Estimation of RTT and Bandwidth for Congestion Control Applications in Communication Networks

Krister Jacobsson, Håkan Hjalmarsson, Niels Möller and Karl Henrik Johansson

Abstract— **Heterogeneous communication networks with their variety of application demands, uncertain time-varying traffic load, and mixture of wired and wireless links pose several challenging problem in modeling and control. In this paper we focus on the round-trip time (RTT), which is a particularly important variable for efficient end-to-end congestion control, and on bandwidth estimation. Based on a simple aggregated model of the network, an algorithm combining a Kalman filter and a change detection algorithm is proposed for RTT estimation. It is illustrated on real data that this algorithm provides estimates of significantly better accuracy as compared to the RTT estimator currently used in TCP, especially in scenarios where new cross-traffic flows cause bottle-neck queues to rapidly build up which in turn induces rapid changes of the RTT.**

Standard techniques for bandwidth estimation is based on measurements of inter-arrival times of packets as the bandwidth is proportional to the inverse of the inter-arrival time. Two main classes of bandwidth estimators are analyzed wrt how variations in the inter-arrival times affect the estimates. It is shown that linear time-invariant filtering of instantaneous bandwidth estimates does not change the bias. In contrast to this, smoothing the inter-arrival-time samples, does give a bias reduction which depends on the smoothing filter. Hence, with such approach, noise attenuation can be traded against tracking ability wrt changes in the actual bandwidth.

I. Introduction

Congestion control is one of the key components that has enabled the dramatic growth of the Internet. The original idea [13] was to adjust the transmission rate based on the loss probability. The first implementation of this mechanism, denoted TCP Tahoe, was later refined into TCP Reno. This algorithm (together with some of its siblings) is now the dominating transport protocol on the Internet. The throughput and delay experienced by individual users are depending on several factors, including the TCP protocol, link capacity and competition from other users. There are also lower layers that may affect the achieved delay and bandwidth, particularly if part of the end-to-end connection is a wireless link, see [24], [30], [5], [29], [7].

There are many studies of the statistics of network quantities [1], [2] and even estimation of RTT at the link level [16]. However, there seems to be minor work on considering RTT estimation from the perspective of TCP.

More work has been published on bandwidth estimation. In active probing, packet pairs/trains are transmitted and their dispersion is used as a measure of the cross-traffic/bandwidth, see [12], [25] and references therein for more information. TCP Westwood [8] is a TCP clone which explicitly estimate the available bandwidth, using the rate of the packet acknowledgments.

The outline of the paper is as follows. In Section II a brief presentation of TCP, and especially the importance of accurate RTT estimates, is given. This motivates the RTT model and estimation scheme presented in Section III. A new algorithm based on a Kalman filter and change detection is proposed for RTT estimation. In scenarios where new cross-traffic flows cause bottle-neck queues to rapidly build up, the algorithm is shown to be particularly useful to track the rapid changes of the RTT. It gives significantly better accuracy compared to the RTT estimator currently used in most TCP versions. In Section IV we discuss how to estimate the bandwidth so that the noise that the short-lived traffic induces does not introduce a bias in the estimate. It has been recognized that e.g. TCP Westwood suffers from this problem [8].

II. TCP and RTT

TCP is window-based which means that each sender has a window that determines how many packets in flight that are allowed at any given time. The transmission rate is regulated by adjusting this window. For a network path with available bandwidth b and RTT τ, the optimal window size is $b\tau$, in the sense that if all users employed this window there would be no queues and the link capacities would be fully utilized. Using loss probability (as in TCP) to measure congestion means that the capacity of the network cannot be fully utilized. With necessity, queues have to build up (causing increased delays) and queues have to overflow (causing loss of throughput). Several methods to cope with these shortcomings have been suggested. In TCP Vegas [3] a source tries to estimate the number of packets buffered along its path and regulates the transmission rate so that this number is low (typically equal to three). One interpretation of this algorithm is that it estimates the round-trip queuing delay and sets the rate proportional to the ratio of the round-trip propagation delay and the queuing delay [23]. Both round-trip delays are obtained from measurements of the RTT.

Congestion can be indicated in more sophisticated manners than just dropping packets. In random early detection

This work was supported by European Commission through the project EURONGI and by Swedish Research Council.

K. Jacobsson is with Department of Signals, Sensors and Systems, KTH, SE-100 44 Stockholm, Sweden krister.jacobsson@s3.kth.se

H. Hjalmarsson is with Department of Signals, Sensors and Systems, KTH, SE-100 44 Stockholm, Sweden hjalmars@s3.kth.se

N. Möller is with Department of Signals, Sensors and Systems, KTH, SE-100 44 Stockholm, Sweden niels@s3.kth.se

K. H. Johansson is with Department of Signals, Sensors and Systems, KTH, SE-100 44 Stockholm, Sweden kallej@s3.kth.se

(RED) [6], packets are dropped before the buffer is full, with a probability that increases with the queue length. RED can thus be seen as a way of indirectly signaling the queue length to the source. In explicit congestion notification (ECN) the links add information concerning their status to the packets. As there is only one bit available in the packet header for ECN, clever coding is required, e.g., random exponential marking (REM) [27].

The transmission rate control problem can be solved in a completely decentralized manner, as was recently shown in [18], [17]. Each source has a (concave) utility function of its rate. The optimization problem is to maximize the sum of the utility functions for all sources. It is shown that in order to solve this problem each source needs to know the sum of the link prices on the path. The link price is a a penalty function corresponding to the capacity constraint. It is a function of the total arrival rate at the link and can thus be computed locally at each router. This optimization perspective of the rate control problem has been taken in a number of contributions [20], [22], [21].

Network state variables such as queueing delays and RTT are essential for efficient congestion control, as has been recognized by many researchers, cf., [28]. This has a simple intuitive interpretation: The ideal window is $b\tau$. Hence, the more accurate estimates of b and τ that are available, the closer to the ideal situation it is possible to keep each flow. It is also clear that when only such indirect measures of congestion can be used, throughput will suffer somewhat, since queues have to start to build up in order for the source to detect an increased delay. The queues can be seen as a way to smooth out the uncertainty in the bandwidth and RTT estimates without loss of throughput. It follows that one can obtain higher throughput than TCP Reno, since TCP Reno fills up the bottle-neck completely before reacting.

III. RTT Estimation

Motivated by the previous discussion on the importance of accurate RTT estimates, we now take a closer look at short-range RTT from a statistical perspective. The raw RTT measurements, obtained from packet acknowledgements (ACK's), include delays caused by transient effects in the network (attributed to short-lived cross-traffic). The short-lived duration of these flows means that their contribution to the RTT can be considered as noise from the point of view of congestion control. It is thus reasonable to filter them out. In present versions of TCP this is done with a first-order low-pass filter. The average RTT (averaged over a few RTT) often makes sudden changes due to the appearance of a long-lived cross-flow somewhere along the path. It is then important for the controller to react quickly to this change, since otherwise buffers will start to build up with enlarged risk of packet loss and increased delay as consequences. It is impossible with a first-order filter to rapidly adjust to these changes. This motivates the use of a filter based on change detection for RTT estimation, as presented in this section. Before introducing the algorithm, we present a simple model for RTT. The section is ended by experimental evaluations.

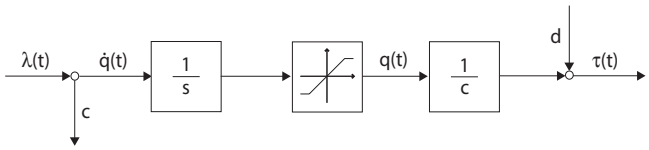

Fig. 1. Network node model.

A. Model

Let $\tau(t)$ denote RTT at the time the ACK is received at the source node. Introduce $d_{l,i}(t)$ for the link delay of link i, $d_{p,i}(t)$ for the corresponding propagation delay, and $d_{q,i}(t)$ for the queue delay. Suppose the considered end-to-end connection has m nodes. Then,

$$\tau(t) = \sum_{i=1}^{m} \left(d_{l,i}(t) + d_{p,i}(t) + d_{q,i}(t) \right).$$

Assume, for now, that the path through the network remains constant during a session. The propagation and link delay is fairly constant and thus only contributes with a constant to the RTT estimate. All the major RTT dynamics is depending on the evolution of the queue states along the paths.

The simple dynamics of the queue length $q_i(t)$ at node i is a saturated integrator, as illustrated in Figure 1. Let c_i denote the link capacity and t_i the time instant for the packet arrival at node i. Then, RTT at the time t when the ACK is received is given by

$$\tau(t) = \sum_{i=1}^{m} \frac{q_i(t_i)}{c_i} + \text{bias},$$

where the bias term represents the link and propagation delays.

B. Change detection

RTT have high-frequency characteristics that are desirable to detect. To be able to follow step changes in the RTT mean value due to increased network load, new competing traffic flows, or sudden path changes, more advanced algorithms are needed than the first-order linear filter currently implemented in most TCP versions. We propose an adaptive filter with change detection.

Regard RTT as being composed of a piece-wise constant desired RTT signal together with an additive high-frequency noise component. We thus model the desired RTT as a noisy observation of a constant exposed to step changes in the mean. If we denote the RTT by y_t and the underlying piecewise constant RTT by x_t, we obtain the model

$$\begin{aligned} x_{t+1} &= x_t + \delta_t v_t, \qquad \delta_t \in \{0, 1\} \\ y_t &= x_t + e_t. \end{aligned}$$

The noisy characteristic of RTT is thus captured by the measurement noise e_t with variance R_e. The step changes

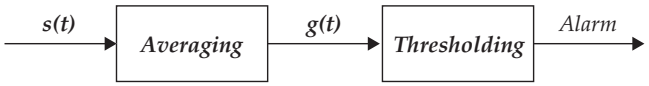

Fig. 2. The principle of change detection.

in the desired RTT x_t is modeled as the process noise v_t with variance R_v and the discrete variable δ_t. If a change occurs at time t, then $\delta_t = 1$ otherwise $\delta_t = 0$. To estimate the sequence δ^N of instances when a change occurred is a segmentation problem. The optimal linear estimator is the Kalman filter which is used to estimate x_t. The principle of change detection is illustrated in Figure 2. To estimate the sequence δ^N of instances of changes, we use a one-sided cumulative sum (CUSUM) algorithm [11]. Combining the Kalman filter with the CUSUM algorithm yields the following adaptive filter, which we denote the CUSUM Kalman filter:

$$\begin{aligned}
\hat{x}_t &= \hat{x}_{t-1} + K_t(y_t - \hat{x}_{t-1}) \\
K_t &= \frac{P_{t-1}}{P_{t-1} + R_e} \\
P_t &= (1 - K_t)P_{t-1} + R_v \\
\epsilon_t &= y_t - \hat{x}_{t-1} \\
g_t &= \max(g_{t-1} + \epsilon_t - \xi, 0) \\
\delta_t &= 1 : R_v \neq 0, \text{alarm if } g_t > h \\
\delta_t &= 0 : R_v = 0 \\
&\text{After an alarm, reset } g_t = 0
\end{aligned}$$

The output of the CUSUM Kalman filter is given by the estimate of the (desired) RTT $\hat{x}_t$. The filter has two design parameters: the negative drift ξ and the alarm threshold h. These parameters are tuned to adjust the sensitivity in the detection procedure. The same values have been used in all our initial experiments with satisfying results, so the filter seems fairly robust. Note that also the variances R_v and R_e influence the filter behavior.

C. Experimental evaluation

The objective is to capture the rapid changes that the RTT undergoes when queues build up when for instance the traffic load increases abruptly. We developed a modified ping tool to monitor RTT time series. By sending a dense stream of small packets the RTT is monitored effectively without affecting the network too much. Note that data is not easily obtained from a TCP session itself, because a rapid traffic load increase would probably trigger the RTO mechanism in TCP (which would mean lack of samples during the transient). Also, if a queue along the path fills up, packet loss occurrences reduces the TCP sending rate with more sporadic time series as a consequence. Recall that the RTT time in congestion control is often in fractions of seconds (typically $10 - 500$ ms).

We measured the RTT between KTH and the CAIDA web site.This path is normally about 20 hops, including

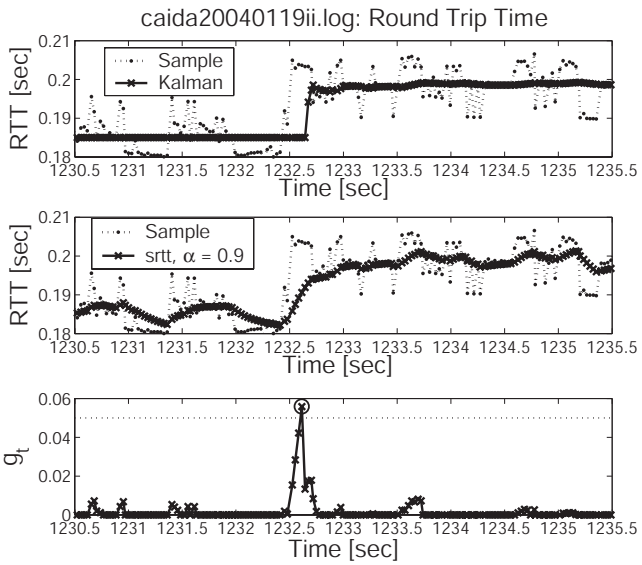

Fig. 3. Experimental evaluation of the proposed RTT estimation algorithm on a small change in RTT.

the Atlantic link. The mean RTT is approximately 190 ms. The sending interval (sampling time) of the ping tool was 30 ms. According to our measurements, the path is normally not congested with the result that the RTT almost shows a deterministic behavior with a low variance. However, sporadically the traffic load increased with the result of suddenly increased and fluctuating queues, which propagated to the RTT. The proposed CUSUM Kalman filter was used on the collected data sets. The variances R_e and R_v were set to the variance of the RTT samples. The CUSUM design parameters were set to $\xi = 0.005$ and $h = 0.05$ in all experiments.

A detection of a sudden step in the RTT mean value can be seen in Figure 3. The two upper plots show the sampled RTT values, together with the output of the CUSUM Kalman filter and the output of the TCP first-order filter with gain set to 0.9 [14]. In the lower plot the test statistic g_t is plotted and the detection alarms are encircled. The CUSUM Kalman filter estimate is smooth, but still manages to detect a mean change as low as 7 %. Note that the TCP filter is more sensitive to noise and periodic fluctuations. As the change in mean is moderate in this example, the TCP filter adapts quite fast. The RTT mean change is probably a result of an abrupt change in the traffic load. An additive static traffic stream as a UDP flow might be an explanation. However a re-routing inside the network is also a possibility.

In Figure 4 we see a large change in the mean of about 100 %. Even if the probing packets are sent with only 30 ms intervals, we do not capture the queue building up. The result becomes a step in the RTT measurements. In this scenario the proposed RTT estimation algorithm reacts after only a few samples, and is much faster than the filter in TCP. Note that this time it was no static mean change, but the queue vanishes after half a second and drops to its original

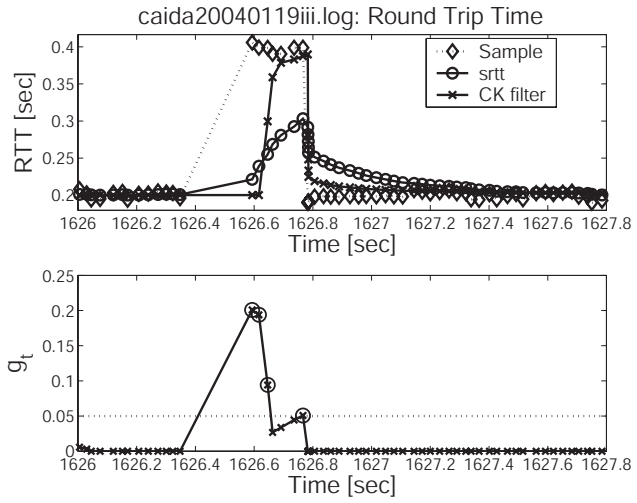

Fig. 4. Experimental evaluation of the proposed RTT estimation algorithm on a large change in RTT. Note how well the proposed algorithm (CK-filter) captures the rapid change, while the conventional filter in TCP (srtt) adjusts much slower.

level. The estimation algorithm reacts on the level reset and adapts almost immediately while the TCP filter is lagging.

The two given examples show the main features of the CUSUM Kalman filter: smooth estimates but still fast adaption when drastic changes occur. In an application within a transmission protocol, the sampling time is typically larger and the measurements tend to be more spiky. The probability of hitting a plateau as in Figure 4 with more than a few samples is then low. By filtering out such events the RTT estimate could be kept at the appropriate level.

IV. Unbiased available bandwidth estimation

In a window based congestion control protocol with feedback, measurements of the returning packet dispersion can be used to reconstruct the experienced throughput. This implicit information can be exploited and post processed into a suitable estimate for use in different congestion control applications. In the TCP clone Westwood (TCP-W) [8] this is recognized, and throughput samples are used in the window recovery mechanism. The rationale is that the smoothed throughput is interpreted as the path available bandwidth and if a congestion event occur, the congestion window is set to match this rate instead of being drastically halved as in standard TCP. Avoiding getting too deep into the discussion if this available bandwidth is the fair share or not, we conclude that if the path available bandwidth (ABW) is defined as in [15], i.e. the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic along that path, TCP-W should ideally strive towards its fair share and the protocol objective seems reasonable.

The Westwood scheme has shown good performance, especially in lossy network environments as wireless applications where packet losses mainly are due to link failure and not a signal of congestion (which is what TCP originally was designed for). However, despite good performance TCP-W interacts with other congestion control applications as its TCP siblings, and should hence be fair in the sense that if two competing flows are sending over the same link the steady state throughput become equal. This issue has been investigated in e.g [8], [10], and in [9] it is concluded that the Westwood scheme overestimate the available bandwidth in the presence of ACK compression [26] which explains empirical problems with fairness towards less aggressive schemes. The observed phenomena is claimed to be due to aliasing effects, and to counteract this it is proposed that one should smooth suitable clusters of inter-arrival-times and then recreate the bandwidth samples that are finally filtered. The bias in bandwidth estimation has also been observed in [4]. It is shown in simulation examples that the bias disappears when the ACK inter-arrival-times are smoothed before the available bandwidth is calculated. The modified TCP scheme is reported to be more friendly towards TCP Reno in wired networks.

As it seems as if the underlying reason for these observed behaviors is not well understood, we will analyze bandwidth estimation in more detail below.

A. Bandwidth reconstruction and estimation

The throughput or average bandwidth b used by a connection is the successfully transferred amount of data normalized with the time interval in consideration. The average bandwidth can be reconstructed at the sender side by logging the ACK inter-arrival-time δ_i and the amount of corresponding acknowledged data ν_i according to

$$b_n = \frac{\sum_{i=1}^{n} \nu_i}{\sum_{i=1}^{n} \delta_i} \tag{1}$$

Since we assume that the used bandwidth also is the fair share we will refer to it as available bandwidth in the sequel.

For stationary conditions on the path, i.e. constant cross-traffic load conditions, (1) is also the best estimate of the available bandwidth at the time when packet n is ACKed. However, (1) is less suitable as estimate of the available bandwidth under varying load conditions. The reason being that the averaging in (1) makes it react slowly to changes in the available bandwidth. Considering the problem of processing (filtering) the inter-arrival-times into an estimate of the available bandwidth at the current time we are thus faced with two requirements. The estimate should

i) resemble (1) under stationary conditions, and
ii) react quickly to changes.

In general these two requirements are conflicting thus enforcing a trade-off.

Consider the path bottle-neck link with capacity c. This link is completely utilized by definition and outgoing packets travels back-to-back. If no interfering traffic is present downstream the bottle-neck link or on the ACK path, the time space between two received ACK:s is the sum of the

packet size ν_i and merging data κ_i scaled by the bottle-neck capacity c, see Figure 5. Under stationary conditions,

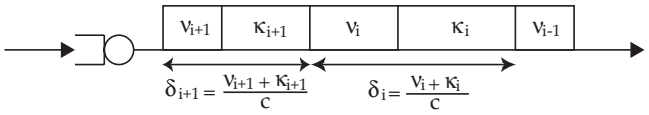

Fig. 5. Packets traveling on the bottle/neck link with capacity c Mbps.

transmitting a constant packet size $\nu_i = \nu$, should ideally result in the merging traffic being of constant size $\kappa_i = \kappa$. This means that the inter-arrival-times would be perfectly evenly distributed, i.e. $\delta_i = \delta$ for some constant $\delta > 0$, and hence, the instantaneous bandwidth estimate

$$\hat{\hat{b}}_i = \frac{\nu_i}{\delta_i} \tag{2}$$

would equal the available bandwidth ν/δ. However, in practice, even with constant packet size $\nu_i = \nu$, the merging traffic κ_i is generally time varying. It is instructive to think of these variations as noise as these variations will average out over a longer time-interval. Packet displacements due to cross-traffic downstream the bottle-neck, as well as ACK delays can also be considered as noise affecting the inter-arrival times. It is thus appropriate to introduce the model

$$\delta_i = \overline{\delta}_i + e_i \tag{3}$$

where $\overline{\delta}_i$ is the inter-arrival time that would be obtained when one packet of size ν_i is transmitted for each ACK and the load from cross-traffic is frozen at the present level and ideal conditions hold. As discussed above, the noise term e_i can be attributed to local variations in the cross-traffic at the bottleneck and the impact of cross-traffic downstream the bottleneck as well as on the ACKs. It seems reasonable to assume that when the cross-traffic is stationary, e_i can be modeled as a stationary stochastic process with zero mean.

Under this assumption the requirements i) and ii) above can be rephrased as that the bandwidth estimate should

- i') reduce the impact of e_i as much as possible, while still be able to
- ii') track variations in $\overline{\delta}_i$.

Below we will analyze some approaches to this problem.

B. Filtering the instantaneous bandwidth estimates

Viewing the instantaneous bandwidth estimates (2) as raw data, an obvious approach is to low-pass filter these estimates in order to obtain a smoother estimate. With $H(q) = \sum_{k=0}^{\infty} h_k q^{-k}$ denoting a linear time-invariant (LTI) filter (here q^{-1} denotes the backward-shift operator $q^{-1}e_i = e_{i-1}$), the bandwidth estimate is given by

$$\hat{b}_{b,n} = H(q)\hat{\hat{b}}_n = \sum_{k=0}^{\infty} h_k \, \frac{\nu_{n-k}}{\delta_{n-k}} \tag{4}$$

Let us now analyze the impact of the noise e_i under stationary conditions. Suppose therefore that the packet size $\nu_i = \nu$, a constant, so that $\overline{\delta}_i = \overline{\delta} = \nu/b$ where b is the available bandwidth.

Under ideal conditions, i.e. when $e_n = 0$, $\hat{\hat{b}}_n = \nu/\overline{\delta}$ is a perfect estimate of the available bandwidth b. Hence, in order to guarantee that the smoothed estimate $\hat{b}_n$ is unbiased, the filter should have static gain 1, i.e. $H(1) = 1$.

Now, notice that when the noise e_i is non-zero, the instantaneous estimate is over-biased. The function $\Phi(x) = \nu/(\overline{\delta} + x)$ is convex and Jensen's inequality [19] implies

$$\mathrm{E}\left[\hat{\hat{b}}_n\right] = \mathrm{E}\left[\Phi(e_n)\right] > \Phi\left(\mathrm{E}\left[e_n\right]\right) = \Phi(0) = \frac{\nu}{\overline{\delta}} = b \tag{5}$$

Hence, also the smoothed estimate is biased since

$$\mathrm{E}\left[\hat{b}_{b,n}\right] = H(q)\left(\mathrm{E}\left[\hat{\hat{b}}_n\right]\right) = \mathrm{E}\left[\hat{\hat{b}}_n\right] > b \tag{6}$$

where the second equality follows from that $H(1) = 1$.

The bias can be quantified by expanding the raw bandwidth estimate (2) in a Taylor series

$$\hat{\hat{b}}_n = \frac{\nu}{\overline{\delta} + e_n} = \frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{e_n^j}{\overline{\delta}^j}\right) \tag{7}$$

Hence

$$\begin{aligned} \hat{b}_{b,n} &= H(q)\left(\frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{e_i^j}{\overline{\delta}^j}\right)\right) \\ &= \frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{H(q)e_n^j}{\overline{\delta}^j}\right) \end{aligned} \tag{8}$$

and

$$\begin{aligned} \mathrm{E}\left[\hat{b}_{b,n}\right] &= \frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{H(q)\mathrm{E}\left[e_n^j\right]}{\overline{\delta}^j}\right) \\ &= \frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{m_j}{\overline{\delta}^j}\right) \end{aligned} \tag{9}$$

where m_j denotes the jth moment $\mathrm{E}[e_n^j]$ of the noise.

Observe that the bias is independent on the properties of the filter H (as long as it has static gain 1) and is the *same* as for the raw bandwidth estimate $\hat{\hat{b}}_n$. We illustrate this in an example.

Example 4.1: Figure 6. shows outputs from different filters with bandwidth samples from a NS-2 simulation as input data. The dashed line is the bandwidth estimate produced by an arithmetic mean filter and the solid line is the bandwidth estimate produced from a first order filter with a pole at 0.99. The experimental setup is a TCP SACK source sending data over a 5 Mbps bottle-neck link. The congestion window bound and end-to-end delay are configured such that the link is completely utilized. At time 10 s a single UDP flow of 1 Mbps enters the bottle-neck link. After the transient has died out we see that both filters reach the same level, and hence have the same bias.

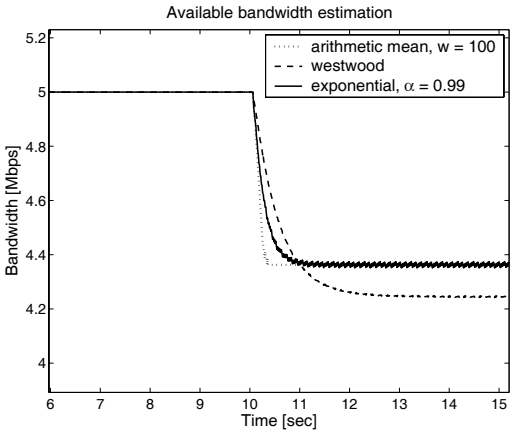

Fig. 6. Estimation by filtering directly on the bandwidth sample. Up to time 10 s the available bandwidth is 5 Mbps, after that 4 Mbps.

C. TCP-Westwood

TCP-W filters the instantaneous bandwidth samples and should, according to the analysis in the previous section, be susceptible to bias. This has also been observed [9], [4] and is also shown in Figure 6 where the bandwidth estimate from the original time dynamic exponential filter in TCP-W (see[8]) is shown as the dotted line. Notice that, after the transient has died out, after the onset of the UDP-flow, this estimate does *not* reach the same level as the arithmetic mean and first order filter estimates. This means that the bias for this estimate is different from that of a LTI filter. To better understand this we investigate the TCP-W filter in [8], which is given by

$$\hat{b}_{w,n} = \alpha_n \hat{b}_{w,n-1} + (1-\alpha_n)\hat{\hat{b}}_n$$

with

$$\alpha_n = \frac{2\tau - \delta_n}{2\tau + \delta_n} \tag{10}$$

where τ is a filter parameter. The filter is thus linear, but *time-varying* and this is the reason for its different behavior. The estimate can be expressed as

$$\hat{b}_{w,n} = \sum_{k=0}^{\infty}(1-\alpha_{n-k}) \prod_{l=0}^{k-1} \alpha_{n-l}\, \hat{\hat{b}}_{n-k} \tag{11}$$

Notice first that if $\alpha_n = \alpha$, then the above is a first order filter with static gain 1. Now, in order to get some intuition for this expression when the filter coefficients vary with time, let us, for the moment, use the simple model

$$\alpha_n = \alpha + v_n \tag{12}$$

where v_n is modeled as a stationary stochastic process with zero mean. Assuming, for simplicity, that $\hat{\hat{b}}_n$ and v_n are processes independent of each other gives that

$$\mathrm{E}\left[\hat{b}_{w,n}\right] = \mathrm{E}\left[\sum_{k=0}^{\infty}(1-\alpha_{n-k}) \prod_{l=0}^{k-1} \alpha_{n-l}\right] \mathrm{E}\left[\hat{\hat{b}}_{n-k}\right] \tag{13}$$

>From this we see that when $\{v_n\}$ is an uncorrelated sequence, the filter has an average static gain 1 since all factors are independent with mean α. However, this will in general not hold when this sequence is correlated. We conclude that the bias in TCP-W will depend on the correlation of the filter coefficients α_n. Thus different scenarios will result in different bias. This is exemplified in Figure 7 where we apply a time varying first order exponential filter with correlated and uncorrelated coefficient sequences respectively to the experimental data just mentioned.

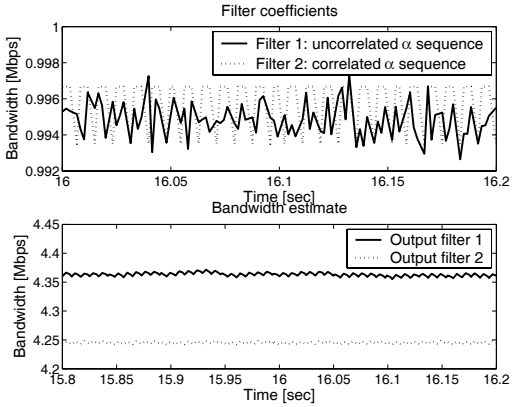

Fig. 7. Coefficients and output of a time varying first order exponential filter with correlated and uncorrelated filter coefficients.

Returning to Figure 6 it can be observed that the the bias for the TCP-W estimate is less than the other two estimates in this scenario. From the above analysis it is clear that this is not an indication of superior performance of TCP-W in general as it is just a coincidence due to the particular correlation of the filter coefficients in this scenario.

D. Smoothing the ACK inter-arrival-time

By rewriting (1) as

$$b_n = \frac{\frac{1}{n}\sum_{i=1}^{n} \nu_i}{\frac{1}{n}\sum_{i=1}^{n} \delta_i} \tag{14}$$

we see that we can interpret (1) as a way of smoothing (low-pass filtering) the variations in the inter-arrival time.

This suggests the general estimate

$$\hat{b}_{t,n} = \frac{F(q)\nu_n}{H(q)\delta_n} \tag{15}$$

In order to for this estimate to be unbiased under ideal conditions, the static gain of the LTI filters F and H should be unity.

We now analyze the bias of (15) under the same assumptions as in Subsection IV-B. We then have

$$\hat{b}_{t,n} = \frac{F(q)\nu_n}{H(q)\delta_n} = \frac{F(q)\nu}{H(q)\overline{\delta} + H(q)e_n} = \frac{\nu}{\overline{\delta} + H(q)e_n} \tag{16}$$

Comparing this with the instantaneous bandwidth estimate (2) we see that the only difference is that e_n is replaced by

the filtered noise $H(q)e_n$, hence the expansion equivalent to (7) is

$$\hat{b}_{t,n} = \frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{(H(q)e_n)^j}{\overline{\delta}^j} \right) \tag{17}$$

Thus, if the filter is such that the noise is attenuated, the inter-arrival-time smoothed estimate $\hat{b}_{t,n}$ will have less bias than the raw bandwidth estimates $\hat{\hat{b}}_n$.

Regarding the smoothed bandwidth estimate $\hat{b}_{b,n}$, defined in (4), we see by comparing (9) with the mean of $\hat{b}_{t,n}$, which from (17) is given by

$$\mathrm{E}\left[\hat{b}_{t,n}\right] = \frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{\mathrm{E}\left[(H(q)e_n)^j\right]}{\overline{\delta}^j} \right), \tag{18}$$

that the filter $H(q)$ does influence the bias of $\hat{b}_{t,n}$. Hence, in contrast to what holds for $\hat{b}_{b,n}$, $H(q)$ can be used as a design variable (subject, of course, to $H(1) = 1$) to reduce the bias for the smoothed inter-arrival-time based bandwidth estimate $\hat{b}_{t,n}$.

We will illustrate this on a simplified example.

Example 4.2: Suppose that e_n is an uncorrelated sequence with variance $\lambda_\circ$ and that it is only the moments up to order 2 that give a contribution to the raw bandwidth estimates $\hat{\hat{b}}_n$. For (2), it follows from (9) that

$$\mathrm{E}\left[\hat{b}_{b,n}\right] = \frac{\nu}{\overline{\delta}} \left(1 + \sum_{j=1}^{\infty} (-1)^j \frac{m_j}{\overline{\delta}^j} \right) = \frac{\nu}{\overline{\delta}} \left(1 + \frac{\lambda_\circ}{\overline{\delta}^2} \right) \tag{19}$$

which gives the bias

$$\mathrm{E}\left[\hat{b}_{b,n}\right] - \frac{\nu}{\overline{\delta}} = \frac{\nu}{\overline{\delta}} \frac{\lambda_\circ}{\overline{\delta}^2} \tag{20}$$

For the smoothed inter-arrival-time estimate $\hat{b}_{t,n}$, suppose that $H(q)$ in (15) is a first order filter with impulse response coefficients $h_k = (1-\alpha)\alpha^k$. Then, since e_n has zero mean,

$$\mathrm{E}\left[\hat{b}_{t,n}\right] = \frac{\nu}{\overline{\delta}} \left(1 + \frac{\mathrm{E}\left[(H(q)e_n)^2\right]}{\overline{\delta}^2} \right) = \frac{\nu}{\overline{\delta}} \left(1 + \frac{\frac{1-\alpha}{1+\alpha}\lambda_\circ}{\overline{\delta}^2} \right)$$

which implies the bias

$$\mathrm{E}\left[\hat{b}_{t,n}\right] - \frac{\nu}{\overline{\delta}} = \frac{\nu}{\overline{\delta}} \frac{\frac{1-\alpha}{1+\alpha}\lambda_\circ}{\overline{\delta}^2}$$

Notice that the bias can be made arbitrarily small by choosing α sufficiently close to 1, i.e. by making the filter more and more low-pass.

We now illustrate the difference between smoothing raw bandwidth samples and smoothing raw inter-arrival-time samples by way of a NS-2 simulation.

Example 4.3: The data originates from the same experimental setup as before but with poisson traffic as disturbance. At time 10 s traffic with mean 1 Mbps is passing the bottle-neck link. After 20 s stochastic traffic with mean 1 Mbps is added to the 5 Mbps link along the ACK path. The results are shown in Figure 8. The robustness of the inter-arrival-time smoothing filter is clearly superior and the effect of the ACK clustering is negligible as illustrated.

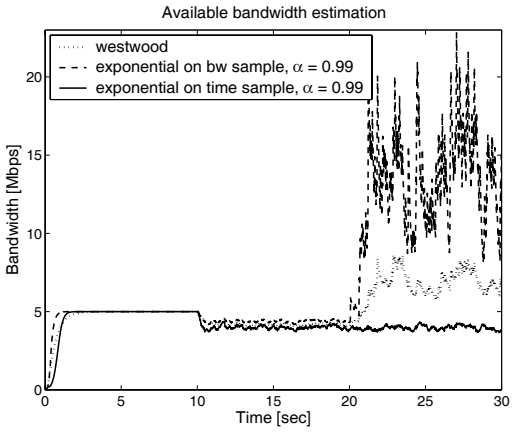

Fig. 8. Same exponential filter applied to both bandwidth samples and samples of the ACK inter-arrival-time. Stochastic traffic of 1 Mbps is entering the bottle-neck link at time 10 s. At 20 s a flow with 1 Mbps stochastic traffic is disturbing the ACK path at a link with 5 Mbps capacity.

V. Conclusions and future work

In this paper we have studied RTT estimation that could be suitable for a congestion control application using network state information. Weighting the characteristic of RTT and the application of the estimate together we concluded that the estimation objective is to keep the estimate smooth but still able to capture sudden mean changes. A Kalman filter combined with CUSUM change detection was proposed and shown to perform well based on results of experiments on real data. Long lived rapid changes are detected but short lived ones are filtered out as noise. Tuning the parameters of the CUSUM Kalman filter were done manually but online tuning is an issue to investigate further. In the paper we have only considered the RTT estimation, not how the network metric itself should be used practically in a congestion control algorithm.

Regarding bandwidth estimation we have argued that the bandwidth estimation problem is a trade-off between noise suppression, required for accurate estimation during stationary conditions, and tracking ability, required to follow rapid changes of the true underlying bandwidth. In this contribution we have focused on analyzing the stationary behavior for the two main approaches to bandwidth estimation: i) smoothing raw bandwidth samples, and ii) smoothing the raw inter-arrival times. For estimators employing LTI filters, we have shown that the first approach leads to a bias that cannot be influenced by the filter itself if it is designed to provide unbiased estimates in the case of perfect raw bandwidth samples. This is in contrast with the second approach where the filter can be used to attenuate the noise and, thus, provide significantly less biased estimates.

This was also illustrated on some examples. We also noted that the time-varying filter in TCP-Westwood may by itself introduce a bias.

In order to capture changes in the underlying $\overline{\delta}_i$, a change-detection approach, similar to the one presented for RTT estimation in Section III, may be an interesting approach. This is currently under investigation and will be reported elsewhere.

The optimal window size setting based on information from accessible implicit information as estimates of RTT and available bandwidth should in the future be investigated in parallel with the development of the estimation procedures.

REFERENCES

[1] M. Allman. A web server's view of the transport layer. *ACM SIGCOMM Computer Communication Review*, 30(5), 2000.

[2] M. Allman and V. Paxson. On estimating end-to-end network path properties. *ACM SIGCOMM Computer Communication Review*, 31(2), 2001.

[3] L.S. Brakmo and L.L. Peterson. TCP Vegas: end-to-end congestion avoidsance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[4] A. Capone, Luigi Fratta, and Fabio Martignon. Enhanced bandwidth estimation algorithms in the TCP congestion control scheme. In *NetCon 2002*, IFIP Conference Proceedings, pages 469–480. Kluwer, 2002.

[5] S. Cen, P.C. Cosman, and G.M. Voelker. End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Trans. on Networking*, 11(5):703–717, 2003.

[6] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. on Networking*, 1(4):397–413, 1993.

[7] C.P. Fu and S.C. Liew. TCP veno: 'TCP enhancement for transmission over wireless access networks. *IEEE Journal on Selected Areas in Communications*, 21(2):216–228, 2003.

[8] M. Gerla, M.Y. Sandidi, R. Wang, A. Zanella, C. Casetti, and S. Mascolo. TCP westwood: Congestion window control using bandwidth estimation. In *IEEE Globecom'01*, San Antonio, Texas, November 2001.

[9] L.A. Grieco and S. Mascolo. End-to-end bandwidth estimation for congestion control in packet networks. In A. Roveri M. Ajmone Marsan, G. Corazza. M. Listanti, editor, *Lecture Notes in Computer Science*, volume 2601/2003, pages 645–658. Springer-Verlag Heidelberg, 2003.

[10] Luigi Alfredo Grieco and Saverio Mascolo. TCP westwood and easy RED to improve fairness in high-speed networks. In *Protocols for High-Speed Networks*, pages 130–146, 2002.

[11] F. Gustafsson. *Adaptive Filtering and Change Detection*. Wiley, West Sussex, 2000.

[12] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, 2003.

[13] V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18:314–329, 1988.

[14] V. Jacobson. Congestion avoidance and control. In *Proc. of SIGCOMM*, volume 18.4, pages 314–329, 1988.

[15] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *IEEE/ACM Transaction on Networking*, 11(4):537–549, August 2003.

[16] H. Jiang and C. Dovrolis. Passive estimation of TCP round-trip times. *ACM SIGCOMM Computer Communication Review*, 32(3), 2002.

[17] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of Operational Research Society*, 49:237–252, 1998.

[18] F.P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[19] E.L. Lehmann. *Theory of Point Estimation*. Wiley series in probability and mathematicall statistics. Wiley, 1983.

[20] S. Liu, T. Basar, and R. Srikant. Controlling the Internet: A survey and some new results. In *Proc. 42nd IEEE Conference on Decision and Control*, pages 3048–3057, Maui, Hawaii USA, 2003.

[21] S. H. Low and R. Srikant. A mathematcial framework for designing a low-loss, low-delay Internet. *Networks and Spatial Economics*, January-February 2003. special issue on "Crossovers between Transportation Planning and Telecommunications".

[22] S.H. Low. A duality model of tcp and queue management algorithms. In *Proceedings of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, 2000.

[23] S.H. Low, L. Peterson, and L. Wang. Understanding Vegas: a duality model. *Journal of ACM*, 49(2):207–235, 2002.

[24] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. TCP Westwood: bandwidth estimation for enhanced transport over wireless links. In *MobiCom*, Rome, Italy, 2001.

[25] B. Melander, M. Björkman, and P. Gunningberg. A new end-to-end probing and analysis method for estimating bandwidth bottlenecks. In *IEEE Globecom'00*, San Francisco, CA, November 2000.

[26] J.C. Mogul. Observing TCP dynamics in real networks. In *Communications, Architectures & protocols*, pages 305–317, August 1992.

[27] S. Athuraliya nad V.H. Li, S.H. Low, and Q. Yin. Rem: active queue management. *IEEE Networking*, 15(3):48–53, 2001.

[28] F. Paganini, Z. Wang, S.H. Low, and J.C. Doyle. A new TCP/AQM for stability and performance in fast networks. In *Proceedings of IEEE Infocom 2003*, 2003.

[29] N.K.G. Samaraweera. Non-congestion packet loss detection for TCP error recovery using wireless links. *IEE Proceedings-Communications*, 146(4):222–230, 1999.

[30] P. Sarolahti, M. Kojo, and K. Raatikainen. F-RTO: an enhanced recovery algorithm for TCP retransmission timeouts. *ACM SIGCOMM Computer Communication Review*, 33(2), 2003.