

Steady State Analysis of Internet Congestion Control Algorithms

Brian Whetten, brian@whetten.net

Abstract

TCP congestion control is widely recognized as the “magic” that has allowed the Internet to scale exponentially without experiencing congestion collapse. By extension, congestion control for reliable multicast has been identified as an important research problem. However, until recently there has been surprisingly little work done defining the theory behind Internet congestion control, the requirements for a congestion control algorithm, what it means to be “fair” with TCP, or even the safe operating range for TCP congestion control. The “drop-towards-zero” problem has been identified as a core challenge for multicast congestion control, but has not been formally defined or quantified. This paper helps address these issues, by summarizing the results of a detailed theoretical analysis of congestion control based on steady state analysis of congestion response functions [WC98, Whetten00]. It proposes a set of requirements that congestion control algorithms should meet, quantifies the drop-to-zero effect, specifies the safe operating range for TCP, analyzes the existing TCP response functions, and defines a notion of TCP fairness.

1. Introduction

The TCP congestion control algorithm was originally specified as a set of heuristics [Jacob88] that have been tuned over the years using simulations and experience. Much attention has been paid to the *dynamic* behaviors of TCP, such as its responsiveness in the face of changing network loads [FJ92], and its interaction with different queuing disciplines [FJ93]. However, despite being one of the most crucial pieces that has allowed the Internet to safely scale, there has been remarkably little fundamental theoretical analysis on the requirements for Internet congestion control, or of the theoretical operating range of TCP congestion control. The core additive increase, multiplicative decrease (AIMD) algorithm behind TCP congestion control is analyzed and justified in [CJ89]. This paper was also one of the first to establish the notion of an internetwork as a control system, and congestion control as a resource allocation problem using a control function and binary feedback from the network. However, it states that congestion is to be avoided rather than controlled, that

congestion avoidance should provide *maxmin fairness*, and that by doing this AIMD algorithms are safe for most any network environment—three conclusions that this paper argues should be revisited.

A major breakthrough occurred in 1997, when the first TCP response functions were proposed. A TCP response function is a model for the throughput of a TCP connection, when averaged over a long period of time. We define this as being applicable to *steady state* analysis of TCP. Steady state analysis is primarily a resource allocation problem, given a network and a set of offered loads. Two models for the TCP response function have been proposed, an initial *simple model* [FF97, MSMO97] and the *complex model* [PFTK98]. These response functions, define TCP's throughput (T, in bytes/second) in terms of packet size (B, in bytes), loss rate (P, as a percentage), round trip time (RTT, in seconds), and TCP timeout (t0, in seconds). The simple model (equation 1) is accurate up to a 1% loss rate. At loss rates above 5%, the TCP timeout t0 plays a dominant role in the performance of a connection, and this is taken in to account in the complex model (equation 2). For common implementations that send an ACK every other packet, C = 1.15 and C' = 2.60. For implementations that send an ACK every packet, C = 0.82 and C' = 1.83. As shown in [Whetten00], if the t0 timeout is set to a constant multiple (t) times RTT (equation 3), the shape of the response function is constant for all values of RTT, as shown in figure 1.1.

$$T \approx \frac{C' B}{RTT \sqrt{P}} \quad (1) \quad T \approx \frac{B}{C \sqrt{RTT \sqrt{P}} + t_0 \min(1, C \sqrt{P}) P (1 + 32 P^2)} \quad (2)$$

$$T \approx \frac{B}{RTT} \frac{1}{C \sqrt{P} + t \min(1, C \sqrt{P}) P (1 + 32 P^2)} \quad (3)$$

Having a response function for an individual TCP connection provides the foundation for steady state analysis of Internet congestion control. Steady state analysis looks at fairness, safety, utilization, and operational range, over long time scales.

In order to perform steady state analysis, [WC98, Whetten00] models the Internet as a combination of a physical topology and a summation of many transport connections, each of which obeys some congestion control response function that varies the offered load of each connection in relationship to the amount of congestion in the network. Under heavy load, routers drop packets as their primary signal of congestion. Often, the round trip time going through routers also increases

as a function of offered load. However, this is primarily a second order variable, because it typically varies over a much smaller range than loss rates can. For example, the practical range of non-zero values for a loss signal is approximately 0.01% to 33%, a range of 3300, while round trip times for a given connection rarely vary by more than a factor of 2 times.

With this model, the network appears as a complex control system, with the losses in the routers as the primary control signal, and the response function as the control function. Figure 1.2 illustrates this model with a simple example. On the left are N senders of TCP traffic, and on the right are N corresponding receivers. In the middle there is a router, which sources traffic to the bottleneck link with bandwidth L. If the sum of the offered loads is less than L, then zero loss will be generated by the network, and it will be running at less than 100% utilization. If the sum of the offered loads is greater than L, and the size of the queue at the router is small relative to the link latency (typical for the Internet), then the router that is sourcing packets to L will have to drop some packets (i.e. create congestion losses) to create equilibrium in the system. If we assume that the offered load for each of the N connections is infinite, that they each have the same constant RTT, and that each is accurately modeled by equation 1 (accurate for P<1%, and reasonable for P<5%), then the loss rate for the bottleneck link can be calculated from equation 4.

$$P \approx \frac{C \cdot B \cdot N}{RTT \cdot L} \quad (4) \quad \text{For RTT=300 ms, B=1512 bytes, N=5, and L=1.5 Mbps, the loss}$$

rate is 1.4%. However, for N=20, the loss rate according to this formula is 22%, and if RTT = 100 ms, then the system can't equalize, as P>100%! This introduces two concepts. First, that certain operating conditions for TCP style congestion control can still cause congestion collapse. Second, that the simple formula in equation 1 is inadequate for use in the Internet. A key insight gained from this simple example is that a network using congestion control operates by "pushing back" on the connections crossing it when it is overloaded. If all connections only run across a single congested gateway, then they all reduce their throughput in a proportional way. As the number of connections increases, the total bandwidth of the link decreases, the RTT of each connection decreases, or the PacketSize of the connection increases, then the loss rate of the network must increase to accommodate this. As the network becomes more complex, the control system becomes

drastically more complex, and solving it in this way becomes infeasible, because the loss rate a connection sees is approximately equal to the sum of the loss rates of all the links it crosses. If connections don't all share the same path, then the solution requires solving a complex linear system.

From this, we see that persistent congestion is a natural feature of internetworks, necessary for all but the rare cases where the offered load is less than the available bandwidth. Many of the works that discuss congestion control [Jacob88, CJ89] imply that congestion is something to be avoided. Indeed, many congestion control algorithms are instead termed congestion avoidance algorithms. Instead, we find that congestion is an essential feature of any IP network, and the primary challenge for a congestion control algorithm is instead to use this control signal as well as possible, to meet the above goals. In order to quantify the relative merits of any congestion control algorithm, this paper proposes a set of metrics that partially characterize a response function, and by extension the congestion control algorithms represented by each response function. Despite TCP having been defined over fifteen years ago as a set of mechanisms and heuristics [Jacob88], and only recently having its behavior modeled as a response function, we find that TCP provides a near ideal solution when measured with these metrics.

[Gole98A] and [Gole98B] provide a general architecture for reliable multicast congestion control, and contain some overlap with the independent work presented in this paper. An important contribution of these papers is a ground-breaking insight in how to use windowed congestion control for reliable multicast in a scalable way, that is also provably fair with TCP. [Gole98b] proposes two potential definitions for fairness. "Rate based fairness" is essentially defined as giving each connection the same rate no matter what the round trip time of the connection is, and "Window based fairness" is defined as giving each connection a throughput inversely proportional to its RTT. The second definition is essentially the same as the definition of global fairness put forth in this dissertation. [WS98] created a new definition of fairness, termed "essentially fair", and proposed a window based algorithm that controls a connection in a way that meets this definition. This definition relaxes the constraints on a multicast connection as the

number of receivers grows. It solves the drop-towards-zero problem by regulating the throughput based on the average loss rate of the congested receivers, as opposed to the worst case among the congested receivers. However, this allows it to be unfair to TCP on the most congested links.

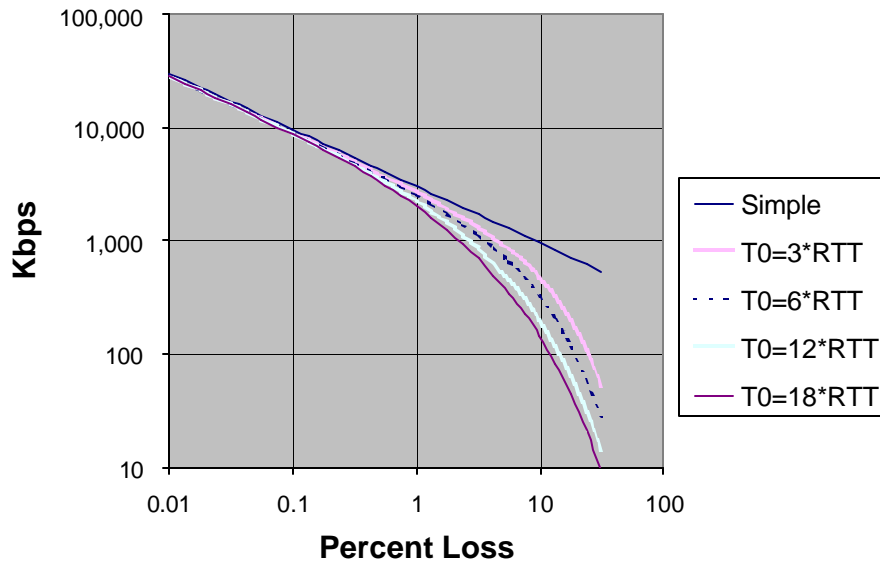


Figure 1.1. Different TCP Response Functions

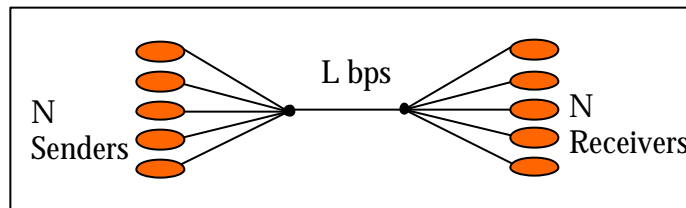


Figure 1.2. Control System Example

2. Requirements for congestion control

There is no generally accepted definition of the requirements for congestion control in either a unicast or multicast environment. This section proposes such a set of requirements, which are formally defined in [Whetten00]. We start by dividing the behavior of congestion control algorithms into two halves: steady state and dynamic. Steady state analysis looks at the behavior of the algorithm in a network where all of the traffic flows on the network have a constant offered load, and the set of traffic flows stays constant. Dynamic analysis looks at the more fine-grained behavior of the algorithm as the network conditions change over short time scales. Steady state analysis is

primarily accomplished through modeling and mathematical analysis, while dynamic analysis requires both network simulations and analysis.

The requirements for the steady state behavior of a congestion control algorithm are as follows.

1) **Safety**, defined as the avoidance of congestion collapse, and is the primary requirement. This is primarily a function of the *effective operating range* of the algorithm covering all existing and planned network deployment scenarios, and is also related to the responsiveness of the algorithm, particularly when new senders are added to the network. 2) **Utilization**, which for any given network link is the ratio of the goodput of the connections crossing it, divided by the bandwidth of the link. 3) **Fairness** between connections, for which both unicast and multicast definitions are offered below.

The requirements for the dynamic behavior of a congestion control algorithm are as follows. 4) **Responsiveness**, which quantifies the speed at which the algorithm responds to changes in network conditions or changes in the offered load at the sender. 5) **Stability**, so that the system cannot get in to unstable conditions through self-reinforcing oscillations, particularly in the face of network failures. 6) **Variance**, which is the amount that the throughput of a given connection oscillates over short time scales, when the network conditions and offered loads are not varying. Variance directly impacts utilization, and may also impact stability and safety.

In designing a congestion control algorithm, the following is a partial set of the challenges that must be dealt with to meet these requirements. 1) In order to be safe, the algorithm must support an extremely heterogeneous range of operating parameters for each connection. 2) There is a fundamental tradeoff between safety and fairness. 3) There is a fundamental tradeoff between responsiveness and variance, as well as between fairness and responsiveness. This is the core of the drop-towards-zero effect quantified in section 5. 4) Any set of network simulations is only able to consider a small fraction of the conditions in the Internet. 5) For multicast connections, it is important to both be fair to TCP, while still offering economic incentives to use multicast.

The first challenge is particularly difficult, for it means that the congestion control algorithm, which governs the throughput of a connection, must be able to adapt over 6 orders of magnitude or more. An ideal congestion control algorithm should be able to allow the backbone of a

supercomputer facility to deliver up to multiple Gbps, while also restricting modem connections connected to a web server halfway around the world to one or two Kbps under heavy load.

3. Safety and Response Function Metrics

The primary requirement for safety is that the *effective operating range* of the congestion control algorithm encompasses all of the deployment scenarios of the network that it will be deployed in. In the case of the Internet, a primary way of determining whether these requirements are met is through defining metrics over a response function that accurately models the congestion control algorithm [WC98, Whetten00]. This can be done either by modeling the algorithm, as in the TCP response functions, or by starting with a response function and building a congestion control algorithm around it [WC98, Whetten00, HWF99, MF97, FHPW00].

The *domain* of a response function provides the set of input values over which the function is defined, for a given connection and network, which produces the *range* of results for that function for that connection and network. For some response functions, such as equations (1) and (3), if we specify a fixed domain for P , the shape of the response function is the same for all values of RTT and B (figure 1.1). For these cases, we define the *adaptive range* as the maximum value of the range of any connection divided by the minimum value of the range of that connection. This provides the magnitude of change in operating conditions that the response function can respond to.

For the Internet, a realistic domain for the measurable loss rate of any given connection (P), which also maintains reasonable network utilization, is [0.01% - 33%]. Given this, the adaptive range of the simple TCP response function (1), is a factor of 57.4. The adaptive range of the complex function (3) is dependent on the value of t , where the TCP timeout $t_0 = t \cdot RTT$. For $t=20$, the adaptive range is 2619, for $t=10$ it is 1341, for $t=6$ it is 828, and for $t=3$ it is 443. There is not a consistent value of t for existing TCP implementations, but using a value of $t=6-10$ in equation based congestion control algorithms appears to provide an adequate tradeoff between adaptive range and TCP fairness.

In addition to having a large adaptive range, the response function must be calibrated well for the networks it is to run across. The metric *maximum connections* is defined as the most simultaneous

homogenous connections that can share a given link, without driving the link in to an unstable operating range where the loss rate above a maximum rate, say 33%. For example, take the example of a large number of clients all trying to download a large file or media stream through the same 1.5 Mbps T1 line, and assume that the RTT for each connection is 100ms. Assume also that the connection is accurately modeled by equation 3 with $t=10$, and that $B=1512$ bytes. In this case, even with $P=33\%$, each connection is sending at a throughput of 7.82 Kbps. Dividing the line capacity by this rate, we get $1500\text{Kbps}/7.82 \text{ Kbps}=192$ maximum connections. Figure 3.1 shows the maximum connections metric as a function of RTT, for $t=10$, $B=1512$ bytes, and a maximum P value of 33%

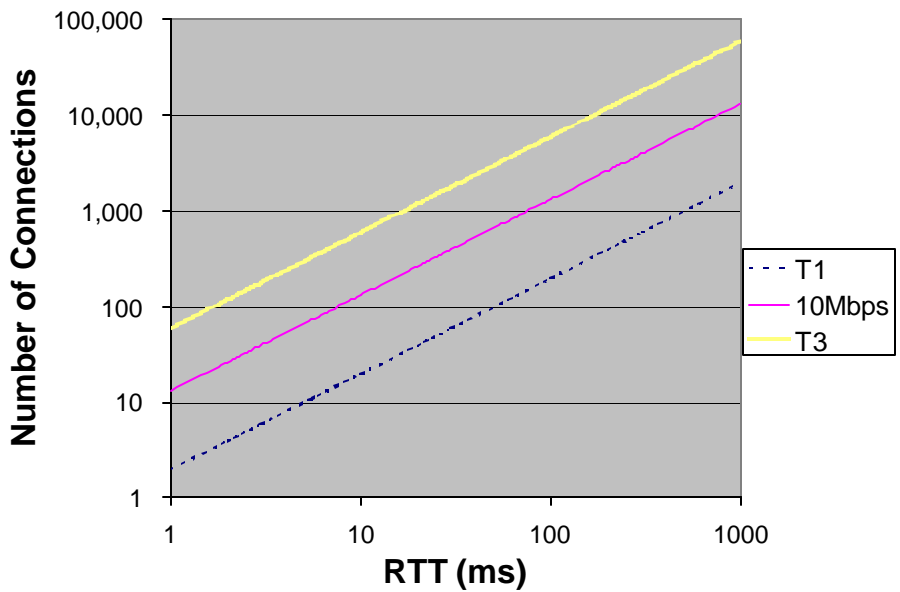


Figure 3.1. Maximum Connections vs. RTT.

Figure 3.2 shows the loss rate that a link will run at, given different link speeds and an RTT of 50 ms, as a function of the number of connections. Notice that as the number of connections increases, the loss rate increases rapidly until approximately 5% loss rate, and then increases slowly. This is due to the accelerating slope of TCP's complex response function, and is much of the reason that TCP can operate safely in heterogeneous environments.

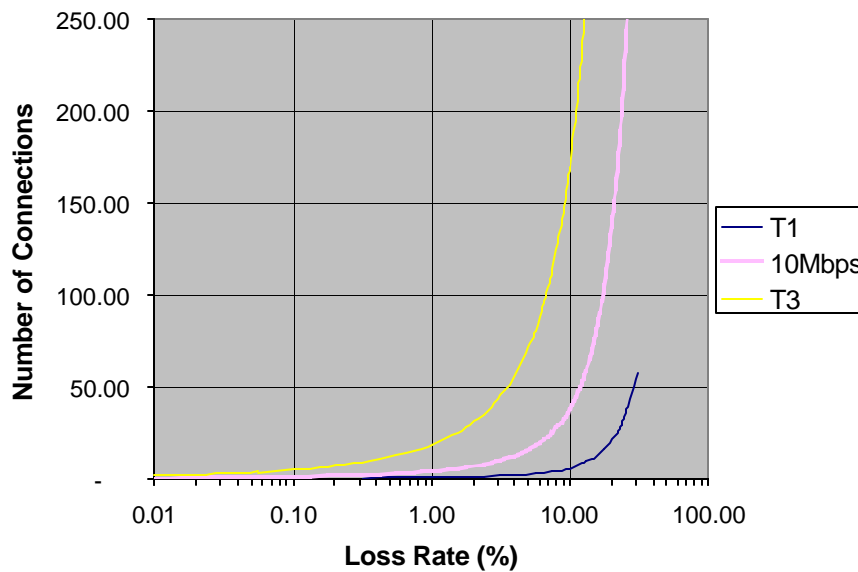


Figure 3.2 Loss Rate vs. Number of Connections

4. Fairness

A key challenge in defining congestion control for multicast has been the lack of an agreed upon definition for fairness. Most early formal work (such as [CJ89]) specified that the network should obey *maxmin fairness*, where the throughput for all connections experiencing the same single bottleneck link (or loss rate) would experience the same throughput, regardless of their round trip times. However, a congestion control algorithm suitable for use in the Internet must have a network adaptive range of at least 6 orders of magnitude. While this theoretically could be done without using RTT as part of the response function, it would be difficult. For example, consider if throughput was proportional to $1/P^2$, instead of $1/\sqrt{P}$, such as in $T \propto \frac{C \cdot B}{P^2}$ (5). Equation 5 has an adaptive range of 7 orders of magnitude. However, it is much more sensitive to errors in measuring

the loss rate, and as shown in section 5, this would dramatically increase the magnitude of the drop-towards-zero effect. The responsiveness of the system would have to be significantly reduced in order to have the same level of throughput variations seen today. Also, this would increase the known bias against connections that pass through multiple congested links. In the Internet, loss is approximately additive across all links in a connection's path. If we take the example of a connection crossing N congested links, each with the same latency and loss rate, then with the simple model in equation 1, throughput is proportional to $1/N^{3/2}$ (equation 6). With equation 5, throughput drops off as a factor of $1/N^2$ (equation 7).

$$T \approx \frac{C \cdot B}{N \cdot RTT \cdot \sqrt{N \cdot P}} \approx \frac{1}{N \sqrt{N}} \quad (6) \quad T \approx \frac{C \cdot B}{(N \cdot P)^2} \approx \frac{1}{N^2} \quad (7)$$

[Whetten00] makes the case that if all bits transmitted have the same value, then in the absence of congestion, the best estimate for the cost for each bit is RTT. While admittedly crude, this estimate is proportional to the number of links crossed, takes at least some account of the added cost of international delivery, can be easily measured in Internet protocols—and is actually the function used by TCP, to the extent that equations (1) and (3) are accurate. Based on this, it defines a two part definition of fairness. A congestion control algorithm meets *resource fairness* if, in the absence of congestion, all connections get the same amount of network resources, where resources for each packet sent are measured by the round trip time of the connection. As congestion occurs on a local link, an algorithm meets *local fairness* if all connections back off by an amount dictated by the same response function. In the absence of multiple congested links, all connections react to loss in a proportional way. In the case of multiple congested links, all connections react to the most congested link they pass.

TCP is more conservative than this two-part definition, because it reacts to loss on the whole path, as opposed to the worst link, which is termed *path fairness*. An algorithm that meets both resource fairness and path fairness, and matches the same response function as TCP, is exactly fair with TCP, under steady state analysis. An easier way of stating this is that there are two types of resource fairness—path based and locally based. With one exception, TCP obeys *path based resource fairness*, because throughput is inversely proportional to RTT and all connections “back off” in

response to a congested link in the same way. While we will not provide the formal proof here, the exception is when a connection crosses multiple congested gateways that push the connection in to the “non linear operating range” of equation 3 (see figure 1.1), where $P > 5\%$. The advantages of resource fairness over maxmin fairness are that resource fairness defines fairness relative to an (admittedly crude) estimate of the cost of network resources, and that it can more easily support the very large adaptive range required for the Internet—and therefore is safer than maxmin fairness.

[Whetten00] proposes that a potentially superior solution to the current Internet would be to use *locally based resource fairness* coupled with a response function that is both proportional to $1/\text{RTT}$ and $1/P$, rather than $1/\sqrt{P}$. This would have the following advantages.

- ?? It does not discriminate against connections that cross multiple congested links. Connections that cross N identical congested links will have throughput that is $1/N$ that of connections which only cross one. A connection will get the same proportional resources if it crosses multiple identically congested gateways as if it crosses only one.
- ?? The steeper slope of the response function ($1/P$) would provide more adaptive range and a safer network. However, if this was used in conjunction with path fairness, it would result in a system that penalized connections as a function of $1/N^2$ for crossing N identically congested multiple gateways.
- ?? It is potentially enforceable at each individual router.

For multicast, it has become generally accepted that for a single-rate congestion control algorithm, the sender must adapt to the speed of the slowest receiver. A primary question is over what time scale this must occur, which is discussed in the next section. The other question is how to select the slowest receiver, using loss and RTT measurements to each.

[Whetten00] defined four ways of selecting the slowest receiver, termed *double worst*, *worst receiver*, *worst edge*, and *restricted worst edge*. Double worst means independently selecting the worst value for RTT for each receiver, and the worst value of P for each receiver, and combining these. This is obviously non-optimal, but may be necessary for protocols where there are not RTT measurements made to each receiver. Worst receiver is the most common method used, where the values of P and

RTT for each receiver are evaluated in a response function, and the slowest receiver is selected from the results of these response function calculations. While they use different specific mechanisms, both TFMCC [WC01] and PGMCC [Rizzo00] conform with worst receiver fairness.

Worst edge and restricted worst edge proposed two ways of doing calculations in hierarchical reliable multicast protocols such as [WT00], to solve the potential economic disincentive against using multicast the drop-towards-zero effect causes in some environments, when compared with TCP, especially in the presence of hierarchical caches and streaming appliances which improve the performance of TCP by reducing the RTT of the TCP connections. Worst edge proposes emulating this hierarchical TCP, evaluating the RTT and P values for each edge in the reliable multicast tree, and selecting the worst edge. However, this may not be safe for all network cases, particularly if there is not perfect topology congruence between the multicast router topology and the reliable multicast tree topology. Restricted worst edge uses the RTT of a single edge in the tree, coupled with the loss rate of the worst receiver of the subtree below that edge. Restricted worst edge is shown to be both fair and safe, offering a typical performance increase of 2x-4x or more, but at the cost of extra implementation complexity.

5. Drop-Towards-Zero Effect and Measuring Loss at Receivers

The majority of the work done to date on multicast congestion control algorithms has attempted to look at the mechanisms of TCP, and generalize those to a reliable multicast environment. However, by themselves the mechanisms of TCP do not appear to scale well to reliable multicast, because of the drop-towards-zero problem. It has been established that a reliable multicast connection must be provably fair with TCP on the worst link that it crosses. If this is enforced on a short time scale (one to two RTT's), then the performance of the RM connection will be significantly worse than TCP's, and this effect will increase as the number of receivers increases. This drop-towards-zero effect occurs for two reasons.

?? Fractal Network Traffic. Because of the self-similar nature of internet traffic [PS95], the packet starvation effect in CSMA/CD LANs [WSF94], and the traffic phase effects of drop tail routers [FJ92], losses on a given link tend to be fractal in nature. Taking the maximum function of

multiple fractal distributions creates a distribution with a higher average than any of the input functions.

?? Information Limits. Because the signal for loss is strictly binary (a packet is dropped or not), any measurement of loss will have a very low signal to noise ratio.

Of these, the second is by far the most important. As explained in section 1, the network can be viewed as a control system, with connections deciding on their fair share of bandwidth based on the loss rate and the RTT measurements from the network. RTT measurements can be made accurately on each packet acknowledged to a sender, but the RTT for a connection rarely changes by more than a factor of two. While this can be useful for fine tuning a connection and avoiding self-reinforcing phase effects that cause instability, this does not provide an adequate adaptive range to deal with the Internet's heterogeneity.

TCP attempts to react as quickly as possible to congestion in the network, reducing its window size (and therefore its throughput) by 50% in response to each lost packet that is detected. While this provides an average behavior congruent with the response function, there is tremendous variation in the throughput when viewed on a shorter time scale.

In digital signal processing, the signal-to-noise ratio is commonly used to quantify the amount of information available from a given signal. This is defined as the ratio of the average amplitude of the signal, divided by the standard deviation of the signal. To understand the theoretical information limits of Internet congestion control, we model packet drops as a Bernoulli process with constant mean rate P . This is roughly accurate for RED gateways [FJ93], but is not accurate for drop-tail gateways. Quantifying just how much worse the effects of drop-tail gateways will be on these averages is outside of the scope of this paper. The assumption that the average loss rate is constant is also optimistic, as large networks tend to show a fractal, constantly varying load and loss rate at each router [PS95]. Therefore, this model provides an optimistic, upper bound on the S/N ratio of the signal coming from the routers.

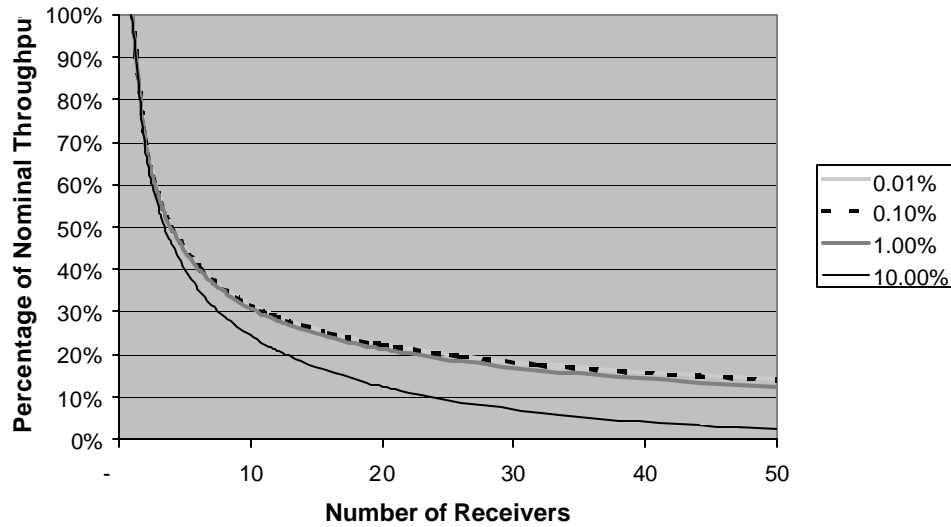


Figure 5.1. Drop-towards-zero at TCP Time Scale

With a Bernoulli process of mean rate P , the number of successes between losses is a geometric distribution with mean P . This is the basic period of measurement and adaptation for a TCP connection. If we define $Q=1-P$, then the expected value of this geometric distribution is Q/P , and the standard deviation is $\sqrt{Q/P^2}$. The S/N ratio is then equal to \sqrt{Q} . Numerically, this value ranges from 1.000005 with $P=0.01\%$, to 1.222 with $P=33\%$. The S/N ratio increases as a factor of \sqrt{N} if N periods are averaged, resulting in a S/N ratio of at least 2 if four periods are averaged, and at least 4 if 16 periods are averaged. No matter what set of mechanisms or response function is used for an actual congestion control algorithm, this provides fundamental theoretical limits on the amount of information that can be derived from the network. By derivation, this also provides the lowest theoretical variance that any set of mechanisms can have, if they are based on loss measurements from the network. For TCP, this means that the average throughput of a connection will be accurate if measured over a very large number of individual loss measurements, but the throughput averaged over a single window of data will vary wildly over the duration of a connection, due to a S/N ratio of at most 1.2.

With reliable multicast congestion control, it has been established that a connection must be provably fair with TCP on the worst link that it crosses. The crucial question is what time scale this fairness must be provided over. If it is provided over the same time scale as TCP, either by using a

similar set of AIMD mechanisms, or a response function which uses a loss report based on the number of packets received before the most recent loss, this alone causes dramatic drop-towards-zero problems. Note that while PGMCC [Rizzo00] adapts to a given receiver on a TCP time scale, it avoids the drop-towards-zero effect by only switching *between* receivers on a longer time scale, similar to that of [WC98] and [WH01]. To the extent that a single receiver at a time consistently provides the traffic bottleneck, this allows PGMCC to emulate TCP's rate to that receiver, with the fast responsiveness and high variability this brings. If multiple receivers are the bottleneck at the same time, then PGMCC starts to look more like TFMCC, adapting between receivers at a slower time scale.

To extend this statistical model to multicast, we assume that each receiver has a fixed RTT (but one which is possibly different from other receivers) for the length of its connection. Since congestion control deals with fundamental network safety, we need to make sure any congestion control algorithm can work in any environment. The worst case we need to examine is when the network conditions to each of R receivers results in the same average throughput to that receiver, and each receiver's conditions are statistically independent (i.e. no correlation in their loss patterns) from each other. To quantify the drop-towards-zero effect in this scenario, we assume that the receivers measure their loss rates over some fixed number of N loss periods (a loss period corresponds to a single loss from the network) using a linear averaging function, and the sender responds to the highest loss report for each period. For the case where $N=1$, the distribution at each receiver is geometric, and we need to find the minimum of a set of R geometric distributions. As shown in [Whetten00], this distribution, X , (of the average number of trials before a success) has the expected value and standard deviation

$$Q=1-P \quad E(X)=\frac{Q^R}{1-Q^R} \quad \sigma(X)=\sqrt{\frac{Q^R}{1-Q^R}}$$

To quantify the drop-towards-zero effect for R receivers, we assume the use of the simple TCP response function where throughput is proportional to $1/\sqrt{P}$, which is equal to $\sqrt{E(X)}$. To get the expected size of the drop-towards-zero effect, $E(D)$, we take the ratio of the expected value of throughput for R receivers over the expected value of throughput for 1 receiver

$$E(T) \approx \sqrt{E(X)} \quad E(D) \approx \frac{E(T)}{E(T) + R + 1} \approx \sqrt{\frac{Q^{R+1}(1+Q)}{1+Q^R}}$$

This is plotted in figure 5.1, for values of P ranging from 0.01% to 10%, and for values of R ranging from 1 to 50. The plotted values for P=10% are conservative due to the use of the simple response function. If the complex response function was used, the drop-towards-zero effect would be even higher. Also, note that if equation 7 was used where throughput was proportional to 1/P, the drop-towards-zero effect would be the square of this curve. The only non-conservative assumptions are that the receivers have the same approximate bottleneck throughput, and that their losses are

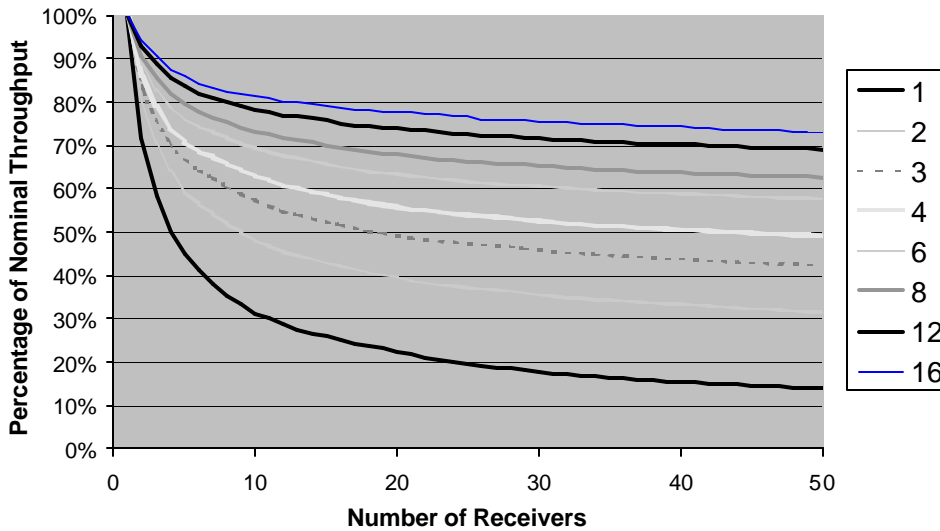


Figure 5.2. Drop-towards-zero Effect for Different Averaging Intervals

independently generated. Given this primarily conservative analysis, the results are still quite dramatic. For R=2 the drop-towards-zero effect reduces average throughput by 30%, for R=4 throughput is reduced by 50%, and for R=16 throughput is reduced 75%. This result demonstrates conclusively that AIMD mechanisms that attempt to work on this time scale, and react to the worst receiver for each loss period, cannot feasibly be used for reliable multicast congestion control.

For cases where the averaging interval N is larger than 1, the receiver has a negative binomial distribution for $E(X)$, and we need to find the minimum of a set of R negative binomial distributions. While a closed form solution to this is difficult, we can simulate it in order to get

numerical results. Figure 5.2 shows this for different values of N and R, for P=0.1%. The simulation is validated by its similarity to the closed form solution in figure 5.6 for N=1. As in figure 5.1, this graph assumes the use of the simple TCP response function, and assumes that the expected value for each receiver's throughput is the same. This graph demonstrates that the drop-towards-zero effect is greatly ameliorated by increasing the value of N, although this comes with the tradeoff of adapting on a slower time scale than TCP. The lowest line, for N=1, is similar to that shown in figure 5.1. In order to limit the drop-towards-zero effect to no more than 50% of the nominal throughput (as compared to the case with a single receiver), N needs to be set to at least 6. Again, these graphs do not take in to account other factors such as the effects due to variances in RTT, the increased variance introduced by drop-tail routers, the fractal nature of Internet loss patterns, or the use of the complex response function. It is only looking at the fundamental lack of information in the communication between the network and the transport connections, and the fundamentally low signal/noise ratio associated with this channel. All of these factors will increase the magnitude of the drop-towards-zero effect.

There are three primary ways that have been proposed to measure the loss rate at the receivers. [FHPW00] proposes a weighted linear average over the last eight loss measurements. With the selected weights, this scheme is approximately equal to using a value of N=6, and so appears to find a near-ideal solution to the tradeoff between responsiveness and variance. Of the three techniques reviewed, this is the most accurate, and provides the best optimization of responsiveness vs. variance. It may have slightly more complexity than might be desired in some systems, however. [WC98, Whetten00] proposes using an exponentially weighted moving average (EWMA), with the value of the weight set dynamically to P/2. For a series of measurements, X_n , this algorithm measures P_n as
$$P_n = \frac{P}{2} * X_n + (1 - \frac{P}{2}) * P_{n-1}$$
 and initializes the weight using a linear average of the first four losses. This sets the S/N ratio to 2. This is potentially simpler than [FHPW00], while being marginally less optimal. [Rizzo00] proposes using an EWMA average with a fixed constant. Since an EWMA average acts as a low pass filter, using a fixed constant means limiting the ability to detect loss rates below the frequency of the constant. Unless a very small constant is used, in which case

responsiveness would overly suffer, this reduces the adaptive range of the system, and we recommend that the small amount of added complexity of either of the two other mechanisms is more than justified by their ability to optimize for a constant S/N ratio in the tradeoff between responsiveness and variance.

By using a value of $N=6$, a congestion control mechanism is up to six times less responsive than TCP to sudden increases in the loss rate. Figure 5.3 shows the numerical results of solving equation 3 for the average number of RTT's it takes to get a complete new loss measurement over six lost packets ($N=6$), for two values of t_0 .

Putting explicit congestion notification in the routers (that returned at least a 13-bit value, and preferably a 16-bit value for the average loss rate a connection sees rather than a 1-bit loss value) would allow feedback to be done on a much shorter time scale. It could result in responsiveness periods as short as a single RTT for networks with high levels of statistical multiplexing. In this case, the responsiveness would be limited by the time it takes to get feedback to the sender and by potential dynamic oscillation effects, rather than the limits on information from the network.

P	$T_0=10*RTT$	$T_0=3*RTT$
0.01%	491.0	490.2
0.03%	277.5	276.1
0.10%	158.4	156.0
0.32%	93.3	89.0
1.00%	60.0	52.3
3.16%	47.8	33.6
10.00%	61.5	29.3
31.62%	260.7	84.3

Figure 5.3. Number of RTT's Required for Loss Measurements

6. Conclusions and Future Work

This paper has presented the results of a formal analysis of Internet congestion control using steady state analysis of congestion control response functions. It defined a set of metrics for evaluating the safety of a response function, and applied this analysis to TCP. This showed that while the simple TCP response function is unsafe, the complex TCP response function does a remarkably good, although not ideal, job of providing both safety and fairness in the existing

Internet. It also points out the dependency TCP has on accurate timer measurements, which current implementations do not necessarily provide.

One of the fundamental issues in multicast congestion control has been dealing with the drop-towards-zero pathology. This document provided a quantification of the scale of this problem, and demonstrated that traditional AIMD congestion control algorithms cannot scale for reliable multicast in cases where there are many receivers with similar, independent loss rates. The fundamental issue is that the signal to noise ratio of any system that adapts on every lost packet is only slightly larger than 1. Hybrid schemes, such as [Rizzo00], solve this problem by selecting the slowest receiver on a slower time scale, and using TCP AIMD mechanisms to rapidly adapt to that receiver over shorter time scales.

This paper summarized a formal definition of fairness based on the resources used by a connection, as a potential reference point by which congestion control algorithms can be judged. It was shown that this definition is compatible with TCP.

7. References

- [CDB00] A. Chaintreau, C. Diot, F. Baccelli. "Impact of Network Delay Variation on Multicast Sessions Performance with TCP-like Congestion Control". Work in progress, <http://www.sprintlabs.com/People/diot/publications.html>.
- [CJ89] D-M. Chiu and R. Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", Computer Networks and ISDN Systems, V. 17, pp.1-14, 1989.
- [DO97] D. DeLucia and K. Obraczka. A Congestion Control Mechanism for Reliable Multicast. IRTF meeting, Sept. 1997. <http://www.east.isi.edu/rm/delucia.ps>.
- [FF97] S. Floyd and K. Fall. "Router Mechanisms to Support End-to-End Congestion Control". Lawrence Berkeley Labs technical report, Feb 1997. <ftp://ftp.ee.lbl.gov/papers/collapse.ps>.
- [FJ92] S. Floyd and V. Jacobson. "On Traffic Phase Effects in Packet-Switched Gateways." Internetworking: Research and Experience, 3(3):115-156, September 1992.
- [FJ93] S. Floyd and V. Jacobson. "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.
- [FHPW00] S. Floyd, M. Handley, J. Padhye, J. Widmer. "Equation Based Congestion Control for Unicast Applications". Proceedings of SIGCOMM 2000, August 2000.
- [Floyd96] S. Floyd. "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic", Computer Communications Review, Vol.21, No.5, October 1991, p. 30-47.
- [HFW99] M. Handley, S. Floyd, B. Whetten, "Strawman Specification for TCP Friendly (Reliable) Multicast Congestion Control (TFMCC)," work in progress.
- [Jacob88] V. Jacobson. "Congestion Avoidance and Control", Proceedings of SIGCOMM 1988, pp. 314-328.
- [MF97] J. Mahdavi and S. Floyd. TCP-Friendly Unicast Rate-Based Flow Control". Technical note sent to the end2end-interest mailing list, 1997. <http://www.psc.edu/networking/papers>.
- [MRBP98] A. Mankin, A. Romanow, S.Brander, V.Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC2357, June 1998.
- [MSMO97] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm". ACM Computer Communication Review, 27(3):67-82, Jul. 1997.

- [OR99] V. Ozdemir, I. Rhee. "TCP Emulation At Receivers (TEAR)" Presentation made at IRTF RM meeting, Washington DC, Nov 1999.
- [PFTK98] J. Padhye, V. Firoiu, D. Towsley, J. Kurose. "Modeling TCP Throughput: A Simple Model and its Empirical Validation". University of Massachusetts Technical Report CMPSCI TR 98-008.
- [PS95] V. Paxson and S. Floyd. "Wide-Area Traffic: The Failure of Poisson Modeling". IEEE/ACM Transactions on Networking, Vol. 3 No. 3, pp. 226-244, June 1995.
- [RBR98] I. Rhee, N. Ballaguru, G. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast", Proceedings of INFOCOM 1999.
- [Rizzo00] L. Rizzo, "PGMCC: A TCP-friendly Single Rate Multicast Congestion Control Scheme", Proceedings of SIGCOMM 2000.
- [SES97] D. Sisalem, F. Emanuel, H. Schulzrinne. "The Direct Adjustment Algorithm: A TCP-Friendly Adaptation Scheme", Preprint, August 1997.
- [SYST98] T. Sano, N. Yamanouchi, T. Shiroshita, O. Takahashi. "Flow and Congestion Control for Bulk Reliable Multicast Protocols – Towards Coexistence with TCP". <http://info.isl.ntt.co.jp/chisho/rmtp/infocomm98.ps.gz>.
- [VRC98] L. Vicisano, L. Rizzo, J. Crowcroft. "TCP-like Congestion Control for Layered Multicast Data Transfer", Proceedings of IEEE INFOCOMM, April, 1998.
- [WC98] B. Whetten, J. Conlan. "A Rate Based Congestion Control Scheme for Reliable Multicast", GlobalCast Communications Technical White Paper, November 1998. <http://www.talarian.com/rmtp-ii>
- [WH01] J. Widmer, M. Handley. "Extending Equation-Based Congestion Control to Multicast Applications". Proceedings of SIGCOMM 2001.
- [Whetten00] B. Whetten. "Reliable Multicast Congestion Control and Tree Based Reliable Multicast". Dissertation, University of California at Berkeley. December 2000.
- [WS98] H. Wang and M. Schwartz. "Achieving Bounded Fairness for Multicast Traffic and TCP Traffic in the Internet", Proceedings of ACM SIGCOMM, September 1998.
- [WSF94] B. Whetten, S. Steinberg, D. Ferrari. "The Packet Starvation Effect in CSMA/CD LANs and a Solution", Proceedings of the 19th Conference on Local Computer Networks, pp. 206-217, October 1994.
- [WT00] B. Whetten, G. Taskale. "An Overview of the Reliable Multicast Transport II". IEEE Network, February 2000.