibits a bias which is significantly larger than Jacobson or Modified, with an MAE *20 times larger* than Modified, and *4 times larger* than Jacobson. Also note that Jacobson tracks RTT fairly well, but has large, positive overshoot when RTT drops suddenly. Modified compensates for the Jacobson "overshoot" problem in cases where the RTT sequence drops suddenly (cf. packet 800 & 840).

VI. Impacts on SACK and FRT

As discussed in Section III-C and Section III-D, SACK and FRT are two important mechanisms to enhance the performance of TCP [21]. Naturally, an important question

of any revision to the current SCTP timeout mechanism is how the revision will impact SACK and FRT. The merits of any proposed revision would be questionable if the revision significantly decreases SACK or incurs unnecessary FRT.

Figure 5 shows the relationship between the number of bytes transmitted and the minimum RTO values (in millisecond). It is clear to see that the number of expired SACKs is only significantly impacted when the minimum RTO value is smaller than 200 msec. Otherwise, the minimum RTO has no direct impact on number of SACKs.

As discussed in Section III-C, SACK is proposed to enhance TCP performance over *long delay paths* [19], [23]. Our algorithm will set mininum RTO to values smaller than 200 msec when it is confident that the values of the current and most recent RTT are very small (than 200 msec). Naturally in that case the TCP communication sessions are not on *long delay paths*. Hence, it is understandable that the number of SACKs decreases.

The impact of our algorithm on FRT is less intuitive. Figure 6 shows that the number of fast retransmits is almost unaffected by the RTO algorithms. In the diagram, the X-axis is the percentage of packet losses, which were created in our simulation in order to force timeout and fast retransmits, and the chunk size is 50 bytes. The Y-axis is the number FRT events per MByte of data transmitted. The green line is the FRT for our algorithm and the red line is for the current RTO algorithm.

Recall that FRT is based on the heuristic that a TCP/SCTP segment must be lost and hence should be retransmitted because another segment sent before that segment has been acknowledged multiple times. This fast retransmission occurs before the timer for that segment expires. Smaller RTT values means that the network is more reliable and it will in turn set smaller minimum RTO values. Similarly larger RTT values implies a less reliable network, which in turn will set larger minimum RTO values. Therefore, regardless of algorithms used, the number of FRTs is directly proportional to the number of timer expirations, which is illustrated in Figure 6.

The current minimum RTO algorithm (Jacbson's Algorithm) starts by setting a 1000 msec minimum RTO value. On the other hand, our algorithm sets minimum RTO values dynamically according its own heuristics. If it sets minimum RTO values smaller than the mininum RTO value of current algorithm, then more timeouts will occur. But it will not incur more FRTs.

More details about the relationships of proposed algorithm and SACKS/FRTs will be presented in a forthcoming writing.

VII. Conclusion and Future Work

The theme of this paper is that the timeout aspects of current TCP and SCTP protocols can no longer reflect the current infrastructure and traffic dynamics of today's Internet. Current and future research should work on new and practical timeout mechanisms that meet the need of changed/changing landscapes of Internet.

This paper analyzes the methods for computing and using RTT and RTO estimates in IP-based transport protocols such as SCTP and TCP. The theoretical basis of the Jacobson Algorithm is discussed and an alternative approach is presented, which retains the fundamentally sound theoretical basis and operational structure of the algorithm, but improves the performance over other well-known techniques without introducing heuristic modifications. It is shown that our alternative approach will not adversely affect the SACK and FRT mechanisms.

Currently, we are implementing an RTO-min optimization algorithm based on the limited past history of RTO values. The implementation is done by modifying the SCTP modules of Linux kernels. The new algorithm is being tested over the Internet under various traffic dynamics.

Future work involves the continued optimization of the modified algorithm as well as investigation into the effects of variable minimum bound for the RTO timer. We believe that the work that we have done is innovative and solid. However, an implementation and testing of our algorithm over a sizable large real network will be helpful to pinpoint the strength and weakness of the algorithm.

Acknowledgement

The authors would like to express appreciation to Mr.Eduardo Gonzalez for providing data used in Section VI.

References

[1] S. McClellan and W. Peng, "Estimating retransmission timeouts in IP-based transport protocols," in *Proc. of ICDT 2013*, Apr. 2013, pp. 26–31.

[2] J. Postel, "Transmission control protocol," RFC 793, Sep. 1981.

[3] R. Stewart *et al.*, "Stream control transmission protocol," RFC 2960, Oct. 2000.

[4] R. Stewart, "Stream control transmission protocol," RFC 4960, Sep. 2007.

[5] V. Jacobson, "Congestion avoidance and control," *in Proc. ACM Comput. Commun. Review* (SIGCOMM'88),vol. 18, Aug. 1988, pp. 314–329.

[6] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," RFC 6298, June 2011.

[7] R. Ludwig and K. Sklower, "The Eifel retransmission timer," *in Proc. ACM Comput. Commun. Review* (SIGCOMM'00), vol. 30, July 2000, pp. 17–27.

[8] H. Ekstrom and R. Ludwig, "The Peak-Hopper: A new end-to-end retransmission timer for reliable unicast transport," in *Proc. 23rd Joint Conf. of the IEEE Computer & Communications Societies (INFOCOM 2004)*, vol. 4, Nov. 2004, pp. 2502–2513.

[9] L. Ma, G. Arce, and K. Barner, "TCP retransmission timeout algorithm using weighted medians," *IEEE Sig. Proc. Letters*, vol. 11, pp. 569–572, June 2004.

[10] J. Pedersen, C. Griwodz, and P. Halvorsen, "Considerations of SCTP retransmission delays for thin streams," in *Proc. 31st IEEE Conf. on Local Computer Networks*, Tampa, FL, Nov. 2006, pp. 135–142.

[11] A. Petlund, P. Beskow, J. Pedersen, E. S. Paaby, C. Griwodz, and P. Halvorsen, "Improving SCTP retransmission delays for time-dependent thin streams," *Multimedia Tools and Applications*, vol. 45, pp. 33–60, Oct. 2009.

[12] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig, "Early retransmit for TCP and stream control transmission protocol (SCTP)," RFC 5827, Apr. 2010.

[13] M. Allman and V. Paxson, "On estimating end-to-end network path properties," in *Proc. Conf. Appl., technol., arch., and protocols for comput. commun.*, ser. SIGCOMM '99. New York, NY, USA: ACM, 1999, pp. 263–274.

[14] L. Coene and J. Pastor-Balbas, "Telephony signaling transport over stream control transmission protocol (SCTP) applicability statement," RFC 4166, Feb. 2006.

[15] I. Psaras and V. Tsaoussidis, "Why TCP timers (still) don't work well," *Computer Networks*, vol. 51, pp. 2033–2048, Nov. 2007.

[16] J. Pedersen, "Evaluation of SCTP retransmission delays," Master's thesis, University of Oslo Department of Informatics, May 2006.

[17] S. Khatri, "SCTP performance improvement based on adaptive retransmission time-out adjustment," Master's thesis, Texas State University Department of Computer Science, Aug. 2011.

[18] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed. New York, NY: McGraw-Hill, 1991.

[19] V. Jacobson and R. Braden, "TCP extensions for long-delay paths," RFC 1072, Oct. 1988.

[20] L. Eggert, "Moving the undeployed TCP extensions RFC 1072, RFC 1106, RFC 1110, RFC 1145, RFC 1146, RFC 1379, RFC 1644, and RFC 1693 to historic status," RFC 6247, May 2011.

[21] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, 1st ed. Boston, MA: Addison-Wesley, 1994.

[22] V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," RFC 1323, May 1992.

[23] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," RFC 2018, Oct. 1996.

[24] R. Bruyeron, B. Hemon, and L. Zhang, "Experimentations with TCP selective acknowledgment," in *Proc. Comput. Commun. Review*, vol. 28. ACM, Apr. 1998, pp. 54–77.

[25] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC 2001, Jan. 1997.

[26] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, Apr. 1999.

[27] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's initial window," RFC 3390, Oct. 2002.

[28] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," RFC 5681, Sep. 2009.