Stability of Congestion Control Algorithms Using Control Theory with an application to XCP

Ioannis Papadimitriou (jpg@stanford.edu)
George Mavromatis (gmavr@stanford.edu)

1. Introduction

During recent years, a lot of work has been done towards the theoretical understanding of TCP and especially its congestion control algorithm. Congestion control is viewed as a feedback system, described by a deterministic flow model approximation. Its stability properties are examined through applying control theory techniques upon the appropriate differential equations. The purpose for this analysis is to develop new algorithms with well-known behavior against the variations of the network parameters, such as round-trip delays, capacity (long-term variations), number of users and requested data rate (short-term variations). Hence, there is a strong motivation for designing new congestion control algorithms that provide stability guarantees, robustness to delay and scalability to large capacities and arbitrary topologies.

There are two forms of congestion control, the primal and the dual. The former is based on the users controlling their sending rates depending on the congestion indication feedback signals they get from the network resources (routers). The latter is implemented by the resources through gathering information from the flows that are using them. Primal algorithms are more popular because they get more easily distributed. TCP for example is basically a primal congestion control algorithm, although its current version has dynamics in both sides.

In [1], which is considered to have initiated all this discussion, the following primal fluid-flow equation is proposed:

$$\dot{x}_r(t) = \kappa\left(w_r - x_r(t) \cdot \sum_{j \varepsilon r} \mu_j(t) \right) \quad (1)$$

$$where \quad \mu_j(t) = p_j\left(\sum_{s:\, j \varepsilon s} x_s(t) \right)$$

There are different ways to describe these equations. From an economic point of view, users try to adjust their own sending rate x_r, so that they equalize the price per unit time w_r they are willing to pay with the total price that the network charges for servicing that much rate through the links that constitute their route. From a feedback viewpoint, let p_j denote the intensity of the signals that resource j sends as an indication of its congestion level. Then, equilibrium is reached when the additive increase term w_r (in bps) becomes equal to the multiplicative decrease factor Σp_j multiplied by $x_r(t)$. It has been proved, using an appropriate Lyapunov function, that this primal problem is globally stable. The analogous dual problem equations are also proposed.

In [2], the authors introduce delays to equation (1). Now, the equations are:

$$\dot{x}_r(t) = \kappa\left(w_r - x_r(t - D_r) \cdot \sum_{j \varepsilon r} \mu_j(t - D_{jr}) \right) \quad (2)$$

$$\mu_j(t - D_{jr}) = p_j\left(\sum_{s:\, j \varepsilon s} x_s(t - D_{jr} - D_{sj}) \right)$$

where D_{jr} is the delay from resource j to user r and D_{sj} is the delay from user s to resource j. The queuing delays are considered negligible (see [17] for a discussion about this), so only the propagation delays are taken into account. Assuming that all users have the same round-trip time (RTT), it is proved that the system is locally stable when each user has gain inversely proportional to the common RTT.

Papers [3] and [4] move one step forward and establish the conditions for having local stability for arbitrary round-trip delays. Nevertheless, it must be pointed out that all these proofs are based on the assumptions that the number of users does not change and that we only have one resource (link) shared by the users.

This discussion concerning the stability of delayed feedback systems found application in the theoretical analysis of congestion control algorithms, especially TCP and AQM implementations. In [6] the authors propose a framework for the description of networks that support active queue management with TCP flows through differential equations. In [7], this methodology is used in order to investigate the stability of RED, assuming that all users have the same RTT. It is proved that Tail Drop is unstable and that RED, although stable under certain conditions, imposes a tradeoff between fast response times and stability margins. Furthermore, for an overloaded system, the flows pay a double penalty of higher delay as well as higher loss. In [8], the authors describe guidelines for design of AQM systems that are both stable and efficient and propose two new algorithms aimed at solving the RED problems.

Following these guidelines, it is proved in [9] that the TCP/RED becomes unstable when users have different RTTs, when the delays increase and surprisingly when the link capacity increases. The authors also prove the stability conditions for one link and several heterogeneous users. Finally, they state that TCP/AQM can be considered to be a distributed primal-dual algorithm: the users adjust their sending rates, and the links

(resources) the loss/marking probabilities, both trying to maximize the aggregate source utility. Motivated by the fact that TCP performance will decline in the future because of increase in propagation delays and link capacities, reference [10] proposes a new set of implementation solutions and proves that the longer the propagation delay the slower the source control must be, to avoid instability.

Coming back to the proofs of stability for the system proposed by F.P. Kelly in [1], Glenn Vinnicombe investigates in [11] the stability criteria for multiple links and multiple heterogeneous users. In doing so, he proves that in such a system there is always a tradeoff between network performance and its speed of response. He finally applies his previous derivations to TCP-like algorithms.

Concluding this literature survey, we would like to present a number of open issues concerning congestion control stability. As mentioned above, all proofs (with the exception of the non-delayed system of [1]) refer to local stability, i.e very close to the equilibrium point. But this tells nothing about the global stability of the system, which means that we don't know if the network is stable, although there is convergence around each resource. Furthermore, strong assumptions and approximations have been used: negligible queuing delay, flow fluid approximation, fixed population of users, only one resource. As a result, there is a lot of discussion about the application of these theories to the real world. On the other hand, there is a new field for further research aimed at proving stability under fewer assumptions, designing new robust algorithms, describing new AQM implementations and investigating their parameters that lead to stable, fair and efficient networks.

The remaining of this report is organized as follows: chapter 2 is a brief summary of the eXplicit Control Protocol (XCP) proposed by Dina Katabi et al. in [12]. In chapter 3 we describe a methodology that can be used for the determination of the XCP parameters so that local stability of a system with one resource and N heterogeneous users is ensured. In chapter 4, we present the simulation results we got through the implementation of XCP in ns-2. Finally, the appendices contain the MATLAB simulink model we developed for two users, as well as the plots showing the instantaneous rate for each user when such a model is used.

2. XCP overview

Explicit Control Protocol (XCP) is a new congestion control algorithm aimed at solving the stability and efficiency problems that TCP and AQM are facing. Its main features are:

- The feedback that users get from a router is descriptive of the level of congestion of the router and not just a binary indication.

- There is an Efficiency controller (implementing a MIMD scheme) and a Fairness controller (implementing an AIMD scheme). They are decoupled, in order to provide stability and efficient utilization of the spare bandwidth.

- Packets carry the RTT and the current congestion window from the user to the routers as well as the feedback from the routers to the users. These three new fields constitute the congestion header.

- XCP has 2 fixed parameters (proposed values: $a = 0.4$ and $b = 0.226$)

When the XCP sender receives an acknowledgement, its congestion window is adjusted by just adding the feedback (can be positive or negative) carried by the acknowledgement to each current congestion window. The XCP receiver just copies the congestion header from the data packet to its acknowledgment.

In routers the algorithm works in three blocks. The first one gathers RTT and congestion window statistics through the incoming packets. The second one calculates the aggregate feedback (that must be sent to the users) and two constants that determine the way that this aggregate feedback will be divided in a fair way among the users. This block is executed once every average RTT, with the average RTT being updated according to the statistics gathered by the first block. The third block writes the appropriate feedback information to each outgoing packet.

The simulations show that XCP is more robust than TCP and most AQM schemes against varying traffic demands and round-trip times. It also seems to perform better as far as high utilization, small queue size and fair bandwidth allocation are concerned. The next chapter is about XCP's stability issues.

3. XCP Stability

In this chapter, we investigate the stability of the eXplicit Control Protocol (XCP). Let c be the capacity of a single link, N be the number of sources using that link (N-constant) and $r_i(t)$

be the sending rate of source i at time t. In [12], it is proved that, if all sources have the same round trip delay d, this system of one link and N users is stable, for any delay, when the parameters a and b satisfy the following conditions:

$$0<a<\frac{\pi}{4\sqrt{2}} \quad \text{and} \quad b=a^2\sqrt{2} \qquad (3)$$

We believe that the constant RTT assumption is very strong and we propose a methodology for finding the margins for a and b that lead the system to local stability.

The protocol can be viewed as a delayed feedback system, where the router informs the sources about the increase/decrease in congestion window size needed for the link to carry aggregate throughput equal to its capacity. The sources then adjust their rate according to this information, which they obtain through the acknowledgment packets that they receive. In [12], equations (15) – (17) describe the system when the RTT is constant. Moving one step forward, the following equations present the response of each heterogeneous user to the incoming feedback:

$$\frac{dr_i(t)}{dt}=\frac{1}{N}\left[\left(\dot{y}(t)\right)^{+}+0.1\frac{\sum_{s=1}^{N}r_i(t-d_{ri}-d_{sr})}{d(t-d_{ri})}\right]-$$

$$-\frac{r_i(t-d_i)}{\sum_{s=1}^{N}r_i(t-d_{ri}-d_{sr})}\left[\left(-\dot{y}(t)\right)^{+}+0.1\frac{\sum_{s=1}^{N}r_i(t-d_{ri}-d_{sr})}{d(t-d_{ri})}\right]$$

where

$$\dot{y}(t)=-\frac{a(\sum_{s=1}^{N}r_i(t-d_{ri}-d_{sr})-c)}{d(t-d_{ri})}-\frac{bq(t-d_{ri})}{d(t-d_{ri})^2}$$

and

$$\dot{q}(t)=\sum_{s=1}^{N}r_s(t)-c$$

where d_{ri} is the propagation delay from the router to source i, d_{ir} is the propagation delay from source i to the router, d_i is the round trip time of source i, $d(t)$ is the average round trip time calculated by the router and $q(t)$ is the size of the queue at time t. The notation $(dy/dt)^+$ is equivalent to:

$$\max(0,\dot{y}) \qquad (5)$$

When the system is exactly at equilibrium, the average round trip delay $d(t)$ calculated by the router is constant and equal to:

$$d(t)=d=\frac{\sum_{i=1}^{N}d_i}{N} \qquad (6)$$

Around the equilibrium we can assume that the above equation still holds. Simulations in Matlab showed that this constitutes a very good approximation. We also consider the queuing delay to be negligible compared to the propagation delay, which is also a good approximation, as stated by F.P. Kelly in [17]. Hence, the round trip delay of each user is constant and equal to $d_i = d_{ri} + d_{ir}$.

Additionally, the fairness rule states that when the link is requested to service more traffic than its capacity all users get the same bandwidth at the equilibrium point. Thus, towards the complete linearization of the system around the equilibrium point, we can also assume that the coefficient of $(-dy/dt)^+$ is equal to 1/N. Simulations in Matlab showed that this also is a reasonable assumption, as the system response remained practically unchanged (see Appendix B - Figure 10). Note that we do not make the same approximation for the next term, because then the system will have an eigenvalue equal to 0, which implies marginal and not asymptotic stability. As expected, Matlab simulations validated the above theoretical result. Taking all these into account, the simplified equation is:

$$\frac{dr_i(t)}{dt}=\frac{1}{N}\left[\dot{y}(t)+0.1\frac{\sum_{s=1}^{N}r_i(t-d_{ri}-d_{sr})}{d}\right]-0.1\frac{r_i(t-d_i)}{d}$$

where

$$\dot{y}(t)=-\frac{a(\sum_{s=1}^{N}r_i(t-d_{ri}-d_{sr})-c)}{d}-\frac{bq(t-d_{ri})}{d^2}$$

and

$$\dot{q}(t)=\sum_{s=1}^{N}r_s(t)-c$$

Note that now the $(dy/dt)^+$ and $(-dy/dt)^+$ terms have been unified into one term and no decision is needed anymore. Then, we write this equation for each user and make the following transformation:

$$x_i(t) = r_i(t) - \frac{c}{N} \qquad n = 1...N$$
$$x_q(t) = q(t)$$

where $x_i(t)$ refers to source i and $x_q(t)$ refers to the queue size. Taking the Laplace transform for each equation and writing everything in matrix form, we get:

$$A \bullet \begin{bmatrix} r_1 & r_2 & \dots & r_N & r_Q \end{bmatrix}^T = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix}^T$$

where

$$A = \begin{bmatrix} x_1 & w_{12} & \dots & w_{1N} & y_1 \\ w_{21} & x_2 & \dots & w_{2N} & y_2 \\ \dots & \dots & \dots & \dots & . \\ w_{N1} & w_{N2} & \dots & x_N & y_N \\ z_1 & z_2 & \dots & z_N & s \end{bmatrix}$$

$$x_i = s + \frac{a + 0.1 \cdot (N-1)}{N \cdot d} e^{-d_i s}$$

$$y_i = \frac{b}{N \cdot d^2} e^{-d_{ri} s}$$

$$z_i = -1$$

$$w_{ij} = \frac{a - 0.1}{N \cdot d} \cdot e^{-(d_{ri} + d_{jr}) s}$$

The system is locally asymptotically stable when all the solutions of the equation det[A] = 0 have real part smaller than 0. The problem from now on is not a control theory one, it is just about investigating the values for a and b that lead all roots to have negative real part. Nevertheless, such a computation is not at all trivial because of the exponentials. From now on, we describe our derivation for 2 users.

When N=2, matrix A is equal to:

$$A = \begin{bmatrix} s + \frac{a+0.1}{2 \cdot d} e^{-d_1 s} & \frac{a-0.1}{2 \cdot d} \cdot e^{-(d_{r1} + d_{2r}) s} & \frac{b}{2d^2} e^{-d_{r1} s} \\ \frac{a-0.1}{2 \cdot d} \cdot e^{-(d_{r2} + d_{1r}) s} & s + \frac{a+0.1}{2 \cdot d} e^{-d_2 s} & \frac{b}{2d^2} e^{-d_{r2} s} \\ -1 & -1 & s \end{bmatrix}$$

The corresponding characteristic function is:

$$0.1 \cdot a \cdot d \cdot e^{(d_{r1} + d_{r2}) s} \cdot s + (0.05 + 0.5a) d^2 (e^{(d_{1r} + 2d_{r1} + d_{r2}) s} +$$
$$+ e^{(d_{2r} + d_{r1} + 2d_{r2}) s}) s^2 + d^3 e^{(d_{1r} + d_{2r} + 2(d_{r1} + d_{r2})) s} s^3 +$$
$$+ b\{0.05 e^{(d_{1r} + d_{r1} + d_{r2}) s} + 0.05 e^{(d_{r1} + d_{r2} + d_{2r}) s} +$$
$$0.5 \cdot d \cdot e^{(d_{1r} + 2d_{r1} + d_{r2} + d_{2r}) s} s + 0.5 \cdot d \cdot e^{(d_{r1} + d_{1r} + d_{2r} + 2d_{r2}) s} s\}$$

In order to avoid the calculations with exponentials, we use the well-known 1st order Padé approximation:

$$e^{-ws} = \frac{-\frac{w}{2} s + 1}{\frac{w}{2} s + 1}$$

This generates a 9-order polynomial, whose coefficients are derived with the aid of Mathematica. The Matlab code we wrote investigates the solutions of the polynomial under different values for a, b and delays. For each specific value of a and b, we examine the roots of det[A] = 0 under different values for the delays d_{1r}, d_{r1}, d_{2r} and d_{r2}. If we derive a root with non-negative real part, this means that the system is unstable. In the following plot the red spots (highlighted region) correspond to a and b values that lead to instability for some combination of delay times and the white spots to a and b values which make the system stable for all the delay values we tried:

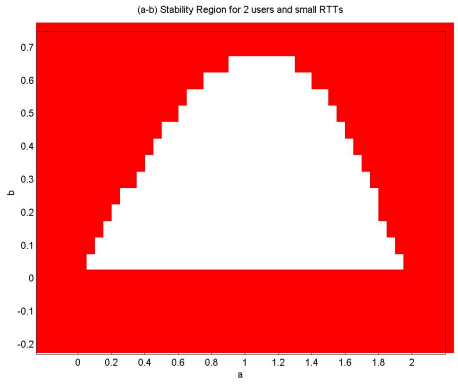

Figure 1: Plot showing the stability region for different values of a and b

This figure has been created with delays ranging from 0 to 20μsec and a 0.05 resolution for parameters a and b. The reason for choosing such values for the RTTs is that Padé's performance for high delays is not very good. Since the increase of propagation delays makes the system less stable, we expect that the region of stability depicted in Figure 1 will be smaller. Nevertheless, we ran the MATLAB program we wrote for delays of up to 6 sec and

found that the stability area is about the same, due to the fact that Padé approximation is "more relaxed" as delays increase. Any other more accurate approximation and methodology can be used in order to simplify the solution of the det[A] = 0 equation.

Furthermore, additional stability simulations for the two-users case have been made using the Simulink model shown in Appendix A. The plots found in Appendix B depict the behavior of the system under different values for the parameters *a* and *b* and the delays d_1 and d_2.

The reason for following the procedure described above to prove the conditions for stability is that we wanted to investigate all the values of a and b which make the system stable. In this way, we avoided imposing constraints in the a and b values as Dina Katabi does in [12] in order to simplify the amplitude and angle equations of the open-loop transfer function.

4. Simulations under ns-2

Design decisions and implementation

We implemented XCP as presented in [12] with the following changes:

- Users never request negative feedback when it occasionally happens that router sends positive updates that make the sender's congestion window larger than what is requested by the sender. This was found to cause instability and limit cycles especially when users are few.

- Estimation of queue occupancy is done over the complete control interval *d*, not just during the "last propagation delay" as mentioned in [12]. This part of the paper is unclear and potentially requires keeping significant state in the router, which defeats the purpose of simplicity of the protocol.

- Each sender maintains a low-pass filtered estimate of RTT. The TCP Tahoe RTT estimation ns-2 code was consistently found to estimate RTT wrongly for all configuration values of tcpTick attempted, and this was a major source of errors.

To the best of our knowledge we built a correct implementation of XCP. The following figure for example is similar to figure 12 of [12] derived for the adversarial case of two users having different RTTs:

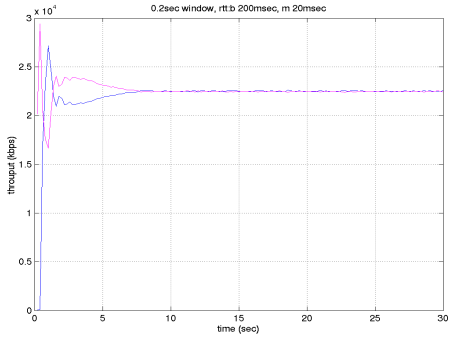

Figure 2

Simulation setup

In all cases mentioned below we used the topology of figure 2 of [12] where the link connecting the XCP router and the traffic sink is the bottleneck link. All users are connected to the traffic sink through this XCP router.

Performance observations

Our simulations were in general in accordance with Dina Katabi's claim that XCP is stable and achieves fairness and high efficiency when parameters *a* and *b* take the values 0.4 and 0.226 respectively. Our goal was to discover deviations from the ideal behavior described in [12] by applying appropriate adversarial traffic (large differences in RTTs and variable with time number of users).

- *Fairness:*

We found that XCP is generally fair to different flows even if they belong to significantly heterogeneous flows. However, when flows with different RTTs are applied, XCP sometimes discriminates among heterogeneous flows. In Figure 3, the average rates for users 1 and 2 were 1007 kbps and 987kbps respectively:

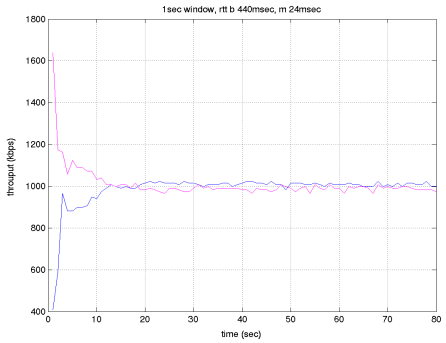

Figure 3

In Figure 4, the rates were 3200kbps and 2800kbps respectively:

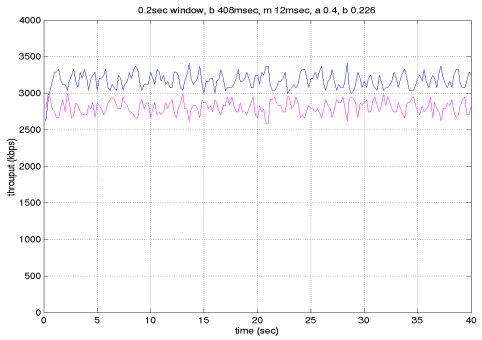

Figure 4

- *Stability and transient behavior:*

We tested the protocol with values for parameters *a* and *b* within and out of the stability region we derived in our previous analysis for two flows. Simulation indeed showed that flows eventually converge to a steady state for values chosen in this region. We also observed that as we come closer to the instability region the limit cycles have greater width, the utilization drops and fairness is not achieved. As expected, if we go far enough inside the unstable area oscillations are huge and throughputs do not converge. Additionally, higher values for parameter *a* lead to faster response of the system to changes of inputs. Therefore, there is a tradeoff in choosing the value for *a* between speed of response and stability margins.

The following scenario illustrates the impact of choosing parameter *a*. Seven users share an underutilized link. At time t = 10 an additional flow with small RTT starts sending packets. The following figures show that both faster response and higher utilization can be achieved for values of *a* that are greater than the maximum value proposed in [12]. Figure 5 is derived with $a = 0.4$ and Figure 6 with $a = 0.7$:

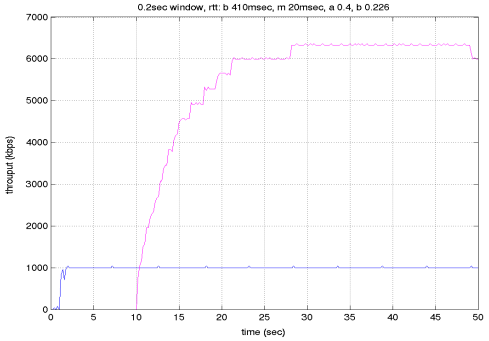

Figure 5

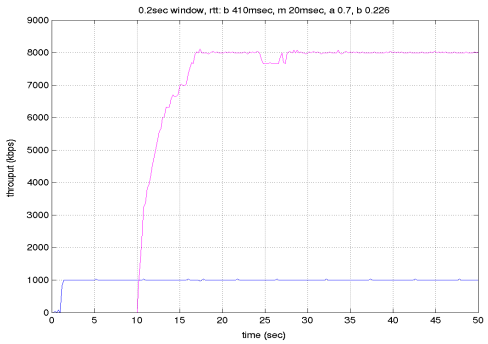

Figure 6

Note 1: The temporary drop in utilization is due to the concept of "shuffled traffic" used in XCP.

During the transient phase XCP exhibits significant oscillations in the users' sending rates, typical of a control system. Eventually these oscillations diminish and throughputs converge. Moreover traffic becomes interleaved and thus no high queue occupancies happen except in the transient phases following changes in the number of users (see the following figures as well as Figure 4):

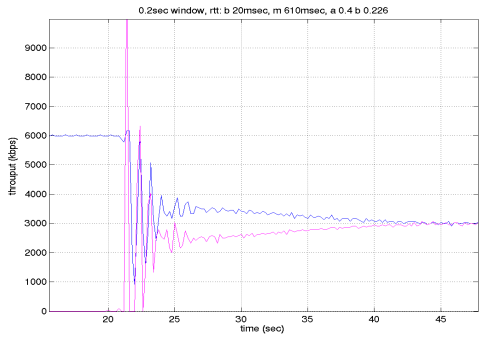

Figure 7

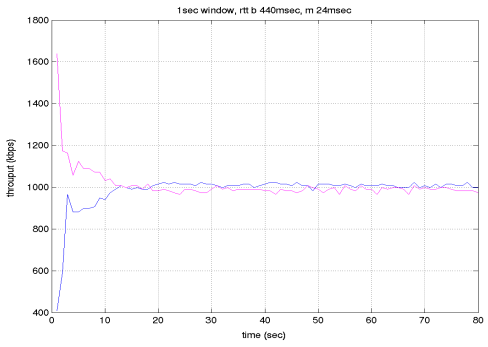

Figure 8

We observe that queueing delay during the transient period is not negligible. Queues grow

very large initially and then converge to small values.

Convergence to the steady state may last long even if there is available bandwidth, contrary to what stated in [12, par. 3.3]. In Figure 5, for example, the flow that enters at t = 10 needs 10-15 sec to reach its steady state data rate.

5. References

[1] F.P. Kelly, A.K. Maulloo, D.K.H. Tan. Rate control in communications networks: shadow prices, proportional fairness, and stability. 1998

[2] R. Johari and D. Tan. End-to-end congestion control for the internet: delays and stability. 2000

[3] L. Massoulie. Stability of distributed congestion control with heterogeneous feedback delays. Nov. 2000

[4] Glenn Vinnicombe. On the stability of end-to-end congestion control for the Internet. Dec. 2000

[5] S. Low, D. Lapsley. Optimization flow control, I: basic algorithm and convergence. Dec. 1999

[6] V. Misra, Wei-Bo Gong, D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. 2000

[7] C.V. Hollot, V. Misra, D. Towsley, Wei-Bo Gong. A control theoretic analysis of RED. April 2001

[8] C.V. Hollot, V. Misra, D. Towsley, Wei-Bo Gong. On designing improved controllers for AQM routers supporting TCP flows. April 2000

[9] S. Low, F. Paganini, J. Wang, S. Adlakha, J.C. Doyle. Dynamics of TCP/RED and a scalable control. July 2001

[10] F. Paganini, J. Doyle, S. Low. Scalable laws for stable network congestion control. 2001

[11] Glenn Vinnicombe. Robust congestion control for the Internet. 2002

[12] Dina Katabi, M. Handley, C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. 2002

[13] D. Bishop. Modern Control Systems, Addison Wesley, 8th edition

[14] J-J Slotine, W. Li. Applied non-linear control. Prentice Hall, 1991

[15] J.E. Marshall. Control of time-delay systems. New York 1979

[16] L. Dugard, E.I. Verriest. Stability and control of time-delay systems. London 1998

[17] F.P. Kelly. Models for a self-managed Internet. 2000

[18] “NS-2 network simulator” found at http://www.isi.edu/nsnam/ns/

APPENDIX A: SIMULINK Model for 2 sources

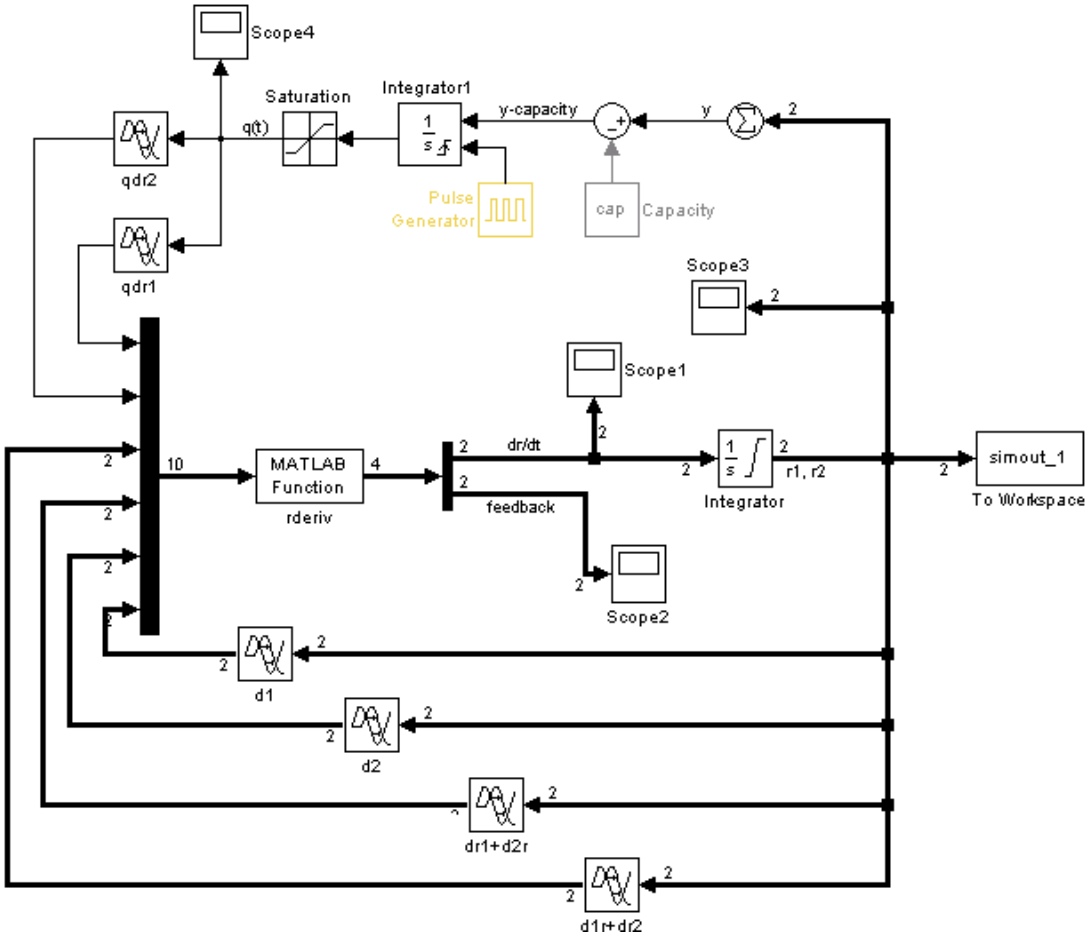

Figure 9: The Simulink model for 2 sources

The previous figure shows the model we developed with the aid of Simulink. Its main components are:

- The rate integrator, which receives the derivatives of rates from the *'rderiv'* component and outputs the instantaneous rates $r_1(t)$ and $r_2(t)$.
- The delay components, which take the two rates $r_1(t)$ and $r_2(t)$ as inputs and output a delayed version of them.
- The *'queue'* subsystem, which calculates the queue rate $q(t)$ and the instantaneous queue size *q(t)*, taking into account that the queue size can never be negative. Then, *q(t)* passes through the appropriate delay components and enters the *'rderiv'* component.
- The *'rderiv'* component, which receives the delayed version of queue size and rates, calculates the feedback and adjusts the rates $r_1(t)$ and $r_2(t)$ according to the XCP protocol. The following Matlab code has been written for the implementation of this functionality:

```
%Calculates the derivatives of rates
%   given the instantaneous queue size (delayed appropriately)
%   and the instantaneous rates (delayed appropriately)
function rderiv = rderiv(input)

global N cap d1 d2 a b;

%help variables
r1_d1=input(9);
r1_dr2d1r=input(3);

r2_d2=input(8);
r2_d2rdr1=input(6);

%queue size
q1 = input(1);
q2 = input(2);

%average rtt
d = (d1+d2)/N;

%feedback
fbk1 = -a/d*(r1_d1+r2_d2rdr1-cap)-b/d^2*q1;
fbk2 = -a/d*(r2_d2+r1_dr2d1r-cap)-b/d^2*q2;


%if choice is 0, then the rate derivatives calculations are done
%   under the assumption that the coefficient for [-y'(t)]+
%   is 1/N and not {r_i/(r_1+r_2)}
%if choice is 1, then no such assumption is made
choice = 1;

%calculation of rate derivatives (assumption is made, see above)
if choice == 0

    if fbk1>=0
        rderiv(1) = 1/N*(fbk1 + 0.1/d*(r1_d1+r2_d2rdr1)) -...
            (1/N)*(0+N*r1_d1*0.1/d);
    else
        rderiv(1) = 1/N*(0 + 0.1/d*(r1_d1+r2_d2rdr1)) -...
            (1/N)*(-fbk1+N*r1_d1*0.1/d);
    end

    if fbk2>=0
        rderiv(2) = 1/N*(fbk2 + 0.1/d*(r1_dr2d1r+r2_d2)) -...
            (1/N)*(0+N*r2_d2*0.1/d);
    else
        rderiv(2) = 1/N*(0 + 0.1/d*(r1_dr2d1r+r2_d2)) -...
            (1/N)*(-fbk2+N*r2_d2*0.1/d);
    end

%calculation of rate derivatives (without the assumption, see above)
else

    if fbk1>=0
        rderiv(1) = 1/N*(fbk1 + 0.1/d*(r1_d1+r2_d2rdr1)) -...
            r1_d1/(r1_d1+r2_d2rdr1)*(0+0.1/d*(r1_d1+r2_d2rdr1));

    else
        rderiv(1) = 1/N*(0 + 0.1/d*(r1_d1+r2_d2rdr1)) -...
            r1_d1/(r1_d1+r2_d2rdr1)*(-fbk1+0.1/d*(r1_d1+r2_d2rdr1));
    end


    if fbk2>=0
        rderiv(2) = 1/N*(fbk2 + 0.1/d*(r1_dr2d1r+r2_d2)) -...
            r2_d2/(r2_d2+r1_dr2d1r)*(0+0.1/d*(r1_dr2d1r+r2_d2));
    else
        rderiv(2) = 1/N*(0 + 0.1/d*(r1_dr2d1r+r2_d2)) -...
            r2_d2/(r2_d2+r1_dr2d1r)*(-fbk2+0.1/d*(r1_dr2d1r+r2_d2));
    end
end


%send feedbacks to output
rderiv(3) = fbk1;
rderiv(4) = fbk2;
```

Appendix B: Instantaneous rate plots

The following plots have been derived with the aid of the 'Scope3' component shown in Figure 9. Please, refer to Appendix B for the details about the Simulink model we used.

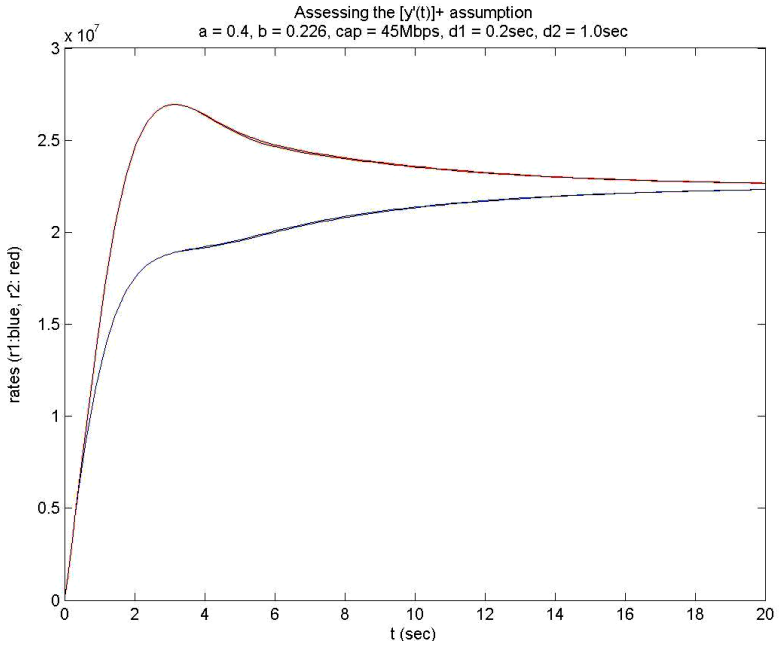

Figure 10: when we make the approximation that $[dy/dy]^+$ and $[-dy/dt]^+$ can be unified, we get the same rates for users 1 and 2 as in the normal case

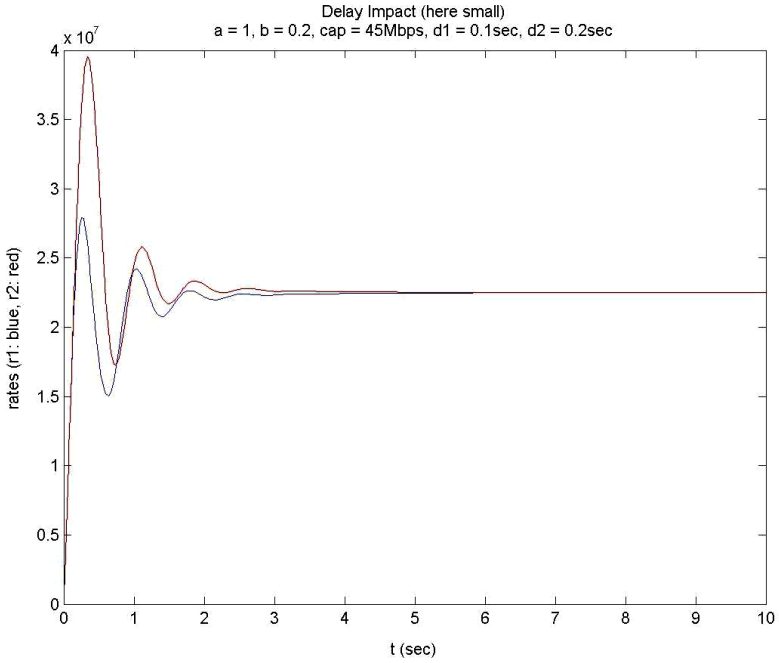

Figure 11: quick convergence when delays are small

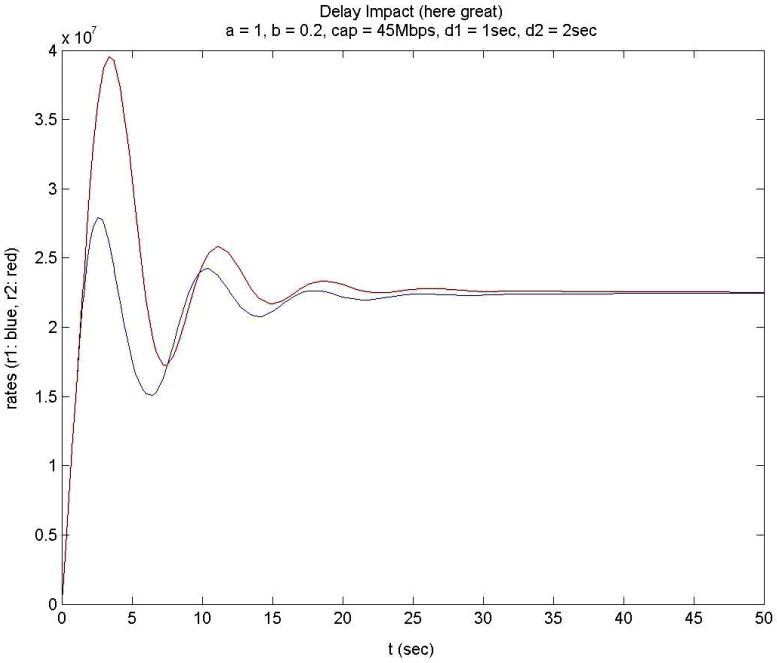

Figure 12: slow convergence for larger delays

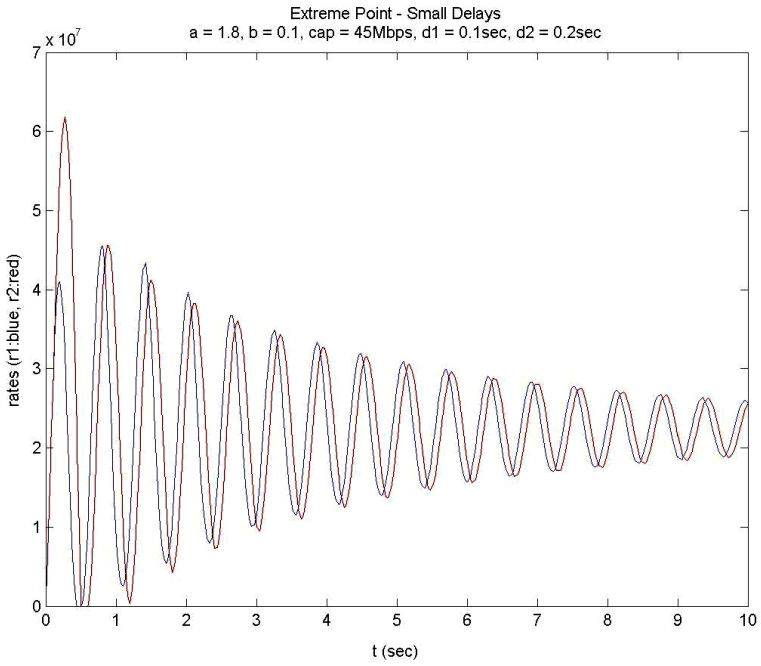

Figure 13: 'oscillatory' convergence when the values for a and b determine a point close to the boundary between stability and instability (small delays)

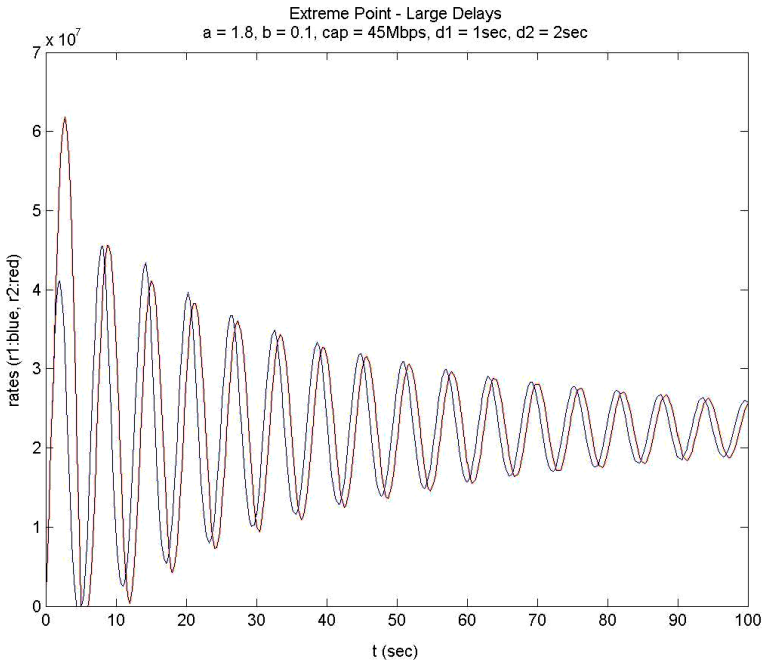

Figure 14: 'oscillatory' and slow convergence when the values for a and b determine a point close to the boundary between stability and instability(large delays)

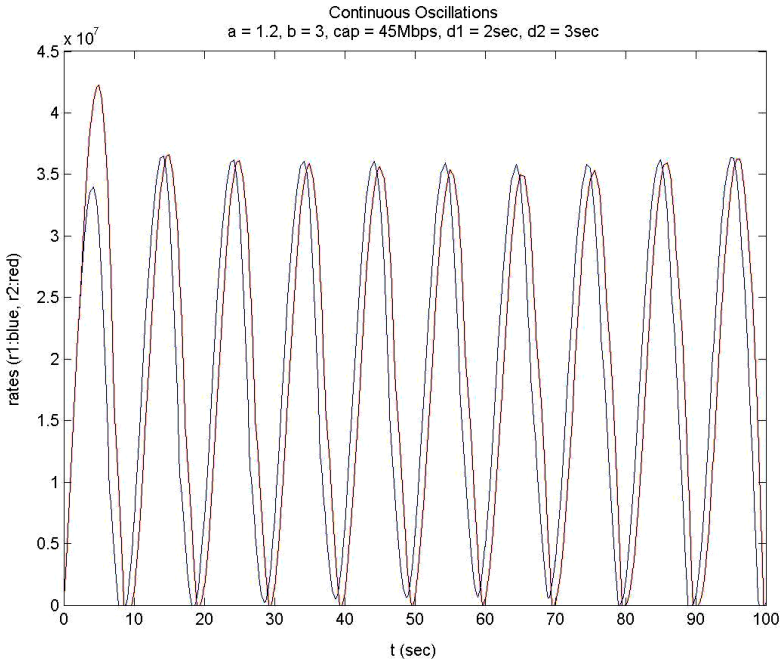

Figure 15: Continuous oscillations for an (a,b) point residing in the area of instability