Appendix A – Understanding, Improving and Applying Fluid-Flow Models

Appendix A Understanding, Improving and Applying Fluid-Flow Models

In order to insure efficient and stable operation of the Internet it is important to be able to estimate network performance characteristics under TCP traffic, which today constitutes the bulk of Internet data freight. The main obstacle to achieving this goal is posed by the dynamic nature of TCP congestion control: very complex collective behavior arises as a result of interactions between congestion control algorithms of concurrent flows. A good analogy that we will return to below is that of gas or fluid dynamics in which relatively simple interactions between molecules comprising the substance lead to familiar but complex bulk properties such as viscosity, temperature, and pressure, which are not easily understood or computed from the microscopic, molecular description.

As in fluid dynamics there are two possible approaches to gauging the properties of an aggregate network — by simulating the microscopic dynamics of transmission and transit of individual packets or by studying heuristically derived high-level approximations describing the packet traffic as a kind of a continuous substance flowing along the network links. The advantage of the former approach (adopted in the main body of this study) is that it yields very detailed information that is easily compared against traces collected on experimental test beds, for example, for verification. On the other hand, simulating a network of a realistic size over a large number of network parameter combinations may prove computationally infeasible. The fluid approximation models by comparison have very modest resource demands, although they are also less detailed. Fluid approximation models have another advantage over simulations in that they can sometimes give precise mathematical relationships between performance and network parameters, which can then be used as a guide in design of future networks and protocols as well as to improve the performance of current systems.

We begin in Sec. A.1 by introducing fluid-flow approximation models for TCP Reno flows and then we discuss the utility and limitations of such models. In Sec. A.2, we use fluid-flow approximation to develop response functions for TCP Reno, as well as CUBIC [56] and Compound [58] TCP and then compare the estimated equilibrium throughput of these alternatives. We close in Sec. A.3, where we outline future work related to fluid-flow approximation of Internet congestion control algorithms.

A.1 Fluid-flow Approximation Models

How would the Internet appear when visualizing packets as points moving along links and through routers? With hundreds of thousands of packets crossing a typical router every second they would appear to be an uninterrupted blur of motion, as if a fluid were flowing through pipes rather than a series of discrete packets flowing along links. This is the basic idea of fluid approximation: if the number of packets in the network is very large and they are moving very fast then the packet traffic will be well approximated by an abstract continuous stream.

Although each individual TCP flow behaves deterministically, aggregate dynamics, for a network of any significant size, will appear as nearly random. A good physical analogy is molecular dynamics in a volume of gas: while each molecule obeys simple Newtonian laws of motion, their collective behavior is essentially stochastic.

Describing the paths of individual molecules is a hopeless and ultimately useless endeavor; on the other hand, bulk properties such as pressure and temperature are neatly connected by the ideal gas equations. Similarly, one may hope that for a large collection of TCP flows the aggregate throughput may be related to the round-trip delay, capacity and buffer size in a simple way. Thus, the ultimate goal of fluid-flow approximation models is to develop a kind of TCP network thermodynamics.

We introduce fluid approximation by briefly describing the congestion window and throughput dynamics in the case of an isolated flow. To keep the presentation as simple and clear as possible we will discuss the most basic version of TCP Reno preserving only the most salient features of the protocol. That is, by TCP we will henceforth mean TCP Reno without selective acknowledgment, fast retransmit and fast recovery. Mathematical models incorporating these advanced features of TCP have been studied elsewhere [127]. For an isolated flow there can only be one bottleneck router along its path and so the rest of the path contributes only in terms of propagation delay. We assume further that the bottleneck router is positioned immediately in front of the source on the outgoing link. The general case is not substantially different. Let the router capacity at the bottleneck be denoted by C packets per second (pps), let its buffer size be B packets and the round-trip propagation delay T sec. If the variation in the round-trip time due to queuing at the router is negligible, the size of the congestion window in the congestion avoidance phase is approximated by the differential equation

$$\frac{dW}{dt} = \frac{1}{T} - \frac{1}{2}W(t)P(t), \tag{1}$$

where $W(t)$ interpolates the discrete congestion window size and $P(t)$ is a sum of delta functions $P(t) = \sum \delta(t - t_j)$, with t_j corresponding to moments of congestion window reduction due to detection of packet losses. In the case of an isolated connection the sequence t_j is periodic and can be computed explicitly. Note, also, that if the buffer is large T must be replaced with the equilibrium round-trip time that includes the equilibrium queuing delay.

If $B \approx CT$ and the variation in queuing delay is significant, then queue length has to be explicitly included in the model. Equation (1) then becomes

$$\frac{dW}{dt}(t) = \frac{1}{T + Q(t-T)/X} - \frac{1}{2}W(t)P(t) \tag{2}$$

$$\frac{dQ}{dt}(t) = \left(\frac{W(t)}{T + Q(t-T)/X} - X\right)\chi_{[0,B]}(Q(t)), \tag{3}$$

where $Q(t)$ is the buffer queue length at the router and $\chi_{[0,B]}(x)$ is the characteristic function of $[0,B]$.

While equations (1) and (2) describe the evolution of the congestion window, the quantity of real interest is usually the transmission rate. It is not hard to see that the rate at which the TCP sliding window[1] advances, and hence the transmission rate, is equal to $W(t)$ divided by the total round-trip delay $T+Q(t)/C$. So long as queuing delay is small relative to the total round-trip propagation time, the transmission rate, $X(t)$, can be approximated by $W(t)/T$. Dividing (1) through by T we have

$$\frac{dX}{dt} = \frac{1}{T^2} - \frac{1}{2}\sum X(t)P(t). \qquad (4)$$

Note that, as expected, transmission rate attains its maximum value C at $W=CT$ and stays at this value even as W continues to increase. This means that when the buffer size is comparable with the bandwidth-delay product $X(t)$ does not carry enough information to reconstruct the system state completely because it is impossible to compute $X(t)$ following a congestion window reduction given only the value of $X(t)$ before the reduction. In this case, throughput has to be computed by solving (2) first.

When considering the case of multiple flows, the complexity of the problem increases in two ways. First, as the number of flows increases, the simple periodic congestion window dynamics described above breaks down because packet loss now depends on the collective behavior of all flows. Consequently, the evolution of congestion windows becomes more and more complex and chaotic. The difficulty of the problem also increases with increasing complexity of the network structure. As the web of interactions among flows becomes increasingly complicated so does the global dynamics of the network. This topological aspect of the problem has so far received comparatively little mathematical treatment, mainly because describing a large number of flows in a simple topology is already a formidable challenge. Most mathematical models in current literature treat very simple network topologies. The one most often considered is that of a single bottleneck link shared by a large number of identical flows. While this simple topology may not exhibit the full range of dynamics that may exist in more complicated networks, it is, nevertheless, an important special case both practically and theoretically. We turn to this case next.

A.1.1 Modeling Many Flows on One Link

We briefly outline the derivation of the fluid approximation for the one-link-many-flows case. Let the number of flows N be large and let capacity and buffer size of the router scale with N as NC and N^aB, $0 \le a \le 1$, respectively. Then the number of packets passing through the shared link per unit time will be large, satisfying the intuitive condition necessary for the fluid approximation to hold. While the system as a whole will remain deterministic as N grows, the packet loss process will be increasingly well approximated by a stochastic one. Thus it makes sense to model evolution of congestion windows with

[1] The sliding window is the interval of packet numbers corresponding to already sent but not yet acknowledged packets. The size of the sliding window is bounded by the size of the congestion window; its right edge advances when a new packet is sent and its left when a previously sent packet is acknowledged [9].

a corresponding collection of random processes. Let $W^N(t)$ be the random process describing aggregate congestion window size when the number of concurrent flows is N

$$W^N(t) = \sum_{i=1}^{N} W_i^N(t), \tag{5}$$

where $W_i^N(t)$ describes the congestion window size of flow i at time t. Since the flows are identical, it is reasonable to assume that the $W_i^N(t)$ are identical random processes. We will also assume that the $W_i^N(t)$ become independent as N tends to infinity. Applying the law of large numbers we have that $1/N\, W^N(t)$ converges to some deterministic process $w(t)$ as N goes to infinity. The deterministic process $w(t)$ is the *fluid approximation* of $W^N(t)$.

Considering the simpler case of small buffers $a<1$ first [120], we have from (1) and (5)

$$\frac{dW^N}{dt}(t) = \frac{N}{T} - \frac{1}{2}\sum_{i=1}^{N} W_i^N(t) P_i^N(t). \tag{6}$$

Note that $P_i^N(t)$ are now coupled random variables. Dividing through by N and letting N go to infinity we obtain the governing equation for $w(t)$

$$\frac{dw}{dt}(t) = \frac{1}{T} - \frac{1}{2} w(t)\, p(t), \tag{7}$$

where $p(t) = \lim_{N\to\infty} 1/N \sum_{i=1}^{N} P_i(t)$ (assuming the limit exists) is the aggregate loss density function. If the variation in round-trip time is small, $p(t)$ can be assumed to depend only on $w(t–T)$ (ignoring the rest of the network parameters for the moment). Furthermore, the number of packets lost per unit time can be approximated as $p(w(t–T))w(t–T)/T$, where $p(w)$ now stands for the probability that an arriving packet will be dropped due to buffer overflow when the aggregate congestion window size is w. Note that because of the round-trip delay the source detects packet loss only after a round-trip time T so that p depends on the transmission rate T seconds in the past. Formula (7) can be approximated as

$$\frac{dw}{dt}(t) = \frac{1}{T} - \frac{1}{2}\frac{w(t)\, p(w(t-T))w(t-T)}{T}. \tag{8}$$

If the buffer is large (a=1) then $p(w)$ will depend not only on $w(t–T)$ but also on buffer content $q(t–T)$. Using equations (2) and (5) gives

$$\frac{dw}{dt}(t) = \frac{1}{T + q(t-T)/C} - \frac{1}{2}\frac{w(t)\,p(w(t-T), q(t-T))w(t-T)}{T + q(t)/C}$$

$$\frac{dq}{dt}(t) = \left(\frac{w(t)}{T + q(t-T)/C} - C\right)\chi_{[0,B]}(q(t)), \qquad (9)$$

where we assume that the per flow buffer content $q(t) = \lim_{N\to\infty} Q^N(t)/N$ converges to a continuous deterministic variable.

A.1.2 Utility of Fluid-flow Approximation Models

There are two main types of results that can be obtained from models such as equations (8) and (9). First, one can deduce existence and uniqueness of the equilibrium, and its dependence on network parameters. Secondly, one can analyze stability properties of the equilibrium solution and the dependence of the equilibrium solution on the network parameters. Both types of results have clear practical applications, the first for optimal resource utilization and the second for stable network design. We will give a brief survey of both types of results.

By setting the right side of (8) and (9) to 0 we can obtain the expression for the equilibrium mean congestion window size w^*

$$w^* = \sqrt{\frac{2}{p^*}}, \qquad (10)$$

where $p^*=p(w^*)$ or $p(w^*,q^*)$ respectively. This shows that an equilibrium exists, since $p^*\neq 0$ is satisfied. Making the natural assumption that $p(w)$ is monotonically increasing in w, equation (10) also shows that the equilibrium is unique [122]. Formula (10) is close to experimental measurements [117] and also agrees with first principles derivations [117]. Unfortunately, the dependence of w^* on T and B is hidden inside the unknown function $p(w)$, which limits the usefulness of (10) for making a priori throughput estimates. Simple forms for $p(w)$ such as $\left(\frac{w}{CT}\right)^B$ corresponding to buffer overflow probability in an M/M/1/B queuing system have been found to be far from accurate [112-113]. Recently, an alternative packet loss model based on the Anick-Mitra-Sondhi on-off fluid queuing model has been proposed [112, 114] and found to be substantially more accurate than the M/M/1/B model at reproducing dependence of packet loss on w and the network parameters.

Even without knowing the exact expression for $p(w)$, however, sufficient conditions for linear stability of equilibrium (10) can be deduced in terms of p^* and $p'^*=p'(w^*)$. Using standard methods of control theory it has been shown [126] that equilibrium (10) of equation (8) will be stable in linear approximation provided that

$$w^*(2p^* + w^* p'^*) < \frac{p}{2}. \quad (11)$$

Inequality (11) is a necessary but not sufficient condition for stability, i.e., if it is violated the equilibrium will definitely be unstable but satisfying (11) does not guarantee (non-linear) stability. Based on (11) one can conclude, for example, that p'^* must decrease with increasing bandwidth-delay product in order for the equilibrium not to lose stability, since larger bandwidth-delay product corresponds to larger equilibrium congestion window size w^*. With more advanced methods it is also possible to derive conditions guaranteeing global stability of equilibrium (10), although, the resulting inequalities [124] are considerably more complicated and less informative than (11).

Fluid approximation models have also been used in a global optimization framework for the Internet, originally developed by F.P. Kelly, et al. [125] to show that per-flow TCP congestion control can be viewed as optimizing a certain global utility function. That is, TCP congestion control can be seen as a decentralized iterative algorithm for solving a network wide optimization problem. This point of view revolutionized understanding of the effect of TCP congestion control on global network dynamics. In particular, it paved the way toward a top-down protocol design, where starting with a desirable global network state first, the end-to-end congestion control can be tailored to achieve this global state.

Finally, fluid approximation models have been used to create fast simulators for large networks [106, 119] by leveraging fast numerical methods for solving systems of differential equations. While not as accurate or detailed as packet level simulators, the results from the first implementations are encouraging.

A.1.3 Limitations of Fluid-flow Approximation Models

In spite of great theoretical value, and even some practical applications, the fluid approximation framework falls short of being truly useful to practitioners of network and protocol design mainly due to lack of accuracy [112-113]. The main obstruction to the accuracy of fluid approximation models is lack of an accurate packet loss process model, which determines the equilibrium as well as dynamic behavior of the network. At the outset, the packet loss process was assumed to be well approximated by the loss process in an M/M/1/B queuing system. However, there is experimental evidence against the Poisson packet arrival hypothesis [118]. Based on packet traces collected from the Internet it was shown that the packet arrival process has rather different statistical properties from a Poisson process. In particular, it was observed that it is much burstier and is, moreover, bursty on all time scales. Due to the difficulty of mathematical analysis of queuing systems fed by such self-similar traffic, relatively little headway has been made toward obtaining a closed form expression for packet loss usable in the fluid approximation framework. In fact, it is still common in current publications [112, 120, 122, 123] to find computations based on the M/M/1/B queuing system.

Elsewhere [112] we proposed and tested a new expression for packet loss based on a queuing model of Anick, Mitra and Sondhi (AMS) [121]. Briefly, the model consists of a single fixed rate server fed by a superposition of fluid, fixed-rate, on-off sources with exponentially distributed “on" and “off" periods. This model is essentially a packet level fluid approximation. Observations [112, 114-115] of *ns2* traces suggest that TCP sources tend to concentrate packets in bursts (corresponding to a single congestion window)

rather than transmitting packets at a uniform rate on the time scale of a round-trip time. Thus setting the mean duration of the "on" periods to the congestion window size allows us to simulate burstiness arising from the non-uniformity in the transmission rate at the round-trip time scale.

The resulting mathematical model turns out to have a closed form solution in terms of the basic system parameters such as the number of sources, the server and source rates and the mean duration of the “on" and “off" periods. While this model is certainly only an approximation, since, for example, the window size distribution is expected to be non-exponential [107, 110], numerically it produces much better results than the commonly used M/M/1/B model [112]. Moreover there are some indications that the AMS packet loss model may be applicable to any unpaced TCP variant and not just TCP Reno, for which it was developed. Assuming this is so we can then use the fluid approximation framework to easily compare alternative congestion control algorithms in a variety of different network set ups with varying link bandwidths, buffer sizes and propagation delays. Next, we show how this can be accomplished.

A.2 Applying Fluid-flow Approximation Models to Compare Alternate Congestion Control Algorithms

In what follows, we briefly illustrate how the fluid approximation framework can be used to compare throughput performance of different TCP variants in a simple network. Specifically we consider an extension of the dumbbell topology — a network with a single link shared by a large number of continuously transmitting TCP flows with similar round-trip times (RTTs). We concentrate our attention on the standard TCP Reno and two other TCP variants — CUBIC [56] and Compound [58] TCP — which are currently increasingly deployed in the Internet due to their inclusion in Linux and Windows® Vista and Server operating systems, respectively [109]. In the following we will be interested only in the equilibrium throughput and so we will ignore the transient convergence dynamics described by the fluid approximation differential equations model and concentrate on the equilibrium solution. As previously explained, because congestion control mechanisms regulate transmission speed by opening and closing the congestion window, it is this window rather than throughput that is typically the main variable in mathematical models of TCP. The throughput is roughly proportional to the congestion window size divided by the round-trip time (including propagation and queuing delays).

The equilibrium mean congestion window size is described by a system of equations of the form

$$w^* = w(p^*) \tag{12}$$

$$p^* = p(w^*, C, B, T, N)$$

where w^* and p^* are the equilibrium congestion window size and packet loss probability, respectively. In special cases $w(p)$ may also depend on other network parameters such as the round-trip time (RTT).

Generally speaking one might expect that the second equation in (12), describing the dependence of packet loss on network parameters and equilibrium congestion window size, is roughly the same for all congestion control algorithms that use ACK self-

clocking, i.e., insert new packets into the network only in response to acknowledgments from the sink. The reason for this is that the sliding window algorithm in combination with ACK self-clocking largely determines the statistics of the aggregate packet arrival process at the bottleneck router, which, in turn, determines the statistics of packet loss. The packet loss statistics will vary with the TCP congestion control algorithm to the degree to which the equilibrium congestion window size distribution varies with congestion control algorithm. Unfortunately, due to the complexity of the problem relatively little is known about the properties of this distribution. For TCP Reno the stationary congestion window distribution has been computed under the assumption of Poisson losses [107, 110] and for more general additive-increase multiplicative-decrease algorithms [116]. On the other hand, the packet loss probability function under the assumption of exponential congestion window size distribution has also been derived [112].

We first, show how TCP variants can be qualitatively compared without knowing the form of the packet loss probability function provided it can be assumed to be approximately independent of the specifics of the congestion control algorithm. Indeed, if the second equation in (12) is independent of the specific algorithm, then the relative position of equilibria of TCP variants is determined by the first equation, which strongly depends on the particular congestion control algorithm used. The function $w(p)$ is often referred to in literature as the response function of the congestion control algorithm. As one would intuitively expect it is a decreasing function of p — the less frequent the losses the larger the equilibrium congestion window. Since the congestion window growth functions used in congestion control typically exhibit polynomial growth due to stability requirements, the response function itself is also typically polynomial in p, i.e., $w(p)=cp^{-\alpha}$ for some c, $\alpha>0$. Since the packet loss instances form a random process the form of $w(p)$ depends not only on the average loss probability but also on the statistics of the loss process. For simplicity, it is usually assumed that each packet is lost with probability p independent of previous losses, i.e., packet loss is a Bernoulli process. Suppose two TCP variants TCP_0 and TCP_1 have corresponding response functions $w_0(p)$ and $w_1(p)$. We will say that $w_0(p)$ *dominates* $w_1(p)$ if $w_0(p) > w_1(p)_1$ for $p \in [0,1]$. If TCP throughput is approximated by $(1-p^*)w^*/RTT$, then if $w_0(p)$ dominates $w_1(p)$ (as in Figure A-1) for $p \in [0,1]$ and p^* not too close to 1, TCP_0 will have strictly higher throughput than TCP_1.

In practice[2], p^* is usually less than 0.10. One must however keep in mind that this comparison in itself is an oversimplification in that it omits certain details such as bandwidth lost due to retransmissions, which may in practice lead to significantly lower overall throughput. For example, if we assume that the packet loss probability is equal to the blocking probability in an M/M/1/B queue (an admittedly optimistic scenario) it is not hard to show that the throughput increases with increasing w^* even when p^* is near 1, which translates into the best congestion control algorithm being no congestion control at all! That is, the faster the sources push packets into the network the higher the predicted throughput. Yet it is equally easy to see that this is a recipe for a congestion collapse,

[2] For example, the global Internet packet loss rate for 24 hours starting at 12:55 PM on April 22, 2010 averaged just below 7 %, as measured by the Internet Traffic Report. For the preceding month, the measured average loss rate did not reach 10 %. http://www.internettrafficreport.com/

since with packet loss near 1 the network will quickly fill with retransmitted copies of lost packets driving overall throughput to zero.

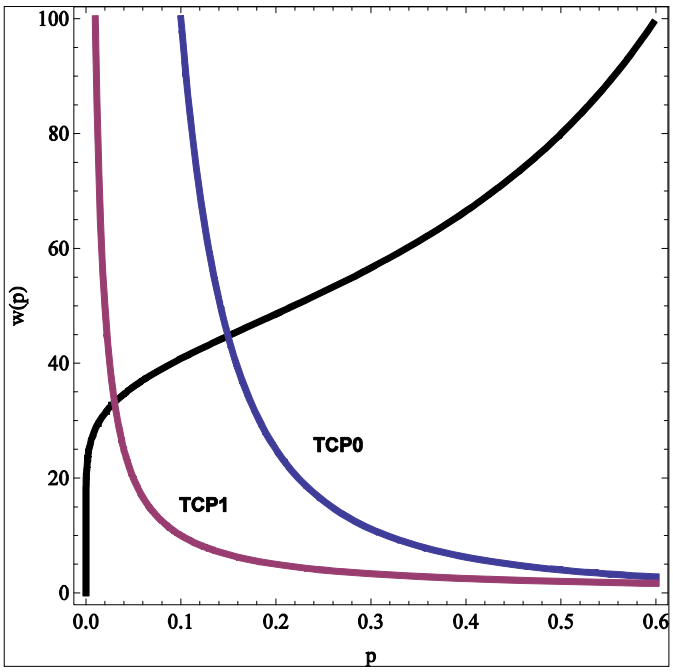

Figure A-1. Response curves of two hypothetical TCP variants TCP0 (blue) and TCP1 (red) and the graph of a hypothetical packet loss function (black).

If a dominance relationship between response functions cannot be established, then information about the packet loss function $p(w,\ldots)$ is necessary to compare congestion control algorithms. We will base our quantitative comparisons of TCP variants below on the packet loss model [112] discussed in Sec. A.1.3 above, since it was shown to produce more accurate results than models based on M/M/1/B queuing systems.

A.2.1 Computing Response Functions

We begin by computing the congestion response functions of TCP Reno, TCP CUBIC and Compound TCP. The response function of TCP Reno is well known and has been extensively experimentally verified [117]. We present a brief outline of the derivation as a simple illustration, since the computations become more involved for the other two variants.

A.2.1.1 TCP Reno. Suppose the per packet loss probability is p, then the average number of packets transmitted before a loss occurs is $N = (1–p)/p$. Suppose the equilibrium congestion window size just after a packet loss is w_0. Let a *round* be the number of packets equal to the current congestion window size and suppose for simplicity that between consecutive losses the number of rounds delivered is always an integer. Then the number of rounds k between consecutive losses is related to N by the equation

$$N=\sum_{i=0}^{k} w_0+k=(k+1)w_0+\frac{1}{2}k(k+1).$$

Solving for k gives

$$k=\frac{1}{2}(-2w_0-1+\sqrt{(2w_0-1)^2+8N}).$$

Thus at the end of a loss free period, just before packet loss occurs, the expected congestion window size will be

$$w_0+k=w_0+\frac{1}{2}(-2w_0-1+\sqrt{(2w_0-1)^2+8N}).$$

Since we assume the connection to be in equilibrium w_0 should be constant on average so

$$w_0=\frac{1}{2}\left(w_0+\frac{1}{2}(-2w_0-1+\sqrt{(2w_0-1)^2+8N})\right).$$

Solving the above equation for w_0 and discarding constant terms small in comparison with N we get

$$w_0=\sqrt{\frac{2/3}{N}}=\sqrt{\frac{2/3(1-p)}{p}}$$

Finally, to obtain the equilibrium mean congestion window size we multiply w_0 by 3/2

$$w(p)=\sqrt{\frac{3/2(1-p)}{p}}.$$

Since in practice p is usually close to 0 an approximation $w(p)=\sqrt{(3/2)/p}$ is often used.

A.2.1.2 Cubic TCP. CUBIC TCP differs from TCP Reno in several important respects, two of which make computation of the response function considerably more difficult as compared to the Reno computations outlined above. First, CUBIC's congestion window growth function depends on time rather than the number of acknowledged packets. Second, the congestion window growth function depends not only on the congestion window size before the packet loss (as in Reno) but also on how this value compares with the congestion window size at the end of the previous loss free period. Thus congestion window size alone no longer determines the full state of the algorithm and an additional parameter — last maximum achieved — must also be tracked to have a full state. Finally, nonlinearity of the congestion window growth function, the very feature that is supposed

to improve performance, makes computation of expectations hard. In view of these difficulties we are forced to content ourselves with the few rough approximations that are computable.

Let us begin by considering a very simple case when the packet losses are periodic in time. In this case the congestion window growth function can be shown to converge to the convex part of the cubic root, which drastically simplifies computations. Let the time between losses be τ and the congestion window just before a loss be w_0 then the congestion window at the end of the loss free period will be

$$c(\tau - k)^3 + w_0$$

where $k = \sqrt[3]{w\beta/c}$ and β and c are constants [56]. Since in equilibrium the w_0 is constant we have

$$c(\tau - \kappa)^3 + w_0 = w_0.$$

Substituting in for k and solving for w_0 we get

$$w_0 = \frac{c\tau^3}{\beta}.$$

Thus in equilibrium $k=\tau$. We can now compute the equilibrium mean congestion window size as a function of τ

$$w(t) = \frac{1}{\tau}\int_0^t c(\tau - t)^3 + w_0 d = c\tau^4\left(\frac{1}{\beta} - \frac{1}{4}\right) \qquad (13)$$

To convert this into a function of p we compute the number of packets sent during a loss free period, which is approximately

$$N = \frac{1}{T}\int_0^\tau c(t-k)^3 + w_0 dt = \frac{c\tau^4}{\mathrm{T}}\left(\frac{1}{\beta} - \frac{1}{4}\right)$$

where T is the round-trip time, which we assume to be approximately constant. Solving for τ we get

$$t = \sqrt[4]{\frac{NT}{c}\frac{4\beta}{4-\beta}}.$$

Substituting for τ in (13) we get

$$w(p) = \left(\frac{c(4-b)(NT)^3}{4\beta} \right)^{1/4}$$

Finally, assuming only one packet is lost in each congestion event so that p=1/N we get

$$w(p) = \left(\frac{c(4-\beta)}{4\beta} \left(\frac{T}{p} \right)^3 \right)^{1/4}$$

which for the default settings of β=.2 and c=.4 gives

$$w(p) \approx 1.17 \left(\frac{T}{p} \right)^{3/4} . \quad (14)$$

This is the formula derived by the designers of CUBIC [56].

In practice, however, packet losses are never periodic even in the case when there is only one connection on the link. Therefore, we consider a more realistic case of a Poisson loss process with rate λ. That is, we assume that a loss event occurs on average every 1/λ seconds. Two possibilities have to be considered because the congestion window at the end of a loss free period can now fall below as well as above the last maximum. We assume that w_0 and k are stationary independent random variables. This is still not enough to make explicit computations possible, so we further replace w_0 and k by deterministic variables equal to the mean values of the respective random variables. With these, admittedly very crude, simplifying assumptions we can write down a pair of fixed point equations for w_0 and k

$$w_0 = (1-\frac{\beta}{2}) \int_0^k \lambda e^{-\lambda t} (c(t-k)^3 + w_0) dt + \int_k^\infty \lambda e^{-\lambda t} (c(t-k)^3 + w_0) dt$$

$$k = \int_0^k \lambda e^{-\lambda t} \sqrt[3]{\frac{\beta}{2c}(c(t-k)^3 + w_0)} dt + \int_k^\infty \lambda e^{-\lambda t} \sqrt[3]{\frac{\beta}{2c}(c(t-k)^3 + w_0)} dt .$$

The first equation can be solved numerically in terms of k. Substituting the resulting equation into the equation for k and solving numerically over a range of λ indicates, as one might expect, that the equilibrium value of k is very nearly proportional to 1/λ with coefficient of about 1.3. This gives

$$w_0 = \frac{c(3+0.8\beta)}{\beta\lambda^3} .$$

for equilibrium congestion window just before a loss and

$$w(\lambda) = \frac{.3b + 3.8}{\beta} \frac{c}{\lambda^3}$$

for mean congestion window size. Expressing the above in terms of losses per number of packets sent and substituting the default values for β and *c* we get

$$w(p) \approx 1.67 \left(\frac{T}{p} \right)^{3/4}, \quad (15)$$

which is remarkably close to the simple formula (14).

A.2.1.3 Compound TCP. Compound TCP (CTCP) attempts to use delay measurements to estimate the number of buffered packets and alleviate congestion. CTCP's congestion window is decomposed into two components: the standard Reno congestion window, which increases by one over the window size for every acknowledgment received, and the delay based component which grows polynomially but only as long as the number of buffered packets, as measured by the increase in the round-trip delay, is below a certain threshold γ, which is itself dynamically adjusted to match available buffer space. Because the queue length measurements are performed for every returning acknowledgment and the congestion window growth rate modified accordingly, the window growth function is tightly coupled to the queue length. Thus, in general, analysis of CTCP must include an explicit model of queue lengths along the connection's path. Unfortunately, modeling queuing dynamics is in itself a complex and largely unsolved problem and so we are again forced to make crude simplifying assumptions. Specifically, we will assume that statistical fluctuations dominate dynamics so that the smoothed round-trip delay remains roughly constant once the network reaches equilibrium. Under this assumption analysis of the congestion window response function simplifies because γ does not change over time. Moreover, dynamic γ tuning insures that on average congestion window growth is polynomial right up to the moment of loss. Proceeding as before we thus have

$$\left(\frac{(1-k)\alpha}{T} \tau + w_0^{1-k} \right)^{\frac{1}{1-k}}$$

for the congestion window size at the end of a loss free period of duration τ given that the starting congestion window size is w_0 [58]. Since the window is reduced by 1−β upon detection of packet loss the equilibrium w_0 is determined by the equation

$$w_0 = (1-\beta) \left(\frac{(1-k)\alpha}{T} \tau + w_0^{1-k} \right)^{\frac{1}{1-k}}.$$

Computing the equilibrium mean congestion window size as a function of τ we get

$$w(\tau) = \frac{1}{\tau}\int_0^{\tau} \left(\frac{(1-k)\alpha}{T}\tau + w_0^{1-k}\right)^{\frac{1}{1-k}} dt$$

$$= \frac{T}{\tau}\frac{\left(w_0^{1-k} + \frac{(1-k)\alpha\tau}{T}\right)^{\frac{2-k}{1-k}} - w_0^{2-k}}{(2-k)\alpha}$$

Substituting for w_0 we have

$$w(\tau) = \frac{T\left(\frac{(1-k)\alpha\tau}{T}\right)^{\frac{2-k}{1-k}}\left((1+\gamma)^{\frac{2-k}{1-k}} - \gamma^{\frac{2-k}{1-k}}\right)}{(2-k)\alpha\tau} \quad (16)$$

$$\gamma = \frac{(1-\beta)^{1-k}}{(1-\beta)^{1-k}-1}$$

The number of packets transmitted in a time interval τ is given by

$$N = \frac{1}{T}\int_0^{\tau} \left(\frac{(1-k)\alpha}{T}t + w_0^{1-k}\right)^{\frac{1}{1-k}} dt$$

$$= \frac{\left(w_0^{1-k} + \frac{(1-k)\alpha\tau}{T}\right)^{\frac{2-k}{1-k}} - w_0^{2-k}}{(2-k)\alpha}$$

$$= \frac{\left(\frac{(1-k)\alpha\tau}{T}\right)^{\frac{2-k}{1-k}}\left((1+\gamma)^{\frac{2-k}{1-k}} - \gamma^{\frac{2-k}{1-k}}\right)}{(2-k)\alpha},$$

which gives

$$\tau = \frac{T}{(1-k)\alpha}\left(\frac{(2-k)\alpha N}{\left(1+\gamma^{\frac{2-k}{1-k}}\right) - \gamma^{\frac{2-k}{1-k}}}\right)^{\frac{2-k}{1-k}}$$

Finally, substituting τ into (16) and assuming $p=1/N$ as before, we obtain

$$w(p) = \frac{1-k}{2-k}\left((1+\gamma)^{\frac{2-k}{1-k}} - \gamma^{\frac{2-k}{1-k}} \right)^{\frac{1-k}{2-k}} \left(\frac{(2-k)\alpha}{p} \right)^{\frac{1}{2-k}}$$

for the response function of CTCP. For the default values of α=1/8, β=1/2 and k=3/4 [58] this is

$$w(p) \approx 0.25 \frac{1}{p^{4/5}}. \quad (17)$$

A.2.2 Comparing Congestion Control Algorithms

We begin performance comparison with the qualitative method described in the A.2 (see Fig. A-1). Since all computed response functions are approximations we must allow for errors in the resulting models. At present, however, there is no theoretical framework for computing fluid model error bounds and we are forced to make a somewhat arbitrary, but we hope conservative, assumption that the model equilibrium congestion window size is within 50% of the average window size that would be observed in a similar physical network.

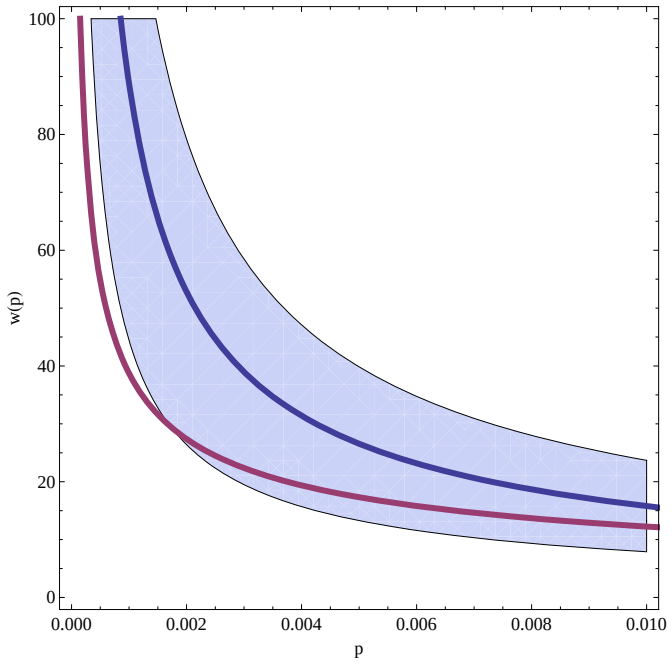

Figure A-2. Response curve for CUBIC (blue) with a ±50% error region (blue) vs. TCP Reno (red).

Fig. A-2 shows congestion window size as a function of packet loss for CUBIC and TCP Reno. For TCP Reno we do not plot the error region because the TCP Reno response function has been shown to be reasonably accurate [117]. As can be seen from the diagram, for the same probability of packet loss, CUBIC is likely to have a larger

congestion window, and so a higher throughput, for equilibrium packet loss rates up to about 1 %. We did not compute the response curve for higher loss rates because both CUBIC and TCP are likely to have limited throughput as loss rates become substantially higher. The response function plot for CTCP (Fig. A-3) shows that it will likely have a higher throughput than TCP Reno if equilibrium packet loss is below about .3 %. The plot also suggests that for equilibrium packet loss rates above about .5 % TCP Reno may actually outperform CTCP.

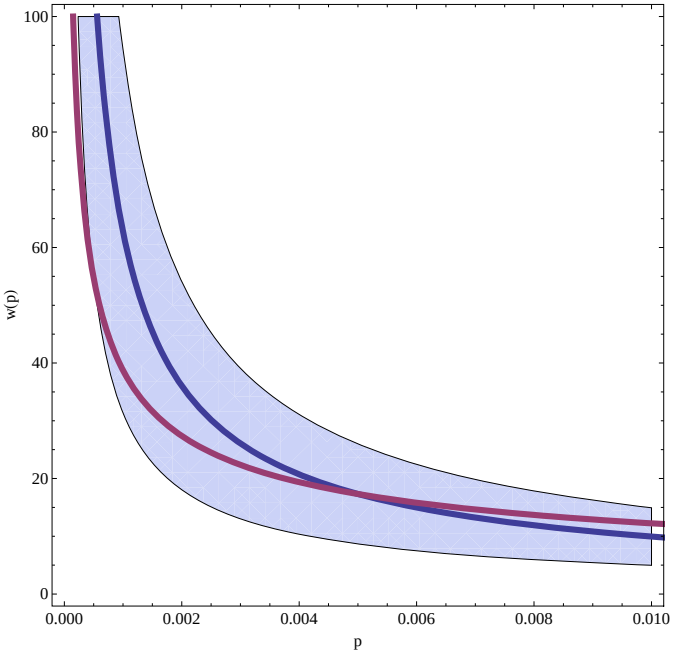

Figure A-3. Response curve for CTCP (blue) with a ±50% error region (blue) vs. TCP Reno (red)

Comparing response functions of CTCP and CUBIC (Fig. A-4) we observe that CUBIC is likely to have higher throughput than CTCP for the same equilibrium packet loss rate. However, because the error regions overlap considerably it is hard to say conclusively which of the two algorithms is likely to achieve higher throughput in practice.

We can also obtain some quantitative measures of the algorithms' performance by using a specific packet loss model, such as the one we introduced [112], to compute equilibrium throughput over a range of network parameters. Specifically we take

$$p(w) = \frac{Nw}{CT} e^{-1.5\left(\frac{C}{Nw} - \frac{1}{T}\right)\frac{BN}{C}}, \qquad (18)$$

where N is the number of concurrent flows, T is round-trip propagation delay, C is router capacity and B is buffer size. Tables A-1 through A-3 show average throughput for 1000 continuously transmitting flows over a 1 Gbps link for a range of propagation delays and buffer sizes. These quantitative results, unsurprisingly, largely agree with the above

qualitative analysis, but they also suggest some unanticipated conclusions. First, for round trip propagation times below 150 ms the alternative TCP variants do no better, and sometimes worse, than TCP Reno. This is presumably because the equilibrium packet loss rate for these scenarios is relatively high. Of course, CUBIC and CTCP were by design optimized for links with high capacity and long propagation delay (also called “long-fat pipes") and so their under-performance on links with relatively low bandwidth-delay product may be a chosen and accepted trade-off. Note, also, that this justifies the Reno-to-alternative mode switch present in most of the new TCP variants, including CUBIC and CTCP.

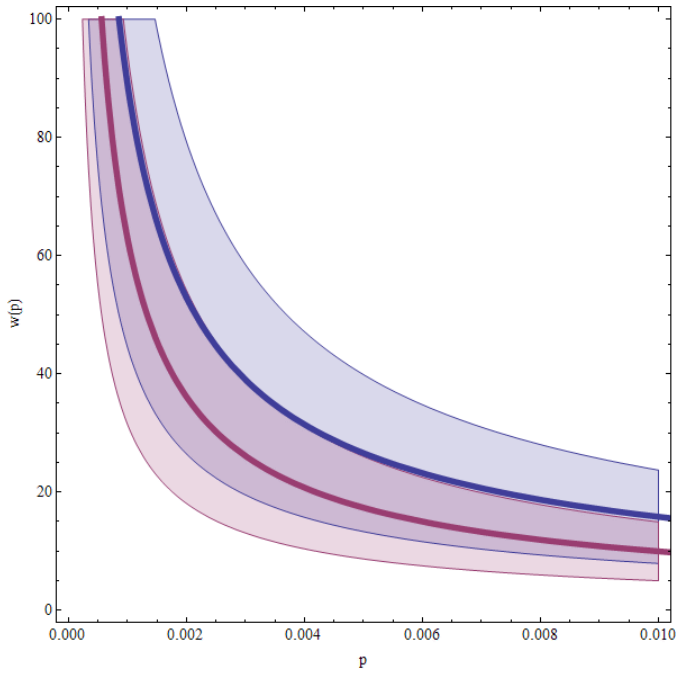

Figure A-4. Response curves for CUBIC (thick blue) and CTCP (thick red) with corresponding error regions

A second unexpected observation is that CTCP does not perform significantly better than TCP Reno even when round-trip delay becomes large, at least for the network parameters considered. Examining the graph of the CTCP response function (Fig. A-3), we see that this is most likely the result of a relatively high equilibrium packet loss rate, in the range of .3 % to .5 %, where the difference between response functions for CTCP and TCP Reno is small. On the other hand, in agreement with the qualitative analysis, CUBIC comes out ahead with an estimated improvement in throughput in the range of 10 % to 15 % over TCP Reno for networks with round-trip propagation delays longer than 150 ms.

While far from being exact or scalable to a network as large and complicated as the Internet, the mathematical models and methods presented here provide a cheap and fast way for evaluating alternative TCP congestion control algorithms even before any code is written. The value of these techniques is even greater when they are used as

design guideposts in the earliest stages of the development process of new TCP congestion algorithms.

Table A-1. Estimated throughput (p/ms) for CUBIC for 1000 concurrent flows on a link with a 122 p/ms capacity (for 1 KB packets)

CUBIC							
		B (pkts)					
		50	**100**	**150**	**200**	**250**	**300**
T (ms)	**50**	108	114	116	118	118	118
	100	98	107	111	113	114	115
	150	90	101	106	109	111	112
	200	84	96	102	105	108	110
	250	79	91	98	102	105	107
	300	76	88	94	99	102	105

Table A-2. Estimated throughput (p/ms) for CTCP for 1000 concurrent flows on a link with a 122 p/ms capacity (for 1KB packets)

CTCP							
		B (pkts)					
		50	**100**	**150**	**200**	**250**	**300**
T (ms)	**50**	112	115	117	117	118	118
	100	98	107	111	113	115	116
	150	87	98	104	108	110	112
	200	78	91	98	102	105	107
	250	70	84	92	97	101	103
	300	65	79	87	92	96	99

Table A-3. Estimated throughput (p/ms) for TCP Reno for 1000 concurrent flows on a link with a 122 p/ms capacity (for 1 KB packets)

TCP Reno							
		B (pkts)					
		50	**100**	**150**	**200**	**250**	**300**
T (ms)	**50**	115	116	116	117	117	118
	100	102	109	112	114	115	116
	150	89	99	105	108	110	112
	200	78	91	97	101	104	107
	250	70	83	90	95	99	102
	300	64	77	85	90	94	97

A.3 Future Work

Given that accurate modeling of packet loss is the key to accurate fluid approximation models, an important direction for future research is in the improvement and refinement of queuing models for TCP traffic. While the new packet loss model described in Sec. A.1.3 performs better than the commonly used, but highly inaccurate, M/M/1/B model, there is still room for improvement. In particular, the packet loss model in Sec. A.1.3 includes a finite buffer correction factor that is a rather crude patch in lieu of a solution for the finite buffer system. The model given in Sec. A.1.3 can also be improved by considering sources with non-exponentially distributed on-off periods since there is reason to expect that congestion window sizes have a non-exponential distribution.

Utility of the fluid approximation framework would also be greatly improved if response functions of the various new alternative congestion control algorithms could be computed more precisely. An important related question is: how do the specifics of the congestion control algorithm affect the congestion window size distribution? Answering this question would determine the sensitivity of the packet loss model to the TCP variant and hence the robustness of the comparison above.

The question of how network topology affects the equilibrium and stability of TCP traffic is another important direction for future work. Recently fluid approximation models have begun to be used for numerical simulations of large networks [108]. The low resource demands and high speed of these simulators permit, for the first time, an extensive exploration of the space of network topologies under a variety of simulated network conditions.