

ACK-clock Dynamics in Network Congestion Control – An Inner Feedback Loop with Implications on Inelastic Flow Impact

Krister Jacobsson, Håkan Hjalmarsson and Niels Möller[†]

Abstract—The focus of this paper is the window mechanism in network congestion control, whereby packet acknowledgments control when new packets are being sent. This constitutes an inner control loop that so far has received little attention. We provide a novel model of this loop that bridges between the standard integrator link model and the more recent static link model. The model is in validation experiments shown to be accurate. It is also shown that as the amount of inelastic cross-traffic increases the dynamics of this inner loop becomes slower. This may influence overall performance and stability in scenarios with heavy inelastic flows.

I. INTRODUCTION

Fluid-flow modeling is by now generally accepted as a viable route to analysis and synthesis of complex network communication systems, including TCP/AQM schemes, [1], [2], [3]. It has been used both for the study of equilibrium properties such as fairness, throughput etc, by the means of (convex) optimization; and the study of dynamical properties such as e.g. stability and convergence using the framework of control theory. As shown by Kelly and coworkers in the late 90s in the remarkable paper [4], the key to success is to choose the appropriate level of aggregation and to explicitly model the congestion measure fed back from the network to the sources. In network fluid-flow modeling, network quantities such as the sources’ sending rates and states of the links are considered as continuous time signals; and the communication network is represented by their interconnection as a feedback system, see e.g. [2] and references therein.

All TCP schemes are window based, meaning that each sender controls its window size—the number of packets that are sent before waiting for an acknowledgment (ACK). The transmission of new packets is controlled or “clocked” by the stream of received ACKs by the transmission control that transmits a new packet for each received ACK, thereby keeping the number of outstanding packets, i.e. the window, constant. A network operating with such a regime is showed in Fig. 1. The window size is controlled in an outer loop, updated on a slower time scale than the inner loop, and the regulation of this quantity has received ample attention in the literature. An increase in the window size makes the inner loop inject additional packets in the network, while a decrease in window size inhibits transmission of new packets for the next few ACKs.

[†]Krister Jacobsson (krister.jacobsson@ee.kth.se), Håkan Hjalmarsson (hakan.hjalmarsson@ee.kth.se) and Niels Möller (niels.moller@ee.kth.se) are with the School of Electrical Engineering, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden.

This work was supported by European Commission through the project EuroNGI, by Swedish Research Council and by the project RUNES.

In equilibrium, the actual sending rate is the ratio between the window size and the round-trip delay. The key point is that it is the source *in cooperation* with the network (and implicitly with other users) that defines the sending rate in window based protocols—the source itself cannot set its rate explicitly by manipulating the window size only. This “auto-regulation” corresponds to the inner loop in Fig. 1. Several models have been proposed to describe the ACK-

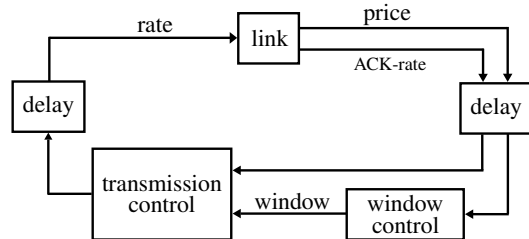


Fig. 1. Network loop in window based congestion control.

clock dynamics (often referred to as link or queue dynamics in the literature), see e.g. [1], [2], [5], [6]. In [7] a static model is proposed and validated. It is derived from the insight that, after a weak transient, the ACK-clock constrains the input rate at a link to the link capacity, independently of the size of the congestion window. In this contribution we refine this analysis and show that this loop can induce significant dynamics in the presence of inelastic cross-traffic, e.g. UDP flows; a scenario not considered for in [7]. Note that large heterogeneity among the window based sources’ round-trip delays may give transient behavior as well¹.

The TCP protocol is the predominant transport protocol on the Internet today, carrying about 83 % of the total traffic volume [8]. Non-TCP traffic includes real-time streaming media and other services with real-time requirements, such as IP telephony and multi-player games. This real-time non-TCP traffic can be expected to grow in the coming years. Congestion control is required for all Internet applications, including real-time applications [9], [10]. Real-time traffic may use congestion control mechanisms that are similar to those of TCP, or admission control and constant sending rates for admitted sessions. For the foreseeable future, Internet traffic is likely to remain a mix of elastic flows from TCP-like sources, and inelastic flows with sending rates independent of the TCP-like flows. We believe that it is important to

¹In fact, for analytical purposes, a UDP source can be considered as a window based source with infinite round-trip time.

take this inelastic cross-traffic into account when analyzing TCP congestion control, and it turns out that the qualitative behavior of a bottleneck queue, as experienced by a TCP endpoint, depends on the amount of inelastic cross-traffic.

The rest of the paper is organized as follows. In Section II we analyze how a change in the window affects the bottleneck queue under static traffic conditions. This is further explored in Section III where an ACK-clock model for the single source single bottleneck case is derived. In Section IV the model of the previous section is generalized to include multiple sources. Stability and convergence of the system is analyzed in Section V before we summarize our efforts and conclude in Section VI.

II. ACK-CLOCK DYNAMICS

A. Preliminaries

We consider N window based sources sending over a bottleneck link that may also be utilized by inelastic cross-traffic such as UDP.

Let $w_n(t)$ denote the congestion window of source n at time t , $n \in \{1, \dots, N\}$. The queue length (in time) at the bottleneck link is denoted $p(t)$, and $c > 0$ corresponds to the capacity of that link. Let a packet that is sent by the n th source at time t appear at the bottleneck queue at time $t + \tau_n^f$. This *forward delay* τ_n^f models the amount of time it takes to travel *from* source n *to* the link, and it accounts for forward latency but not queuing delays since (not modeled) non-bottleneck links are assumed to have empty queues. The *round-trip delay* seen by source n , is the elapsed time between a packet is sent and the corresponding acknowledgment is received. The *latency* of source n is denoted d_n and is defined as the minimum achievable round-trip delay, i.e. the round-trip delay when the bottleneck queue is empty. Let $x_c(t) \in [0, c]$ be the amount of *inelastic* cross-traffic (averaged over suitable time interval) utilizing the link. This implies that the *available bandwidth* in the perspective of the elastic sources sending over the link is $x_a(t) := c - x_c(t)$.

Equilibrium values are denoted with superscript $*$. When working in discrete time the convention $p_k := p(t_k)$ will be used.

B. A motivating example

In principal, previous work on modeling the ACK-clock dynamics can be divided into two different branches. Most existing literature assumes that a source's sending rate is proportional to its window size divided by the round-trip time, and models the link queue as a simple integrator, integrating the excess rate at the link [11], [12], [13], [14], [2]. More recent work, however, claims that due to the so called "self-clocking" (a new packet is sent into the network only when an ACK is received and the congestion window allows it) transient effects are negligible and it is more appropriate to model the link with a static relation [7], [15], [5]. The two different modeling strategies are fundamentally different: in the case of a change in a source's congestion window, the former gives a smooth queue transition while

the latter suggests an immediate proportional change in the queue. Both types of behaviors can be observed in the following example.

Example 2.1 (Convergence of the ACK-clock): In a window based algorithm the congestion window w is initially set to a constant size $w^* = 200$ packets and *not* updated dynamically (i.e. the outer feedback loop in Fig. 1 is broken). The source, having a round-trip latency $d = 100$ ms, is sending over a link with capacity $c = 50$ Mbit/s also utilized by 40 Mbit/s inelastic (UDP) cross-traffic which implies that the available bandwidth is $x_a^* = 10$ Mbit/s. We will also assume a packet size of 1040 bytes.

In equilibrium the sending rate of the source is equal to the available bandwidth, so

$$x_a^* = \frac{w^*}{d + p^*}. \quad (1)$$

Solving for the equilibrium queuing delay from (1) we get $p^* \approx 66.4$ ms corresponding to 399 packets. At time $t = 20$ s, the window size w is increased from 200 to 210 packets, i.e. $w(t) = w^* + \delta w$ for all $t \geq 20$ where $\delta w = 10$ packets. The new queue equilibrium becomes $p' = p^* + \frac{\delta w}{x_a^*} \approx 75$ ms corresponding to $p' = 449$ packets. The above described scenario is simulated in NS-2 and the evolution of the queue is shown as the solid grey line in Fig. 2. It is observed that it

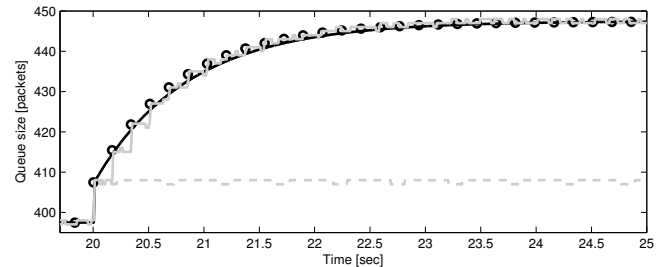


Fig. 2. Dynamics of the ACK-clock when $x_a^* = c/5 = 10$ Mbit/s. Solid grey line: NS-2 simulation. Circles: Discrete model (8). Solid black line: Continuous model (11). Dashed grey line: NS-2 simulation when $x_a^* = c = 10$ Mbit/s.

takes about three to four seconds for the system to converge after the perturbation, or around 20 round-trip times.

The dashed grey line in Fig. 2 shows the queue size in a similar NS-2 simulation as previously described but where the available bandwidth is set equal to the capacity $x_a^* = c$, so no cross-traffic is present, and $w^* = 500$ packets. In this case the system settles in one round-trip time. ■

From Example 2.1 it is observed that the dynamics of the system is fundamentally changed when non-window based cross-traffic is applied. To authors' knowledge this behavior is not captured in previous work, and it seem to be a key in the understanding of the system and suggests that such traffic should be taken into consideration in the modeling. A key observation is also that it may take several round-trip times for the ACK-clock to settle, implying that the ACK-clock dynamics cannot be neglected when evaluating dynamical properties such as stability. In the following subsection we explain the observations made in the example by deriving a

model for the simplest case possible, a single window based source sending over a single bottleneck link.

C. The effect of a window change

Consider the case of a single source sending over a bottleneck link, i.e. $N = 1$. Assume the system is in equilibrium initially and that the window control is disabled (or working on a substantially slower time scale than the ACK-clock). To ease notation, ignore for now the subscripts n . The effect of a congestion window change is studied by applying a perturbation δw to the congestion window at time $t = 0$. It is assumed that the transmission control realizes the change immediately by injecting δw packets into the network if $\delta w > 0$, or simply disregards $|\delta w|$ arriving acknowledgments if $\delta w < 0$. Since the bottleneck link is completely utilized initially the immediate effect of the window perturbation is a proportional change in the queue size at time $t_0 = \tau^f$, so

$$p(t_0) = p^* + \frac{\delta w}{c}. \quad (2)$$

This implies that during the next $d + p(t_0)$ seconds $w^* + \delta w$ packets deriving from the elastic source will arrive at the bottleneck, or the average rate over that time interval is

$$\begin{aligned} \bar{x}(t_0) &:= \frac{w^* + \delta w}{d + p(t_0)} = \frac{w^* + \delta w}{\frac{w_a^*}{x_a^*} + \frac{\delta w}{c}} = x_a^* \frac{w^* + \delta w}{w^* + \frac{x_a^*}{c} \delta w} \\ &= \begin{cases} = x_a^* & \text{if } \delta w = 0 \quad \text{or} \quad x_a^* = c, \\ > x_a^* & \text{if } \delta w > 0 \quad \text{and} \quad x_a^* < c, \\ < x_a^* & \text{if } \delta w < 0 \quad \text{and} \quad x_a^* < c. \end{cases} \quad (3) \end{aligned}$$

Note that the available bandwidth x_a^* is bounded above by the capacity c by definition. We observe that when no inelastic cross-traffic is present, i.e. $x_a^* = c$, the queue input rate equals the capacity $\bar{x}(t_0) = c$, and the queue settles already at time t_0 (the instant the window perturbation affects the queue), $p(t) = p(t_0)$ for all $t > t_0$.

In Example 2.1 we noted that when inelastic sources are sending over the bottleneck the effect of the window perturbation reaches beyond one round-trip time. The evolution of the bottleneck queue when started in equilibrium and subject to a change in the congestion window at $t = 0$, is described by the discrete time dynamical system

$$t_{k+1} = t_k + d + p_k \quad (4)$$

$$p_{k+1} = p_k + \frac{1}{c}(t_{k+1} - t_k)(\bar{x}_k - x_a^*) \quad (5)$$

$$\bar{x}_k = \frac{w^* + \delta w}{t_{k+1} - t_k} = \frac{w^* + \delta w}{d + p_k} \quad (6)$$

initiated at $(t_0, \bar{x}_0, p_0) = (\tau^f, x_a^*, p^* + \frac{\delta w}{c})$. Solving this system for p_k yields

$$p_k = p^* + \frac{\delta w}{x_a^*} - \frac{\delta w}{x_a^*} \left(1 - \frac{x_a^*}{c}\right)^{k+1} \quad (7)$$

where the two first terms correspond to the new equilibrium queue and the last product to the transient.

Based on the previous discussion it seem clear that for the single source single link case with *no* cross-traffic, the

ACK-clock dynamics is a pure time-delay as suggested in [5]. This is a consequence of the properties of the ACK-clock motivating the authors in [7], naturally since the ACK-rate never can exceed the capacity of the bottleneck link, the sending rate of the source is bounded as well. From (7) it is evident that the convergence of the queue depends crucially on the ratio $\frac{x_a^*}{c}$ only, i.e., the proportion of the link capacity which is available for the elastic traffic. This is due to that, when cross-traffic is present, it actually is possible for a window based source to (at least temporary) affect its packet rate through the bottleneck and hence the ACK rate. In fact, the more cross-traffic the heavier the transient is, and the more distinguishing the integrating effect in the buffer becomes. We will quantify this further in Section V-B.

III. SINGLE SOURCE – SINGLE BOTTLENECK MODEL

In this section we derive and validate a model of the dynamical interaction between a single source's ACK-clock and the queue of the bottleneck it utilizes.

A. Discrete time ACK-clock model

Study the evolution of the queue at time instants t_k where $t_{k+1} = t_k + d + p_k$. The amount of data from the source that arrives to the queue in the interval $[t_k, t_{k+1}]$ corresponds to the window size at the end of that interval, or more precisely, the source's window size at time $t_{k+1} - \tau^f$. The net flow of the queue corresponding to the difference between the inelastic cross-traffic and the serving rate $x_{c,k} - c = -x_{a,k}$ is also contributing to the change in the queue with its integral over time. Here $x_{c,k}$ and $x_{a,k}$ are the average rates over the interval $[t_k, t_{k+1}]$. Summing up, the queue is updated according to

$$\begin{aligned} p_{k+1} &= p_k + \frac{\bar{w}_{k+1}}{c} - \frac{d + p_k}{c} x_{a,k} \\ &= \left(1 - \frac{x_{a,k}}{c}\right) p_k + \frac{\bar{w}_{k+1}}{c} - \frac{d}{c} x_{a,k}, \quad (8) \end{aligned}$$

where we have introduced

$$\bar{w}_k = w(t_k - \tau^f) \quad (9)$$

to model that the window size change does not affect the queue until after τ^f seconds. We emphasize that the sample time $T_k := d + p_k$ of the discrete time dynamical system (8) is non-uniform, and that the system is nonlinear since the available bandwidth $x_{a,k}$ is considered as an external input.

B. Continuous time ACK-clock model

In network fluid-flow modeling working in continuous time is common practice [1], [2], [4]; to capture in a model that congestion control protocols often operates in different (time varying) time scales, this is often convenient.

With (4) in mind, rewriting (8) as

$$\frac{p_{k+1} - p_k}{t_{k+1} - t_k} = \frac{1}{c} \left(\frac{\bar{w}_k}{d + p_k} - x_{a,k} \right) + \frac{1}{c} \frac{\bar{w}_{k+1} - \bar{w}_k}{t_{k+1} - t_k}, \quad (10)$$

and using a first order Euler approximation of the derivative together with (9), an approximative continuous time model of the ACK-clock dynamics becomes

$$\dot{p}(t) = \frac{1}{c} \left(\frac{w(t - \tau^f)}{d + p(t)} - x_a(t) \right) + \frac{1}{c} \dot{w}(t - \tau^f). \quad (11)$$

C. Model validation

The discrete time model (8) and the continuous time model (11) are validated against packet-level data in the following examples. The scenario when no cross-traffic is present is not presented here since both models become pure delays in that case (as they should).

Example 3.1 (A positive window step): The two models (8) and (11) are simulated using the first experiment configuration described in Example 2.1 (a positive step in the window size, cross-traffic present). The circles in Fig. 2 show the queue size when the discrete model (8) is simulated. The solid black line is the queue size when the continuous model (11) is simulated. Comparing these two lines with the solid grey line corresponding to the queue size when the system is simulated in NS-2, we see that both models capture the dynamics accurately. ■

Example 3.2 (A negative window step): The scenario is the same as in Example 2.1 but with $x_a^* = 100$ Mbit/s, $c = 1000$ Mbit/s, $d = 200$ ms, $w^* = 2510$ packets, $\delta w = -100$ packets. The solid grey line in Fig. 3 shows the queue size when the system is simulated in NS-2, the circles the model (8) and the solid black line (11). The model fit is

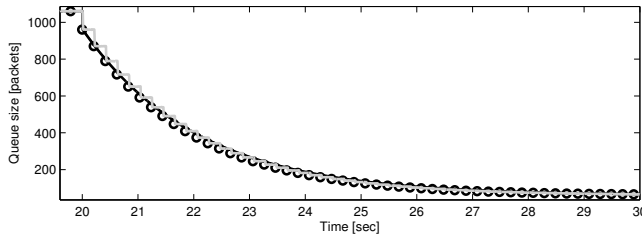


Fig. 3. Validation experiment. Solid grey line: NS-2 simulation. Circles: Discrete model (8). Solid black line: Continuous model (11).

very good for both models for this configuration. ■

Example 3.3 (A dynamic cross-traffic scenario): In this example the window size is kept constant at $w^* = 1510$ packets while the inelastic cross-traffic (UDP) is varied according to $x_a(t) = 125$ Mbit/s for $t < 20$ and $x_a(t) = 50$ Mbit/s when $t \geq 20$. Capacity is set to $c = 200$ Mbit/s and latency is $d = 100$ ms. The system is simulated in NS-2. The solid grey line in Fig. 4 represents the NS-2 simulation, and circles and the solid black line simulations of (8) and (11) respectively. The discrete model (8) is very accurate, while the Euler approximation (11) lags. This is explained by the fact that the packets in flight at time t , that defines the queue input rate during the interval $[t, t + d + p(t)]$, is independent of the evolution of the queue during that interval. Noting that $w^*/(d + p(t))$ is convex w.r.t t in this scenario, it is straightforward to verify this behavior

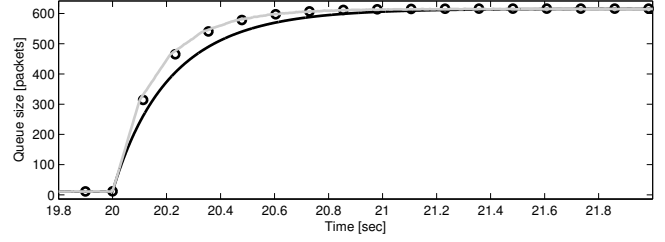


Fig. 4. Validation experiment. Solid grey line: NS-2 simulation. Circles: Discrete model (8). Solid black line: Continuous model (11).

theoretically; the continuous time model queue size is less than or equal to the discrete version in this case. ■

From the given examples and additional experiments not presented here due to space limitations we conclude that both the discrete time model and its continuous time approximation captures the ACK-clock dynamics accurately. However, (8) is slightly superior to (11), this is natural due to the additional approximation made when deriving (11).

IV. MULTIPLE SOURCES – SINGLE BOTTLENECK MODEL

We continue this section by generalizing the single flow single bottleneck ACK-clock model to also incorporate several flows, i.e. $N \geq 1$.

A. ACK-clock modeling

When deriving the single source single bottleneck discrete time ACK-clock model (8) the only assumption made was that changes in window control are realized instantaneously by the transmission control. For that case, the sampling time was naturally chosen as the round-trip time. This is, however, more intricate when studying multiple sources. Individual window based flows operate in per round-trip time time scale which may be heterogeneously distributed in a network. We will hence derive a continuous time model. At this stage we make the additional assumption that packets are perfectly evenly spaced out in time and, hence, that flows are considered as fluids. This yields that for a time interval less than or equal to the smallest round-trip time, i.e. $t_{k+1} - t_k \leq \min(d_n) + p_k$, the queue is updated according to

$$p_{k+1} = p_k + \frac{1}{c} (t_{k+1} - t_k) \sum_{n=1}^N \frac{\bar{w}_{n,k}}{d_n + p_k} - \frac{1}{c} (t_{k+1} - t_k) x_{a,k} + \frac{1}{c} \sum_{n=1}^N (\bar{w}_{n,k+1} - \bar{w}_{n,k}). \quad (12)$$

This can be seen as the generalized version of (8). Observe that if latency of all sources are homogeneous, $d_n = d$ for all n , flows are synchronous and the results in Section III holds. Subtracting $p(t_k)$ from both sides of (12) and dividing with the sample interval, it is straightforward to use an Euler approximation of the first order derivative to arrive at

$$\dot{p}(t) = \frac{1}{c} \sum_{n=1}^N \left(\frac{w_n(t - \tau_n^f)}{d_n + p(t)} + \dot{w}_n(t - \tau_n^f) \right) - \frac{x_a(t)}{c} \quad (13)$$

recalling (9).

B. Model validation

Example 4.1 (Steps in windows): In this example three sources, i.e. $N = 3$, with window sizes $w_n^* = \frac{380}{n}$ packets are sending over the bottleneck. At times 20, 22 and 24 s respectively the congestion windows are subject to perturbations $\delta w_n = \frac{(-1)^{n+1}}{n} 75$ packets. Furthermore $x_a^* = 60$ Mbit/s, $c = 300$ Mbit/s, and $d_n = \frac{150}{n}$ ms. Fig. 5 shows the queue size when the system is simulated in NS-2 (solid grey line) and when the model in (13) is simulated (solid black line). It is observed that despite that we have the (not modeled)

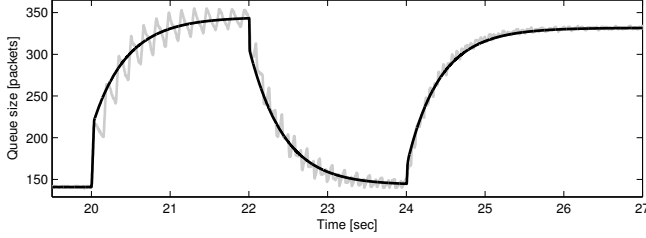


Fig. 5. Validation experiment. Solid grey line: NS-2 simulation. Solid line: Continuous model (13).

packet level high frequency queue fluctuations deriving from that packets are not perfectly smoothed out in time, the model fit is good. ■

Example 4.2 (Sinusoidal windows): Five window based sources ($N = 5$) are sending over a bottleneck link with capacity $c = 75$ Mbit/s and where UDP cross-traffic is such that the available bandwidth is $x_a^* = 50$ Mbit/s. Each flow n has latency $d_n = \frac{120}{n}$ ms. Let $t_n = 0.5(n - 1) + 20$. The window function varies over time as: $w_n(t) = \frac{250}{n}$ for $t < t_n$, and $w_n(t) = \frac{250}{n} + 60 \frac{(-1)^{n+1}}{n+1} \sin\left(\frac{\pi}{n}(t - t_n)\right)$ otherwise. The solid grey line in Fig. 6 shows the queue size when the system is simulated in NS-2. The solid black line shows (13) when simulated. According to the plot the model seems

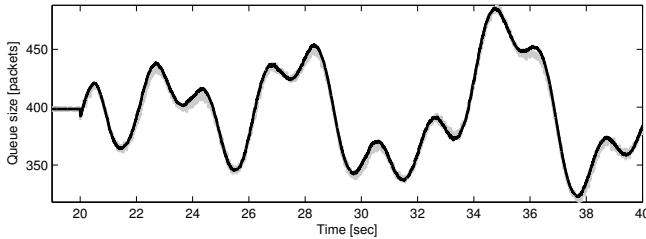


Fig. 6. Validation experiment. Solid grey line: NS-2 simulation. Solid line: Continuous model (13).

accurate. ■

C. Model summary

In Fig. 5 the direct proportional effect of a change in the window is characteristic; it is, however, also evident that the dynamics is *not* static when UDP cross-traffic is present. Recalling the result in [7], that ACK-clock dynamics is static, we conclude that: inelastic cross-traffic slows down the ACK-clock dynamics considerably in the presence of inelastic cross-traffic. According to the validation experiments, (13) is

an accurate description of the ACK-clock dynamics for the case of multiple sources sending over a single bottleneck; taking into account both the immediate effect a window change has on a bottleneck link *and* the more long term effect it has on the sources sending rates. We therefore propose

$$\dot{p}(t) = \begin{cases} \frac{x(t) - x_a(t)}{c}, & \text{if } p(t) > 0 \text{ or } x(t) > x_a(t), \\ 0, & \text{otherwise.} \end{cases} \quad (14)$$

$$x(t) := \sum_{n=1}^N \frac{w_n(t - \tau_n^f)}{d_n + p(t)} + \sum_{n=1}^N \dot{w}_n(t - \tau_n^f) \quad (15)$$

as a novel fluid-flow model describing the inner loop dynamics in Fig. 1, i.e. the ACK-clock-link interaction.

V. ANALYSIS

A. Stability

Since $x_a(t_k) \leq c$ for all t_k by definition it is trivial to see that the single source single bottleneck model (8) is stable (note that the case $x_a(t_k) = 0$ must be treated separately). Clearly working in discrete time using the transformed time scale defined by the sequence $\{t_k\}$ is suitable for stability analysis in this case. Considering the multi source case the analysis gets slightly more involved due to that we need to work in absolute time scale; the system, however, is still stable, according to the following theorem.

Theorem 5.1: The ACK-clock-link dynamics of multiple window based sources sending over a single bottleneck link modeled by (14) is globally stable.

Proof: When studying stability of (14) the congestion windows w_n and the available bandwidth x_a are considered as constants, and naturally only $p(t) \geq 0$ is accounted for.

Start with the case when the configuration is such that

$$\sum_{n=1}^N \frac{w_n}{d_n} \leq x_a, \quad (16)$$

then (14) becomes

$$\dot{p}(t) = \begin{cases} \frac{1}{c} \left(\sum_{n=1}^N \frac{w_n}{d_n + p(t)} - x_a \right), & \text{if } p(t) > 0, \\ 0, & \text{if } p(t) = 0. \end{cases} \quad (17)$$

Since $\dot{p}(t)$ is negative for all $p(t) > 0$ we conclude that $p^* = 0$ is an equilibrium, and furthermore that it is unique and stable.

When (16) is not fulfilled, for all $p(t) \geq 0$, (14) reduces to

$$\dot{p}(t) = \frac{1}{c} \left(\sum_{n=1}^N \frac{w_n}{d_n + p(t)} - x_a \right). \quad (18)$$

As the right hand side of (18) is monotonically decreasing as a function of p , it exists a unique positive equilibrium p^* for this case. Since that $\dot{p}(t) > 0$ for $p(t) < p^*$ and $\dot{p}(t) < 0$ when $p(t) > p^*$, it follows that the system converges towards p^* for all nonnegative initial conditions. ■

Remark 5.1: Based on the previous proof we can furthermore conclude that there exist no oscillatory trajectories.

We sum up by concluding that the ACK-clock, i.e. the inner loop in Fig. 1, is stable. This is also in line with all simulations performed and with the results in [16].

B. Convergence

It is of interest to quantify how the amount of inelastic cross-traffic affects the convergence of the inner loop, the ACK-clock, when the system is subject to a step change in the congestion window.

Assume that the cross-traffic is static, $x_a(t) = x_a^*$, then the queue evolves according to (7) after a window change of δw packets at time $t = -\tau^f$.

Definition 5.1 (Rise time): Consider a system initially settled at the origin and subject to a step in the input, define the *rise time* T_r as the time it takes for the output to first reach 90% of its final value. ■

Definition 5.2 (Settling time): Consider a system initially settled at the origin and subject to a step in the input, define the *settling time* T_s as the earliest time after which the output remains within $\pm 1\%$ of its final value. ■

From (7) we get that

$$T_r = \frac{\ln(0.1)}{\ln\left(1 - \frac{x_a^*}{c}\right)} - 1, \quad T_s = \frac{\ln(0.01)}{\ln\left(1 - \frac{x_a^*}{c}\right)} - 1, \quad (19)$$

which are plotted as functions of the available bandwidth (in fractions of the capacity) in Fig. 7. We see that for available

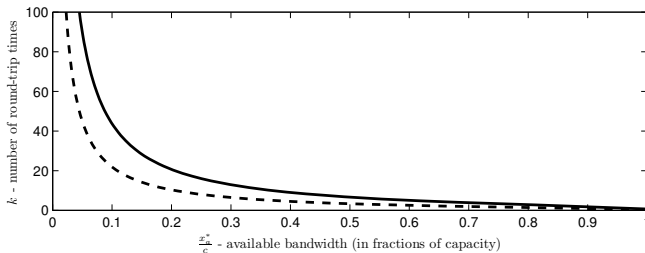


Fig. 7. Convergence of the ACK-clock. Dashed line: rise time T_r . Solid line: settling time T_s .

bandwidths less than about half the capacity, more than six round-trip times are needed for the system to converge. Furthermore, when only a fraction of ten per cent and less of the capacity is available for the elastic source the convergence is very slow.

Example 5.1: In Example 2.1 the available bandwidth was configured to one fifth of the capacity, i.e. $\frac{x_a^*}{c} = 0.2$. From Fig. 7 (solid line) or directly from (19) we expect that it takes about 20 round-trip times before the system converges to the final value. This also agrees with the conclusion based on the observations made in the example. ■

VI. SUMMARY AND CONCLUSIONS

In this paper we have provided a novel accurate description of the dynamical interaction between the sources' ACK-clocks and a bottleneck link in a packet-switched communication network. The model is derived from a careful study of the system at packet-level and it takes into account both

the immediate effect a window change has on a bottleneck link and the more long term effect it has on the sources' sending rates. It agrees with the known static link model when sources' round-trip delays are similar and no inelastic cross traffic is present, and is analogous to the standard integrator link model when the heterogeneity of round-trip delays is significant and/or in the presence of heavy inelastic flows. The model is rigorously validated towards packet-level data using NS-2.

It is observed and analyzed what impact non-responsive flows have on the convergence of the bottleneck queue (and hence implicitly the sources' sending rates). Quantitative results show that with increased proportion of inelastic cross-traffic, it takes longer time for the queue to stabilize after a window change. This may influence overall performance and stability and should be taken into consideration in the synthesis and analysis of the window control mechanism.

We remark that all results are independent on the window control algorithm and hence valid for *all* congestion control protocols applying a congestion window mechanism.

REFERENCES

- [1] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proceedings of SIGCOMM '00*, 2000, pp. 151–160.
- [2] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," *Control Systems Magazine*, vol. 22, no. 1, pp. 28–43, February 2002.
- [3] F. Paganini, Z. Wang, J. C. Doyle, and S. H. Low, "Congestion control for high performance, stability, and fairness in general networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 43–56, 2005.
- [4] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.
- [5] J. Y. Choi, K. Koo, J. S. Lee, and S. Low, "Global stability of FAST TCP in single-link single-source network," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Seville, Spain, December 2005, pp. 1837–1841.
- [6] P. Tinnakornsrisuphap and A. M. Makowski, "Limit behavior of ECN/RED gateways under a large number of TCP flows," in *Proceedings of IEEE Infocom 2003*, 2003.
- [7] J. Wang, D. X. Wei, and S. H. Low, "Modelling and stability of FAST TCP," in *Proceedings of IEEE Infocom 2005*. IEEE, March 2005.
- [8] M. Fomenkov, K. Keys, D. Moore, and kc claffy, "Longitudinal study of Internet traffic from 1998–2003," <http://www.caida.org/outreach/papers/2003/nlanr/>, 2004.
- [9] S. Floyd, "Congestion control principles," RFC 2914, September 2000.
- [10] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 3550, July 2003.
- [11] E. Altman, C. Barakat, and V. Ramos, "Analysis of AIMD protocols over paths with variable delay," in *Proceedings of IEEE Infocom 2004*, 2004.
- [12] F. Baccelli and D. Hong, "AIMD, fairness and fractal scaling of TCP traffic," in *Proceedings of IEEE Infocom 2002*, 2002.
- [13] C. Hollot, V. Misra, D. Towsley, and W. B. Gong, "A control theoretic analysis of RED," in *Proceedings of IEEE Infocom 2001*, Anchorage, USA, April 2001, pp. 1510–1519.
- [14] S. Liu, T. Basar, and R. Srikant, "Pitfalls in the fluid modeling of RTT variations in window-based congestion control," in *Proceedings of IEEE Infocom 2005*. IEEE, March 2005.
- [15] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proceedings of IEEE Infocom*. IEEE, March 2004.
- [16] N. Möller, K. H. Johansson, and K. Jacobsson, "Stability of window-based queue control with application to mobile terminal download," in *CD-ROM Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, July 2006.